Rev. 1.0 — 1 October 2024

Application note

Document information

Information	Content
Keywords	AN14367, OpenOCD, Cortex-A debug, GUI, i.MX 93 EVK, i.MX 8M Plus, GDB, U-Boot debugging, RTOS debugging, Native RTOS, JLink, JTAG
Abstract	This application note describes how to debug the U-Boot and Native RTOS on Cortex A Core by using OpenOCD together with the command-line tool GDB or GUI tool Eclipse on i.MX 93 and i.MX 8M Plus EVK



1 Introduction

This application note focuses on the NXP MPU Cortex-A Core debugging method and flow. It provides two methods, one is to use the command-line debugging tool (GDB), the other one is to use the GUI debugging tool (Eclipse in this AN), and takes U-Boot and RTOS debugging as examples. There is no fixed binding relationship, and projects debugged using the JTAG flow can also be debugged using J-Link and vice versa.

i.MX 8M Plus EVK and i.MX 93 EVK provide a local JTAG port (a standard 10-pin JTAG header on the board is used for local debug, it is paralleled with JTAG from the FTDI chip) and a remote JTAG port (the A-bus of FT4232 is numerated as JTAG), so this AN provides two variants of hardware connection: a local JTAG port with the JLink debug probe or a remote debug port used directly.

This AN applies to:

- i.MX 8M Plus EVK and i.MX93 series EVK debugging
- U-Boot debugging
- With/without J-link debugging
- Cortex-A core RTOS debugging
- OpenOCD debugging
- GDB debugging
- OpenOCD + Eclipse debugging

2 Definitions, acronyms, and abbreviations

Table 1 describes the definitions, acronyms, and abbreviations used in this application note.

Table 1. Definitions, acronyms, and abbreviations

Acronyms	Meaning
AN	Application Note
BSP	Board Support Package
EVK	Evaluation Kit
GDB	GNU Project Debugger
GUI	Graphical User Interface
JTAG	Joint Test Action Group
MPSSE	Multi-Protocol Synchronous Serial Engine
OpenOCD	Open On-Chip Debugger
RTOS	Right Real-Time Operating System

3 Hardware and software setup

This chapter describes the software and hardware required for U-Boot and RTOS debugging that are to be prepared before proceeding to the next step.

3.1 Hardware materials

- For i.MX 8M Plus EVK
 - i.MX 8M Plus EVK board
 - Host PC with Ubuntu18/Ubuntu 20/ Ubuntu 22

- i.MX 8M Plus board power cable
- micro-USB debug cable
- USB-Type C cable
- Micro SD card
- (Only for J-Link debugging flow) A SEGGER J-Link debug probe
- For i.MX 93 EVK
 - i.MX93EVK board
 - Host PC with Ubuntu18/Ubuntu 20/ Ubuntu 22
 - i.MX93 EVK board power cable
 - USB-Type C debug cable
 - USB-Type C cable
 - Micro SD card
 - (Only for J-Link debugging flow) A SEGGER J-Link debug probe

3.2 Hardware connection

If a remote JTAG port is used, the following connection is enough for OpenOCD debug:

- 1. Insert the Micro SD card to the card slot on the EVK board (in this process, the image is booted from the Micro SD card);
- 2. Connect the USB debug cable to the debug port on the EVK board for debugging, connect the other end to the Host PC with Ubuntu. Do not use a USB hub to connect to avoid signal interference;
- 3. Connect the USB-Type C cable to the USB port on the EVK board for flashing the image, connect the other end to the host PC with Ubuntu;
- 4. Connect the power cable to the power interface.
- 5. (Optional) If J-Link is used, an extra step is to connect the J-Link probe to a local JTAG port. For details, refer to Figure 1



3.3 Software setup

A Linux PC host is needed to set up the debugging environment. Ubuntu 20 or Ubuntu 22 is recommended because the example in this AN has been verified on them.

3.3.1 Software setup on Ubuntu PC

Refer to <u>Table 2</u> to install the following software on an Ubuntu host PC: bcu, uuu, OpenOCD, and Tabby (or other serial console tools).

Install the GDB client for command-line debugging or Eclipse for GUI debugging. For J-Link debugging, install the J-Link probe driver and extra patch:

Table 2. Software requirements on Ubuntu PC

BCU	BCU is used for board debug interface enablement. The current version is 1.1.92.
	<pre>\$ wget https://github.com/nxp-imx/bcu/releases/download/bcu_1.1.92/bcu_ Ubuntu18no-check-certificate</pre>
	For Ubuntu_20:
	<pre>\$ wget <u>https://github.com/nxp-imx/bcu/releases/download/bcu_1.1.92/</u> <u>bcu_Ubuntu20</u>no-check-certificate For Ubuntu 22:</pre>
	For Obunu_22.
	bcu_Ubuntu22
	The latest bcu is available at the following link: <u>https://github.com/nxp-imx/bcu/releases.</u>
OpenOCD	OpenOCD is used for debugging remote targets that work with GDB. There are cfg files for i.MX 8M Plus, but for i.MX 93, the patch must be applied.
	<pre>\$ git clone https://github.com/openocd-org/openocd.git \$ sudo apt-get install make libtool pkg-config autoconf automake texinfo</pre>
	<pre>\$ sudo apt-get install libusb-1.0-0-dev libitdi-dev libitdil-2 \$ sudo apt-get install autotools-dev build-essential swig cmake python-dev libconfuse-dev libboost-all-dev libtool-bin libjaylink- dev</pre>
	<pre>\$ cd openocd \$ git checkout 12ff36bd19e4f25dd7505c46a77d9f2c47dc350a \$./bootstrap \$./configureenable-ftdienable-openjtagenable-jlink prefix=/usr/local/share \$ make \$ make \$ sudo make install</pre>
	Save the following debugging scripts from the <u>OpenOCD 93 debugging scripts</u> and <u>OpenOCD 8MP</u>
	<u>debugging scripts</u> to the specified directory. Note: By default, OpenOCD debugging scripts are installed in the target or board directory. Debugging script nxp_imx93-evk-reset.cfg includes an additional initial reset that is needed when a rebuilt image is downloaded and debugged.
	<pre># cp -a imx93.cfg openocd/tcl/target/imx93.cfg # cp -a nxp_imx93-evk.cfg openocd/tcl/board/nxp_imx93-evk.cfg # cp -a nxp_imx93-evk-reset.cfg openocd/tcl/board/nxp_imx93-evk- reset.cfg # cp -a nxp_imx8mp-evk.cfg openocd/tcl/board/nxp_imx8mp-evk.cfg # cp -a imx8mp.cfg openocd/tcl/target/imx8mp.cfg</pre>
GDB	GDB is the main debugging tool that can be installed by using the apt command.
	\$ sudo apt install -y gdb-multiarch

Table 2. Sollware requirements on Obuntu PGcontinu	Table 2.	Software	requirements	on Ubur	tu PCcontinue
--	----------	----------	--------------	---------	---------------

uuu	<pre>uuu is used for the BSP image or bootloader image downloading, the current version is 1.5.125. \$ wget https://github.com/nxp-imx/mfgtools/releases/download/ uuu_1.5.125/uuu \$ chmod +x uuu</pre>
Eclipse IDE	Eclipse IDE is the GUI platform, download, and install Eclipse IDE on Ubuntu. The current test software version is 4.29.0. The download link is: <u>https://www.eclipse.org/downloads/</u> .
Tabby	Tabby is a serial console tool on Ubuntu. The download link is: <u>https://tabby.sh/</u> . You can also use your favorite console tool.
J-Link Debugging	flow (Optional):
J-Link driver and patch	 Download the J-Link driver (J-link Version: V7.96) from https://www.segger.com/downloads/jlink/. Install the driver: # dpkg -i JLink_Linux_V796f_x86_64.deb. Download the SEGGER J-Link patch for i.MX 93 and save it in directory /opt/SEGGER/JLink. Note: For step-to-step guidelines, refer to readme in the downloaded folder.
libjaylink	If "libjaylink" is not present during building, install the library manually by following the steps below, then build OpenOCD again:
	<pre>\$ git clone https://gitlab.zapb.de/libjaylink/libjaylink.git \$ cd libjaylink \$./autogen.sh \$./configure \$ make && sudo make install</pre>

4 U-Boot OpenOCD debugging

This chapter introduces the method of using JTAG for U-Boot debugging.

4.1 OpenOCD Debugging with JTAG Introduction

Most i.MX 8 and i.MX 9 series EVKs are equipped with a Joint Test Action Group (JTAG) interface and an FTDI chip allowing debugging for Cortex-A core and Cortex-M core. Open On-Chip Debugger (OpenOCD) is a tool that utilizes the JTAG interface to perform chip debugging. J-link is not needed under this combination. For i.MX 93 EVK (taken as an example), the general debugging process outline is:

- 1. The i.MX 93 EVK has an FTDI 4232H chip that has Multi-Protocol Synchronous Serial Engine (MPSSE) supporting USB signals to JTAG signals;
- 2. bcu controls RC_nSEL to enable FTDI chip to transmit JTAG signal through USB debug cable;
- 3. OpenOCD starts running as a server waiting for the connections from GDB or Telnet clients and handling the commands issued through those channels;
- 4. Users use the GDB client to connect to the server, through the corresponding port(3333 for Cortex-A on i.MX 93 by default) for debugging.



4.2 Debugging U-Boot with Eclipse IDE

This chapter introduces the preparation work before starting, which is necessary. By executing it step by step, unnecessary troubles can be avoided. The basic steps of debugging are described in the later sections.

4.2.1 Software requirements

Table 5. Soltwal	e requirements			
Build (get) the Linux BSP	Linux BSP release L6.1.36_2.1.0 is used in this application note. Specifically:			
image	• i.MX 8M Plus: imx-image-full-imx8mpevk.wic			
	• i.MX 93: imx-image-full-imx93evk.wic			
	Download the official release BSP from nxp.com or refer to <u>i.MX Yocto Project User's Guide (nxp.com)</u> to build on a host Ubuntu PC by yourself.			
Download the Linux BSP	Method 1: download it to the board using uuu (change the bootmode to serial download mode and use a USB-Type C cable):			
image	<pre>\$ sudo ./uuu -b sd imx-image-full-imx8mpevk.wic # for 8mp \$ sudo ./uuu -b sd imx-image-full-imx93evk.wic # for 93</pre>			
Method 2: flash it to the SD card directly:				
	<pre>\$ sudo dd if=.wic of=/dev/sd[x] bs=1M status=progress conv=fsync</pre>			
	Note: Check your card reader partition and replace sd[x] with your corresponding partition.			
Setup U-Boot	U-Boot is the project that is debugged.			
	Before building the U-Boot:			
	 Use your cross-compile toolchain, refer to <u>i.MX Linux User's Guide (nxp.com</u>) for more toolchain information. 			
	2. Choose one of two defconfig files (for 8MP or for 93).			
	<pre>\$ git clone https://github.com/nxp-imx/uboot-imx.git</pre>			
AN14367	All information provided in this document is subject to legal disclaimers. © 2024 NXP B.V. All rights reserved.			

