NP

Freescale Semiconductor Application Note Document Number: AN4876

CodeWarrior U-Boot Debugging

1. Introduction

This document describes the steps required for U-Boot debugging using the CodeWarrior IDE.

This document includes the following sections:

- Configuring and building U-Boot.
- Creating a CodeWarrior project to debug Uboot.
- Specifying the launch configuration settings.
- Debugging U-Boot from NOR, NAND, SPI, and SD card flash devices for low-end and high-end Power Architecture CPU.

2. Preliminary background

U-Boot resides in flash memory on target systems and boots an embedded Linux image or other OS image (vxworks) or an elf, developed for those systems.

Before debugging U-Boot on a target system, follow these steps:

- 1. Install the Board Support Package (BSP) for a target system you want to debug on the host Linux machine.
- 2. Configure the BSP U-Boot package to place

© Freescale Semiconductor, Inc., 2014. All rights reserved.

Contents

۱.	Introduction1
2	Preliminary background1
3.	Configuring and building U-Boot
ŀ.	Configuring a CodeWarrior project2
5.	Debugging U-Boot from NOR for e500v28
) .	Debugging U-Boot from NAND for e500v214
7.	Debugging U-Boot from SPI/SD/MMC for e500v2
	27

- 8. Debugging U-Boot from NOR for e500mc32
- 9. Debugging U-Boot from NAND for e500mc..38
- 10. Debugging U-Boot from SPI/SD/MMC for e500mc
- 12. Troubleshooting Tips51



Configuring and building U-Boot

debugger symbolic information in the U-Boot binary executable file.

- 3. Configure hardware to use the U-Boot image. (For more information, see Chapter 7.5.3 of Targeting PA Processor.pdf)
- 4. Create a new CodeWarrior project that you will use to debug U-Boot on the target system.

3. Configuring and building U-Boot

After installing BSP, configure and build U-Boot images for CodeWarrior debug. For more information on configuring and building U-Boot with CodeWarrior debugger support, see the SDK User Manual.

4. Configuring a CodeWarrior project

This section covers:

- Creating a CodeWarrior project
- Configure initialization file of project for debugging
- Board hardware configuration

4.1. Creating a CodeWarrior project

- 1. Open CodeWarrior IDE.
- 2. Choose **File** > **Import**, to import the U-Boot .elf file generated during the U-Boot compilation. It can be found in u-boot folder.

Figure 1. CodeWarrior File menu



3. Choose the source to import and select Next.



Figure 2. Import executable file dialog

* Import	
elect	
Import a CodeWarrior Executable file and create a project	
Select an import source:	
type filter text	
🕨 🧽 General	
▷ 🗁 C/C++	
🔺 🗁 CodeWarrior	
CodeWarrior Classic Project	
CodeWarrior Executable Importer	
Example Project	
▷ 😂 CVS	
> Constall	
Processor Expert	
Software Analysis	
> 🕞 Team	
Other	

4. Specify **Project name** and **Location**, or use the default location and select **Next**.

Figure 3. Import executable file dialog



5. Browse to the U-Boot .elf file and select open. By default, CodeWarrior looks for an .elf extension, so change the file type in lower right corner of select file dialog, as shown in Figure 4.



Configuring a CodeWarrior project

Figure 4. Select U-Boot elf file

Select the file to import	bler Executable Files			
File to import ace\pa	131024_u-boot_debug\P1010 ed file to current project folder	RDB_NOR\Debug\u-b	oot Browse	
Select file				_ ×
🔘 🖓 🖉 🖉	010RDB_NOR + Debug	- 49	Search Debug	
Organize 🕶 Ne	w folder		J== •	
 ☆ Favorites Desktop Downloads Recent Places Libraries Documents Music Wisic Videos 			06-Dec-13 9:41 Å	Type
🕵 Computer 🖗 Primary (C:)	* •	m		
	File name:	•	*	•

6. Select processor type for the project and select Next.

Figure 5. Select processor type

Processor Choose the processor for this pr	oject		
Processor			
type filter text			
> Qonverge			*
A QorIO P1 P1010			
P1011			m
P1012			
P1015			
P1014			
P1016			1
P1017			
P1020			
P1021			
P1022			
P1023			
P1024			
P1025			
b QorlQ_P2			
Toolchain			
Bareboard Application			
C Linux Application			
C and the second			
Target OS			
None			
C Linux Kernel			

7. Select Debugger Connection Types, Board, and Connection Type.



Figure 6. Target settings dialog

	et Settings			
Target Settin	igs			
Debugger C Har	onnection Types dware ilator			De
Board	P1010RDB	-		
Launch	Connectio	on		
Dov	mload 🖉 Defa	ault		
🔽 Atta	ch 🦨 Defa	ault	+	
Connection			1	
TAP address	use loss the		<u> </u>	

8. Select the configuration that you want to create and then, select **Finish** to close the wizard.

Figure 7. Select configuration dialog

Import a CodeWarrior Executable file	
Configurations	
Choose the configurations you want to create	
Core index	
Core 0	
< Back Next >	Finish Cancel

4.2. Configure initialization file of project for debugging

- 1. Choose **Run** > **Debug configurations**, to open the **Debug configurations** dialog.
- 2. Select Project name from the left pane and from the right pane, under Main tab **Target** settings, select Edit, as shown in Figure 8.



Configuring a CodeWarrior project

Figure 8. Debug Configurations dialog

🗱 🗎 🕆 Name: P1010RDB_NOR_Debug_P1010_Attach						
Main (4): Argum	ents 🕸 Debugger 🖐 Trace	and Profile by Source ma	Environment	Common		
Debug session type Choose a predefined debug session type or custom type for maximum flexibility Download Connect a datach Custom						
▼ C/C++ application						
Project:	P1010RDB_NOR			Browse		
Application:	Debug/u-boot	Search Project	Browse	Variables		
Build (if required) before launching						
▼ Target settings						
Connection: Execute reset seque	- P1010RDB_NOR_De	tbug_P1010_Attach ⊤	Edit	New		
				10.00		
	Name P1010RDB_NOR Main OF-Argum Debug session type Choose a preddined d Attach C(C+ application Project: Application: Build (if required) b Target settings Connection: Execute initialization Execute initialization	Name: P1010RD8_NOR_Debug_P1010_Attach Main 69-Arguments Trace Debug session type Concet Concet Openhoad Connect Connect # Attach © Custom Connect # Openhoad Connect Connect # Attach © Custom Connect # C/C++ application Debug/u-boot Ebug/u-boot # Build (if required) before launching Target settings Connection: # P1010RDB_NOR_Dr Execute reset sequence Execute reset sequence Execute initialization script(s)	Name: P1010RDB_NOR_Debug_P1010_Attach Main Or Arguments Sp Debugger Trace and Profile Sp Source Sp Debugger Debug session type Connect Connect Source Sp Debugger Source Source<	Name P1010RDB_NOR_Debug_P1010_Attach Image: Constant in the image: Constant in t		

3. From the **Hardware or Simulator Connection** dialog, select **Edit** to edit the target as shown in the Figure 9.

Figure 9. Hardware or Simulator Connection dialog

Hardware or Simulator Conr	Hardware or Sim	ulator Connection	$\phi \bullet \phi \bullet$
	Parent profile:	rro03-ws466	
	Name:	P1010RDB_NOR_Debug_P1010_Attach	
	Template:	None •	Apply Defaults
	Target:	P1010RDB_NOR_Debug_P1010_Attach Target	New
	Connection type:	USB TAP	
	Connection Ad	vanced	
	USB TAP	mber:	
	JTAG settings	1020 IO20	
	CCS server	cu (M) (K). A VERV	
	Automatic I	lunch	
	Server port	number: 41475	
	CCS exe	utable:	
	Manual laur	ch	
	Server host	ате/ЛР: 127.0.0.1	
	Server port	number: 41475	
	Connect	server to TAP	

4. From the **Initialization** tab, browse to the location of U_Boot initialization file and add its location in the **Initialize target**, as shown in <u>Figure 10</u>.



Hardware or Simulator Targe	Hardware or	Simulator Target		$\diamondsuit \bullet \diamondsuit$
	Parent profile:	zro03-ws466		
	Name:	P1010RDB_NOR_Debug_P1010_/	Attach Target	
	Description:			
	Template:	None		Apply Defaul
	Target type:	P1010		- Edit
	Initialization			
	mitianzation	Memory Advanced		
	Run e	eset out of reset	La .	
	✓ Initialize	target: \${PA_TOOLS_HOME}/P	A_Support/Initialization_Files/QorIQ_P1/P10	10RDB_uboot_3 🛄

Figure 10. Hardware or Simulator Target dialog

5. Navigate to **Memory** tab and deselect **Memory configuration**.

Figure 11. Hardware or Simulator Target dialog

Hardware or Simulator Targe	Hardware or	Simulator Target	$\Leftrightarrow \bullet \Leftrightarrow \bullet$
	Parent profile Name:	: zro03-ws466 P1010RDB_NOR_Debug_P1010_Attach Target	
	Description: Template:	None	Apply Defaults
	Target type:	P1010 -	Edit
	Internet	Advanced	
	Memory	y configuration: S[PA_TOOLS_HOME]/PA_Support/Initialization_Files/Memory/P	1010RDB.n
	Memory	y configuration: S[PA_TOOLS_HOME]/PA_Support/Initialization_Files/Memory/P	1010RDB.n

6. Select **OK** to exit the **Debug configurations** dialog.

4.3. Board hardware configuration

See the SDK User Guide for the correct board configuration and switch settings.

U-Boot debug is JTAG-based and a probe needs to be connected to the board. NOTE

CodeWarrior U-Boot Debugging Application Note

Freescale Semiconductor

Debugging U-Boot from NOR for e500v2

4.4. Useful hints and tips

Refer to Chapter 12, for useful hints and tips.

5. Debugging U-Boot from NOR for e500v2

5.1. Debug environment

Use the following setup for U-Boot debugging on e500v2 core:

- P1010RDB board.
- Compiled U-Boot for the NOR FLASH target.
- Flash U-Boot on the target board. (See SDK documentation, for more information on how to program the U-Boot to NOR flash.)
- Switches set for NOR boot. (See SDK documentation, for more information on how to set switches.)
- Latest release of CodeWarrior IDE.
- P1010RDB_uboot_32.tcl initialization file.
- USB TAP or other probe.

5.2. U-Boot NOR debugging

The U-Boot .elf file generated during the U-Boot compilation should be imported as CodeWarrior project. (See <u>Configuring a CodeWarrior project</u>, for more information.)

