

MICROCHIP







First Thursday's

- May 2 Webinar 1: Discover Renode for PolarFire[™] SoC Design and Debug
- June 6 Webinar 2: How to Get Started with Renode for PolarFire SoC
- July 4 Webinar 3: Learn to Debug a Bare-Metal PolarFire SoC Application with Renode
- Aug. 1 Webinar 4: Tips and Tricks for Even Easier PolarFire SoC Debug with Renode
- Sept. 5 Webinar 5: Add and Debug PolarFire SoC Peripherals with Renode
- Oct. 3 Webinar 6: Intro to PolarFire SoC MSS Configuration and Software Flow



Supporting Content

Містозеті в Міскосн е сотра	pany Search Microsemi.com
	Ordering Company Partners Support
PGA & SOC	Libero SoC Design Suite v12.0
Antifuse FPGAs FPGA & SoC Design Tools FPGA Applications	• 60% nurtime reduction for Timing and 20% nurtime reduction for Power • 25% nurtime improvement for Place and Route
FPGA Boards, Kits, and Solutions	Home / Products & Services / FPGA & SoC / Mi-V RISC-V Ecosystem
Mi-V RISC-V Ecosystem	Mi-V RISC-V Ecosystem
Design Tools Operating Systems RISC-V CPUs	Overview MI-V Partners Tutorials Renode Webinar Series Articles and News
Rad-Tolerant FPGAs SoC FPGAs Technology Solutions	Getting Started with the RISC-V Based PolarFire ™ SoC FPGA Webinar Series Learn how to get started with the PolarFire SoC FPGA, the world's first RISC-V based SoC FPGA, to create fully deterministic, real-time systems alongside the Linux® operating system. We are holding a series of webinars to introduce you to the free Renode™ development platform from Mi-V
	partner Antmicro that is available with our SoftConsole v6.0 software development environment. You will see demo applications, learn how to create projects, and find out how to set up and configure your own systems targeting the new SoC FPGA architecture.



Webinar 1 (May 2): Discover Renode for PolarFire[™] SoC Design and Debug

In this introductory session, we will provide you with an overview of SoftConsole 6.0 with Renode¹⁰⁴ integration. We will introduce you to the Renode development framework and provide an orview of the platform and its features. You will also learn about the PolarFire¹⁰⁴ SoC architecture and how to use Renode to develop your application.

www.microsemi.com/Mi-V "Renode Webinar Series"



Learn to Debug a Bare-Metal PolarFire SoC Application with Renode

- Configuring debug sessions, external tools and launch groups
- PolarFire SoC on Renode
- PSE-Blinky project
- · Demo



Learn to Debug a Bare-Metal PolarFire SoC Application with Renode



- Renode is an external tool that needs to be running before you try to debug
- Launching a debug session without Renode is like trying to debug with the board off
- Two ways to do this:
 - a) Manually launch Renode followed by GDB
 - b) Use a launch group to start Renode and then GDB

SC External Tools Configurations	
Create, manage, and run configurations	
Run a program	
	Name: PolarFire-SoC-Renode-emulation-platform
v 💁 Program	🖻 Main 🧇 Refresh 🗟 Build 🖾 Environment 🔲 Common
Mi-V-Renode-emulation-platform	Location: \${env_var:RENODE_BINARY_LAUNCHER}
PolarFire-SoC-Renode-emulation-platform	Browse Workspace Browse
	Working Directory:
	\${eclipse_home}/renode/bin
	Browse Workspace Browse
	Arguments:
	\${env_var:RENODE_BINARY_ARGUMENT}/scripts/single-node/po
	Note: Enclose an argument containing spaces using double-quotes
< >	
Filter matched 3 of 3 items	
0	



- Renode launches and a script is passed as an argument to tell it what platform to load
- It can be launched independently or as part of a launch group