Table 3. Software requirements

Table 3. Software requirements...continued

	\$ cd uboot-imx
	\$ git checkout lf-6.1.36-2.1.0
	<pre>\$ source /opt/fsl-imx-internal-xwayland/6.1-langdale/</pre>
	environment-setup-armv8a-poky-linux
	<pre>\$ make imx8mp_evk_defconfig # for 8MP</pre>
	<pre>\$ make imx93_11x11_evk_defconfig # for 93</pre>
	\$ make
Build the	Build flash.bin using the built files from the U-Boot project in the imx-mkimage project,
bootloader	including:
	• u-boot.bin
	• spl/u-boot-spl.bin
	The Building must be done on the host Ubuntu PC. The command in imx-mkimage is:
	<pre>\$ make SOC=iMX8MP flash_evk # for 8mp</pre>
	<pre>\$ make SOC=iMX9 REV=A1 flash_singleboot_m33 # for 93</pre>
	Pofer to: i MX Linux Lloor's Guide for more fleach, bin building information
Download the bootloader	Method 1: Download it to the board by using uuu (change the bootmode to serial download mode and use a USB-Type C cable):
	\$ sudo ./uuu -b sd flash.bin
	Method 2: Flash it to the SD card directly:
	<pre>\$ sudo dd if=flash.bin of=/dev/sd[x] bs=1k seek=32</pre>
	Note: Check your card reader partition and replace sd[x] with your corresponding partition.

4.2.2 Eclipse OpenOCD Configuration for U-Boot debugging

In terms of usage, OpenOCD can be used in the terminal as an independent tool or integrated into certain GUI software, for example, Eclipse. Compared to terminal debugging, the benefits of integrating into a GUI include:

- · Viewing the call stack and variables more intuitively;
- Setting the breakpoints and single-step debug in the code directly;
- Using the advantages of the GUI platform to view and modify the code more conveniently.

The following are Eclipse OpenOCD Configuration steps:

1. Open Eclipse IDE, select a directory as a workspace, then click Launch.

AN14367

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs

Ecu	pse IDE Launcher ×
Select a directory as workspace	
Eclipse IDE uses the workspace directory to	store its preferences and development artifacts.
	1
/home/mpuse/eclipse-u-boot-debug	V Browse
Use this as the default and do not ask ag	jain
 Recent Workspaces 	(2)
	Cancel
	Cancer

2. In the main menu, left-click File and then click Open Projects from Files Systems.

AN14367

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs

(I)	ecli	pse-u-boot-debug - Eclipse IDE		(×
Eile Edit Source Refactor	Navigate Search Project	Run Window Help					
New	Shift+Alt+N →	• • • • • • • • • • • • • • • • • • •	· 🍃 🗀 🔗 🗸 🔛 🔲	π : 🗨)Ø	ð	
Open File	2				. :		
🥥 Open Projects from File	System				~ :	8	
Recent Files	>						6
Close Editor	Ctrl+W						E
Close All Editors	Shift+Ctrl+W						
🗐 Save	Ctrl+S						
🖳 Save As							
🐚 Save All	Shift+Ctrl+S						
Revert							
Move							
🖻 Rename	F2						
🐑 Refresh	F5						
Convert Line Delimiters To	>						
👜 Print	Ctrl+P						
🔤 Import							
🖾 Export							
Properties	Alt+Enter						
Switch Workspace	>						
Restart							
Exit	J						
🖹 Problems 🗙 繥 Tasks 📮 C	onsole 🔲 Properties 🕮 Call	Graph	7	6 9 8	3 -		
0 items							
Description	Resource	e Path Location	п Туре				

Figure 4. Open Projects from Files System or Archive

3. On the **Import Projects from Files System or Archive** tab, click **Directory...**. To browse the folder, choose the path of the U-Boot source code. Confirm that the project already includes this uboot-imx folder, then click **Finish**.

AN14367

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs

e	clipse-u-boot-debug - Eclipse IDE	- • ×
File Edit Source Defactor Navigate Search Brois	ect - Dup - Window - Help rt Projects from File System or Archive	×
☆ Import Projects from File System or Archiv	10	
This wizard analyzes the content of your folder o	or archive file to find projects and import them in t	he IDE.
Import source: //home/mpuse/openocd_eclip	ose_test/uboot-imx	1 Directory Archive
type filter text		Select All
Folder	Import as	Deselect All
l uboot-imx		1 of 1 selected
Use installed projects open completed projects upon completed projects to:	uon	
Working sets		New
Working sets:		~ Select
	Show	other specialized import wizards
Probl items		2
?	< Back Next >	Cancel Finish
Figure 5. Choose uboot-imx source code di	rectory	

4. Click Run, then click Debug Configurations.

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs

eclips	e-u-boot-debug - Eclipse IDE				×
File Edit Source Refactor Navigate Search Project	Run Window Help				
🗂 🗸 🗐 🐚 🕲 🗸 🗞 v 🖉 🖓 🍙 v 🚳 v 🕤 v	🗞 Run Last Launched	Ctrl+F11	} : ₽ 🗉 T : 📮	0	5
	🎋 Debug Last Launched	F11		0 : =	
	🚱 Profile Last Launched			≪ :⊞	
	Profile History	>			3 8
	🚱 Profile As	>			
	Profile Configurations				۲
	Run History	>			-
	Run As	>			
	Run Configurations				
	Debug History	>			
	🎋 Debug As 🛛 🕗	>			
	Debug Configurations				
	Breakpoint Types	>			
	 Toggle Breakpoint 	Shift+Ctrl+B			
	 Toggle Line Breakpoint 				
	🌮 Toggle Watchpoint				
	 Toggle Method Breakpoint 				
	Skip All Breakpoints	Ctrl+Alt+B			
	🗞 Remove All Breakpoints		-		
	🤷 External Tools	>	J		
					_
Problems Z Tasks Console X Properties in Call G	raph			~ _ [
No consoles to display at this time.					_
Figure 6. Open the Debug Configurations tab					

5. Create a GDB OpenOCD Debugging interface and name the configuration in Name as shown in Figure 7 In the Main tab in Project, name the project, in C/C++ Application, select the U-Boot file compiled from the U-Boot source code.

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs

	Debug Config	gurations	>
reate, manage, and run configurations			1
3 B 🕫 🖿 🗙 🖻 🏹 🗸	1) Name: uboot-imx-debug		2
type filter text	🗍 Main 🌶 Debugger 🕨 Startup 🦻 Source 🔲 Commo	on 🔀 SVD Path	
 C/C++ Application C/C++ Attach to Application C/C++ Container Launcher C/C++ Postmortem Debugger C/C++ Remote Application 	Project: (boot_openocd) C/C++ Application: /home/mpuse/openocd_eclipse_test/uboot-imx/u-boo	t	Browse
C CC/C++ Unit C GDB Hardware Debugging C GDB OpenOCD Debugging Ubbot-imx-debug	Build (if required) before launching	Variables Search Proj	ect Browse
© GDB PyOCD Debugging © GDB QEMU aarch64 Debugging © GDB QEMU arm Debugging	Enable auto build Use workspace settings	O Disable auto build Configure Workspace Settings	•
© GDB QEMU gnuarmeclipse Debugging (Dep © GDB QEMU riscv32 Debugging © GDB QEMU riscv64 Debugging © GDB SEGGER J-Link Debugging © Launch Group			
ilter matched 18 of 18 items		Rever	t Apply
2		Clos	e Debug

Figure 7. Configure the main tab

6. In the **Debugger** tab, OpenOCD setup section, click the **Browse...** button, select the built openocd location. Enter the following commands in **Config Options**.

Note: Modify the command according to your actual path: For i.MX 8M Plus

-f
<pre>/home/mpuse/openocd eclipse test/openocd/tcl/interface/ftdi/imx8mp-evk.cfg -f</pre>
/home/mpuse/openocd_eclipse_test/openocd/tcl/board/nxp_imx8mp-evk.cfg -s
/home/mpuse/openocd_eclipse_test/openocd/tcl/

For i.MX 93

-f

```
/home/mpuse/openocd_eclipse_test/openocd/tcl/interface/ftdi/imx93-evk.cfg -f
/home/mpuse/openocd_eclipse_test/openocd/tcl/board/nxp_imx93-evk.cfg -s
/home/mpuse/openocd_eclipse_test/openocd/tcl/
```

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs

		Debug Configurations	
reate, manage, and run configurations			Ś
C 🖻 🕫 🗎 🗶 🖻 🍸 🗸	Name: uboot-imx-d	ebug	
type filter text	🖹 Main 🎯 Debugge	🌮 🕨 Startup 🦻 Source 🔲 Common 🔀 SVD Path	
C/C++ Application C/C++ Attach to Application	OpenOCD Setup	locally	1
C/C++ Container Launcher	Executable path:	/home/mpuse/openocd_eclipse_test/openocd/src/openocd	Browse Variables
 C/C++ Postmortem Debugger C/C++ Remote Application 	Actual executable:	/home/mpuse/openocd_eclipse_test/openocd/src/openocd	
^C ⁱⁱ C/C++ Unit			
GDB Hardware Debugging	GDB port:	3333	
GDB OpenOCD Debugging Uboot-imx-debug	Telnet port:	4444	
GDB PyOCD Debugging	Tcl port:	0000	
GDB QEMU aarch64 Debugging GDB QEMU arm Debugging GDB QEMU arm celipse Debugging GDB OEMU gnuarmeclipse Debugging (Depr	Config options:	-f/home/mpuse/8MP_openocd/openocd/tcl/interface/ftdi/imx8mp-evk.cfg -f/home/mpuse/8MP_ imx8mp-evk.cfg -s /home/mpuse/8MP_openocd/openocd/tcl/	openocd/openocd/tcl/board/
GDB QEMU riscv32 Debugging	Allocate conso	le for OpenOCD Allocate console for the telnet conne	ction
GDB QEMU riscv64 Debugging	GDB Client Setup		
GDB SEGGER J-Link Debugging	Start GDB sess	ion	
Lauren Group	Executable name:	/usr/bin/gdb-multiarch	Browse Variables
			Revert Apply
lter matched 18 of 18 items			
3)			Close
2			Debug

Figure 8. Configure the Debugger tab - 1

7. In the GDB Client Setup section, click Browse to choose the GDB (use which gdb-multiarch to check the gdb path)

	Debug Configurations	×
Create, manage, and run configurations		- A
Image: Construction Image: Cycle+ Application Image: Cycle+ Attach to Application Image: Cycle+ Attach to Application Image: Cycle+ Attach to Application Image: Cycle+ Container Launcher Image: Cycle+ Postmortem Debugger Image: Cycle+ Postmortem Debugging Image: Cycle+ Debuge: Cycle+ Debugging Image: Cycle+ Debuge: Cycle+ Debuge: Cycle+ Debuge: Cycle+ Debuge: Cycle+ Debuge	Name: ubootimx-debug Main Debugger Startup Source Common So SVD Path Allocate console for OpenOCD Allocate console for the telnet connection OBE Client Setup Start CDB session Executable name: /usr/bin/gdb-multiarch Actual executable: /usr/bin/gdb-multiarch Other options: Commands: commands: set mem inaccessible-by-default off Remote Target Host name or IP address: Host name or IP address: localhost Port number: 3333 Force thread list update on suspend	Browse Variables Browse Variables Restore defaults
Filter matched 17 of 17 items	Rey	Арріу
0	c	lose Debug

Figure 9. Configure the Debugger tab - 2

8. In the Startup tab, click load Symbol and Executable.

- Make sure that Initial Reset has been selected, and the Type is init.
- Check the Load Symbols, then use the U-Boot compiled from the U-Boot source folder.
- The Symbol offset(hex) offset is necessary. The value can be got at the U-Boot stage:

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs

Create, manage, and run configurations		Debug Configurations	×
1	e, and run configurations		Ť.
Image: Status Image: Status<	X Image: Solution of the constraint of the constrated of the constraint of the constraint of the constraint	ubootimx:debug n Source Common Source Comm	File System
Filter matched 17 of 17 items	7 of 17 items	Revert	Apply
Close Debug		Close	Debug

