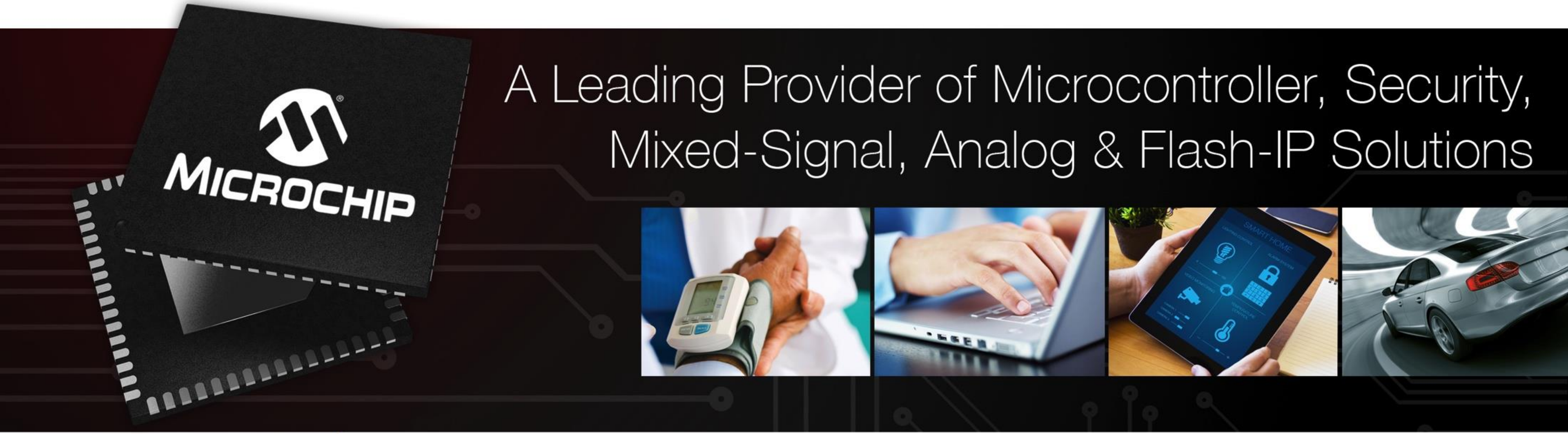
MICROCHIP


MICROCHIP
A Leading Provider of Microcontroller, Security,
Mixed-Signal, Analog & Flash-IP Solutions

Getting Started with the RISC-V Based PolarFire® SoC FPGA Webinar Series

Session 11: Handling Binaries

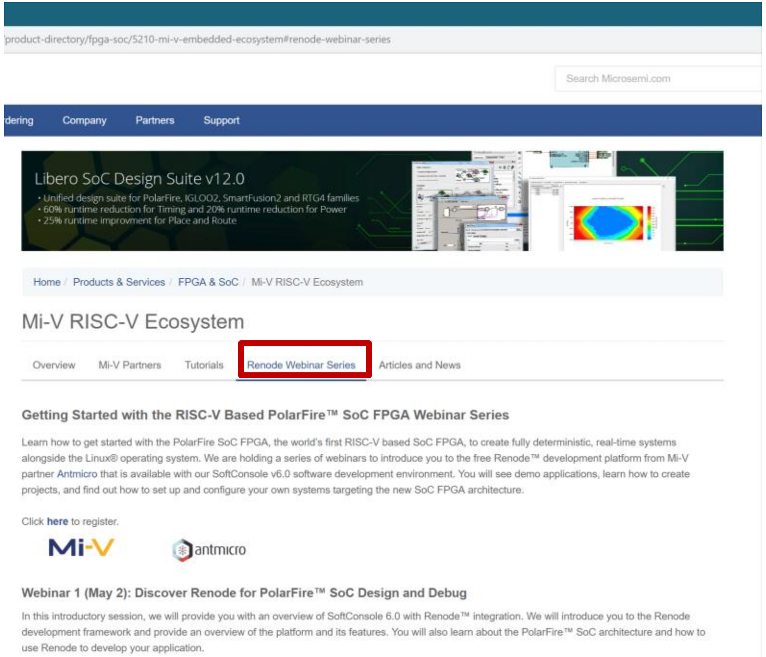
Hugh Breslin, Design Engineer
Thursday Mar. 12, 2020