5.2.1. Stage 0 – Connect CodeWarrior to a board

Before debugging, run the board in the debug mode.

1. Choose **Run** > **Debug configurations**, to open **Debug configurations** dialog and select **Debug**, as shown in Figure 12.



A C/C++ - - CodeWarrior Development Stu 0 (11 E C/C++ CodeWal St Debug Configurations the second Create, manage, and run configurations D File Name Debug or run an application to a target P101 P101 P101 3 🗈 🗙 | 🖻 🕸 • Name: P1010RDB NOR Debug P1010 Attach 📄 Main 💀 Arguments 🕸 Debugger 👼 Trace and Profile 🤤 Source 🖉 Environment 🗔 Common C CodeWarrior C P1010RD8_NOR_Debug_P1010_Attach Launch Group Target Communication Framework Dobug session type
Choose a predefined debug session type or custom type for maximum flexibility
 Download
 Connect Custom Attach ▼ C/C++ application P1010RDB NOR Browse... Project Application: Debug/u-boot Search Project... Browse... Variables... Build (if required) before launching - P1010RDB_NOR_Debug_P1010_Attach + Edit... New... Connection: Execute reset sequence Execute initialization script(s) Comm Filter matched 4 of 4 items · Proiec impor CodeV Filter by Project: P1010RDB NAND RAMBOOT P1010RDB_NAND_SPL → Build/D ≪ Bui ♥ Cle 尊 Det Apply Revert ? Close Debug

Figure 12. Debug configurations dialog

2. The connection initializes and configures the TAP, and then it will attach to board.

Figure 13. Debug window



3. To reinitialize the target from CodeWarrior, select **Reset**, as shown in Figure 14.

Figure 14. Reset dialog



5.2.2. Stage 1 – Debug NOR for AS0

1. Set PIC load address as 0xfff80000, using Debugger Shell command setpicloadaddr 0xfff80000.

Figure 15. File location

C (AsmSection)() at /home/freescale	/SDK/QorIQ-SDK-V1.4-20	130625-yocto/build_p1010rdb_release	/tmp/work/p1010rdb-fsl_network	ing 🛿 🗖 🗖	Disassembl	y 🕄 📲 Outline		- 1
Can't find a source file at "/home/free	scale/SDK/QorIQ-SDK-V1	.4-20130625-yocto/build p1010rdb n	elease/tmp/work/p1010rdb-fsl_ne	working-linux-		Enter location here	- 2 6	E C C
View Diassembly Locate File Edit Source Lookup Path Apply to Common Source Lookup I	caaeouw Qonq-sux-vi	4-2013062-yoctorbuild_pittindb_n	eessertmprwork/pJJJJVrab-Hilne	working-inux-	fffffh4: fffffh8: fffffh8: fffffh8: fffffh6: ffffffc8: ffffffc8: ffffffc8: ffffffd8: ffffffd8: ffffffc4: ffffffc4: ffffffc4: ffffffc4: ffffffc6: fffffc6: fffffc6: fffffc6: fffffc6: fffffc6: ffffffffff	Enter Docadon Here dc.1 0xffffffff dc.1 0xfffffff dc.1 0xffffffff dc.1 0xffffffff dc.1 0xffffffff dc.1 0xffffffff dc.1 0xffffffff	fff000); 0x	FFFFF000
Communities 50		Convert 22 R Tarley 8 M		0	Denne (18	Dahuman Chall 52	-	
Project Creation Import project	Miscellaneous Welcome screen Output	EPPC, u-boot, core 0		Proble ■ • □ •	-o Progress	p bebugger anen 23		
Codewarrior bareboard Project	Flash programme							
Build (Al) Crean (Al) Crean (Al) Debug Settings Project settings Debug settings Debug settings					%>setpicload Debugger nov address.	iaddr 0xfff80000 v assumes 'u-boot' is	: loaded at t	he specified
*m		4			%>			

2. After the path is provided, source will become available in CodeWarrior. (See Figure 15, for more details.)

Figure 16. File editor



3. Now debugging (step, run, or breakpoint) can be done before switching to AS1.



Debugging U-Boot from NOR for e500v2

Figure 17. File editor

🛐 start.S 🕱	
89 */	*
90	
91 .section .bootpg,"ax"	
92 .globl_start_e500	
93 45	
94_start_e500:	
95/* Enable debug exception */	
96 li r1,MSR_DE	
97 mtmsr r1	
98	
99 #ifdef CONFIG_SYS_FSL_ERRATUM_A004809	
100 mfspr r3,PVR	
101 rlwinm r3,r3,28,0xf /* major_rev */	
102 cmpwi r3,0x1 /* is rev 1? */	
103 bne 1f	
104	
105 msync	
106 isync	
107 mfspr r3, SPRN_HDBCR0	
108 oris r3, r3, 0x0100	
109 ori r3, r3, 0x8000	
110 mtspr SPRN_HDBCR0, r3	-
*	•

4. In file start.S, last instruction before moving to AS1 is rfi before switch_as. (See <u>Stage 2</u> – <u>Debug NOR for AS1</u>, for more information.)

Figure 18. File editor



5.2.3. Stage 2 – Debug NOR for AS1

- 1. Step Into rfi instruction.
- 2. Reset PIC load address, using Debugger Shell command setpicloadaddr reset.



Debugging U-Boot from NOR for e500v2

Figure 19. Debugger shell view



3. Debugging (step, run, or breakpoint) can be done until code is relocated in DDR.a) Run to Line: board_init_f and Step Into.

Figure 20. File editor

🔁 board.c 🔀	- 8
<pre>369 unsigned long logbuffer_base(void) 370 { 371 return CONFIG_SYS_SDRAM_BASE + get_effective_memsize() - LOGBUFF_LEN; 372 } 373 #endif</pre>	*
374 375 void board_init_f(ulong bootflag) 376 Ø	
377 bd_t *bd; 378 ulong ten, addr, addr_sp; 379 ulong ten; 380 gd_t *id; 381 init_fnc_t **init_fnc_ptr; 382	
383 #ifdef CONFIG_PRAM 384 ulong reg; 385 #endif	
<pre>386 387 /* Pointer is writable since we allocated a register for it */ 388 gd = (gd_t *) (CONFIG_SYS_INIT_RAM_ADDR + CONFIG_SYS_GBL_DATA_OFFSET); 389 /* compiler optimization barrier needed for GCC >= 3.4 */ 390asmvolatile_(^*:::memory*);</pre>	+
	P

b) Run to Line: relocate_code and Step Into.

Figure 21. File editor





c) In start.S, last instruction before relocate to DDR is relocate_code.

Figure 22. File editor



d) Step Into blr, it shows the code in assembly. (See <u>Stage 3 – Debug in DDR's higher address</u>, for more information.)

5.2.4. Stage 3 – Debug in DDR's higher address

1. Set the PIC load address as 0x3ff30000 using Debugger Shell command setpicloadaddr 0x3ff30000. (See <u>How to calculate PIC load address</u>, for more information.)

Figure 23. Debugger shell view



- 2. You can debug until U-Boot is running.
 - a) Run to Line: board_init_r and do Step into.

Figure 24. File editor

🔁 board.c 🔀	- 0
627/* 628 * This is the next part if the initialization sequence: we are now 629 * running from RAM and have a "normal" C environment, i. e. global 639 * data can be written, BSS has been cleared, the stack size in not 631 * that critical any more, etc. 632 */ 633 void board_init_r(gd_t *id, ulong dest_addr)	*
 634 k bd_t *bd; 635 vlong malloc_start; 637 638 #ifndef CONFIG SVS_NO_FLASH 639 ulong flash_size; 640 #rendif 	
641 gd = id; /* initialize RAM version of global data */ 643 bd = gd->bd; 644	
645 gd->flags = GD_FLG_RELOC; /* tell others: relocation done */ 646 647 /* The Malloc area is immediately below the monitor copy in DRAM */ 648 malloc_start = dest_addr - TOTAL_MALLOC_LEN;	*
· ·	E State

b) Run to Line: main_loop()

Figure 25. File editor



6. Debugging U-Boot from NAND for e500v2

U-Boot NAND boot is a 2-stage booting process:

- First stage (U-Boot NAND SPL) when turned on and on reset, U-Boot NAND SPL gets the control. It runs from IFC's internal SRAM and it copies U-Boot RAMBOOT to DDR and transfers control to it.
- Second stage (U-Boot NAND RAMBOOT) RAMBOOT code.

Depending upon the booting stage, U-Boot NAND debugging can be classified into two modes:

- <u>U-Boot NAND SPL debugging</u>
- <u>U-Boot NAND RAMBOOT debugging</u>

6.1. Debug environment

Use the following setup for U-Boot NAND debugging on e500v2 core:

- 1. P1010RBD board.
- 2. Compiled U-Boot for the NAND FLASH target.

Debugging U-Boot from NAND for e500v2



- 3. Flash U-Boot on the target board. (See SDK documentation, for more information on how to program the U-Boot to NAND flash.)
- 4. Switches set for NAND boot (See SDK documentation, for more information on how to set switches.)
- 5. Latest release of CodeWarrior IDE.
- 6. P1010RDB_uboot_32.tcl initialization file.
- 7. USB TAP or other probe.

6.2. U-Boot NAND SPL debugging

For this stage the U-Boot-spl elf file generated during U-Boot compilation should be imported as a CodeWarrior project. (See <u>Configuring a CodeWarrior project</u>, for more details.)

6.2.1. Stage 0 – Connect CodeWarrior to board

Before starting debugging, run the project in debug mode.