Figure 10. Configure the Startup tab - 1

9. Type bdinfo at the U-Boot stage to check the relocaddr and use it as an offset.

AN14367

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs

```
[*]-Video Link 0adv7535_mipi2hdmi hdmi@3d: Can't find cec device id=0x3c
fail to probe panel device hdmi@3d
fail to get display timings
probe video device failed, ret -19
[0] lcd-controller@4ae30000, video
[1] dsi@4ae10000, video_bridge
[2] hdmi@3d, panel
adv7535_mipi2hdmi hdmi@3d: Can't find cec device id=0x3c
fail to probe panel device hdmi@3d
fail to get display timings
probe video device failed, ret -19
tre: coriol
In: serial
Out: serial
         serial
BuildInfo:
   - ELE firmware version 0.0.16-44880904
switch to partitions #0, OK
mmc1 is current device
UID: 0x48c94f17 0x4049810c 0xceec07ab 0xc3e4b276
Net: eth0: ethernet@42890000, eth1: ethernet@428a0000 [PRIME]
Fastboot: Normal
Normal Boot
Hit any key to stop autoboot: 0
=> bdinfo
= 0x000000008000000
baudrate = 115200 bps
relocaddr = 0x00000000feef8000
reloc off = 0x000000007ecf8000
Build = 64-bit
current eth = ethernet@428a0000
ethaddr = 00:04:9f:07:b2:8b
IP addr = <NULL>
fdt_blob = 0x00000000fceebf90
new_fdt = 0x00000000fceebf90
fdt_size = 0x000000000000000000
                    = lcd-controller@4ae30000 inactive
Video
lmb_dump_all:
 cmb_admp_att:
memory.cnt = 0x1
memory[0] [0x800000000-0xffffffff], 0x800000000 bytes flags: 0
reserved.cnt = 0x2
reserved[0] [0xa4120000-0xa421ffff], 0x001000000 bytes flags: 4
reserved[1] [0xfcee7b10-0xffffffff], 0x031184f0 bytes flags: 0
dvicetres
sp start = 0x00000000fceebf80
Early malloc usage: 15ee8 / 18000
```

Figure 11. Type bdinfo to obtain the relocaddr value

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs

	Debug Configurations 5
Create, manage, and run configurations	
Image: Construction C/C++ Application C/C++ Attach to Application C/C++ Attach to Application C/C++ Perstore Launcher C/C++ Perstore Application C+/C++ Perstore Debugging C GDB Hardware Debugging C GDB OpenOCD Debugging C GDB QEMU arch64 Debugging C GDB QEMU arch64 Debugging C GDB QEMU and Debugging C GDB QEMU arch64 Debugging C GDB QEMU siscv64 Debugging C GDB QEMU riscv32 Debugging C GDB QEMU riscv32 Debugging C GDB QEMU riscv32 Debugging C GDB SEGGER J-Link Debugging C HU riscv32 Debugging C HU riscv32 Debugging C GDB SEGGER J-Link Debugging C HU riscv32 Debugging C HU riscv44 C HU riscv44 <tr< th=""><th>Name: ubootimx-debug Main * Debugger > Startup > Source Common > SVD Path Duse project binary: home/mpuse/openocd_eclipse_test/93/uboot-imx/u-boot Use file: Executable offset (hex): Runtime Options Debug in RAM Run/Restart Commands Pre-run/Restart reset Type: halt (always executed at Restart) Pre-run/Restart reset Type: halt (always executed at Restart) Restore defaults Restore defaults Revert Apply</th></tr<>	Name: ubootimx-debug Main * Debugger > Startup > Source Common > SVD Path Duse project binary: home/mpuse/openocd_eclipse_test/93/uboot-imx/u-boot Use file: Executable offset (hex): Runtime Options Debug in RAM Run/Restart Commands Pre-run/Restart reset Type: halt (always executed at Restart) Pre-run/Restart reset Type: halt (always executed at Restart) Restore defaults Restore defaults Revert Apply
0	Close Debug
Figure 12. Configure the S	tartup tab - 2

Run/Restart Commands

- · Make sure Pre-run/Restart reset has been selected, and the type is halt.
- Set program counter at (hex), Set breakpoint at, Continue do not need to be selected.
- Click Apply and Close.

4.2.3 Eclipse debugging steps

Note: Disconnect other boards with the FTDI JTAG interface to avoid cross-impact.

- 1. Open the serial console software, choose the third COM (Cortex-A);
- 2. Power on the board through the ON/OFF switch;
- 3. Stop at the U-Boot stage, check the time when you build U-Boot to make sure builds u-boot.bin and u-boot-spl.bin are used.

AN14367

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs

U-Boot SPL 2022.04 (Mar 14 2024 - 16:05:16 +0800) DDRINFO: start DRAM init DDRINFO: DRAM rate 4000MTS DDRINFO:ddrphy calibration done DDRINFO: ddrmix config done SEC0: RNG instantiated Normal Boot Trying to boot from BOOTROM Boot Stage: Primary boot image offset 0x8000, pagesize 0x200, ivt offset 0x0 NOTICE: Do not release JR0 to NS as it can be used by HAB NOTICE: BL31: v2.8(release):android-14.0.0_1.0.0-rc1-1-g08e9d4eef NOTICE: BL31: Built : 06:43:30, Nov 21 2023 U-Boot 2022.04 (Mar 14 2024 - 16:05:16 +0800) i.MX8MP[8] rev1.1 1800 MHz (running at 1200 MHz) CPU: Commercial temperature grade (OC to 95C) at 35C CPU: Reset cause: POR Model: NXP i.MX8MPlus LPDDR4 EVK board DRAM: 6 GiB TCPC: Vendor ID [0x1fc9], Product ID [0x5110], Addr [I2C2 0x50] SNK.Power3.0 on CC2 PDO 0: type 0, 5000 mV, 3000 mA [E] PDO 1: type 0, 9000 mV, 3000 mA [] PDO 2: type 0, 15000 mV, 3000 mA [] PDO 3: type 0, 20000 mV, 2250 mA [] Requesting PDO 3: 20000 mV, 2250 mA Source accept request PD source ready! tcpc_pd_receive_message: Polling ALERT register, TCPC_ALERT_RX_STATUS bit failed, ret = -62 Power supply on USB2 TCPC: Vendor ID [0x1fc9], Product ID [0x5110], Addr [I2C1 0x50] Core: 203 devices, 30 uclasses, devicetree: separate MMC: FSL_SDHC: 1, FSL_SDHC: 2 Loading Environment from MMC... *** Warning - bad CRC, using default environment [*]-Video Link Oprobe video device failed, ret -2 [0] lcd-controller@32e80000, video
[1] mipi_dsi@32e60000, video_bridge
[2] adv7535@3d, panel mipi_dsi@32e60000, video_bridge probe video device failed, ret -2 In: serial Figure 13. Check the SPL and U-Boot build time

4. Enable the JTAG remote debug interface with bcu:

sudo ./bcu_Ubuntu20 set_gpio remote_en 1 -board=imx8mpevk
sudo ./bcu_Ubuntu20 set_gpio remote_en 1 -board=imx93evk11b1

Note: When the board must be restarted, first use BCU to disable the JTAG interface. If DEBUG is still needed, re-enable the JTAG interface.

- 5. (Optional) Disable the JTAG remote debug interface with BCU:
- Attention: After running the following commands, JTAG remote debug will no longer be available.

sudo ./bcu_Ubuntu20 set_gpio remote_en 0 -board=imx8mpevk
sudo ./bcu_Ubuntu20 set_gpio remote_en 0 -board=imx93evk11b1

6. Debugging pathway test

Before using OpenOCD GUI debugging, use OpenOCD on a terminal to check if the debugging pathway is available.

```
$ sudo src/openocd -s ./tcl -f interface/ftdi/imx93-evk.cfg -f board/
nxp imx93-evk.cfg
Open On-Chip Debugger 0.11.0+dev-00651-g9de084e00 (2022-04-26-15:55)
Licensed under GNU GPL v2
For bug reports, read
http://openocd.org/doc/doxygen/bugs.html
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
Info : clock speed 1000 kHz
Info : JTAG tap: imx93.cpu tap/device found: 0x0892801d (mfg: 0x00e
 (Freescale (Motorola)), part: 0x8928, ver: 0x0)
Info : imx93.a55.0: hardware has 6 breakpoints, 4 watchpoints
Info : starting qdb server for imx93.a55.0 on 3333
Info : Listening on port 3333 for gdb connections
Info : starting gdb server for imx93.m33 on 3334
Info : Listening on port 3334 for gdb connections
Info : qdb port disabled
```

7. Open a new terminal to test the GDB connection

```
$ gdb-multiarch
(qdb) set architecture auto
(gdb) target remote localhost:3333
Remote debugging using localhost:3333
warning: No executable has been specified and target does not support
determining executable automatically. Try using the "file" command.
0x0000000feefa200 in ?? ()
(qdb) symbol-file u-boot
Reading symbols from u-boot...
(gdb) set $offset = ((gd t *)$x18)->relocaddr
(qdb) symbol-file
Discard symbol table from `/home/mpuse/openocd eclipse test/93/uboot-imx/u-
boot'? (y or n) y
No symbol file now.
(qdb) add-symbol-file u-boot $offset
add symbol table from file "u-boot" at
 .text addr = 0xfeef8000
(y or n) y
Reading symbols from u-boot...
(gdb) bt
#0 ?? () at arch/arm/cpu/armv8/exceptions.S:139
#1 0x000000000000000 in ?? ()
```

8. (Optional) Open another terminal to test the Telnet connection,

```
$ telnet localhost 4444
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Open On-Chip Debugger
> resume
> halt
imx93.a55.0 halted in AArch64 state due to debug-request, current mode: EL2H
cpsr: 0x200002c9 pc: 0xfef4419c
MMU: enabled, D-Cache: enabled, I-Cache: enabled
> step
imx93.a55.0 halted in AArch64 state due to single-step, current mode: EL2H
cpsr: 0x200002c9 pc: 0xfef441a0
```

```
MMU: enabled, D-Cache: enabled, I-Cache: enabled
> step
imx93.a55.0 halted in AArch64 state due to single-step, current mode: EL2H
cpsr: 0x200002c9 pc: 0xfef441a4
MMU: enabled, D-Cache: enabled, I-Cache: enabled
> step
imx93.a55.0 halted in AArch64 state due to single-step, current mode: EL2H
cpsr: 0x200002c9 pc: 0xfef441a8
MMU: enabled, D-Cache: enabled, I-Cache: enabled>
```

- 9. If the previous step works well, it confirms the pathway test pass. To run the Eclipse OpenOCD debugging:
 - Close the terminal windows in step5;
 - Reboot the EVK and let it stop at the U-Boot stage again;
 - Make sure the Eclipse OpenOCD configuration is finished.