Create, manage, and run configurations	Q_
Run a program	2
〕 🕼 🔕 🗎 🗶 📄 🔆 ▼	Name: PolarFire-SoC-Renode-emulation-platform
type filter text	🖻 Main 🤣 Refresh 👦 Build 🖾 Environment 💷 Common
v 🌯 Program	Location:
Mi-V-Renode-emulation-platform PolarFire-SoC-Renode-emulation-platform	\${env_var:RENODE_BINARY_LAUNCHER}
Polarrire-SoC-Renode-emulation-platform	Browse Workspace Browse File System Varjables
	Working Directory:
	\${eclipse_home}/renode/bin
	Browse Workspace Browse File System Variables
	Arguments:
	\${env_var:RENODE_BINARY_ARGUMENT/scripts/single-node/polarfire-soc-multiple-
	· · · · · · · · · · · · · · · · · · ·
	Variable <u>s</u>
	Note: Enclose an argument containing spaces using double-quotes (").
< >	
ilter matched 3 of 3 items	Re <u>v</u> ert Appl <u>y</u>



 External tool in group 	launch	SC Debug Configurations Create, manage, and run configurations Launch several other configurations sequentially			- - -	×
9.046			Name: pse-blinky Start-Renode-emulator-and-attach G Launches Common			
SC External Tools Configurations Create, manage, and run configurations		 © GDB OpenOCD Debugging © pse-blinky hart0 Attach-to-Renode © pse-blinky hart1 Attach-to-Renode © pse-blinky hart1 Attach-to-running-Renode ✓ 🔹 Launch Group 	🔽 💁 Program::PolarFire-SoC-Renode-emulation-p Run	Action Wait for output matc Wait until terminated	Up Down Edit	
Run a program	Repose-blinky Start-Renode-emulator-and-attach			Add Remove	à	
\$0	orking <u>D</u> irectory: (eclipse_home}/renode/bin guments:	Filter matched 6 of 12 items	<	Revert Debug	Apply	
\$((env_var:RENODE_BINARY_AR)					



Debug configuration for a hardware target	GDB startup configuration for a hardware target	SoftConsole Debug Session
Starts openOCD (1)	Symbols and executable loaded from ELF	GDB (3) OpenOCD (1)
OpenOCD acts as GDB server (2)	Breakpoint set on main	server (2)
GDB starts (3) and connects (4)	Continue selected	Target (4)



Debug configuration for a Renode target	GDB startup configuration for a Renode target	SoftConsole	Debug Session
Starts Renode (1)	Initial reset selected	GDB (3)]
Renode acts as GDB server (2)	Symbols and executable loaded from ELF		
GDB starts (3) and connects (4)	Breakpoint set on main		GDB Renode (1)
	Continue selected		Target (4)



Debug configuration for a hardware target

- 1. Starts openOCD
- 2. openOCD creates a port for GDB
- 3. GDB starts and connects

C Debug Configurations	
° 🖉 🖗 🎘 🖻 🗯 ▾	Name: miv-rv32im-systick-blinky Debug
type filter text	Debugger I Startup Source III foremation in the provide of the p
	 Allocate console for OpenOCD Allocate console for the telnet connection GDB Client Setup Start GDB session Executable name: \$(cross_prefix)gdb\$(cross_suffix) Browse
	Other options: Commands: set \$target_r/iscv = 1 set mem inaccessible-by-default off
	Remote Target Host name or IP address localhost Port number: 3333
	Force thread list update on suspend
< >> ilter matched 5 of 11 items	Reyert Apply
0	<u>D</u> ebug Close



Debug configuration for a <u>Renode target</u>

- Assumes Renode is running and has created a port for GDB
- 1. openOCD not needed
- 2. GDB starts and connects