Second Thursdays

Mar. 12	**-**	**Webinar 11: Handling Binaries**
April 9	**-**	**Webinar 12: Two Bare-Metal Applications on PolarFire® SoC**
May 14	**-**	**Webinar 13: Linux® on Renode**
June 11	**-**	**Webinar 14: Building Applications for Linux on PolarFire SoC**
July 9	**-**	**Webinar 15: Real-Time (AMP Mode) on PolarFire SoC**

Supporting Content

www.microsemi.com/Mi-V “Renode Webinar Series”



'product-directory/fpga-soc/5210-mi-v-embedded-ecosystem#renode-webinar-series
Search Microsemi.com
rdering
Company
Partners
Support
Home / Products & Services / FPGA & SoC / Mi-V RISC-V Ecosystem
Overview
Mi-V Partners
Tutorials
Renode Webinar Series
Articles and News
Mi-V
antmicro

Libero SoC Design Suite v12.0

- Unified design suite for PolarFire, IGLOO2, SmartFusion2 and RTG4 families
- 60% runtime reduction for Timing and 20% runtime reduction for Power
- 25% runtime improvment for Place and Route

Mi-V RISC-V Ecosystem

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.

Click **here** to register.

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 Renode™ integration. We will introduce you to the Renode development framework and provide an overview of the platform and its features. You will also learn about the PolarFire™ SoC architecture and how to use Renode to develop your application.

Webinar 1: Discover Renode for PolarFire® SoC Design and Debug

Webinar 2: How to Get Started with Renode for PolarFire SoC

Webinar 3: Learn to Debug a Bare-Metal PolarFire SoC Application with Renode

Webinar 4: Tips and Tricks for Even Easier PolarFire SoC Debug with Renode

Webinar 5: Add and Debug PolarFire SoC Models with Renode

Webinar 6: Add and Debug Pre-Existing Model in PolarFire SoC

Webinar 7: How to Write Custom Models

Webinar 8: What’s New in SoftConsole v6.2

Webinar 9: Getting Started with PolarFire SoC

Webinar 10: Introduction to the PolarFire SoC Bare-Metal Library

Agenda

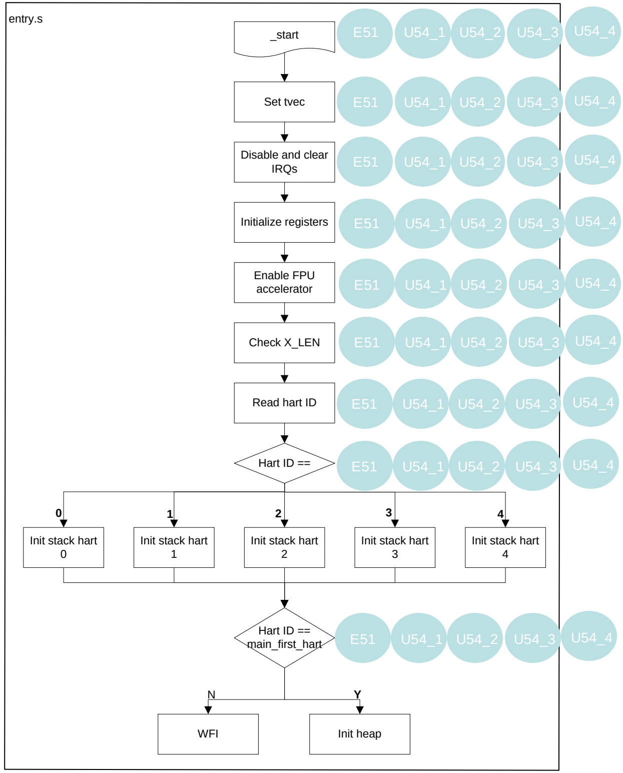
- The PolarFire® SoC Start Up Sequence
- Setting Breakpoints on Specific Harts
- Demo PolarFire ® SoC Start Up Sequence
- MPFS_HAL_FIRST_HART
- MPFS_HAL_LAST_HART