If so, click the **debug** button, wait a few seconds for Eclipse to start the server, connect the target and start the debugging session.

							eclip	se-u-b	oot-debu	ug - Ec	lipse II	DE						-			×
Fil	e	Edit	Source	Refactor	Navigate	Search	Project	Run	Window	Help											
1	} ~	·	6 8	~ % ~	4 6 :	🖆 ~ 😂	~ 🖻 ~	@ ~		0 ~	₽ ~	9	- 1 🤌	Þ	🔗 🗸 i	R I	Π	₿ :)Ø	ð	
: 2	~		~ 🏷 🗘		\sim													Q	. 1	E	
																					8

Figure 14. Debug button

• In case of a similar situation, press the **Resume** button as shown in Figure 15

	93test1 - Source not found Eclipse IDE	. ø ×
File Edit Navigate Search Project Run Window Help		
9. @ # N = II 🖉 # II 🖬 II = N 3. 9. A	19 馬 武 🕹 株 < 0 < 9 < 1 😂 / 株 < 1 😂 🖋 < 1 🖄 < 初 < 1 🖉 < 1 < 1 < 1	Q i 🖻 🗟 🏘
🂠 Debug 🗙 🔁 Project Explorer 🦯 🖻 🖹 😫 🔋 🗖	© 0xfef449f4 x □ serial_lpuart.c □ console.c □ generic_timer.c □ board_r.c □ time.c "22 □ □	🚥 Variabl 💁 Breakp 🗙 🛠 Expres 🐨 Disass 🚼 Periph 👘 🗖
✓	Break at address "0xfef449f4" with no debug information available, or outside of program code.	# 🔆 🖉 🕾 🗷 💌 🖽 📚 😫
 ² P-boot ² 	View Disassembly	
Some state of the state of		
≣ oxo ⊾i openocd ⊾i gdb-multiarch	Configure when this editor is shown Preferences	
	🖾 Console 🗙 🗰 Registers 😰 Problems 🕡 Executables 🙀 Debugger Console 🚺 Memory	• X % & B @ @ Ø Ø = O ~ C ~ = D
	<pre>ubookimw.debug(CDB OpenoCCD Debugging) Info : [ims3].abb Examination succeed Info : istarting gdb server for ims3].a55.8 on 3333 Info : Listening on port 333 for gdb connections Info : istarting gdb server for ims3].a53 on 3344 Info : Listening on port 334 for gdb connections Info : percepting 'gdb' connection on tcp/333 Info : New Gdb Connection. I. Target ims3].a53.0, state: halted Warm : Prefer GdB command 'target extended-remote :3333' instead of 'target remote :3333' Error: Can't assert SRT: InSRT signal is not defined Info : Defering arp examine of ims3].a53.1 Info : Defering arp examine of ims3].a54.1 Info :</pre>	(9)
		5

Figure 15. Resume

- Resume the debugging process
- Press Suspend

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs

	93test1 - Source not found Eclipse IDE	- ø x
File Edit Navigate Search Project Run Window Help		
	1+馬派 ②[株×0×9 ₄ ×]島 ダ×目目 > 別 > ひ ひ ひ ひ へ ○ □	Q i 🖻 🗟 🕸
🎄 Debug 🗙 🍋 Project Explorer 🛛 🎘 🐘 🖇 😑 🗖	© 0xfef449f4 x □ serial lpuart.c □ console.c □ generic timer.c □ board r.c □ time.c "12 □ 0 Wariabl • Breakp x ☆ Expres # D	isass 😤 Periph 😐 🗖
~ 🗈 uboot-imx-debug [GDB OpenOpD Debugging]	Break at address "Oxfef449f4" with no debug information available, or outside of program code. 🏾 🗮 👻	
~ @u-boot	View Dicaseembly	
Thread #1 (Running: User Request)	view obsessering	
⊨ii openoco ⊨ii odb-multiarch	Configure when this editor is shown Preferences	
	🖸 Console 🗴 🏙 Registers 😭 Problems Q Executables 🖶 Debugger Console 🚺 Memory 👛 🕷 🌸 📓 🐼 🕼 🖉	
	uboot-imx-debug [CDB OpenOCD Debugging]	
	<pre>Inf : Ims3.acb it semination succeed Inf: itariing gdb server for ims3.acb.en a333 Inf: itariing gdb server for ims3.acb.en a334 Inf: itariing gdb server for ims3.acb.en a334 Inf: itariing gdb server for ims3.acb.en a334 Inf: itariing gdb server for ims3.acb.en as a set itariing a server for ims3.acb.en as a set itariing a server for se</pre>	
		5
Figure 16. Suspend		

• Halt the debugging process

10. If everything goes well, you can see:

Figure 17. The main debug interface

On Figure 17, the sections are as follows:

- a. The Call stack tab is on the left
- b. The **Source code** tab is in the middle.
- c. The Variables tab is on the right.
- d. The **Console log** window is at the bottom. To see different information, switch tabs. For example, you can see the breakpoint you set on the left.

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs



Figure 19. Disassembly tab on the right side of main interface

e. Variable information can be accessed by hovering the mouse over a variable. *Note: For the errors in this region:*

Error 1: Cannot assert SRST: nSRST signal is not defined. We did not define nSRST in the cfg files, but the nSRST gpio does indeed exist, it is behind an I2C GPIO expander. Use bcu to reset the chip: bcu reset sd/emmc/usb/... -board=imx93evk11

Error 2: [imx93.m33] The target is not examined, it will not halt after reset! There is no m33 image in our build flash.bin, this error will be reported.

11. (Optional) You can use terminal debug mode, press **Window**, **Show View**, and choose **Debugger Console**



12. (Optional) The terminal debugging can be done in **Debugger Console**, like this:

b board_init_f c



Figure 21. Use GDB command in Debugger Console

 (Optional) Click the button to switch the Console. This gdb traces window contains lots of gdb configuration information, for example, the log in the red box corresponding to the setting in the **Startup** tab, use it to debug your customized configuration.

File Edit Source Refactor Navigate Search Project Run	Window Help	
3. 전 2. 전 4. 8. 10 10 10 10 10 10 10 10 10 10 10 10 10	1+ 특 값 값 数 + ★ × 0 × 4 × 1 ★ ☆ ☆ × 1 → ☆ 1 월 × 전 × ♡ ♡ ♡ ◊ × ○ × 2	Q. i 🖻 🖷 🕸
🏚 Debug 🗙 🔂 Project Explorer 🛛 📄 🗮 🔹 📟 🖽	@ serial_lpuart.c @ stdio.c □ cli_readline.c @ board_init_f() □ serial_lpuart.c x □ cli_hush.c *s ** **	👓 Variabl 🎭 Breakp 😤 Expres 🔡 Disass 🧏 Periph 🗴 👘 🖽
Chickey & Chickey Explore Chickey & Chickey & Chickey & Chickey & Chickey Chickey & Chickey & Chickey & Chickey & Chickey & Chickey Explore Chickey & Chickey & Chickey & Chickey & Chickey Explore Chickey & Chickey & Chickey & Chickey & Chickey Chickey & Chickey & Chickey & Chickey & Chickey & Chickey Chickey & Chickey & Chickey & Chickey & Chickey & Chickey Explore Chickey & C	If If (Figs L Dust Tube (Forward) and L Dust (L Dust) Seriel (Just C) C (Just C	In Variabl % Breaks of Epores III Dians (), Ave X = 1 Registeral Address Description

Figure 22. GDB traces

14. Click the **Suspend** button above the IDE to pause and start step-by-step debugging. From the left to right, these buttons are **Resume**, **Suspend**, **Terminate**, **Disconnect**, **StepInto**, **StepOver**, **Step Return**.

File Edit <u>S</u> ource Refactor Navigate Search Project Run Windo	w Help)
i 🖆 🗸 🐘 💀 i 😌 i 🖳 i 🖉 i 🖉 i 🖉 👘 💷 🖬 🖏 🖓 iè 🗮	T	& : ☆ ~ Ο
☆ Debug × A Project Explorer Suspend i→ i→ i→ i→ i→ i→ i→ i→ i→ i	s ex	ceptions.S
✓ ☑ uboot-imx-debug [GDB OpenOCD Debugging]	358	return G
∼ 🎛 u-boot	359	}
✓ Phread #1 [imx8mp.a53.0] 1 (Name: imx8mp.a53.0, state: debug-r	e 361	static int m
mxc_serial_pending() at serial_mxc.c:365 0xfef34b3c	362	{
fgetc() at console.c:513 0xfef0fd78	363	struct m
cread line() at cli readline.c:286 0xfef0f044	€ 365	uint32 t
<pre>cli readline into huffer() at cli readline c:468 0xfef0f044</pre>	366	
uboot cli readline() at cli bush c:004 0xfef07448	367	if (inpu
$= abbot _cti_leadine() at cti_liash.c.) 54 0x1e107440$	368	retu
$= get_uset_input() at cli_nush.c. 1,027 0x1e107448$	370	retu
	371	}
	372	
= parse_stream() at cli_hush.c:2,954 0xfef088dc	373	static const
parse_stream_outer() at cli_hush.c:3,188 0xfef088dc	375	.pending
<more frames=""></more>	376	.getc =

Figure 23. Debug buttons

15. Always make sure that your U-Boot is alive, if it reports an error or the terminal does not response to any input, disable the remote debug function by using bcu, reset the board and re-enable the remote debug function again.

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs

438 4	1 mpuse@mpuse:~ 2 串口:FTDI + 1〇
439 struct mipi dsi panel plat *plat = dev get plat(dev):	
440 struct rad platform data *data = (struct rad platform	
1/11 ctruct mini dei device *device - nlat-sdevice.	● /dev/tty0582 (115200)
😫 Console 🗙 🕵 Problems 🕡 Executables 📓 Debugger Console	Fastboot: Normal
<terminated>New_configuration [GDB OpenOCD Debugging] openocd.1 (Te</terminated>	Normal Boot
Error: Invalid ACK (7) in DAP response	Hit any key to stop autoboot: 0
Error: JTAG-DP STICKY ERROR	u-boot=>
Error: Invalid ACK (7) in DAP response	u-boot=>
Error: JTAG-DP STICKY ERROR	u-boot=>
Error: Invalid ACK (7) in DAP response	u-boot=>_"Synchronous Abort" handler, esr 0x86000004
Error: JTAG-DP STICKY ERROR	elr: 527fffff7739c060 lr : 00000004022f070 (reloc)
Error: Invalid ACK (7) in DAP response	elr: 5280000036080060 lr : 000000000tet13070
Error: JTAG-DP STICKY ERROR	x0 : 5280000036080060 x1 : 000000001/TTTTTC
Error: Invalid ACK (7) in DAP response	
Error: JTAG-DP STICKY ERROR	
Error: Invalid ACK (7) in DAP response	
Error: JTAG-DP STICKY ERROR	
Error: Invalid ACK (7) in DAP response	x12: 000000000000000000000000000000000000
Error: JTAG-DP STICKY ERROR	x14: 00000000fced5300 x15: 00000000000000
Error: Invalid ACK (7) in DAP response	x16: 5280000036080060 x17: 00000000004340
Error: JTAG-DP STICKY ERROR	x18: 00000000fcee3da0 x19: 000000000000000
Error: Invalid ACK (7) in DAP response	x20: 0000000000000000 x21: 00000000fcef6470
Error: JTAG-DP STICKY ERROR	x22: 00000000feff86a0 x23: 00000000feff8000
Error: Invalid ACK (7) in DAP response	x24: 00000000feff8690 x25: 000000000000000
Error: ITAG-DP STICKY ERROR	x26: 0000000000000002 x27: 00000000fefade51
Error: Invalid ACK (7) in DAP response	x28: 00000006cf0df90 x29: 00000000fced5060
Error: ITAG-DP STICKY ERROR	
Error: Invalid ACK (7) in DAP response	Code: "Synchronous Abort" handler, esr 0x96000004
Error: JTAG-DP STICKY ERROR	elr: 0000000040202b88 [r : 0000000040202b88 (reloc)
Error: Invalid ACK (7) in DAP response	elr: 00000000tee6b88 Lr : 00000000tee6b88
Error: JTAG-DP STICKY ERROR	X0 : 000000000000000 X1 : 00000000000000
Error: Invalid ACK (7) in DAP response	X2 : 000000001613356 X3 : 00000000000001
Error: JTAG-DP STICKY ERROR	X4 . 0000000016614013 X7 . 000000001604070
Error: Invalid ACK (7) in DAP response	
Error: JTAG-DP STICKY ERROR	x10: 000000000000000 x11: 00000000000000
Error: Invalid ACK (7) in DAP response	x12: 00000000001869f x13: 0000000fced51f8
Error: JTAG-DP STICKY ERROR	x14: 0000000fced5300 x15: 000000000000000
Error: Invalid ACK (7) in DAP response	x16: 00000000fef359d0 x17: 0000000000004340
Error: JTAG-DP STICKY ERROR	x18: 0000000fcee3da0 x19: ffffffffffffff
Error: Invalid ACK (7) in DAP response	x20: 0000000fefade53 x21: 00000000fefa2c14
Error: JTAG-DP STICKY ERROR	x22: 00000000fefafe61 x23: 5280000036080060
Error: Invalid ACK (7) in DAP response	x24: 00000000tett8690 x25: 0000000000000000
Error: JTAG-DP STICKY ERROR	x26: 0000000000000002 x27: 00000000tetade51
Error: Invalid ACK (7) in DAP response	x28: 000000000000000000000000000000000000
Error: JTAG-DP STICKY ERROR	Codo: 03800072 h0000640 012h0000 04030000 (h8727001)
Error: Invalid ACK (7) in DAP response	Code. 32000073 00000040 91300000 94029000 (06737841)
Error: JTAG-DP STICKY ERROR	Resetting tro
Error: Invalid ACK (7) in DAP response	resetting
Error: JTAG-DP STICKY ERROR	resetting
Error: Invalid ACK (7) in DAP response	
Figure 24 Error reported in Folince concels and	