1. Choose **Run** > **Debug** configurations, to open **Debug configurations** dialog and select **Debug**.

Figure 26. Debug configurations dialog

🗅 📴 🗙 🖻 🌟 🗕	Name: P1010RDB_NA	ND_SPL_Debug_P1010_Attach				
type filter text	Main (x)= Argun	nents 🕸 Debugger 💺 Trace and P	rofile 🤤 Source 🚾 Environ	ment 🔲 Comr	mon	
 CodeWarrior P1010RDB_NAND_SPL_Debug_P1010_Attach Launch Group Target Communication Framework 	Debug session type Choose a predefined Download Attach	debug session type or custom type fo Connect Custom	r maximum flexibility			
	▼ C/C++ application					
	Project: P1010RDB_NAND_SPL Browse					
	Application:	Debug/u-boot-spl	Search Project	Browse	Variables	
	Build (if required)	before launching				
	Connection:	P1010RDB_NAND_SPL_De ence on script(s)	ebug_P1010_Attach 🔻	Edit	New	
Filter matched 4 of 4 items						
ilter by Project:						
P1010RDB NAND SPL						

2. This initializes and configures the TAP, then attaches to board.



Debugging U-Boot from NAND for e500v2

Figure 27. Debug view



3. Reinitialize the target from CodeWarrior, using the U-Boot initialization file.

Figure 28. Reset dialog



6.2.2. Stage 1 – Debug NAND SPL in IFC SRAM for AS0

1. Browse to the location, where the source file is saved, as shown in Figure 29.





🌣 Debug 🛙 🦳 🧮	🖾 🖉 Variables 🖾 🤇	💩 Breakpoints 🧻 Cache 👯 Re
	~	
â ⋈ Ə 🕨 🗉 📕 🖓 🕹 🥄 🕫 এ ≂ i≯ 🕱 🛷 🗰 🖽 📲 🛲 - ≣ + 🛼 🖫 🖫	Name	Value
- ▼ 事		
C P1010RDB_NAND_SPL_Debug_P1010_Attach [CodeWarrior]		
EPPC, u-boot-spl, core 0 (Suspended) Thread IID: 0-01 (Suspended: Single 'Evention 5276' received, Description, DewarDC Evention)		
I (AsmSection)() resetvec.S:2 0xffffffc		
D:\workspace\pa\131024_u-boot_debug\P1010RDB_NAND_SPL\Debug\u-boot-spl (11/20/13 4:15 PM)		
(AsmSection)() at /home/freescale/SDK/QorlQ-SDK-V1.4-20130625-yocto/build_p1010rdb_release/tmp/work/p Can't find a source file at '/home/freescale/SDK/QorlQ-SDK-V1.4-20130625-yocto/build_p1010rdb_release/tmp/w r33/git/arch/powerpc/cpu/mpc85xx/resetvec.S'	<pre>1010rdb-fsl_networking-linu ork/p1010rdb-fsl_networking</pre>	x-gnuspe/u-boot/ 🙁 🖳 [g-linux-gnuspe/u-boot/git-
(AsmSection)() at /home/freescale/SDK/QorlQ-SDK-V1.4-20130625-yocto/build_p1010rdb_release/tmp/work/p Can't find a source file at '/home/freescale/SDK/QorlQ-SDK-V1.4-20130625-yocto/build_p1010rdb_release/tmp/w 133/git/arch/powerpc/cpu/mpc85xx/resetvec.5" Locate the file or edit the source lookup path to include its location. View Disassembly	4 1010rdb-fsl_networking-linu ork/p1010rdb-fsl_networking	x-gnuspe/u-boot/ 🛛 🗖 🛛
(AsmSection)() at /home/freescale/SDK/QorlQ-SDK-V1.4-20130625-yocto/build_p1010rdb_release/tmp/work/p Can't find a source file at '/home/freescale/SDK/QorlQ-SDK-V1.4-20130625-yocto/build_p1010rdb_release/tmp/w 13/ git/arch/powerpc/cpu/mpc85xx/resetvec.5" Locate the file or edit the source lookup path to include its location. View Disassembly Locate File	< 1010rdb-fsl_networking-linu ork/p1010rdb-fsl_networking	x-gnuspe/u-boot/ 🛛 🦳 🗖
(AsmSection)() at /home/freescale/SDK/QorlQ-SDK-V1.4-20130625-yocto/build_p1010rdb_release/tmp/work/p Can't find a source file at '/home/freescale/SDK/QorlQ-SDK-V1.4-20130625-yocto/build_p1010rdb_release/tmp/w 19/git/arch/pwerpc/cpu/mpc85sc/restvec.5" Locate the file or edit the source lookup path to include its location. View Disassembly Locate File Edit Source Lookup Path	4 1010rdb-fsl_networking-linu ork/p1010rdb-fsl_networking	x-gnuspe/u-boot/ 🛛 🦳 🗖
	<pre>def control contr</pre>	x-gnuspe/u-boot/ 🛛 🦳 🗖 [g-linux-gnuspe/u-boot/git-
	<pre>def control contr</pre>	x-gnuspe/u-boot/ 🛛 🦳 🖶
	<pre>definition of the second second</pre>	x-gnuspe/u-boot/ 🖾 🦳 🖻
	<pre>definition of the second second</pre>	x-gnuspe/u-boot/ 🛛 📼 g-linux-gnuspe/u-boot/git-
(AsmSection)() at /home/freescale/SDK/QorlQ-SDK-V1.4-20130625-yocto/build_p1010rdb_release/tmp/work/p Can't find a source file at "/home/freescale/SDK/QorlQ-SDK-V1.4-20130625-yocto/build_p1010rdb_release/tmp/w r33/git/arch/powerpc/cpu/mpc35x0/reservec.S" Locate the file or edit the source lookup path to include its location. View Disassembly Locate File Edit Source Lookup Path Apply to Common Source Lookup Path	<pre>dullordb-fsl_networking ork/p1010rdb-fsl_networking</pre>	x-gnuspe/u-boot/ 🛛 📼
(AsmSection)() at /home/freescale/SDK/QorlQ-SDK-V1.4-20130625-yocto/build_p1010rdb_release/tmp/work/p Can't find a source file at "/home/freescale/SDK/QorlQ-SDK-V1.4-20130625-yocto/build_p1010rdb_release/tmp/w r33/git/arch/powerpc/cpu/mpd5xo/reservec.S" Locate the file or edit the source lookup path to include its location. View Disassembly Locate File Edit Source Lookup Path Apply to Common Source Lookup Path	4 1010rdb-fsl_networking ork/p1010rdb-fsl_networking	x-gnuspe/u-boot/ 🛛 📼

2. After the path is specified, the source is available in CodeWarrior.

Figure 30. File editor



3. Now, debugging (step, run, or breakpoint) can be done before switching to AS1.



Figure 31. File editor

	start.S	23		- E
	89 * 90 91 92 93	.sectio .globl	n .bootpg,"ax" _start_e500	^
	94_5	start_e500	:	
5	957	li c1	Mob DE	
Ĩ	97	mtmsr	r1	
	90 #1	fdef CONE	TG SYS ESI ERRATIM A004809	
	100	mfspr	r3,PVR	
	101	cmpwi	r3,0x1 /* is rev 1? */	
	103	bne 1f		
	105	msync		
	106	isync		
	107	mfspr	r3, SPRN_HDBCR0	
	108	oris	r3, r3, 0x0100	
	109	ori r3,	r3, 0x8000	
	110	mtspr	SPRN_HDBCR0, r3	*
	4			N

4. In file start.S, last instruction before moving to AS1 is rfi before switch_as. (See <u>Stage 2</u> – <u>Debug NAND SPL in IFC SRAM for AS1</u>, for more information.)

Figure 32. File editor



6.2.3. Stage 2 – Debug NAND SPL in IFC SRAM for AS1

- 1. Step Into this instruction.
- 2. Debugging is possible until the code is relocated to DDR.
 - a) Run to Line: board_init_f and Step Into: board_init_f.



Figure 33. File editor

🖸 spl_n	ninimal.c 🕱	- 0
94 95 96	<pre>/* Let the controller go */ out_be32(&ddr->sdram_cfg, in_be32(&ddr->sdram_cfg) SDRAM_CF6_MEM_EN);</pre>	*
97 98 } 99	<pre>set_next_law(CONFIG_SYS_NAND_DDR_LAW, LAW_SIZE_1G, LAW_TRGT_IF_DDR_1);</pre>	
100 vo	<pre>bid board_init_f(ulong bootflag)</pre>	
101 {		
102	u32 plat_ratio;	
103 104	ccsr_gur_t *gur = (võid *)CONFIG_SYS_MPC85xx_GUTS_ADDR;	
105	/* initialize selected port with appropriate baud rate */	
106	<pre>plat_ratio = in_be32(&gur->porpllsr) & MPC85xx_PORPLLSR_PLAT_RATIO;</pre>	
107	<pre>plat_ratio >>= 1;</pre>	
108 109	gd->bus_clk = CONFIG_SYS_CLK_FREQ * plat_ratio;	III -
110	NS16550_init((NS16550_t)CONFIG_SYS_NS16550_COM1,	
111	<pre>gd->bus_clk / 16 / CONFIG_BAUDRATE);</pre>	
112		
113	<pre>puts("\nNAND boot ");</pre>	
114		
112	/* INITIALIZE THE DUKS */	
4		*

b) Run to Line: relocate_code and do Step Into.

Figure 34. File editor



c) In file start.S, last code before relocate to DDR is relocate_code.

Figure 35. File editor

S start.S	23		- 8
1879 *	*/		*
1880			
1881	addi	r0,r10,in_ramstart + _START_OFFSET	
1882			
1883	/*		
1884	* As 1	CVPR is going to point RAM address,	
1885	* Make	e sure IVOR15 has valid opcode to support debugger	
1886	*/	N	
1887	mtspr	IV:R15, r0	
1888			
1889	/*		
1890	* Re-p	point the IVPR at RAM	
1891	*/		
1892	mtspr	IVPR,r10 .	
1893			
1894	mtlr	r0	
1895	blr	/* NEVER RETURNS! */	
1896	.globl	in_ram	
1897 ir	n_ram:		
1898			
1899	/*		
1900	* Relo	ocation Function, r12 point to got2+0x8000	-
4			F