1 🛙 🕫 🖹 🗮 🛪 🗖 🔿 🕶				
		ystick-blinky Attach-to-Renode		
type filter text	🗎 Main 🅸 Debugge	er 🕨 Startup 🦆 Source 🔲 Common 🔀 SVD Path		
 C GDB OpenOCD Debugging C miv-rv32im-systick-blinky Attach-to-Renode C miv-rv32im-systick-blinky Debug 	OpenOCD Setup	locally		
 Kit Kozini Systek binky bebug Kaunch Group 	Executable path:	{openocd_path}/{{openocd_executable}	Browse	Variables
miv-rv32im-systick-blinky Start-Renode-emulator-and	Actual executable:	C:\Microsemi\SoftConsole_v6.0\eclipse\//openocd/bin/openocd.exe		
		(to change it use the global or workspace preferences pages	or the <u>project</u> properties pa	ge)
	GDB port:	3333		
	Telnet port:	4444		
	Tcl port:	6666		
	Config options:	file board/microsemi-riscv.cfg		~
	Allocate consol	e for OpenOCD Allocate console	for the telnet connection	Ý
	GDB Client Setup	2		
	Start GDB sessio			
	Start ODD sessio			
		{cross_prefix}gdb{cross_suffix}	Browse	Variables
	Executable name:	\${cross_prefix}gdb\${cross_suffix} riscv64-unknown-elf-gdb	Browse	Variables
	Executable name:		Browse	Variables
	Executable name: Actual executable:		Browse	Variables
	Executable name: Actual executable: Other options:	riscv64-unknown-elf-gdb set \$target_riscv = 1 set mem inaccessible-by-default off	Browse	Variables
	Executable name: Actual executable: Other options: Commands:	riscv64-unknown-elf-gdb set \$target_riscv = 1	Browse	Variables
	Executable name: Actual executable: Other options: Commands: Remote Target	riscv64-unknown-elf-gdb set Starget_riscv = 1 set mem inaccessible-by-default off set arch riscvrv32	Browse	Variables
	Executable name: Actual executable: Other options: Commands: Remote Target Host name or IP ac	riscv64-unknown-elf-gdb set \$target_riscv = 1 set mem inaccessible-by-default off set arch riscvrv32 idress: localhost	Browse	Variables
	Executable name: Actual executable: Other options: Commands: Remote Target	riscv64-unknown-elf-gdb set Starget_riscv = 1 set mem inaccessible-by-default off set arch riscvrv32	Browse	Variables
	Executable name: Actual executable: Other options: Commands: Remote Target Host name or IP ac	riscv64-unknown-elf-gdb set \$target_riscv = 1 set men inaccessible-by-default off set arch riscvrv32 idress: [ocalhost 3333	Browse	<u></u>
<	Executable name: Actual executable: Other options: Commands: Remote Target Host name or IP ac Port number:	riscv64-unknown-elf-gdb set \$target_riscv = 1 set men inaccessible-by-default off set arch riscvrv32 idress: [ocalhost 3333	Browse	Variables



Startup configuration for a hardware target

- 1. Symbols loaded from ELF
- 2. Executable loaded from ELF
- 3. Breakpoint set at main
- 4. Continue selected

ame: miv-rv32im-systick-blinky Debug Main Debugger Startup Source Common SSVD Path Initialization Commands Initial Reset: Type: Init Load Symbols and Executable Load symbols SUSe project blinay: miv-rv32im-systick-blinky.elf		~
Initialization Commands Initialization Initialization Commands Initialization		
Initial Reset: Type: Init Enable ARM semihosting Load Symbols and Executable Use project binary: miv-rv32lm-systick-blinky.elf		
Enable ARM semihosting Load Symbols and Executable Load symbols Use project binary: miv-rv32im-systick-blinky.elf		
Load Symbols and Executable Load symbols (1) Use project binary: miv-rv32im-systick-blinky.elf		
Load Symbols and Executable Load symbols (1) Use project binary: miv-rv32im-systick-blinky.elf		~
Load Symbols and Executable Load symbols (1) Use project binary: miv-rv32im-systick-blinky.elf		
Load Symbols and Executable Load symbols (1) Use project binary: miv-rv32im-systick-blinky.elf		
 ✓ Load symbols ⑥ Use project binary: miv-rv32im-systick-blinky.elf 		
Use project binary: miv-rv32im-systick-blinky.elf		
O Use file:	Workspace	File System
	workspace	The System
Symbols offset (hex):		
—		
	14/	File System
	workspace	rife system
Executable offset (hex):		
Runtime Options		
Debug in RAM		
Run/Restart Commands		
Pre-run/Restart reset Type: (always executed at Restart)		
		^
		~
Set program counter at (hev):		
Continue		
1	Debug in RAM Run/Restart Commands	Use project binary: mix-rv32im-systick-binky.elf Use file: Executable offset (hex): Executable offset (hex): Debug in RAM Run/Restart Commands Pre-run/Restart reset Type: (always executed at Restart) Set program counter at (hex): Set breakpoint at: main