Figure 24. Error reported in Eclipse console and serial console

4.2.4 Debugging U-Boot with on GDB terminal

There are a lot of GDB commands. Use **help** to check these commands. Then these commands classes are listed:

```
Type "apropos -v word" for full documentation of commands related to "word".

Command name abbreviations are allowed if unambiguous.

(gdb) help

List of classes of commands:

aliases -- User-defined aliases of other commands.

breakpoints -- Making program stop at certain points.

data -- Examining data.

files -- Specifying and examining files.

internals -- Maintenance commands.

obscure -- Obscure features.

running -- Running the program.

stack -- Examining the stack.

status -- Status inquiries.

support -- Support facilities.

text-user-interface -- TUI is the GDB text based interface.

tracepoints -- Tracing of program execution without stopping the program.

user-defined -- User-defined commands.
```

```
Figure 25. GDB help command
```

Use help stack to check the GDB commands in the stack class:

```
(gdb) help stack
Examining the stack.
The stack is made up of stack frames. Gdb assigns numbers to stack frames
counting from zero for the innermost (currently executing) frame.
At any time gdb identifies one frame as the "selected" frame.
Variable lookups are done with respect to the selected frame.
When the program being debugged stops, gdb selects the innermost frame.
The commands below can be used to select other frames by number or address.
List of commands:
backtrace, where, bt -- Print backtrace of all stack frames, or innermost COUNT frames.
down, dow, do -- Select and print stack frame called by this one.
faas -- Apply a command to all frames (ignoring errors and empty output).
frame, f -- Select and print a stack frame.
frame address -- Select and print a stack frame by stack address.
frame apply -- Apply a command to a number of frames.
frame apply all -- Apply a command to all frames.
frame apply level -- Apply a command to a list of frames.
frame function -- Select and print a stack frame by function name.
frame level -- Select and print a stack frame by level.
frame view -- View a stack frame that might be outside the current backtrace.
return -- Make selected stack frame return to its caller.
select-frame -- Select a stack frame without printing anything.
select-frame address -- Select a stack frame by stack address.
select-frame function -- Select a stack frame by function name.
select-frame level -- Select a stack frame by level.
select-frame view -- Select a stack frame that might be outside the current backtrace.
up -- Select and print stack frame that called this one.
```

Figure 26. GDB help stack command

4.2.4.1 Common commands

Common commands are the following:

4.2.4.2 Stack commands

backtrace(bt) Print backtrace of all stack frames or innermost COUNT frames.

frame n Select and print a stack frame.

up Select and print the stack frame that called this one.

down Select and print the stack frame called by this one.

info stack Backtrace of the stack, or innermost COUNT frames.



```
(gdb) down
#18 board_init_r (new_gd=<optimized out>, dest_addr=<optimized out>) at common/board_r.c:880
880
                   (initcall_run_list(init_sequence_r
(gdb) down
#17 0x00000000fef1debc in initcall_run_list (init_sequence=0xfefc1118) at include/initcall.h:46
46
                         ret =
                               (*init_fnc_ptr
(gdb) down
#16 0x00000000fef1dbb0 in run main loop () at common/board r.c:627
                        main_loop
627
(gdb) down
#15 0x0000000fef1b380 in main loop () at common/main.c:66
66
                cli_loop
(gdb) l
61
                if (cli_process_fdt(&s)
62
                         cli_secure_boot_cmd(s)
63
64
                autoboot_command(s);
65
66
                cli_loop
67
                panic(
68
(gdb) up
#16 0x00000000fef1dbb0 in run_main_loop () at common/board_r.c:627
627
                        main_loop
(gdb) up
#17 0x00000000fef1debc in initcall_run_list (init_sequence=0xfefc1118) at include/initcall.h:46
46
                               (*init_fnc_ptr
                         ret =
(gdb) l
41
                                                  (char *)*init_fnc_ptr
                                                                           reloc_ofs
42
                                                          *init_fnc_ptr
43
                                        initcall: %p\n", (char *)*init_fnc_ptr -
44
                                 debug
                                                                                   reloc ofs)
45
```

Figure 28. GDB stack example in U-Boot project -2

- 1. Use **backtrace(bt)** to check the stack as shown in Figure 27
- 2. Use frame 19 to select the #19 frame
- 3. Use list to check the code.
- 4. Use **down/up** to switch the frame
- Use list(I) to check the code anytime.
 Combining these commands can easily browse stack information and related code.

4.2.4.3 breakpoint

breakpoint(b) Set the breakpoint at a specified location.

delete(d) Delete some breakpoints or autodisplay expressions.

clear Clear breakpoint at specified location.

info b Status of specified breakpoints.

disable/enable Disable/Enable some breakpoints.

AN14367

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs

(adb) bt #0 console_tstc (file=file@entry=0) at common/console.c:302
#1 0x00000000ff23fa4 in Tgetc (file=0) at common/console.c:490 #2 0x0000000fff23fa0 in cread-line (timeout=-16806800, len=csynthetic pointer>, buf=0xfeff8404 "sssh\n", prompt=0xfcf181e0 "\003") at common/cli_readline.c:277 #3 cli_readline_into_buffer (prompt=0xfcf181e0 "\003", buffer=0xfeff8404 "sssh\n", timeout=-16806880) at common/cli_readline.c:554
#4 0x00000006f1b904 in uboot_cli_readline (i=0xfceebef0) at common/cli_hush.c:995 #5 get_user_input (i=0xfceebef0) at common/cli_hush.c:1028 #6 file oct (i=continized out) at common (cli_hush.c:1107
<pre>## orce_get (l=soptimized out>) at common/cit_nush.c:1090 ## orce_get (l=soptimized out>, dest=<optimized out="">) at common/cit_hush.c:2955 ## 0x000000000fef1cda4 in parse_stream (end_trigger=<optimized out="">, input=<optimized out="">, ctx=<optimized out="">, dest=<optimized out="">) at common/cit_hush.c:2955</optimized></optimized></optimized></optimized></optimized></pre>
<pre>#9 parse_stream_outer (inp=inp@entry=0xfceebef0, flag=flag@entry=2) at common/cli_hush.c:3189 #10 0x000000006ff1d344 in parse_file_outer () at common/cli_hush.c:3289</pre>
#11 0x00000000fef1b380 in main_loop () at common/main.c:66 #12 0x0000000fef1b380 in main_loop () at common/main.c:66 #13 0x0000000fef1dbbb in run main_loop () at common/board r.c:627
#14 0x00000006fff1debc in initall_run_list (init_sequence=0s/fefc1118) at include/initcall.h:46 #15 board_init_r (new_gd= <optimized out="">, dest_addr=<optimized out="">) at common/board_r.c:880</optimized></optimized>
#16 0x0000000feefa580 in _main () at arch/arm/lib/crt0_64.S:140 Backtrace stopped: previous frame identical to this frame (corrupt stack?) (-ub) b forecome
(gob) b fgetc Breakpoint 1 at 0xfef23f7c: fgetc. (2 locations) (adb) b act user input
Breakpoint Z at 0xfef1b8d0: file common/cli_hush.c, line 1044. (gdb) b console.c:300
Breakpoint 3 at 0xfef23b6c: file common/console.c, line 301. (gdb) info b Num Tune Dico Enb Address What
1 breakpoint keep y <multiple> 1.1 y 0x00000000fef23f7c in fgetc at common/console.c:475</multiple>
1.2 y 0x00000000fef23fdc in fgetc at common/console.c:477 2 breakpoint keep y 0x0000000fef1bad0 in get_user_input at common/cli_hush.c:1044
(gdb) c Continuing.
Breakpoint 3, console_tstc (file=file@entry=0) at common/console.c:838
<pre>838 int prev = ctrlc_disabled; /* save previous state */</pre>
Figure 29. GDB breakpoint example in U-Boot project -1
(gdb) d 1 (adb) info b
Num Type Disp Enb Address What
2 breakpoint keep y 0x00000000fef1b8d0 in get_user_input at common/cli_hush.c:1044 breakpoint already hit 1 time
3 breakpoint keep y 0x00000000fef23b6c in console_tstc at common/console.c:301
breakpoint already hit 5 times (adb) n
301 prev = disable_ctrlc(1);
(gdb) n 840 ctrlc disabled = disable:
(gdb) n
301 prev = disable_ctrlc(1); (adb) p
302 for_each_console_dev(i, file, dev) {
(gdb) n 303 if (dev_state = NULL) {
(gdb) n
304 ret = dev-> tstc (dev);
Continuing.
Breakpoint 3. console tstc (file=file@entry=0) at common/console c:838
838 int prev = ctrlc_disabled; /* save previous state */
(gdb) n 301 prev = disable_ctrlc(1):
(gdb) n
840 ctrlc_disabled = disable; (adb) p
301 prev = disable_ctrlc(1);
(gdb) n 202 for each carcala daw(i file daw) (
Soz ror_each_console_dev(1, rite, dev) {
Figure 30. GDB breakpoint example in U-Boot project -2

1. Use backtrace(bt) to check the stack as shown in Figure 29

- 2. Use **breakpoint(b)** to set the breakpoint
- 3. Use **b getc** and **b get_user_input** to set the breakpoint on the function
- 4. Use **b console.c:300** to set the breakpoint on line 300 of the console.c file

- 5. Use info b to check the current breakpoint setting
- 6. Use **delete(d)** to delete the breakpoint
- 7. Use **c** to continue the process, then the program stops at the next breakpoint.

4.2.4.4 Control and view commands

next(n) Step program, proceeding through subroutine calls.

continue(c) Execute to the next breakpoint or program end.

run(r) Start the debugged program.

whatis Print the data type of the expression EXP.

print(p) Print the value of the expression EXP.