3. Step Into: blr, it shows code in assembly. (See <u>Stage 3 – Debug in RAM</u>, for more information.)

6.2.4. Stage 3 – Debug in RAM

1. Set PIC load address as 0x100000 in Debugger Shell using setpicloadaddr 0x100000 command.

Figure 36. Debugger shell view

S start.S 🛛			- 0	E Disassembly	83 E Outline	
1898 1899 /* 1900 * Relocation Funct 1901 * 1902 * Adjust got2 poin 1903 * already puts a f	ion, r12 point to g ters, no need to ch ew entries in the t	bt2+0x8000 tck for 0, this code bble.	•	 0010018c: 00100190: 00100194: 00100198: 0010019c: 	Enterlocation here Ii r0,13 subi r3,r12,32768 lwz r11,-32768(r12) mtctr r0 subf r11,r11,r3	0 6 30 13 3 7
1005 11 c0get2.entc1 1006 n.e7.50T(0072.7A 1007 1006 mtctr P0 1008 1018 add r3,r3,r11 1018 1018 add r3,r3,r4 1011 1018 add r0,r0,r4 1011 1013 add r0,r0,r11 1015 1014 add r0,r0,r11 1015 1015 stw r0,0(r3) 1015 1017 1018 /* 1019 * Now adjust the f	1993 * already puts a few entries in the table. 1995 * already puts a few entries find table. 1996 la * obs(0 (6072, TABLE)) 1987 hur n11,607(6072, TABLE) 1988 mctr r0 1990 sub r11,r3,r11 1911 hur n4,(r3) 1911 hur n4,(r3) 1913 beq - 2f 1914 add r0,rd,r11 1915 the r0, eft, r11 1915 the r0, eft, r11 1915 the r0, eft, r11 19162 beq - 2f 1917 /* 1917 /* 1917 /* 1918 filme addict the filmer and the minimum to the filmer.			0010014019c; 001001400; 00100140; 00100140; 00100140; 00100140; 00100140; 00100140; 00100140; 00100140; 00100140; 00100140; 00100140; 00100140;	Subf 71,71,71 Lucu (P0,4(73) Lucu (P0,4(73) Lucu (P0,4(73) add (P0,4(73) add (P0,4(73) add (P0,4(73) add (P0,4(73) add (P0,4(73) add (P0,4(73) add (P0,4(73) add (P0,4(73)) Lucu (P1,4(73)) Lucu (P1,4(73)) Lucu (P1,4(73)) Lucu (P1,4(71)) Lucu (P1,	; 0x00100188 ; 0x001001A4 ; 0x001001F4
A Commander 83	<u>@ ▽ □ □</u>	🔄 Consol 🕴 🖉 Tasks 🔋 Memor 🚜 Remot 🌹	Target 💽 Proble 🛛 🗖 🗖	Rogress	Debugger Shell 83	
Project Creation Import project Mayor project CodeWarnice Bareboard Project Build/Ocbug Build (All) Clean (All) Clean (All) CodeWarning Debug Portect settings Debug settings Debug settings	Miscellaneous Welcome screen Quick access Flash programme	EPPC, u-boot-spl, core 0 🔳 😹 强 📴		CodeWarrior D %>setpicloada Debugger now address.	bebugger Shell v1.0 ddr 0x10000 assumes 'u-boot-spl' is low	ded at the specified
* m		4	F	3>		

- 2. Debug until U-Boot RAMBOOT code is copied from NAND to RAM and control is transferred to it.
 - a) Run to Line: board_init_r and do Step Into.

Figure 37. File editor



b) Step Into: nand_boot() function.

Figure 38.



c) This is the last function before control is transferred to u-boot RAMBOOT. Run to Line: uboot() and do Step Into. As soon as we Step Into uboot() function, control is transferred to U-Boot NAND RAMBOOT, that is, 0x00200000. This address is used to set PIC load address for U-Boot NAND RAMBOOT debug.

Figure 39.

C 0x00200000 (0x00200000)() 🛙	- D = Di	sassembly 🕄 📴 Outline	- 0
No source available for "0x00200000 (0x00200000)() *	-	Enter location here	- 2 6 3 8 0 0 0 7
View Disassembly		00000: 11 rsp;512 00000: 12 rsp;120, r0 00000: 15 rsp;120, r0 00010: mtspr sp;120, r0 00011: mtspr sp;120, r0 00011: mtspr sp;34, rsp 00011: mtspr sp;34, rsp 00021: mtspr sp;43, r4 00031: mtspr sp;43, r4 00031: mtspr sp;43, r4 00031: mtspr sp;43, r4 00041: mtspr sp;43, r4	•
Community (2) (2 C C C C C C C C C C C C C C C C C C		(
Figure Commander & Kemot & larget Proble	-9 -	ogress Debugger Shell 23	
Project (Velation	*		

 d) Further debugging is not possible with this u-boot-spl.elf and a new project needs to be created for U-Boot NAND debugging. (See <u>U-Boot NAND RAMBOOT debugging</u>, for more information.)

6.3. U-Boot NAND RAMBOOT debugging

For this stage, the U-Boot elf file generated during U-Boot compilation should be imported as a CodeWarrior project. (See <u>Configuring a CodeWarrior project</u>, for more information.)



6.3.1. Stage 0 – Connect CodeWarrior to board

- 1. Restart the board. U-Boot starts and relocates itself into RAM.
- 2. Before Debugging, run the board in Debug mode.

Figure 40. Debug configurations dialog

C/C++ - CodeWarrio	or Development Studio					
File Edit Source Ri	efactor Navigate Search Project Run ProcessorExpert V	Nindow Help				
	19 • % • % (Q / # • 111 • 111 • 111 • 111	8 - 8 - 8 -	\$ · Q · Q · B /·	n in - 5 - 6 - 6	• -	E E C/C
		1 40 144 144				P- a u m) @
LC CodeWarrior Project	Debug Configurations					-×
						15 .
File Name	Create, manage, and run configurations		-0			1
3041_PBL P1010RDB_N/	Debug or run an application to a target.					·∕₽°
🐉 Binaries 🗁 Debug		Name: P1010RDB_NAN	ID_RAMBOOT_Debug_P1010_Attach			
P1010RDB_NA	type filter text	Main 00+ Argum	ents 🕸 Debugger 👼 Trace and Profile	Source 📠 Environment	Common	
P1010RDB_NO	a CodeWarrior	Debug session type				
P1010RDB_SPI	P1010RDB_NAND_RAMBOOT_Debug_P1010_Attacl	Choose a predefined d	lebug session type or custom type for ma	imum flexibility		
P3041DS_NAM	Launch Group	O Download	Connect			
P304105_NOR	Target Communication Framework	Attach	Custom			
		▼ C/C++ application				
		Project	P1010RDB NAND RAMBOOT			Browse
		TTO A VILLATION			n. 1975	
		Application:	Debug/u-boot	Search Project	Browse	vanables
		 Build (if required) b 	before launching			
		▼ Target settings				
		Connection:	- P1010RDB_NAND_RAMBOOT	Debug_P1010_Attach +	Edit	New
		Execute reset seque	DCE			
(Everyte initialization	a script(r)			
Commander 52			(activity)			
Commanue is	Thu maked of them.					
 Project Creation 	Filter matched 4 of 5 items					
Import project	Filter by Project:					
- Codewarrior Bar	12-3041_PBL	(<u>e</u>	"	57	_	
▼ Build/Debug	P1010RDB_NAND_RAMBOOT					
≪ Build (All) ∉ Clean (All)	P1010RDB_NAND_SPL +				Apply	Revert
参 Debug					_	
▼ Settings	O				Debug	Close
Tel Destant anti-						

3. This initializes and configures the TAP, then attaches to the board.

Figure 41. Debug view



4. Reinitialize the target from CodeWarrior, using the U-Boot initialization file.







6.3.2. Stage 1 – Debug NAND RAMBOOT until U-Boot is relocated to DDR's higher address

- 1. Set PIC load address as 0x00200000 in Debugger Shell, using setpicloadaddr 0x00200000 command.
- 2. Set break point at address 0x00200008 in Debugger Shell using bp -hw 0x00200008 command.

NOTE The break point's address is needed to be offset with 8 bytes because it is required to jump over the instructions that enables the MSR[DE] bit, otherwise the break point will not hit.

3. Resume core operation.

Figure 43. Debug view



4. When break point is hit, source code location is asked by CodeWarrior. After the path is specified, it shows the source code in CodeWarrior.

Figure 44. File editor



- 5. Now debugging (step, run, or breakpoint) can be done until U-Boot code is relocated to the higher address of DDR.
 - a) Run to Line: board_init_f and do Step Into.

Figure 45. File editor



b) Run to Line: relocate_code and do Step Into.



Figure 46. File editor



6. In file *start*.*S*, last instruction before moving to the higher address of DDR is relocate_code.

Figure 47. File editor



7. Now Step Into blr, CodeWarrior will show the code in assembly. (See <u>Stage 2 – Debug in</u> <u>DDR's higher address</u>, for more information.)

6.3.3. Stage 2 – Debug in DDR's higher address

1. Set PIC load address as 0x3ff2f000 using Debugger Shell command setpicloadaddr 0x3ff2f000. (See <u>How to calculate PIC load address</u>, for more information.)



Debugging U-Boot from NAND for e500v2

Figure 48. Debugger shell view

S start.S 🕄					- 0	Disassembly 🕄 🗄 Outline		- 0
1898						Enter location here	- 2 6 SB	9 59 7
1899 /* 1990 * Relocation Funct: 1991 * Adjust got2 point 1992 * Adjust got2 point 1993 * already puts a for 1994 */ 1995 11 r0, got2_entrie 1996 1a r3,GOT (GOT2 TAL	ion, r12 point to go ters, no need to che w entries in the ta segsectoff@1 BLE)	xt2+0x8000 €ck for 0, this code bble.				3ff315b4: li r0,2549 3ff315b8: subi r3,r12,32768 3ff315b6: lwz r11,-32768(r12 3ff315c8: mtctr r0 3ff315c4: subf r11,r11,r3 3ff315c8: subf r3,r4, 3ff315c6: lwz r0,4(r3))	*
1997 lac r11,607(607,7; 1965 stdr r0 1956 sub r11,r3,r11 1910 addi r3,r3,-4 1911 lacu r04,6(r3) 1912 cnput r04,0 1913 beq-2f 1914 ad0 r0, r0, r11 1915 stw r0,0(r3) 1916 ztw r0,0(r3) 1917 "Now adjust the f 4 divatist the f	HBLE_) Lxups and the pointe	ers to the fixups				3ff315de: cmpuf n0, 0c0000 3ff315de: bqt 0x2F513560 (%) 3ff315de: add n0, n0, n11 3ff315de: bdn 0x3F7135CC (%) 3ff315de: bdn2 0x3F7135CC (%) 3ff315de: cmpuf n0, 0c0000 3ff315de: bdn2 0x3F7135CC (%) 3ff315de: cmpuf n0, 0c0000 3ff315de: sbb1 r3, r3, 2764 (r10) 3ff315de: sbb1 r3, r3, 4 (%) 3ff315de: laru r4, 4(r3) 3ff315de: laru r4, r4, r4, r1	3ff315e0) ; 0x3FF315E0 x3ff315cc) ; 0x3FF315C0 3ff3161c) ; 0x3FF3161C	c ,
A Commander 🛿		🖾 Consol 🕴 🖉 Tasks 🚺 Me	nor 🗿 Remot 🔞 Ta	arget 🔡 Proble	- 0	🕫 Progress 🔯 Debugger Shell 🖾	11 Bu	<u>}</u> - 0
Project Creation Import project CodeWarrior Bareboard Project D. 7400	Miscellaneous Welcome screen Quick access Flash programme	EPPC, u-boot, core 0	■ × ½ 0 ₄ ፩1		• 23 •	id instance address description #164 #1 v:0x00200008 start.S, line 273, (AsmSection) %>setpicloadaddr 0x0	type enabled? proces -hw ENABLED 0x [u-boot]	55 A x0
Sund (All) Clean (All) Bebug						Debugger now assumes 'u-boot' is address. %>bp -hw in_ram id instance address description	loaded at the specific	ss ≣
Settings Project settings Build settings Debug settings						<pre>#167 #1 v:0x000025b4 start.S, line 1905 [u-boot] %>setpicloadadd 0x3f72f000 Debugger now assumes 'u-boot' is address.</pre>	-hw ENABLED 09	k0 ed
4		141				K N		

- 2. You can debug until U-Boot is running.
 - a) Run to Line: board_init_r and Step Into.