Startup configuration for a Renode target

- 1. Does an initial reset
- 2. Symbols loaded from ELF
- 3. Executable loaded from ELF
- 4. Breakpoint set at main
- 5. Continue selected

· 🖻 💫 🗎 🗮 🕈 🗸	Name: miv-rv32im-systick-blinky Attach-to-Renode	
pe filter text GDB OpenOCD Debugging C miv-rv32im-systick-blinky Attach-to-Renode miv-rv32im-systick-blinky Debug Lanch Group miv-rv32im-systick-blinky Start-Renode-emulator-anc	Main * Debugger Startup Source Common SSVD Path Initialization Commands Initial Reset. Type: Init	~
	Enable ARM semihosting	~
	Load Symbols and Executable Load symbols Suse project binary: miv-rv32im-systick-blinky.elf Use file: Symbols offset (hex): Symbols	
	Executable offset (hex): Runtime Options	
	Debug in RAM Run/Restart Commands Pre-run/Restart reset Type: (aiways executed at Restart)	
	Set program counter at (hex): Set breakpoint at: Continue ■	*
x > x > x > x > x > x > x > x > x > x	5	Revert Apply



Debug configuration for a hardware target	Debug configuration for a Renode target		Startup configuration for a hardware target	Startup configuration for a Renode target
Start openOCD	Start Renode			Initial reset selected
OpenOCD acts as GDB server			Symbols and executable loaded from	Symbols and executable loaded from
GDB starts and			ELF	ELF
connects			Breakpoint set on main	Breakpoint set on main
			Continue selected	Continue selected



SC Debug Configurations					×
Create, manage, and run configurations Launch several other configurations sequentially					Ť.
Image: Second state Image: Second state ✓ Image: Second state ✓	Name: pse-blinky Start-Renode-emulator-and-attach	Mode	Action	Up	
 pse-blinky hart1 Attach-to-Renode pse-blinky hart1 Attach-to-running-Renode se-blinky hart1 Attach-to-running-Renode pse-blinky Start-Renode-emulator-and-attach 			Wait for output matc Wait until terminated	Dowr Edit Add Remov	
	<		>		
Filter matched 6 of 12 items	L		Revert	Appl	У
?			Debug	Clo	se



- Multiple tools can be launched as part of a launch group
- Flow to use a launch group
 - 1. Start Renode external tool
 - Wait for expression in console "Renode\ has\ been\ started\ successfully\ and\ is\ ready\ for\ a\ gdb\ connection"
 - 3. Start GDB debug session

SC Debug Configurations			-		
reate, manage, and run configurations Launch several other configurations sequentially			Ŕ		
〕 2 ◎ ■ ¥ ⊟ ≱ ▼	Name: pse-blinky Start-Renode-emulator-and-attach				
 CDB OpenOCD Debugging pse-blinky hart0 Attach-to-Renode pse-blinky hart1 Attach-to-Renode pse-blinky hart1 Attach-to-running-Renode Launch Group pse-blinky Start-Renode-emulator-and-attach 	ame Mode ✓ ۹. Program::PolarFire-SoC-Renode-emulation-p Run GDB OpenOCD Debugging::pse-blinky hart0. Inherit	Action 2 Wait for output matc Wait until terminated	Up Down Edit Add Remove		
	٢	>			
ilter matched 6 of 12 items		Revert	Apply		
0		Debug	Close		



Adding a Renode platform to a launch group

- 1. Add the Renode platform to launch as a program
- 2. Launch mode: "Run"
- 3. Post launch action: "Wait for console output (regexp)"
- 4. Regular Expression:

"Renode\ has\ been\ started\ successfully\ and\ is\ ready\ for\ a\ gdb\ connection"

SC Edit Launch Con	figuration	
Edit an existing	entry in the launch group	
type filter text		
> GDB OpenO	CD Debugging	
v 🌯 Program	4	
	ode-emulation-platform	
隆 PolarFire-	SoC-Renode-emulation-platform	
L	0	
Launch mode:	Run 2	
Post launch action:	Wait for console output (regexp) 🤦 Regular Ex	pression: Renode\ has\ been\ started\ successfully\ a
	Adopt launch if already running	
(?)		
0		