1. Use **whatis** *prev* to print the data type of *prev* as shown in Figure 31

2. Use **print(p)** prev to print the data value of *prev*.

4.2.4.5 GDB layout split commands

🖪 mpuse@mpuse:-jopenocd_clipue_iest/%j/ubochinx Q 🗄 😋 Ø 🗴
to the second se
46 54<
minute minut minut minut
<pre>rests Respond: The part serial packing</pre>

Figure 32. GDB split layout example in U-Boot project

Use the **layout split**, **bt** combination to see the source code, assembly code, and gdb console in the same window.

5 Native RTOS OpenOCD debugging

This chapter introduces the method of using J-link for native RTOS debugging.

5.1 Native RTOS on Cortex-A Core introduction

Native RTOS on Cortex-A Core refers to the RTOS running on Cortex-A Core without a hypervisor and is kicked to a specified Cortex-A Core by U-Boot commands.

Native RTOS is supported in <u>Real-time Edge Software</u>. The key technology components of Real-time Edge Software include Real-time System, Heterogeneous Multicore Framework, Heterogeneous Multi-SoC Framework, Real-time Networking, and Protocols.

Native RTOS currently supports FreeRTOS and Zephyr on Cortex-A Core, it can use both Cortex-A Core's high performance and RTOS's low latency schedule and interrupt ability.

There are some example applications in Heterogeneous Multicore Framework of Real-time Edge Software. Refer to *Real-time Edge Software User Guide* (document <u>REALTIMEEDGEUG</u>) for more details.

5.2 EVK board software setup

Download a Real-time Edge pre-build SD card binary image from <u>https://www.nxp.com/rtedge</u>, the image for i.MX 93 EVK is nxp-image-real-time-edge-imx93evk.wic after decompression and flash this image to SD by using the following method:

Method 1: download it to the board by using uuu (change the bootmode to serial download mode and use the USB-Type C cable):

\$ sudo ./uuu -b sd nxp-image-real-time-edge-imx93evk.wic # for 93

Method 2: flash it to the SD card directly:

```
$ sudo dd if=nxp-image-real-time-edge-imx93evk.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.

5.3 Debugging Native RTOS with Eclipse IDE

This example demonstrates how to debug Native RTOS by using OpenOCD together with Eclipse IDE on the i.MX 93 EVK board. In this example, J-Link is used. There is a J-Link link issue on several versions of i.MX 93 EVK boards. A workaround is to remove the Wi-Fi/Bluetooth module from the M.2 connector.

5.3.1 Building Native RTOS with Eclipse IDE

1. Download the source code of Native RTOS in a heterogeneous-multicore project.

```
# mkdir ~/a_core_jlink
# cd ~/a_core_jlink/
# pip install west
# west init -m https://github.com/nxp-real-time-edge-sw/heterogeneous-
multicore.git workspace
# cd workspace & west update
```

2. Download and set up the toolchain:

```
# mkdir ~/toolchains/
# cd ~/toolchains/
# wget https://developer.arm.com/-/media/Files/downloads/gnu-a/10.3-
2021.07/binrel/gcc-arm-10.3-2021.07-x86_64-aarch64-none-elf.tar.xz
# tar xf gcc-arm-10.3-2021.07-x86_64-aarch64-none-elf.tar.xz
```

Note: For detailed information on the environment setting and image building, refer to Section 3.2 "Building Heterogeneous Multicore RTOS Application" of Real-time Edge Software User Guide (document <u>REALTIMEEDGEUG</u>).

3. Open Eclipse and select the directory of the workspace.

	uncher	×
Select a directory as workspace		
Eclipse IDE uses the workspace directory to store its p	preferences and development	artifacts.
/home/gsv/a core ilink/workspace		▼ Browse
_		
Use this as the default and do not ask again		
Use this as the default and do not ask again Recent Workspaces		
Use this as the default and do not ask again Recent Workspaces	Cancel	Launch

. . .

To import the project, click **Project Explorer => Import projects** and select **Existing Code as Makefile Project** as shown in <u>Figure 34</u>.

workspace - /home/gsy/a_core_jlink/workspace/heterog	eneous-multicore/apps/hello_world/freertos/boards/mcimx93evk_ca55/hello_world_board.c - Eclipse IDE ×
File Edit Source Refactor Navigate Search Project Run Window	Help Import ×
Debug Project Explorer ×	Select Create a new Makefile project in a directory containing existing code
There are no projects in your workspace. 3 * To add a project. 4 * SPDX-License-I	dent Select an import wizard:
P3 Create automet 7 # include "fsl_dev import projects 9 # include "fsl_dev 10 # include "clock c 1 2 # include "clock c 1 2 # include "clock c 1 4 # include "clock c 1 5 # include "clock c 1 6 # include "clock c 1 7 # include "clock c 1 8 # include "load c 1 9 # include "load c 1 16 # include "load c 1 17 # include c 1 18 # include c 1 19 # include c 1 10 # include c 1 11 # include c 1 10 # include c 1 11	ice Image: Projects from Folder of Archive Image: Projects from Folder of Archive Image: Projects from Folder of Archive Image: Project for Project for Project Image: Project for Project for Project Image: Project for Project for Project Image: Project
	Sebr
Program received signa vApplicationIdleHook (33 Exception ignored in: Traceback (most recent File */usr/share/gd def flush(self): KeyboardInterrupt	cgd ② < Back Next > Cancel Finish
Figure 34. Import existing project	

5. Import the existing code.

To import the existing code, click **Browse** to select the project to build and click **Finish** as shown in Figure 35.

In this case, the project is apps/hello_world/freertos/boards/mcimx93evk_ca55

workspace - /home/gsy/a_core_jlink/worksp	New Project x	o_world_board.c - Eclipse IDE X
File Edit Source Refactor Navigate Search Project Ru I 그 그 데 데 너 것 이 데 데 N 가 가 다 다 문 Be Debug Re Project Explorer X 우리 B startup	Import Existing Code Create a new Makefile project from existing code in that same directory	egist Se Break X & Expre 5, Perioh C C
P Debug is Project Explorer X Image: Startup, Im	Project Name armgcc_aarch64 Existing Code Location /home/gsy/a_core_jlink/workspace/heterogeneous-multicore/apps/hello_wor Induges C C C ++ Toolchain for Indexer Settings <pre>cnone> Arm Cross GCC GNU Autotools Toolchain Linux GCC RISE-V Cross GCC</pre>	egist is Break × is Expre is Periph is 0
0 items selected	< Back Next > Cancel Finish	
Figure 35. Import existing code		

6. Edit the environment setting.

To edit the environment setting, right-click the imported project and Select **Properties** as shown in Figure <u>36</u>.

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs



7. Add a building environment.

To do this, follow the steps shown in Figure 37:

- a. Click **Builders => New** to add a building configuration.
- b. Click Main => Click Browse to select the building script. Note: build ddr debug.sh is used as an example.

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs

r	Properties for armgcc_aarch64	×
type filter text	E Edit Configuration ×	() + () + §
Resource	Edit launch configuration properties	
Builders	Create a configuration that will run a program during builds	New
 C/C++ General 	Name: New_Builder	Import
CMake Git	Main P Refresh 🖾 Environment 😂 Build Options	Edit
Linux Tools Path	Location:	Remove
Project Natures Project References	\${workspace_loc:/armge_aarch64/build_ddr_debug.sn}	
Run/Debug Settings	Browse Workspace Browse Hie System Variables	Up
 Task Repository Task Tags 	Working Directory: \${workspace loc:/armgcc aarch64}	Down
 Validation WikiTaxt 	Browse Workspace Browse File System Variables	
WIRTIERC	Arguments:	
	Variables	
	Note: Enclose an argument containing spaces using double-quotes (").	
	Show Command Line Revert Apply	
	? Cancel OK	
(?)	Cancel	Apply and Close
Figure 37. Add b	uilding environment	

- c. Click Environment and Add to include the GCC toolchain that is to be used as shown in Figure 38.
- d. Enter the variable using for the toolchain:
 - Name: ARMGCC_DIR

Value: /home/gsy/toolchains/gcc-arm-10.3-2021.07-x86_64-aarch64-none-elf *Note: Edit the absolute path with one actual using.*

	Edit Configuration ×	×
type filter text Resource	Bu Edit launch configuration properties Create a configuration that will run a program during builds	$(\varphi + \varphi) \neq \frac{1}{2}$
Builders C/C++ Build C/C++ General CMake	Name: New_Builder	New
Git Linux Tools Path Project Natures	Variable Value Add	Bdit
Project References_ Run/Debug Setting: > Task Repository	Name: ARMGCC_DIR	Up
Task Tags > Validation WikiText	Value: /home/gsy/toolchains/gcc-a Variables	
	Cancel	
	Snow Commany Line Apply	
	⑦ Cancel OX	
٢	Cancel	Apply and Close
set the toolch	ain environment value	

Figure 38. Set the toolchain environment value

e. Click the **Ok** button to save the configuration of the toolchain.

f. Select the building configuration that is added and click Apply and Close as shown in Figure 39.

type filter text	Builders	(4 + 4)	+ 1
Resource	Configure the builders for the project:		
Builders	CDT Builder	New	
C/C++ General	CDT Core Builder	Impor	t
CMake	Subset and a subset of the su	Edit	
Linux Tools Path			
▶ MCU		Teme	ve