Figure 49. File editor



b) Run to Line: main_loop()



Figure 50. File editor



7. Debugging U-Boot from SPI/SD/MMC for e500v2

Booting from SPI and SD\MMC are similar, the only difference between these is how the final image is build. This chapter provides necessary steps for SPI U-Boot debugging.

7.1. Debugging environment

Given below is the setup used for U-Boot debugging on e500v2 core:

- 1. P1010RDB board.
- 2. Compiled U-Boot for the SPI FLASH target.
- 3. Flash U-Boot on the target board. (See SDK documentation, for more information on how to program the U-Boot to SPI flash.)
- 4. Switches set for SPI boot. (See SDK documentation for more information on how to set switches.)
- 5. Latest release of CodeWarrior IDE.
- 6. P1010RDB_uboot_32.tcl initialization file.
- 7. USB TAP or other probe.

7.2. U-Boot SPI debugging

Import the U-Boot elf file, generated during U-Boot compilation as a CodeWarrior project. (See <u>Configuring a CodeWarrior project</u>, for more information.)

7.2.1. Stage 0 – Connect CodeWarrior to board

1. Before debugging, run the board in debug mode.



Debugging U-Boot from SPI/SD/MMC for e500v2

Figure 51. Debug configurations dialog



2. Start the debugging session using the setup provided in <u>Figure 51</u>. This initializes and configures the TAP, then attaches to the board.

Figure 52. Debug view



3. Reinitialize the target from CodeWarrior.





7.2.2. Stage 1 – Debug SPI until U-Boot relocated to DDR's higher address

- 1. Set hardware break point at address 0x1107f008 in Debugger Shell using *bp* -hw 0x1107f008 command.
- 2. Resume core operation.

Figure 54. Debug view



3. Once the break point is hit, specify the source code location. After the path is specified, it shows the source code in CodeWarrior.

Figure 55. File editor



4. Now debugging (step, run, or breakpoint) can be done until U-Boot code is relocated to the higher address of DDR.



- Debugging U-Boot from SPI/SD/MMC for e500v2
 - a) Run to Line: board_init_f and do Step Into.

Figure 56. File editor



b) Run to Line: relocate_code and do Step Into.

Figure 57. File editor



c) In file *start.S*, last instruction before moving to the higher address of DDR is relocate_code.

Figure 58. File editor



d) Step Into blr, it shows the code in assembly. (See <u>Stage 2– Debug SPI in DDR's higher</u> <u>address</u>, for more information.)



NOTE To find the correct address for hardware break point, that is, 0x1107008, disassembly on u-boot.elf is done and the _start_e500 address is searched for. Add 8 to this address to have the correct address for hardware break point.

7.2.3. Stage 2– Debug SPI in DDR's higher address

1. Set PIC load address as 0x3ff30000 using Debugger Shell command setpicloadaddr 0x3ff30000. (See How to calculate PIC load address, for more information.)

Figure 59. File editor

start.S	B	- 8
1898		*
1899	/*	
1900	* Relocation Function, r12 point to got2+0x8000	
1901	*	
1902	* Adjust got2 pointers, no need to check for 0, this code	
1903	* already puts a few entries in the table.	
1904	*/	
1905	li r0, got2 entries@sectoff@l	
1906	la r3,GOT(GOT2 TABLE)	
1907	lwz r11,GOT(GOT2 TABLE)	
1908	mtctr r0 b	
1909	sub r11,r3,r11	
1910	addi r3,r3,-4	
1911 1 :	lwzu r0,4(r3)	
1912	cmpwi r0,0	
1913	beq- 2f	
1914	add r0,r0,r11	
1915	stw r0,0(r3)	
1916 2:	bdnz 1b	
1917		
1918	/*	
1919	* Now adjust the fixups and the pointers to the fixups	T
4		F

2. Run to Line: board_init_r and do Step Into.

Figure 60. File editor



3. Run to Line: main_loop().



Figure 61. File editor



8. Debugging U-Boot from NOR for e500mc

8.1. Debug environment

Setup used for U-Boot debugging on e500mc core:

- 1. P3041DS Hydra board.
- 2. Compiled U-Boot for the NOR FLASH target.
- 3. Flash U-Boot on the target board. (See SDK documentation, for more information on how to program the U-Boot to NOR flash.)
- 4. Switches set for NOR boot (See SDK documentation, for more information on how to set switches.)
- 5. Latest release of CodeWarrior IDE.
- 6. P3041DS_uboot_36.tcl initialization file.
- 7. USB TAP or other probe.

8.2. U-Boot NOR debugging

U-Boot elf file generated during U-Boot compilation should be imported as CodeWarrior project. (See <u>Configuring a CodeWarrior project</u>, for more information.)

8.2.1. Stage 0 – Connect CodeWarrior to board

1. Before debugging, run the board in debug mode.



Figure 62. Debug configurations dialog



2. Start the debugging session using the setup provided in <u>Figure 62</u>. This initializes and configures the TAP, then attaches to the board.

Figure 63. Debug view



3. Reinitialize the target from CodeWarrior.



Figure 64. Reset dialog



8.2.2. Stage 1 – Debug NOR for AS0

1. Set PIC load address as 0xfff80000 using Debugger Shell command setpicloadaddr 0xfff80000.

Figure 65. Debugger shell view

🏶 Debug 😫			(x)= Variables	tpoints 🖾 🚺 Cache 🛲 Rei	gisters 🛋 Modules	- 0
12. st 📣 👞 m 🗰 st 🖈 🖢		······································			x %	∿ # ⊴ × ⊞ ⊟ 🕏 [×]
			Name		Contex	t Address
	ttach [CodeWarrior] nded) ded: Signal 'Halt' received etvec.Sr2 0xffffffc boot_debug\P3041DS_N4	I. Description: User halted thread.) DR\Debug u-boot (11/27/13 3:38 PM)				
				III /		,
(AsmSection)() at /home/freescale	/SDK/u-boot/arch/power	pc/cpu/mpc85xx/resetvec.S: 2 🛛 🥖 P3041DS_NOF	FLASH	er (Disassembly 🕄	🗄 Outline 👘 🗇
Can't find a source file at "/home/free	scale/SDK/u-boot/arch/p	owerpc/cpu/mpc85xx/resetvec.S"				~
View Disassembly					0x11000000	
Locate File		R			C1 61	
Edit Source Lookup Path	Path				ffffffe0: fnmad ffffffe0: fnmad ffffffe0: fnmad ffffffe0: fnmad fffffff0: fnmad fffffff0: fnmad fffffff0: fnmad fffffff0: fnmad fffffff0: b (As	0. (p., p., p., p., p., p., (d. fp., fp., fp., fp., fp., (d. fp., fp., fp., fp., fp., fp., fp., (d. fp., fp., fp., fp., fp., fp., fp., (d. fp., fp., fp., fp., fp., fp., fp., fp.
A Commander	(🕒 Con 😫 🤕 Task 🚺 Mem 📓 Rem 🛸 Tas	rg 🖹 Prob 💽 Exec	🗝 🗖 🗐 Progress 🔯 Debi	agger Shell	
Project Creation Project Creation Manopart project CodeWarner Barboard Project Build/Debug Colam (All) Chan (All) Chan (All) Chang Project sattings Project sattings Podd sattings	Miscellaneous Welcome screen Quick access Flash programmy	EPPC, u-boot, core 0 🔳 🗶 🙀 🕞		Códekarrior Debug Xisetpicloadadd Debugger row asi	gger Shell v1.0 0⊀fffa0000 mes'u-boct' is loade	d at the specified
71				* address.		

2. After the path is specified, source code is available in CodeWarrior.



Debugging U-Boot from NOR for e500mc

Figure 66. File editor

🕼 resetvec.S 🕱	- 8
1 .section .resetvec,"ax"	*
<pre> 2 b _start_e500 </pre>	
3	
N	
45	
	-
· · · · · · · · · · · · · · · · · · ·	E. F.

3. Now debugging (step, run, or breakpoint) can be done before switching to AS1.

Figure 67. File editor



4. In start.S, last instruction before moving to AS1 is rfi before switch_as. (See <u>Stage 2</u> – <u>Debug NOR for AS1</u>, for more information.)