Adding a debug configuration to a launch group

- 1. Select the debug session to be run
- 2. Launch mode: "Inherit"
- 3. Post launch action: "Wait until terminated"

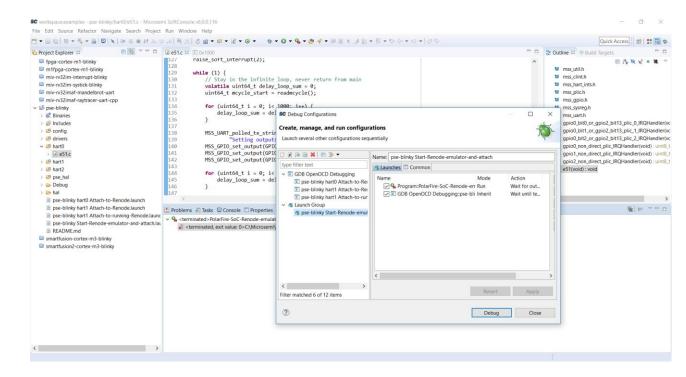
SC Edit Launch Configuration	×
Edit an existing entry in the launch group	
 C GDB OpenOCD Debugging miv-rv32im-systick-blinky Attach-to-Renode 	
miv-rv32im-systick-blinky Debug	
> 💁 Program	
Launch mode: Inherit <u>2</u> Post launch action: Wait until terminated <u>3</u>	
Post launch action: Wait until terminated V	
Adopt launch if already running	
?	OK Cancel



SC Debug Configurations		– 🗆 X	SC External Tools Configurations
Create, manage, and run configurations Launch several other configurations sequentially			Create, manage, and run configurations Run a program
 GDB OpenOCD Debugging pse-blinky hart0 Attach-to-Renode pse-blinky hart1 Attach-to-Renode pse-blinky hart1 Attach-to-running-Renode Launch Group pse-blinky Start-Renode-emulator-and-attach 	Name: pse-blinky Start-Renode-emulator-and-attach Launches Common Name Mode Separation Mode GDB OpenOCD Debugging::pse-blinky hart0. Inherit Inherit	Action Up Wait for output mate Wait until terminated Edit Add Remove	Image: Second state st
Filter matched 6 of 12 items		Revert Apply	Filter matched 3 of 3 items
0		Debug Close	?



 All Mi-V example projects have an included preconfigured launch group to use with Renode





Learn to Debug a Bare-Metal PolarFire SoC Application with Renode



Deterministic, Coherent CPU Cluster RV64IMAC Monitor Core: E51 Performance / Event RV64GC Quad Core: U54 Counters ٠ 128K Boot Flash RISC-V RISC-V Quad core: 4 x harts . Instruction Trace **RV64IMAC** RV64GC AXI Bus Monitors Secure Boot Monitor Core Quad Core U54 1, U54 2, U54 3, U54 4 50 Break Points • Fabric Logic Monitor SmartDebug Debug Locks **Deterministic L2 Memory Subsystem** Coherent Switch Memory mapped peripherals ٠ and storage DDR4/LPDDR4 Controller AMBA Switch with Memory Protection and QoS GPIO • O UART • DDRIO PHY Low Power CAN • **PolarFire[™] FPGA** MAC • Architecture DDR •



- Renode Console
 - Enter commands here
 - See connected peripherals / memory map
- UART Analyzer
 - Shows UART output for a given instance