Project Natures Project References		U.S.	
Run/Debug Settings		Dow	0
Task Repository		<u></u>	
Task Tags			
WikiText			
~			_
C		Cancel Apply and 0	

Figure 39. Builder tab

8. Build the Native RTOS image.

To do this, right-click the imported project and select Build Project to start building as shown in Figure 40.

we		/heterogeneous-multicore/apps/hello_world/freertos/boards/mcimx93evk_ca55/hello_world_board.c - Eclipse IDE	×
File Edit Sourc	Source +	/indow Help	-
	Move		
	Rename F2		
🍄 Debug 🎦 Proje	🗽 Import	is main.c Thello_world_boa X To idle.c To main.c To to Variab in Regist To Break X of Expre to Periph of the to th	2
	🗠 Export	ght 2023 NXP X 월 윤 일 독 문 문 역	00
👻 📸 > armgcc_aa	Build Project	rense-Identifier: BSD-3-Clause	
Binaries	Clean Project Incremental Build of Selected Projects	acense acentation populations	
build	F5 Class Project	*fsl_device_registers.h*	
CMakeFile:	Close Uprelated Project	"fsl_debug_console.h"	
ddr_debug	Ruild Targets	"board.h"	
🕨 🍙 > Default	Index •	rctock_contag.n~ "pin mux.h"	
) 🛐 1.5	Build Configurations	"rtos_memory.h"	
Is dump.s	Source	o_world_board_init(void)	
We build ddr	Profiling Tools	it board cpu and hardware. */	
🛱 clean.sh	O Bun As	<pre>InitMemory(); edetait();</pre>	
🖉 cmake_ins	🎄 Debug As	Bale GIC before register any interrupt handler*/	
🖹 CMakeCac	🚱 Profile As	ID == 0) nable(1):	
CMakeList	Restore from Local History	onsole 🗟 Console × 😧 Executables 🚺 Memory	
MIMX9352	💖 Run <u>C</u> /C++ Code Analysis		
MIMX9352	Team •	New_Builder [Program] /home/gsy/a_core_jlink/workspace/heterogeneous-mu	
🖹 output.ma	Compare With	g C object (MakeFiles/hello world ca55 RTOS1 UART1.elf.dir/home/gsy/a	
	Replace With	g C object CMakeFiles/helle world_ca55_RTOSI_UART1.elf.dir/home/gsy/a	
	✓ Validate	ig C object CMakeFiles/hello world ca55 RTO51 uART1.elf.dir/home/gsy/a C executable ddr debug/hello world ca55 RTO51 UART1.elf	
	Configure F	arget hello_world_ca55_RTOS1_UART1.elf	
	Source P		
😚 armgcc_aarch6	▼ Alt+Enter		
Figure 40.	Build project		

5.3.2 Eclipse OpenOCD configuration for Native RTOS debugging

1. Open Eclipse and select a directory as a workspace.

Eclipse IDE Launcher	×			
Select a directory as workspace				
Eclipse IDE uses the workspace directory to store its preferences and devel	lopment artifacts.			
%	- Browse			
/nome/gsy/a_core_link/workspace	Browse			
Use this as the default and do not ask again				
Recent Workspaces				
Can	cel Launch			
Figure 41. Launch the workspace				

2. Add a Debug Configuration.

To do this, Click **Run** and then click **Debug Configurations** as shown in Figure 42

	workspace - Eclipse I	DE ×
File Edit Source Refactor Navigate Search Proj Image: Comparison of the search project Explorer Image: Comparison of the	ect Run i+ Instruction Stepping Mode i Move to Line (C/C++) startup.S i+ 38 HMU Suspend 39 I Terminate 40 I Terminate 41 /* ARM Noisconnect 42 Statue IS the Over 47 I Step Neturn 48 * This IR step Return 49 * Mana If Run to Line 51 void AR Qual Last Launched Ctrl+F1 55 uin Run History I 56 A Run Configurations Image: Statue Statue 61 A Debug History Image: Statue Statue	Image:
No	Console × Debug Configurations Breakpoint Types • Toggle Breakpoint Shift+Ctrl+I • Toggle Line Breakpoint • Toggle Watchpoint • Toggle Method Breakpoint	
Figure 42 Pup and click Dob	Ctrl+Alt+I	L I

Figure 42. Run and click Debug Configurations

Right-click GDB OpenOCD Debugging, then click New Configuration as shown in Figure 43

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs

				Debug Configurations x	×
File	Edit Source F	Refacto	or Navigate Search	Create manage and sup configurations	
-9		G -	1001 m - 10 -	Create, manage, and run configurations	
				1	
	welcome x			🗅 🖻 😼 🖹 🗶 🏳 🖌 Configure launch settings from this dialog:	
ч <u>с</u>	eclir e	DSE	Welcome to th	type filter text - Press the 'New Configuratiouration of the selected type.	
	- comp			 Press the 'New Prototype' bototype of the selected type. 	Hide
				C/C++ Attach to Application - Press the 'Export' button tot the selected configurations.	
				C/C++ Container Launcher - Press the 'Duplicate' buttonpy the selected configuration.	
		=	Tutorial: Import an ex	C/C++ Postmortem Debugger Press the 'Delete' button tove the selected configuration.	
		-	A guided walk-through how	CijC/C++ Unit	
				GDB Hardware Debugging Select launch configuration(end item to unit a prototype.	
		Ċ,	Review IDE configur	Configuration reset with prototype values	
			Review the IDE's most fier	GDB OEM	
				CGDB QEMI New launch configuration an existing configuration by selecting it.	
	(0	Create a new Embde	CGDB QEMI Duplicate	
			Create a new Eclipse proje	CGDB QEMI Collete age.	
		_		CGDB SEGG	
	•	Ð	Open the New item wizard	Filter matched	
				Reset with Prototype Values	
		~	Checkout projects fr	Close Debug	
			Checkout Eclipse projects	osted in a Git repository	-
		•	Import existing proie	IS	Always show Welcome at start up

Figure 43. Create the debug configuration

3. Set Main of Debug Configuration.

To do this, enter the **Name** and **Project**. To select the directory of RTOS that runs on the A core, click **Browse**.

Note: hello_world_ca55_RTOS0_UART2.elf is used as an example here. Other demo apps or customized apps can also be used.

Debug Configurations			
Create, manage, and run configurations		Ť.	
Image: Second	Name: RTOS_A_debug Main Project: RTOS_A_debug C/C++ Application: /home/gsy/a_core_llink/workspace/heterogeneous-multicore/apps/hello_world/freertos/boards/m Variables Search Pro Build (if required) before launching Build Configuration: Use Active Enable auto build Disable auto build • Use workspace settings Configure Workspace Settings	Browse cimx93evk_ca55/armgcc oject Browse	
0	Clo	se Debug	

Figure 44. Main tab in Debug Configuration

^{4.} Set OpenOCD in Debugger.

To do this, click **Browse** to select the directory of OpenOCD and enter **Config options** to select 'Jlink' and a specific board that is to be used:

-s <Path of tcl folder> -f <Path of jlink.cfg> -f <Path of board script> -c "gdb breakpoint override hard"

Debug Configurations ×			
Create, manage, and run configurations			Ť.
C 2 0 0 X 8 7 ·	Name: RTOS_A_deb	ug	
type filter text	📄 Main 🏇 Debugge	r 🕨 Startup 🦃 Source 🔲 Common 🔀 SVD Path	
© C/C++ Application © C/C++ Attach to Application © C/C++ Container Launcher	OpenOCD Setup Start OpenOCD Executable path:	locally /home/gsy/a_core_jlink/openocd/src/openocd	Browse Variables
C/C++ Postmortem Debugger C/C++ Remote Application	Actual executable:	/home/gsy/a_core_jlink/openocd/src/openocd	
CiC/C++ Unit ⓒ GDB Hardware Debugging ♥ ⓒ GDB OpenOCD Debugaing	GDB port:	3333	g properties page)
C RTOS_A_debug	Telnet port:	4444	
CGDB PyOCD Debugging	Tcl port:	6666	
© GDB QEMU aarch64 Debugging © GDB QEMU arm Debugging © GDB QEMU gnuarmeclipse Debugging (Deprecated)	Config options:	-s /home/gsy/a_core_jlink/openocd/tcl -f /home/gsy/a_core_jlink/openocd/t home/gsy/a_core_jlink/openocd/tcl/board/nxp_mcimx93-evk.cfg -c "gdb_br	cl/interface/jlink.cfg -f / eakpoint_override hard"
CGDB QEMU riscv32 Debugging	Allocate console	e for OpenOCD	connection

Figure 45. Debugger tab in Debug Configuration - 1

Note: The debugging script with initial reset (nxp_mcimx93-evk-reset.cfg) is used here. If the initial reset is not wanted, the nxp_mcimx93-evk.cfg script could be used.

Click **Browse** to select the directory of gdb as shown in <u>Figure 46</u> Enter **Port number** with **3334** as '3334' is assigned to imx93.a55.1

Info : starting gdb server for imx93.a55.1 on 3334

	Name: RTOS_A_debug
type filter text	🗈 Main 🅸 Debugger 🕨 Startup 🦞 Source 🔲 Common 🚼 SVD Path
CC/C++ Application CC/C++ Attach to Application	Allocate console for OpenOCD
C/C++ Container Launcher	GDB Client Setup
C/C++ Postmortem Debugger	Start GDB session
C/C++ Remote Application	Executable name: /usr/bin/gdb-multiarch Browse Variables
CGDB Hardware Debugging	Actual executable: /usr/bin/gdb-multiarch
CGDB OpenOCD Debugging CRTOS_A_debug	Other options:
CGDB PyOCD Debugging CGDB QEMU aarch64 Debugging	commands: set mem inaccessible-by-default on
© GDB QEMU arm Debugging © GDB QEMU gnuarmeclipse Debugging (Deprecated) © GDB QEMU riscv32 Debugging	Remote Target Host name or IP address: localhost
CIGDB SEGGER J-Link Debugging	Port number: 3334
a Launch Group	Restore defaults

Figure 46. Debugger tab in Debug Configuration - 2

5. Unflag Initial Reset and Pre-run/Restart reset in Startup to avoid board resetting as shown in Figure 47

Create, manage, and run configurations	
	Name: RTOS_A_debug
C/C++ Application C/C++ Attach to Application C/C++ Container Launcher C/C++ Postmortem Debugger C/C++ Remote Application	Initialization Commands
CEC/C++ Unit CGDB OpenOCD Debugging CGDB PyOCD Debugging CGDB PyOCD Debugging Eiguro 47 Startup tab in Dobug Configuration	Carl Stranger Strang

5.3.3 Eclipse debugging steps

1. Flash <u>v2.8 Real-Time Image</u> into eMMC or the SD card:

```
.\uuu.exe -b sd_all nxp-image-real-time-edge-imx93evk.wic
```

- 2. Boot the board and stop booting in U-Boot.
- 3. Enter the following command to boot the A core.

```
$ uboot => setenv boot_a1 "mw 0xA0000000 14000000; dcache flush; icache
flush; cpu 1 release 0xA0000000"
$ uboot => setenv bootcmd "run boot_a1"
$ uboot => saveenv
```

- 4. If the configuration is set successfully in previous sections, power on the board.
- 5. Click **debug** to start debug.

Figure 48. Debug button

Note: In the Console window, a successfully connected log could be observed.

6. The program is set to stop at the main function automatically. Then, the address of the pc pointer that is pointing can be found.



7. Click Step Into, to go into the pointing function and execute the next command.

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs



Figure 52. Instruction stepping mode

10. Instruction stepping mode can be selected to check the assembly.

File Edit Source Refactor Navigate Search	Project Run Window Help			
	● 20 1 2 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3			
🎄 Debug 🗙 🍋 Project Explorer	□ □ (gdb[108].proc[42000 🕆 main.c 🕆 hello_world_board.c × □ 1	Debugger Console S Console X	👀 Varia 🔠 Regi 🍨 Brea 🛠 Expr 🔤 Disa 🗙 🖓 Perip 📮 🗖	
E % ♥ CRTOS_A_debug [GDB OpenOCD Debugging] ♥ Phello_world_ca55_RTOS0_UART2.eff ♥ Thread #1 (Suspended : Breakpoint) E hello_world_board_init() at hello_world	<pre>b } 7 finctude "fil_device registers.h" 9 finctude "fil_device registers.h" 9 finctude "fil_device registers.h" 9 finctude "fil_device consile.h" 9 finctude "pin muc.h" 12 finctude "pin muc.h" 13 finctude "pin muc.h" 14 finctude "consile.h" 14 finctude "pin muc.h" 15 evoid hell_world_beard_init(void) 16 { 17 /* Init board cpu and hardware. */ 18 goods Internation (Consile.h) 16 { 17 /* Init board cpu and hardware. */ 18 goods Internation (Consile.h) 17 /* Enable (Consile.h) 18 goods Internation (Consile.h) 19 goods Internation (Consile.h) 19 goods Internation (Consile.h) 19 goods Internation (Console Inter</pre>	Too S, Adeus (Gob Qered) Co Debugging) Critor Cleack (Gob Qered) Co Debugging) Critor Cleack (Gob Qered) Co Debugging) Critor Cleack (Gob Qered) Co Debugging) Error: Triying to use configured scan chain any Error: Triying to use configured scan chain any Error: Im33.a55.1 hardware has 6 breakpoints. Info: [im33.a55.1] braination succeed Info: [im33.a53] Li Examination succeed Info: [im33.a53] Li Examination succeed Info: [im33.a53] Li Examination succeed Info: [im33.a5] Li Examination succeed Info: [im33.a5] Li Examination succeed Info: [im33.a5] Li Examination succeed Info: starting qdb server for im33.a55.40 or Info: im33.a55.1 cluster 1 or eo mult core Oser: molosof5.40 ccmoecolosof0 Wu disabled, 0-Cache: disabled, 1-Cache: disabled, Info: Sing Ose Connection 1, Target Im39.a55.	Enter location herer Image: Solution is a state of the image: Soluti	

Figure 53. Instruction stepping mode

11. If editing and image rebuilding are required, click **Terminate** to terminate the debugging. After that, code editing and rebuilding could be done. For more information of image building, refer to <u>Section 5.3.1</u>.

File Edit Navigate Search Project Run Window Help	
╡┍╩╺┍╗╚╗╡╝╲╲╡╚╝╡╲╡┡╸┉ <mark>╺</mark> ┙┇╲╔╡╆╼┋╗╡ <mark>╝╡╬╼┇┙┩╸┊╔╶╝╶╚╶╗</mark> ╺╚╺┥╸ <mark>┍</mark>	Q 🛛 😰 🕸 🏘
Figure 54. Terminate button	
12. After the image is successfully re-build, click debug to start the debugging again.	

File Edit Source Refactor Navigate Search Project Run Window Help	
☐ ▼ 🛛 🖏 ♡ ♡ ♥ 🖳 ▼ 🖉 ♥ 🖉 🗮 😹 😒 🕹 🗰 🥄 🕹 🗰 🖏 😒 🕹 🗰 🖏 🕹 ♥ 🖉 ♥ 🖉 ♥ 🖉 ♥ 🖉 ♥ 🖉 ♥ 🖉 ♥ 🖉 ♥	Q : 😰 🖬 🔯
Figure 55. Debug button	

5.4 Debugging Native RTOS by using OpenOCD with GDB

This example shows how to debug Native RTOS by using OpenOCD with GDB client command line on i.MX 8M Plus LPDDR4 EVK or i.MX 93 EVK. In this example, the J-Link probe is used to connect to the local JTAG port.

5.4.1 Setup and build RTOS

To keep debug information in RTOS images, change the gcc compile optimization level. "-O0" or "-Og" must be used.

For FreeRTOS applications in heterogeneous-multicore (<u>https://github.com/nxp-real-time-edge-sw/</u><u>heterogeneous-multicore</u>), build debug images using <code>build_ddr_debug.sh</code>. The bin and elf images can be found in the <code>ddr_debug</code> directory.

For Zephyr, enable CONFIG_DEBUG_OPTIMIZATIONS=y to select the optimization option in applications prj.conf.

To check the code address and instruction, the image can be disassembled to get the dump information, for example:

~\$ aarch64-none-linux-gnu-objdump -alD zephyr.elf > dump.s

Then open dump.s to check the code address and assemble instructions.

5.4.2 GDB Debugging steps

The heterogeneous-multicore (<u>https://github.com/nxp-real-time-edge-sw/heterogeneous-multicore</u>) hello_world application on the i.MX 8M Plus LPDDR4 EVK is used as an example.

- 1. Follow the steps in <u>Section 5.4.1</u> to build the application debug image: ddr_debug/hello_world_ca53_ RTOS0 UART4.bin
- 2. Connect the J-Link debugger to the EVK board and Linux host, power on the board and stop it at the U-Boot command line.
- 3. To boot the RTOS image, put the following in the U-Boot command line:

```
u-boot=> mw 0xA0000000 14000000; dcache flush; icache flush; cpu 1 release 0xA0000000
```

4. Start OpenOCD from Linux Host.

```
~$ sudo openocd -f interface/jlink.cfg -f board/nxp_imx8mp-evk.cfg -c
"gdb_breakpoint_override hard"
```

Note: After MMU is enabled on the Cortex-A Core, gdb soft breakpoint cannot be used, use hard breakpoints by running openocd with parameter -c "gdb_breakpoint_override hard. The following log can be found in Linux Host:

```
$ sudo openocd -f interface/jlink.cfg -f board/nxp imx8mp-evk.cfg -c
"qdb breakpoint override hard"
Open On-Chip Debugger 0.12.0+dev-00559-g04154af5d (2024-04-29-12:00)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
force hard breakpoints
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
Info : J-Link Ultra V6 compiled Apr 15 2024 17:37:59
Info : Hardware version: 6.00
Info : VTarget = 1.797 V
Info : clock speed 1000 kHz
Info : JTAG tap: imx8mp.cpu tap/device found: 0x5ba00477 (mfg: 0x23b (ARM
Ltd), part: 0xba00, ver: 0x5)
Info : imx8mp.a53.0: hardware has 6 breakpoints, 4 watchpoints
Info : [imx8mp.a53.0] Examination succeed
Error: JTAG-DP STICKY ERROR
Error: [imx8mp.a53.1] Examination failed
Warn : target imx8mp.a53.1 examination failed
Error: JTAG-DP STICKY ERROR
Error: [imx8mp.a53.2] Examination failed
Warn : target imx8mp.a53.2 examination failed
Info : imx8mp.a53.3: hardware has 6 breakpoints, 4 watchpoints
Info : imx8mp.a53.3 cluster 0 core 3 multi core
Info : [imx8mp.a53.3] Examination succeed
Info : [imx8mp.m7] Cortex-M7 r1p2 processor detected
Info : [imx8mp.m7] target has 8 breakpoints, 4 watchpoints
Info : [imx8mp.m7] Examination succeed
Info : [imx8mp.ahb] Examination succeed
Info : starting gdb server for imx8mp.a53.0 on 3333
Info : Listening on port 3333 for gdb connections
Info : starting gdb server for imx8mp.a53.1 on 3334
Info : Listening on port 3334 for gdb connections
Info : starting gdb server for imx8mp.a53.2 on 3335
Info : Listening on port 3335 for gdb connections
Info : starting gdb server for imx8mp.a53.3 on 3336
Info : Listening on port 3336 for gdb connections
Info : starting gdb server for imx8mp.m7 on 3337
Info : Listening on port 3337 for gdb connections
Info : gdb port disabled
```

From the log, find the GDB server port for each CPU Core: four A53 Cores use 3333 3334 3335 and 3336 ports, and the M4 Core uses 3337.

5. Open the GDB client.

Open the GDB client in another terminal window on Linux Host:

~\$ gdb-multiarch -f hello world ca53 RTOSO UART4.elf

Use "-f" to specify the RTOS elf file, if the file is not in the current directory, add the location directory before the filename.

Connect to the remote GDB server:

```
~$ gdb-multiarch -f hello world ca53 RTOSO UART4.elf
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86 64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from hello world ca53 RTOSO UART4.elf...
(gdb) target remote :3334
Remote debugging using :3334
Reset Handler () at /work/work/realtimeedge/hypervisor-less-virtio/
workspace2/heterogeneous-multicore/os/freertos/Core AArch64/core/armv8a/
startup.S:56
/work/work/realtimeedge/hypervisor-less-virtio/workspace2/
heterogeneous-multicore/os/freertos/Core AArch64/core/armv8a/
startup.S:56:1001:beg:0xc0000004
(gdb) load
Loading section .interrupts, size 0x17d4 lma 0xc000000
Loading section .text, size 0x13034 lma 0xc0002000
Loading section .init array, size 0x40 lma 0xc0015040
Loading section .fini_array, size 0xf80 lma 0xc0015080
Loading section .data, size 0x1f0 lma 0xc0016000
Loading section .got, size 0x40 lma 0xc00161f0
Loading section .got.plt, size 0x18 lma 0xc0016230
Start address 0x000000000000000, load size 88592
Transfer rate: 63 KB/sec, 8053 bytes/write.
(qdb)
```

In this demo, RTOS is running on Core1 of the Cortex-A53 Core on i.MX 8M Plus, connect to port 3334. 6. Set Breakpoint and Debug

For example, set the breakpoint at the function $hello_task()$, change the "pc" register to 0xc0000008 to jump out of the "wfe" dead loop, from the below log find the CPU core stops at this breakpoint, and use the "next" command to run a single step:

```
(gdb) b hello_task
Breakpoint 1 at 0xc0002be0: file /work/work/realtimeedge/hypervisor-less-
virtio/workspace2/heterogeneous-multicore/apps/hello_world/freertos/main.c,
line 49.
(gdb) c
Continuing.
```

```
Breakpoint 1, hello_task (pvParameters=0x0) at /work/work/realtimeedge/
hypervisor-less-
virtio/workspace2/heterogeneous-multicore/apps/hello_world/freertos/main.c:49
/work/work/realtimeedge/hypervisor-less-virtio/workspace2/heterogeneous-
multicore/apps/hello_world/freertos/main.c:49:1187:beg:0xc0002be0
(gdb) next
/work/work/realtimeedge/hypervisor-less-virtio/workspace2/heterogeneous-
multicore/apps/hello_world/freertos/main.c:52:1217:beg:0xc0002bec
```

Continue to use gdb commands to debug the RTOS application.

6 Reference materials

- 1. Board Remote Control Utilities(BCU) Release Notes (document BCU)
- 2. OpenOCD User's Guide (document Openocd)
- 3. Real-time Edge Software User Guide (document REALTIMEEDGEUG)
- 4. i.MX Linux User's Guide (document IMXLUG_6.6.23_2.0.0)
- 5. i.MX Yocto Project User's Guide (document <u>IMXLXYOCTOUG_6.6.23_2.0.0</u>)

7 Note about the source code in the document

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

Copyright 2024 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.

8 Revision history

Table 4. Revision history

Document ID	Release date	Description
AN14367 v.1.0	01 October 2024	Initial version

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.

Amazon Web Services, AWS, the Powered by AWS logo, and FreeRTOS — are trademarks of Amazon.com, Inc. or its affiliates.

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, μ Vision, 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.

Debugging Cortex-A U-Boot and Native RTOS on i.MX 8M Plus and i.MX 93 EVKs

Contents

1	Introduction	2
2	Definitions, acronyms, and	
	abbreviations	2
3	Hardware and software setup	2
3.1	Hardware materials	2
3.2	Hardware connection	3
3.3	Software setup	3
3.3.1	Software setup on Ubuntu PC	4
4	U-Boot OpenOCD debugging	5
4.1	OpenOCD Debugging with JTAG	
	Introduction	5
4.2	Debugging U-Boot with Eclipse IDE	6
4.2.1	Software requirements	6
4.2.2	Eclipse OpenOCD Configuration for U-Boot	
	debugging	7
4.2.3	Eclipse debugging steps	16
4.2.4	Debugging U-Boot with on GDB terminal	25
4.2.4.1	Common commands	26
4.2.4.2	Stack commands	27
4.2.4.3	breakpoint	28
4.2.4.4	Control and view commands	30
4.2.4.5	GDB layout split commands	31
5	Native RTOS OpenOCD debugging	31
5.1	Native RTOS on Cortex-A Core introduction .	31
5.2	EVK board software setup	31
5.3	Debugging Native RTOS with Eclipse IDE	32
5.3.1	Building Native RTOS with Eclipse IDE	32
5.3.2	Eclipse OpenOCD configuration for Native	
	RTOS debugging	36
5.3.3	Eclipse debugging steps	40
5.4	Debugging Native RTOS by using	
	OpenOCD with GDB	42
5.4.1	Setup and build RTOS	42
5.4.2	GDB Debugging steps	42
6	Reference materials	45
7	Note about the source code in the	
	document	45
8	Revision history	45
	Legal information	46

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

© 2024 NXP B.V.

All rights reserved.

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

Document feedback Date of release: 1 October 2024 Document identifier: AN14367