The PolarFire® SoC Start Up Sequence

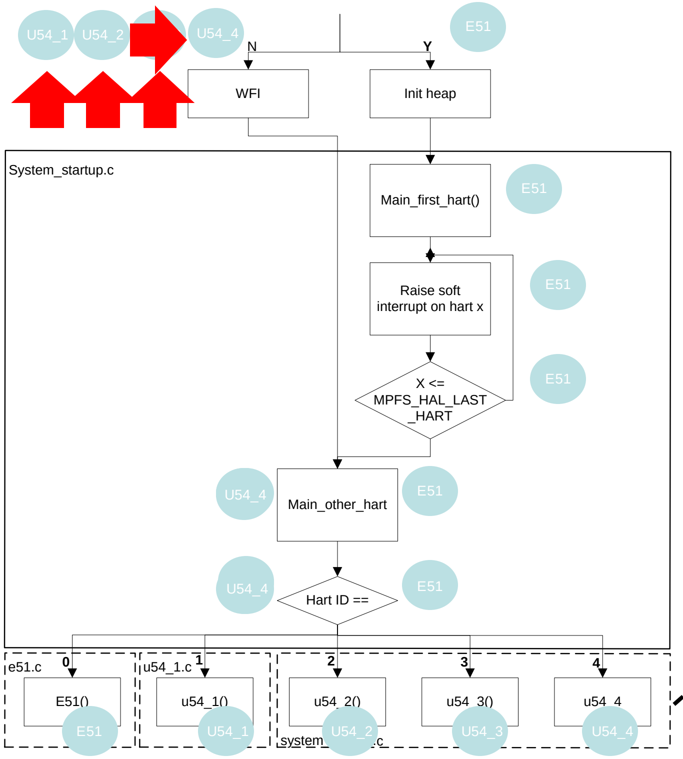
The PolarFire® SoC Start Up Sequence

1. **Harts start from the same “_start” function**
2. **They diverge after initializing their stacks**
   - By default the U54 harts go to WFI
   - E51 initializes the heap and wakes up harts in use
3. **Harts jump to dedicated main functions**

The PolarFire® SoC Start Up Sequence



The PolarFire® SoC Start Up Sequence

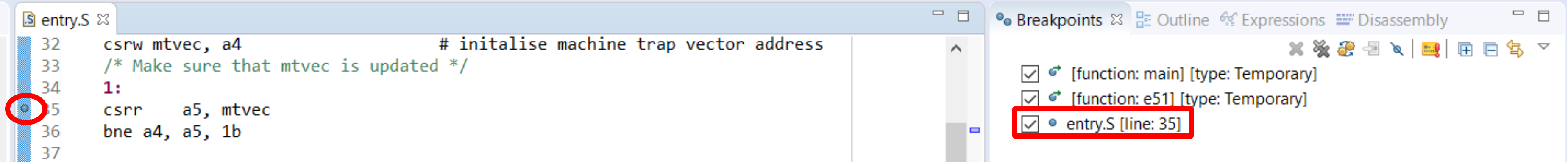


This is the flow used for MPFS Blinky. This can differ between applications

Setting Breakpoints on Specific Harts

Setting Breakpoints on Specific Harts

- Breakpoints can be set for all threads by double clicking on the line the breakpoint is needed