Renode – – × SC workspace.examples				ace.examples - Microsemi SoftCon	sole v6.0.0.116		
RENODE		~		Source Refactor Navigate Sea) 誦 ! 🖳 ! 🍬 🕪 🕕 🔳 💐 🔍 🤋		dow Help ☆ ▼ O ▼	9a - 🍅 🛷
enode, version 1.6.0.30082 (3b6a18a4-201811221641)							
<pre>wonitor) i \$CWD//scripts/single-node/polarfire-soc-multiple-servers.resc machine-0) peripherals ailable peripherals:</pre>					¥ i→ マ □ □ ation-platform [Program] e_v6.0\renode\bin\Renode.exe	🕬 Varia ९७ Break প্র	
sysbus (SystemBus)							
 			€ e51.c ⊠ 142	SS_GPI0_set_output(GPI01_LO, MSS_GPI0	2, 1); ^	BE Outline mss
 			143 144 145	<pre>for (uint64_t i = 0;</pre>	i< 1000; i++) { delav loop sum + :	i:	mss mss mss
 ├─ clint (CoreLevelInterruptor) 	 machine 	-0:sysbus.mmuart0				- 🗆	× mss
 ├─ ddr (MappedMemory) 							gpi gpi
 ├- e51 (RiscV64) Slot: 0							gpi gpi gpi
 envmCfg (PSE_eNVM) <8x282888880, 8x282881FF>							gpi e51
- envmData (MappedMemory) <0x20220000, 0x2023FFF>							
— gpio0 (PSE_GPIO) <0x20120000, 0x20120FFF>							34
⊢ button0 (Button) │ │ button1 (Button)							n\Re nod
button1 (Button) button2 (Button)							is
⊢ gpio1 (PSE_GPIO) ≪8x20121000, 8x20121FFF>							
 			<				



monitor

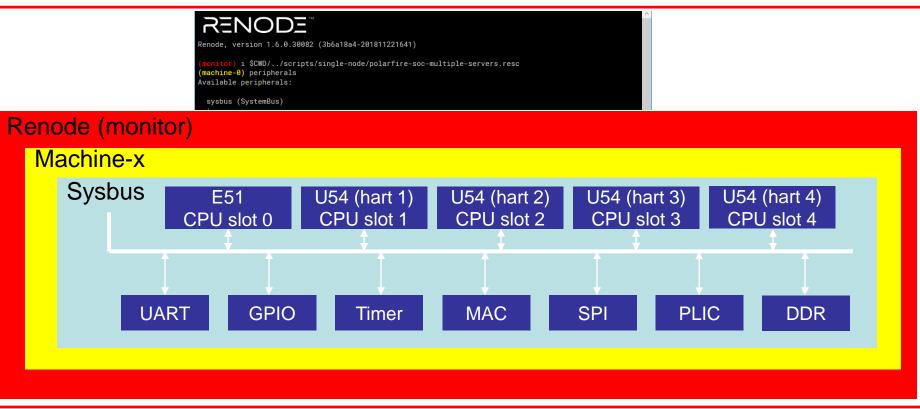
• Initial Renode launch with no system loaded

machine-0

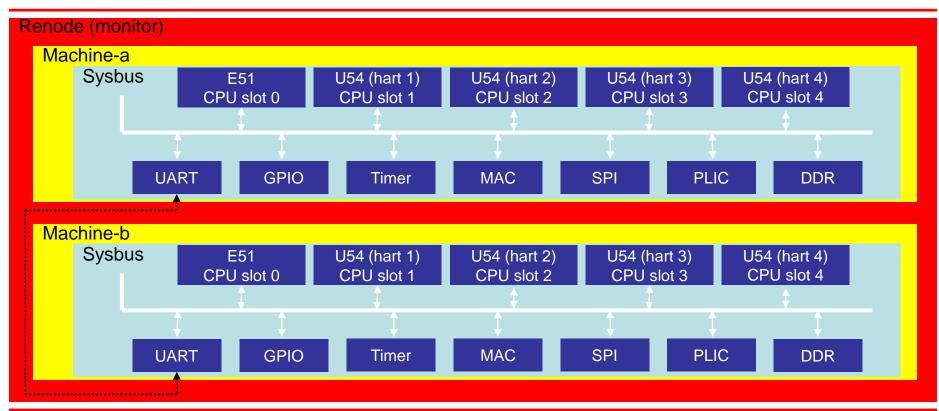
- PolarFire SoC instance
- · Each machine has a sysbus (system bus)
- CPUs and peripherals are connected to sysbus
- Sysbus
 - E51 and U54 harts are connected to sysbus
 - Memory mapped peripherals are connected to sysbus