Figure 68. File editor

S start.S	5 83	- 8
1242	lis r6,MSR_IS MSR_DS MSR_DE@h	*
1243	ori r6,r6,MSR_IS MSR_DS MSR_DE@l	
1244	lis r7,switch_as@h	
1245	ori r7,r7,switch_as@l	
1246		
1247	mtspr SPRN_SRR0,r7	
1248	mtspr SPRN_SRR1,r6	
2 1249	rfi	
1250	er i i i i i i i i i i i i i i i i i i i	
1251 5	witch_as:	
1252 /	* L1 DCache is used for initial RAM */	
1253	The second second second second because	
1254	/* Allocate Initial RAM in data cache.	
1255	*/	
1256	IIS r3, CONFIG_SYS_INIT_RAM_ADDR@h	
1257	OF1 F3,F3,CONFIG_SYS_INIT_RAM_ADDR@I	
1258	mtspr r2, Licrov	
1259	and1. r2, r2, 0x1TT	
1260	/* cache size * 1024 / (2 * LI line size) */	
1261	SIWI 72, 72, (10 - I - LI_CAURE_SHIFT)	
1262	mtctr r2	
1203	11 1.0'0	T
1	e	• •

8.2.3. Stage 2 – Debug NOR for AS1

1. Step Into this instruction.



2. Reset PIC load address using Debugger Shell command setpicloadaddr reset.

Figure 69. Debugger shell view

S start.S			- 0	Disassembly	83 🗄 Outline	
1249 rfi			^		Enter location here	• 2 1 5 5 6 °
1251 switch as: 1252 /* L1 DCache is used for 1253 1254 /* Allocate Initial	r initial RAM */ RAM in data cache.			<pre>effff860: effff864: effff868: effff86c: effff870:</pre>	lis r3,-560 ori r3,r3,0x0000 mfspr r2,spr515 andi. r2,r2,0x01ff slwi r2,r2,3	e and the second s
▶ 1256 11: r3,r3,r0/FIG.5Y3 1257 or1 r3,r3,r0/FIG.5Y1 1258 mfspr r2, LICF00 1259 mdi. r2, r2, 041 1260 /* cache size * 100/ 1261 11: r0,02 1264 11: r0,02 1264 11: r0,0- 1266 dcbz r0,r3 1266 dcbz r0,r3 1266 dcbz r0,r3 1266 dcbz r0,r3 1266 dcbz r0,r3 1267 addi r3,r3,CONFI 1268 bdbn 1b 1268 bdbn 1b	UIT RAM_ADDR@H S_INIT_RAM_ADDR@1 ff + / (2 * L1 line si - 1 - L1_CACHE_SHI S_SYS_CACHELINE_SIZ t 4K page and conti	ie) */ T) E nue to 'normal' start */		effffa74: effffa76: effff876: effff884: effff884: effff884: effff896: effff896: effff896: effff896: effff894: effff884: effff884: effff884:	<pre>mtctr r2 li r0,0 dcbz r0,r3 dcbz r0,r3 addi r3,r3,c4 bdbz (AsmEction)+00. lis r3,r4304 ori r3,r3,000000 addi r3,r3,00000 addi r3,r3,00000 mtlr r3 blr fnmadd. fp31,fp31,fp fnmadd. fn fnmadd. fn fnmadd. fn fnmadd. fn fn fnmadd. fn fn fnmadd. fn fn fn fn fnmadd. fn fn fn fn fn fn fn fn fn fn fn fn fn f</pre>	0xeffff87c (0xeffff87c)) € fp31,fp31 fp31,fp31 fp31,fp33 fp33,fp33 fp33,fp33
A Commander 🕄	<u>⊮ ⊽ ⊓ D</u>	🔄 Consol 🕴 🅢 Tasks 🟮 Memor 🔏 Remot 🐞 Target 🗽 Pro	ble 🗆 🗆	Progress	Debugger Shell	- C
Project Creation Import project GedWanner Barnboard Project Bulld/Debug Bulld/Debug Clean (Al) Debug Forcet:settings Project:settings Debud settings Debud settings	✓ Miscellaneous ④ Welcome screen Quick access J Flash programme	EPPC, u-boot, core 0 💼 🛪 🙀 📴 💽 🕑		CodeWarrior 1 %>setpicload Debugger now address. %>setpicload Debugger now	Debugger Shell v1.0 Mdr 0xff80000 assumes 'u-boot' is addr reset'u-boot' is	loaded at the specified
		4		address.		

- 3. Now debugging (step, run, or breakpoint) can be done until code is relocated in DDR.
 - a) Run to Line: board_init_f and Step Into.

Figure 70. File editor



b) Run to Line: relocate_code and Step Into.

Figure 71. File editor

📓 start.S 🕱	P C
1786 * r5 = length in bytes 1797 * r6 = cachelinesize 1798 */ 1799 .globl relocate_code 1800 relocate code:	*
1801 mr r1,r3 /* Set new stack pointer */	
1802 mr r9,r4 /* Save copy of Init Data pointer */	
1803 mr r10,r5 /* Save copy of Destination Address */	
1804	
1805 GET_GOT	
1806 mr r3,r5 /* Destination Address */	
1807 lis r4,CONFIG_SYS_MONITOR_BASE@h /* Source Address */	
1808 ori r4,r4,CONFIG_SYS_MONITOR_BASE@1	
1809 lwz r5,GOT(init_end)	
1810 sub r5,r5,r4	
1811 li r6,CONFIG_SYS_CACHELINE_SIZE /* Cache Line Size */	
1812	
1813 /*	-
1014 * Eiu GAT naistan:	(12)



c) In start.S, last instruction before relocate to DDR is relocate_code.

Figure 72. File editor



d) Step Into blr, it shows assembly code. (See <u>Stage 3 – Debug in DDR's higher address</u>, for more information.)

8.2.4. Stage 3 – Debug in DDR's higher address

1. Set PIC load address as 0x7ff30000 using Debugger Shell command setpicloadaddr 0x7ff30000. (See <u>How to calculate PIC load address</u>, for more information.)

Figure 73. Debugger shell view



- 2. We can debug until U-Boot is running.
 - a) Run to Line: board_init_r and do Step into.

Figure 74. File editor

ci) -
-

b) Run to Line: main_loop().

Figure 75. File editor



9. Debugging U-Boot from NAND for e500mc

9.1. Debug environment

Setup used for U-Boot NAND debugging on e500mc core:

- 1. P3041DS Hydra board.
- 2. Compiled U-Boot for the NAND FLASH target. (See <u>PBL configuration tool</u>, for more information.)
- 3. Flash U-Boot on the target board. (See SDK documentation, for more information on how to program the U-Boot to NAND flash.)
- 4. Switches set for NAND boot (See SDK documentation, for more information on how to set switches.)
- 5. Latest release of CodeWarrior IDE.
- 6. P3041_uboot_36.tcl initialization file.
- 7. USB TAP or other probe.



9.2. U-Boot NAND debugging

Import the U-Boot elf file generated during U-Boot compilation as a CodeWarrior project. (See <u>Configuring a CodeWarrior project</u>, for more information.)

9.2.1. Stage 0 – Connect CodeWarrior to board

1. Before debugging, run the project in debug mode.

Figure 76. Debug configurations dialog

ıg	2 Debug Configurations						B	
* ∰	Create, manage, and [] an configurations Debug or run an application to a target.							
		E D v Name P3041DS_NAND_Debug_P3041_Attach						
	type filter text	Main 🕪- Argun	Main 🕺 Arguments 🕸 Debugger 💺 Trace and Profile 💱 Source 🌉 Environment 🔲 Common					
	C CodeWarrior C 93041DS, NAND_Debug, 93041_Attach Launch Group Target Communication Framework	Debug session type Choose a predefined © Download	debug session type or custom t Connect Custom	type for maximum flexibility				
			▼ C/C++ application					
		Project:	P3041DS_NAND			Browse		
		Application:	Debug/u-boot	Search Project	Browse	Variables		
_		Build (if required) before launching						
		Connection:	- P3041DS_NAND_D	ebug_P3041_Attach +	Edit	New		
		Execute reset seque	n script(s) a multicore target. Please selec	ct a core, or multiple cores in	the case of SMP:			
	Filter matched 4 of 5 items	Target						
	Filter by Project:	✓ P3041 ✓ e500mc-0						
	3041_PBL P010RDB_NAND_RAMBOOT P1010RDB_NAND_SPL	*	5			Apply	Revert	

2. Start the debugging session using the setup provided in <u>Figure 76</u>. This initializes and configures the TAP, then attaches to the board.

Figure 77. Debug view



3. Reinitialize the target from CodeWarrior, using U-Boot initialization file.



Debugging U-Boot from NAND for e500mc

Figure 78. Reset dialog

Se El	DS_NAND_Debug_P304	1_Attach [CodeW	(arrior]		02.010	
D	Reset	u-boot_debug\P	3041DS_NAND\Debu	g\u-boot (12/2/13 4	1:03 PM)	
	Core reset Target	Core reset	Run out of reset	Initialize target	Initialize target script	Move Up
	P3041					
	e500mc-0				D:\Freescale\CW_PA_v10	Move Down
	e500mc-1	127				Restore Order
	e500mc-2			(m)		
	e500mc-3					
			2 March 198			

9.2.2. Stage 1 – Debug NAND SRAM for AS0

- 1. Reset PIC load address using Debugger Shell command setpicloadaddr reset.
- 2. Source code location is asked by CodeWarrior.

Figure 79. Debug view



3. After the path is specified, sources are available in CodeWarrior.



Debugging U-Boot from NAND for e500mc

Figure 80. File editor

S resetvec.S	- 8
1 .section .resetvec, "ax"	*
Z D_Start_esod	
Þ	

4. Now debugging (step, run, or breakpoint) can be done before switching to AS1.

Figure 81. File editor



5. In file start.S, last instruction before moving to AS1 is rfi before switch_as. (See <u>Stage 2</u> – <u>Debug NAND for AS1</u>, for more information.)

Figure 82. File editor

S start.S	5 83	- 8
795 796 797 798 799 800	lis r6,MSR_IS MSR_D5 MSR_DE@h ori r6,r6,MSR_IS MSR_D5 MSR_DE@l lis r7,switch as@l ori r7,r7,switch_as@l mtspr SPRN_SRR0,r7	
801	mtspr SPRN_SRR1,r6	
802	rfi	
805 / 805 / 806	witch as: * L1 DCache is used for initial RAM */	
807	/* Allocate Initial RAM in data cache.	
808	*/	
809	lis r3,CONFIG_SYS_INIT_RAM_ADDR@h	
810	ori r3,r3,CONFIG_SYS_INIT_RAM_ADDR@1	
811	mfspr r2, L1CFG0	
812	andi. r2, r2, 0x1ff	
813	/* cache size * 1024 / (2 * L1 line size) */	
814	slwi r2, r2, (10 - 1 - L1_CACHE_SHIFT)	
815	mtctr r2	
816	li r0,0	T
1		• •

9.2.3. Stage 2 – Debug NAND for AS1

1. Step Into this instruction.



Debugging U-Boot from NAND for e500mc

Figure 83. File editor

S start.S	5 🛛	- 6
798	ori r7,r7,switch_as@l	*
799		
800	mtspr SPRN_SRR0,r7	
801	mtspr SPRN_SRR1,r6	
802	rfi	
803		
80 51	witch_as:	
805 /*	* L1 DCache is used for initial RAM */	
806		
807	/* Allocate Initial RAM in data cache.	
808	*/	
809	lis r3,CONFIG_SYS_INIT_RAM_ADDR@h	
810	ori r3,r3,CONFIG_SYS_INIT_RAM_ADDR@1	
811	mfspr r2, L1CFG0	
812	andi. r2, r2, 0x1ff	
813	/* cache size * 1024 / (2 * L1 line size) */	
814	slwi r2, r2, (10 - 1 - L1_CACHE_SHIFT)	
815	mtctr r2	
816	li r0,0	
817 1:	· · · · · · · · · · · · · · · · · · ·	
818	dcbz r0,r3	
819	dcbtls 0,r0,r3	+
4		

- 2. Now debugging is be possible, before the code is relocated in DDR.
- a) Run to Line: board_init_f and Step into: board_init_f.