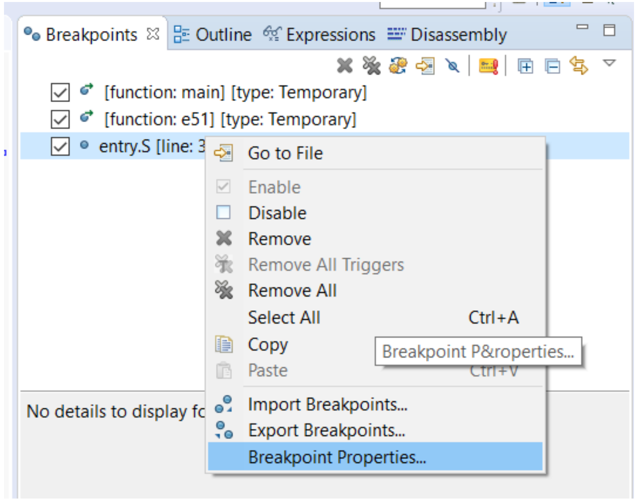
```
    csrw mtvec, a4                  # initalise machine trap vector address
    /* Make sure that mtvec is updated */
    1:
    csrr    a5, mtvec
    bne a4, a5, 1b
```

☑ [function: main] [type: Temporary]
☑ [function: e51] [type: Temporary]
☑ entry.S [line: 35]

- They can also be filtered so that only specific harts halt at the breakpoint

Setting Breakpoints on Specific Harts

- <u>**While**</u> **debugging right click on the breakpoint**
- **Select "Breakpoint Properties…"**



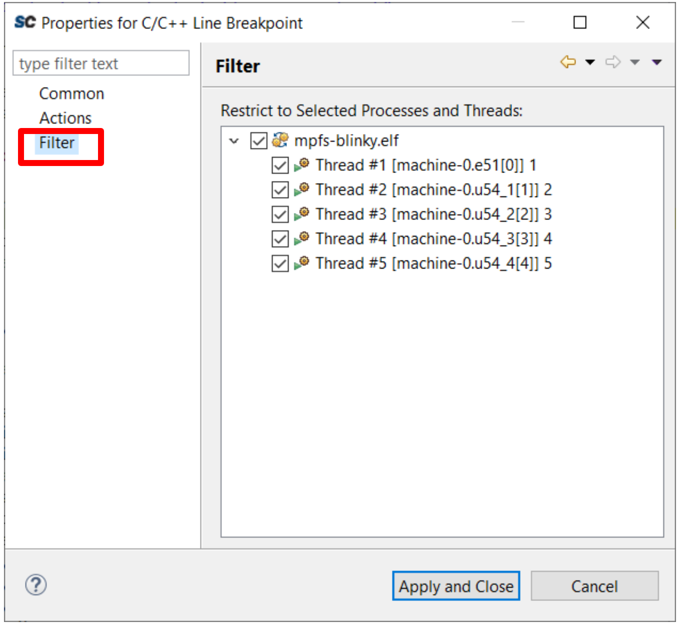
☑ [function: main] [type: Temporary]
☑ [function: e51] [type: Temporary]
☑ entry.S [line: 3