Renode RENODE Renode, version 1.6.0.30082 (3b6a18a4-201811221641) i \$CWD/../scripts/single-node/polarfire-soc-multiple-servers.resc machine-0) peripherals Available peripherals: sysbus (SystemBus) — can0 (PSE_CAN) <0x2010C000, 0x2010CFFF> — can1 (PSE_CAN) <0x2010D000, 0x2010DFFF> — clint (CoreLevelInterruptor) <0x02000000. 0x0200FFFF> ddr (MappedMemory) <0x80000000. 0xBFFFFFFF> - e51 (RiscV64) Slot: 0 envmCfg (PSE_eNVM) <0x20200000. 0x202001FF> envmData (MappedMemory) <0x20220000. 0x2023FFFF>



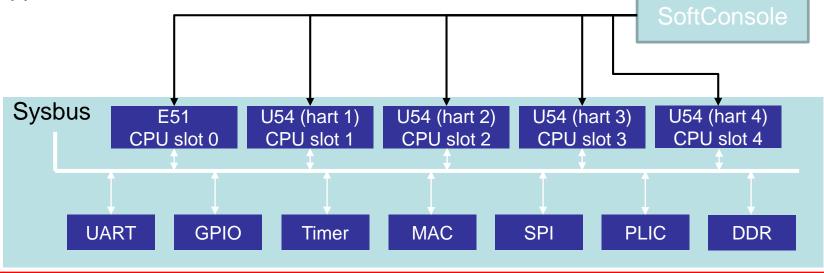








- If one hart is being debugged and is suspended all harts are suspended
- I.E reaching a breakpoint and halting on the E51 will also cause the U54 harts to appear halted





Learn to Debug a Bare-Metal PolarFire SoC Application with Renode

PSE-BLINKY



PSE-BLINKY

- Project configured to run on the PolarFire SoC Renode Emulation Platform
- Can debug a single hart or multiple
- Configured
 - Hart 0: E51
 - Hart 1: U54_1

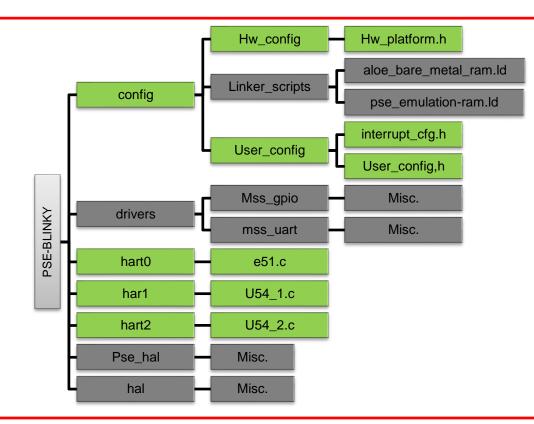
SC workspace.examples - pse-blinky/hart0/e51.c - Microsemi SoftConsole v6.0.0.116