Figure 84. File editor

b) Run to Line: relocate_code and do Step Into.

Figure 85. File editor

c) In file start.S, last code before relocate to DDR is relocate_code.

Debugging U-Boot from NAND for e500mc

Figure 86. File editor

S start.S	S 83	- 8
1447 #	tendif	*
1448	/*	
1449	* Re-point the IVPR at RAM	
1450	*/	
1451	mtspr IVPR,r10	
1452		
1453	mtlr r0	
1454	blr /* NEVER RETURNS! */	
1455	.globl in_ram	
1456 i	n_ram:	
1457	a	
1458	/*	
1459	* Relocation Function, r12 point to got2+0x8000	
1460	*	
1461	* Adjust got2 pointers, no need to check for 0, this code	
1462	* already puts a few entries in the table.	
1463	*/	
1464	li r0,got2_entries@sectoff@l	
1465	la r3,GOT(_GOT2_TABLE_)	
1466	Iwz r11,GOT(_GOT2_TABLE_)	
1467	mtctr r0	
1468	sub r11,r3,r11	
4	4 · · · · · · · · · · · · · · · · · · ·	

3. Step Into: blr, it shows code in assembly. (See <u>Stage 3 – Debug in RAM</u>, for more information.)

9.2.4. Stage 3 – Debug in RAM

 Set PIC load address as 0xFFFFFFF7FFB0020, using Debugger Shell command setpicloadaddr 0xFFFFFFF7FFB0020. (See <u>How to calculate PIC load address</u>, for more information.)

Figure 87. Debugger shell view

S start.S 23				 E Disassembly	83 🗄 Outline		- 0
1457					Enter location here	- 265Q	C 1 1 7
1458 /* 1459 * Relocation Funct 1460 * 1461 * Adjust got2 poin 1462 * already puts a fo	ion, r12 point to go ters, no need to che ew entries in the ta	ot2+0x8000 eck for 0, this code able.		7ff315b4: 7ff315b8: 7ff315bc: 7ff315bc: 7ff315c0: 7ff315c4:	li r0,2761 subi r3,r12,32768 lwz r11,-32768(r1) mtctr r0 subf r11,r11,r3	2)	â
1464 li r0, got2_entri 1465 la r3,GOT(_GOT2_TA) 1466 lwz r11,GOT(_GOT2_TA) 1467 mtctr r0	es@sectoff@l BLE_) ABLE_)			7ff315c8: 7ff315cc: 7ff315d0: 7ff315d4: 7ff315d4:	subi r3,r3,4 lwzu r0,4(r3) cmpwi r0,0x0000 beg 0x7FF315E0 (0: add r0 r0 r11	x7ff315e0) ; 0x7FF315E	:0
1066 sub fl, f, s, r11 1469 addi r3, r3, r4 1470 lwzu r0, 4(r3) 1471 cmpxi r0, 0 1472 beq-2f 1473 add r0, r0, r11 1474 stw r0, 0(r3) 1475 bdm1 lb 1476 add r0, r0, r11 1477 stw r0, 0(r3) 1478 stw r0, 0(r3) 1476 lb 1477 stw r0, 0(r3)	ivens and the points	Re to the flying		7ff315dc: 7ff315dc: 7ff315e0: 7ff315e6: 7ff315e6: 7ff315e6: 7ff315f0: 7ff315f6: 7ff315f6: 7ff31600:	stw r0,0(r3) bdrz 0x7F315CC (l lir r0,1150 lwz r3,-32764(r12) cmpwi r0,0x0000 mtctr r0 subi r3,r3,4 beq 0x7F53161C (0: lwzu r4,4(r3) lwzu r0,r4,r11	hx7ff315cc) ; 0x7FF315) x7ff3161c) ; 0x7FF3161	сс .с
			1		<[•
A Commander 🕄	₩ •	Consol 23 Tasks U Memor	Remot Target Proble	To Progress	Debugger Shell 🔀	H Lin	
Project Creation Import project CodeWarrior Bareboard Project Build/Debug Build (All)	Miscellaneous Welcome screen Quick access Flash programme			CodeWarrior 1 %>setpicload Debugger now address. %>setpicload Debugger now address.	Debugger Shell v1.0 addr reset assumes 'u-boot.elf addr 0x0 assumes 'u-boot.elf	f' is loaded at its li f' is loaded at the sp	nk-time ecified
🐲 Clean (All) 🕸 Debug				%>bp -hw in_ id inst	ram ance address	type enabled? proc	ess
Settings Project settings Build settings Debug settings				#120 start.S, lin %>setpicload Debugger now address.	<pre>#1 v:0xfff81594 e 1464 [u-boot.elf] addr 0xFFFFFFFFFFFF86 assumes 'u-boot.elf</pre>	-hw ENABLED 1020 F' is loaded at the sp	0x0 ecified
5 L III				197			

- 2. Debug until U-Boot code is copied from NAND to RAM and control is transferred to it.
 - a) Run to Line: board_init_r and do Step Into.

Debugging U-Boot from SPI/SD/MMC for e500mc

Figure 88. File editor

🔂 board.c 🛛	
<pre>621/* 622 * This is the next part if the initialization sequence: we are now 623 * running from RAM and have a "normal" C environment, i. e. global 624 * data can be written, BSS has been cleared, the stack size in not 625 * that critical any more, etc. 626 */ 627 void board_init_r(gd_t *id, ulong dest_addr) 628 629 bd_t *bd; 630 ulong malloc start;</pre>	•
631 632 #indef CONFIG SYS NO_FLASH 633 ulong flash_size; 634 #endif 635 636 gd = id; /* initialize RAM version of global data */	
63/ bd = gd->bd; 638 638 639 gd->flags = GD_FLG_RELOC; /* tell others: relocation done */ 640 641 641 /* The Malloc area is immediately below the monitor copy in DRAM */ 642 malloc_start = dest_addr - TOTAL_MALLOC_LEN;	a start

b) Step Into: main_loop() function.

Figure 89. File editor

10. Debugging U-Boot from SPI/SD/MMC for e500mc

Booting from SPI and SD\MMC are similar, the only difference between these is, how the final image is build. This chapter provides steps for SPI U-Boot debugging.

10.1. Debugging environment

Given below is the setup used for U-Boot debugging on e500mc core:

- 1. Compiled U-Boot for SPI FLASH target.
- 2. Flash U-Boot on the target board. (For more information, see Chapter 7.6.1.1 Using the Boot Format Tool, of Targeting PA Processor.pdf)
- 3. Switches set for SPI boot. (See SDK documentation, for more information on how to set switches.)
- 4. Latest release of CodeWarrior IDE.
- 5. P3041_uboot_36.tcl initialization file.
- 6. USB TAP or other probe.

10.2. U-Boot SPI debugging

Import the U-Boot elf file generated during U-Boot compilation as CodeWarrior project. (See <u>Configuring a CodeWarrior project</u>, for more information.)

10.2.1. Stage 0 – Connect CodeWarrior to board

1. Before debugging, run the board in debug mode.

Figure 90. Debug configurations dialog

Debug Configurations						C Di tem	
Create, manage, and run configurations Debug or run an application to a target.							
🖸 🗎 🗶 🗎 🔿 🔸	Name: P3041DS_SPI_D	ebug_P3041_Attach					
type filter text	Main 🕪 Argun	nents) 🕸 Debugger) 💺 Traci	e and Profile by Source	The Environment	Common		
C CodeWarrior P3041DS_SPLDebug_P3041_Attach Launch Group Target Communication Framework	Debug session type O Download O Attach	 Connect Custom 					
	Project:	P3041DS_SPI			Browse		
	[] Application:	Debug/u-boot	Search Project	Browse	Variables		
	Build (if required)	before launching					
		▼ Target settings					
	Connection:	- P3041DS_SPI_Deb	ug_P3041_Attach	Edit	New		
	Execute reset seque Execute initializatio The connection is for	n script(s) a multicore target. P ase sele	ect a core, or multiple cores	in the case of SI	ИР:		
Filter matched 4 of 5 items	Target						
Filter by Project:							
2041_PBL PI010RDB_NAND_RAMBOOT PI010RDB_NAND_SPL	* 500mc-1				Apply	Re	

2. Start the debugging session using the setup provided in <u>Figure 87</u>. This initializes and configures the TAP, then attaches to the board.

Figure 91. Debug view

3. Reinitialize the target from CodeWarrior.

Debugging U-Boot from SPI/SD/MMC for e500mc

Figure 92. Reset dialog

C P	3041DS SPI Debug P3041 A	Attach (CodeWarr	iorl			
3	EPPC, u-boot.elf, core 0					
	D:\workspace\pa\131024	_u-boot_debug\F	3041DS_SPI\Debug\	u-boot.elf (12/4/13 4	4:48 PM)	
ſ	A Reset					
			11			
	Execute a target reset or o	one or more core	resets:			
	Surtem recet			N		
	System reset			43		
	Core reset					
₿ P304	Target	Core reset	Run out of reset	Initialize target	Initialize target script	Move Up
P304 79 80	Target P3041	Core reset	Run out of reset	Initialize target	Initialize target script	Move Up
3 P304 79 80 81	Target ⊿ P3041 €500mc-0	Core reset	Run out of reset	Initialize target	D:\Freescale\CW_PA_v	Move Up Move Down
P304 79 80 81 82 82	Target ▲ P3041 €500mc-0 €500mc-1	Core reset	Run out of reset	Initialize target	Initialize target script D:\Freescale\CW_PA_vi	10 Restore Order
3 P304 79 80 81 82 83 84	Target ▲ P3041 €500mc-0 €500mc-1 €500mc-2	Core reset	Run out of reset	Initialize target	Initialize target script D:\Freescale\CW_PA_v2	10 Move Up Move Down Restore Order
P304 79 80 81 82 83 84	Target P3041 e500mc-0 e500mc-1 e500mc-2 e500mc-3	Core reset	Run out of reset	Initialize target	Initialize target script D:\Freescale\CW_PA_v2	Move Up Move Down Restore Order
P304 79 80 81 82 83 84 85 86	Target P3041 e500mc-0 e500mc-1 e500mc-2 e500mc-3	Core reset	Run out of reset	Initialize target	Initialize target script	Move Up Move Down Restore Orde
P304 79 80 81 82 83 84 85 86 87	Target P3041 e500mc-0 e500mc-1 e500mc-2 e500mc-3	Core reset	Run out of reset	Initialize target	Initialize target script D:\Freescale\CW_PA_vi	Move Up Move Down Restore Orde
P304 79 80 81 82 83 84 85 86 87 88	Target P3041 e500mc-0 e500mc-1 e500mc-2 e500mc-3 Note: Target initialization	Core reset	Run out of reset	Initialize target	Initialize target script D:\Freescale\CW_PA_v/	Move Up Move Down Restore Orde
P304 79 80 81 82 83 84 85 86 87 88 89	Target a P3041 e500mc-0 e500mc-1 e500mc-2 e500mc-3 Note: Target initialization	Core reset	Run out of reset	Initialize target	Initialize target script	10 Move Up Move Down Restore Orde
P304 79 80 81 82 83 84 85 86 87 88 89	Target P3041 e500mc-0 e500mc-1 e500mc-2 e500mc-3 Note: Target initialization Paload actions from the	Core reset	Run out of reset	Initialize target	Initialize target script	Move Up Move Down Restore Ord
P304 79 80 81 82 83 84 85 86 87 88 99 90 91	Target # P3041 e500mc-0 e500mc-1 e500mc-2 e500mc-3 Note: Target initialization Reload settings from the	Core reset	Run out of reset	Initialize target	Initialize target script D:\Freescale\CW_PA_v/	Move Up Move Down Restore Orde

NOTE If Reset Failed error appears go to **Debug Configurations**, edit **Target settings connection**, and then go to **Advanced** tab, select **Reset delay** (**ms**) and set the value to 2000.

10.2.2. Stage 1 – Debug SPI until U-Boot relocated to DDR's higher address

- 1. Reset PIC load address, using Debugger Shell command setpicloadaddr reset.
- 2. Source code location is asked by CodeWarrior.

	💛 🕅 🕬= Variabl	les 🛛 💙 👋 Breakpoi
	▽	
\$\$ #\$ ❸ ▶ 0 ■ # \$\$ \$. \$ 10 £ # \$\$ # \$\$ # # # # \$ + \$\$ \$\$ \$\$ \$	Name	
▶ * 專 插 み		
C P3041DS_SPI_Debug_P3041_Attach [CodeWarrior]		
EPPC, u-boot.elf, core 0 (Suspended) Theory (ID: 0.01) (Suspended)		
Inread [D: 0x0] (suspended: signal mail: received. Description: Oser naited thread.) 1 (AsmSection)() start.5:86.0xffff000		
D:\workspace\pa\131024_u-boot_debug\P3041DS_SPI\Debug\u-boot.elf (12/4/13 5:09 PM)		
could the me of care the source lookup pain to include its location.		
View Disassembly) Locate File Edit Source Lookup Path		
View Disaszembly Locate File Edit Source Lookup Path Apply to Common Source Lookup Path		
View Diassembly Locate File Edit Source Lookup Path Apply to Common Source Lookup Path		
View Disassembly] Locate File] Edit Source Lookup Path] Apply to Common Source Lookup Path		
View Disassembly Locate File Edit Source Lookup Path] Apply to Common Source Lookup Path		
View Disassembly Locate File Edit Source Lookup Path Apply to Common Source Lookup Path		
View Disassembly Locate File Edit Source Lookup Path Apply to Common Source Lookup Path		
View Disassembly Locate File Edit Source Lookup Path Apply to Common Source Lookup Path		

3. After the path is specified, source will be in CodeWarrior.

Figure 94. File editor

🖻 resetvec.S 🕱	-
1 .section .resetvec, "ax"	
2 b_start_e500	
3	

- 4. Now debugging (step, run, or breakpoint) can be done until U-Boot code will be relocated to the higher address of DDR.
 - a) Step Into:

Figure 95. File editor

b) Run to Line: board_ini_f and do Step Into.

Figure 96. File editor

- ----
 - c) Run to Line: relocate_code and do Step Into.

Figure 97. File editor

d) In file start.S, last instruction before moving to DDR's higher address is relocate_code.

Figure 98. File editor

e) Step Into: blr, it shows code in assembly. (See <u>Stage 2 – Debug SPI in DDR's higher</u> address, for more information.)

10.2.3. Stage 2 – Debug SPI in DDR's higher address

 Set PIC load address as 0xffffffffffb0020 using Debugger Shell command setpicloadaddr 0xffffffffb0020. (See <u>How to calculate PIC load address</u>, for more information.)

Debugging U-Boot from SPI/SD/MMC for e500mc

Figure 99. Debugger shell view

2. Run to Line: board_init_r and do Step Into.

Figure 100. File editor

3. Run to Line: main_loop().

Figure 101. File editor

How to calculate PIC load address

11. How to calculate PIC load address

To set the PIC load address, apply this formula:

```
PIC address = Runtime symbol address (RAM symbol address in our case)
- Compile time symbol address
```

After Step Into: *blr*, in Debugger Shell perform these operations:

1. %>setpicloadaddr 0x0: It tells the debugger that the main executables are loaded at 0x0.

NOTE This is not the same as setpicloadaddr reset command, which tells the debugger that the main executables are loaded at the address set in the ELF.

2. %>bp -hw in_ram: It shows the compile time symbol address.

Figure 102. Debugger shell view

3. Calculate the difference between PC address (single step after blr instruction) and compile time symbol address.

Figure 103. Disassembly view

PIC address = 0X7FF315B8 (PC address) - 0x000015B8 (in_ram break point address) = 0x7FF30000.

12. Troubleshooting Tips

This section explains:

- <u>Selecting the correct breakpoint type</u>
- <u>Risky memory maps</u>
- <u>Setting multiple hardware breakpoints</u>
- <u>Skipping U-Boot stages effectively</u>
- <u>Setting correct absolute addresses</u>
- Secure Boot and U-Boot debug

12.1. Selecting the correct breakpoint type

To avoid issues with incorrect interpretation of memory access during the various U-Boot stages, ensure you use hardware breakpoints only when you have successfully reached the first breakpoint in RAM. The debugger tries to do modify the target memory map and breakpoints, but you can avoid risks by sticking to hardware breakpoints during initial bring-up.

12.2. Risky memory maps

Some SoCs do not provide access to invalid memory ranges and get locked due to unfinished transactions. In such cases, the debug session needs to be restarted. When performing early U-Boot, consider the following points:

- Do not open the **Memory** or **Memory Browser** views for ranges that are not actually readable yet and do not leave them open if you know that the next reset will render them inaccessible.
- For U-Boot debug, your debugger init script should be nearly empty, but it should contain at least a reg sp=1 line. This prohibits the debugger in the very early stages from trying to show a stack back trace that causes invalid accesses, if there is no stack yet.

12.3. Setting multiple hardware breakpoints

The number of active hardware breakpoints is limited, but you can use the **Breakpoints** view to disable those that are not relevant right now and then add more. This way you can create a library of breakpoints that persists across project debug cycles. Whenever you need a specific one, you can enable it and disable others to stay within the limits of the available hardware breakpoints. Also remember that the debugger requires a free hardware breakpoint to do specific operations like step over. To avoid error messages, monitor how many hardware breakpoints you have enabled at a specific point of time.

12.4. Skipping U-Boot stages effectively

Remember that setpiclaodaddr automatically relocates all active source related breakpoints to the space where a PIC executable is loaded. This means that you can pick a specific breakpoint from your library of source related hardware breakpoints and use setpiclaodaddr to instantiate it for an appropriate stage of U-Boot debug. For example, if you have determined that U-Boot will relocate to 0x7ff30000 in RAM, run the following sequence:

Troubleshooting Tips

- 1. reset hard
- 2. %>bp -hw in_ram: Assuming this breakpoint is not yet enabled in your **Breakpoints** view.
- 3. %>setpicloadaddr 0x7ff30000: It instantiates the hardware breakpoint at the right address. Check the **Breakpoints** view.
- 4. %>go: It runs through all the various memory map changes and stops on in_ram breakpoint in RAM.

Similarly, you can go straight to board_init_f breakpoint:

- 1. reset hard
- 2. %>bp -hw board_init_f: Assuming that this breakpoint is not yet enabled in your **Breakpoints** view.
- 3. %>setpicloadaddr reset: For a NOR flash setup, board_init_f runs in the address range to which U-Boot was linked to. So, reset is ok.
- 4. %>go: It runs through all the various memory map changes and stops on board_init_f breakpoint in NOR.

12.5. Setting correct absolute addresses

Absolute hex addresses shown in this application note for the setpiclaodaddr command or breakpoint operations are common for Freescale provided setups. For example, a 512KB U-Boot starts in NOR flash at 0xfff80000 and is linked to 0xeff80000. Relocation to RAM is based on RAM sizes. All these perceived absolute values can change depending on the U-Boot size and configuration. So, if your U-Boot configuration differs from the one shown, adjust the addresses used appropriately. Go manually from one debugging stage to another debugging stage during debug, and you will see to what extent addresses may be different for your setup. Then you will know all the required values for subsequent runs.

12.6. Secure Boot and U-Boot debug

When using Secure Boot, remember that ESBC starts at a different virtual address as configured using CSF after ISBC has verified it, and not from 0xffffffc. If you try to debug U-Boot without considering this, the debugger shows you the ESBC code starting at 0xffffffc when it is internally executing an invisible ROM ISBC at those addresses. This shows a discrepancy in the assembly code and execution behavior. If by using CSF you get, for example, 0xcfffffc as ESBC entry vector, then set an initial hardware breakpoint on the ESBC entry and adjust the source mapping with setpiclaodaddr appropriately. Then run from the original reset vector to your breakpoint and skip the invisible ISBC from ROM completely. This procedure can also be entered into lines of a debugger initialization files so that ISBC is automatically skipped when you start debugging ESBC.

How to Reach Us:

Home Page: www.freescale.com

E-mail: support@freescale.com Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: <u>freescale.com/SalesTermsandConditions</u>.

Freescale, the Freescale logo, AltiVec, C-5, CodeTest, CodeWarrior, ColdFire, ColdFire+, C-Ware, Energy Efficient Solutions logo, Kinetis, mobileGT, PowerQUICC, Processor Expert, QorlQ, Qorivva, StarCore, Symphony, and VortiQa are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Airfast, BeeKit, BeeStack, CoreNet, Flexis, Layerscape, MagniV, MXC, Platform in a Package, QorlQ Qonverge, QUICC Engine, Ready Play, SafeAssure, SafeAssure logo, SMARTMOS, Tower, TurboLink, Vybrid, and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2014.