<u>File Edit Source Refactor Navigate Search Project Run Window Help</u>

📅 두 📓 🐘 👻 🔻 💊 두 🐘 💷 🔍 🕨 🗉 🖬 🕫 그 요. (특 곳) (총 🎕 수 없 수 없 수 없 수 없 수 있 수 🍫 수 🌭 🦉 수 🚱 🖉 🧃 🚺 수 🕅 수 🕅 수 🖓 수 🖓 수 👘 🖪 🔩 🔻 🗖 🍐 Project Explorer 🔅 ie e51.c ⊠ i u54_1.c ⊠ i u54_2.c ∞ fpga-cortex-m1-blinky m1fpga-cortex-m1-blinky 2 * (c) Copyright 2018 Microsemi-PR 2 * (c) Copyright 2018 Microsemi-PRC 2 * (c) Copyright 2018 Micros miv-rv32im-interrupt-blinky miv-rv32im-systick-blinky 4 * Code running on E51 hart 0 4 * Code running on U54 hart 1 4 * code running on U54 secon ******** miv-rv32imaf-mandelbrot-uart 5 5 6 * SVN \$Revision: 9661 \$ 6 * SVN \$Revision: 10516 \$ miv-rv32imaf-ravtracer-uart-cpp #include "mss util.h" * SVN \$Date: 2018-01-15 10:43:33 + * SVN \$Date: 2018-11-08 18 ✓ ⅔ pse-blinky 8 #include "mss clint.h" 8 8 Includes 9 #include "mss hart ints.h" > 🤒 config 10 #include "mss plic.h" 10 #include "mss util.h" 10 #include "mss util.h" Description of the second s 11 #include "mss_gpio.h" 11 #include "mss clint.h" \$12 volatile uint32 t count_sw_i ✓ 29 hart0 12 #include "mss sysreg.h" 12 #include "mss uart.h" e51.c 13 #include "mss_uart.h" 14 314 volatile uint32 t count sw ints h1 14^e void u54 2(void) { ✓ ₩ hart1 3 15e uint8 t gpio0_bit0_or_gpio2_bit13 \$415 volatile uint32 t loop_count_h1 = 0 ₹15 uint32 t hartid = read c ⇒ 🔏 u54_1.c **X** 16 MSS UART polled tx string(&g m \$\$16 volatile uint64 t dummy h1 = 0; \$16 volatile uint32 t i = 0; ✓ [™] hart2 17 "\r\nSetting output 0 > li u54 2.c 18 18 18 while (1) { > 🥴 pse_hal **X** 19 19@ void Software_h1_IRQHandler(void) { /* add code here */ MSS GPIO set output(GPIO1 LO, 🗸 🗁 hal uint32 t hart id = read csr(mha i++: /* added some c **2**0 MSS GPIO clear irg(GPIO0 LO, M \$20 20 **X** 21 return EXT IRO KEEP ENABLED: 21 if (hart id == 1) { 21 if (i == 0x1000) { cpu_types.h 22 } 22 count sw ints h1++; 22 i = 0; hal_assert.h 23 23 23 la hal irg.c 24 } ≥ 24⊕ uint8 t gpio0 bit1 or gpio2 bit13 24 🖹 hal.h **a** 25 MSS UART polled tx string(&g m 25 25 hw macros.h 26 '\r\nSetting output 1 26 26 hw reg access.h 27 27@ void u54_1(void) { **X** 28 MSS GPIO set output(GPIO1 LO, 28 volatile int i; hw reg access.S **\$**29 pse-blinky hart0 Attach-to-Renode.launch **X** 29 MSS GPIO clear irg(GPIO0 LO, M uint64 t mcycle start = 0; **3**0 return EXT IRO KEEP ENABLED; 2530 uint64 t mcvcle end = 0; pse-blinky hart1 Attach-to-Renode.launch 31 } **\$**31 uint64 t delta_mcycle = 0; pse-blinky hart1 Attach-to-running-Renode.launc 32 \$32 uint32 t num loops = 100000; pse-blinky Start-Renode-emulator-and-attach.lau > < README.md smartfusion-cortex-m3-blinkv 🖹 Problems 🧟 Tasks 📮 Console 🔲 Properties 🎋 Debug 🛛 smartfusion2-cortex-m3-blinky <terminated, exit value: 0>C:\Microsemi\SoftConsole v6.0\renode\bin\Renode.exe Writable Smart Insert 78:1



PSE-BLINKY





PSE-BLINKY

- Build the project
- Run the "pse-blinky Start-Renode-emulator-and-attach" launch group
 - Starts Renode followed by a GDB debug session that connects to Renode
 - Loads ELF
 - Boots to the E51 hart and sets a breakpoint on the E51 main
 - Allow the program to raise two soft interrupts
 - These start U54 hart 0 and hart 1
 - Run the E51 program and allow it to loop
- Run the "pse-blinky hart1 Attach-to-Renode" debug configuration
 - This connects to the U54 hart and resets its program counter
 - Sets a breakpoint on U54 main



Debug a Bare-Metal PolarFire SoC Application with Renode

• Demo



First Thursday's

- May 2 Webinar 1: Discover Renode for PolarFire[™] SoC Design and Debug
- June 6 Webinar 2: How to Get Started with Renode for PolarFire SoC
- July 4 Webinar 3: Learn to Debug a Bare-Metal PolarFire SoC Application with Renode
- Aug. 1 Webinar 4: Tips and Tricks for Even Easier PolarFire SoC Debug with Renode
- Sept. 5 Webinar 5: Add and Debug PolarFire SoC Peripherals with Renode
- Oct. 3 Webinar 6: Intro to PolarFire SoC MSS Configuration and Software Flow



Thank You