No details to display fc

- **Note: breakpoint properties will reset between debug sessions**

Setting Breakpoints on Specific Harts

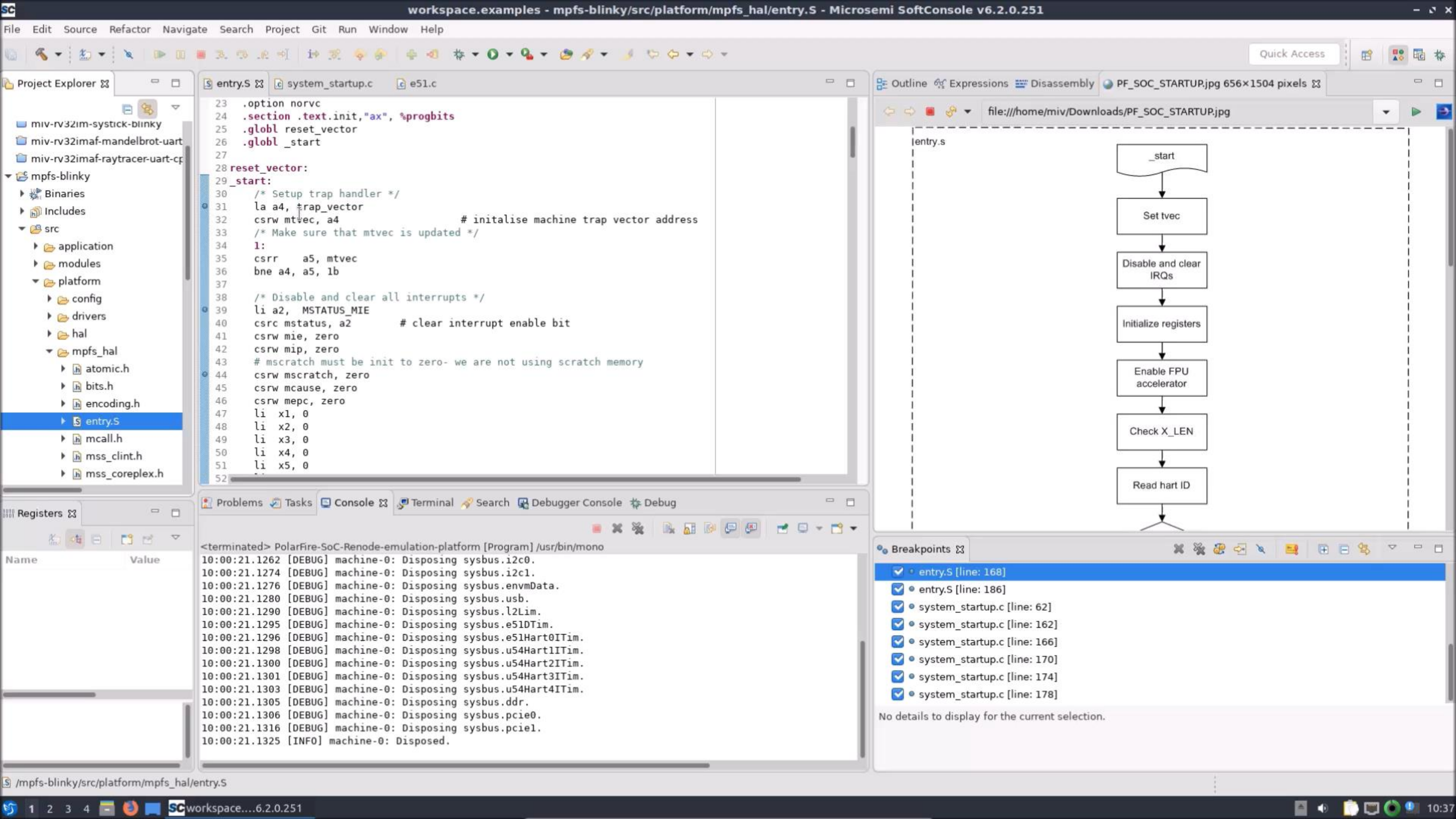
- Open the “Filter” heading
- All threads are selected by default, deselect the threads (harts) you don’t want to halt on the breakpoint



Restrict to Selected Processes and Threads:

- ☑ mpfs-blinky.elf
  - ☑ Thread #1 [machine-0.e51[0]] 1
  - ☑ Thread #2 [machine-0.u54_1[1]] 2
  - ☑ Thread #3 [machine-0.u54_2[2]] 3
  - ☑ Thread #4 [machine-0.u54_3[3]] 4
  - ☑ Thread #5 [machine-0.u54_4[4]] 5

Demo PolarFire® SoC Start Up Sequence



```
.option norvc
.section .text.init,"ax", %progbits
.globl reset_vector
.globl _start

reset_vector:
_start:
    /* Setup trap handler */
    la a4, trap_vector
    csrw mtvec, a4                 # initalise machine trap vector address
    /* Make sure that mtvec is updated */
    1:
    csrr    a5, mtvec
    bne a4, a5, 1b

    /* Disable and clear all interrupts */
    li a2,  MSTATUS_MIE
    csrc mstatus, a2        # clear interrupt enable bit
    csrw mie, zero
    csrw mip, zero
    # mscratch must be init to zero- we are not using scratch memory
    csrw mscratch, zero
    csrw mcause, zero
    csrw mepc, zero
    li  x1, 0
    li  x2, 0
    li  x3, 0
    li  x4, 0
    li  x5, 0
```

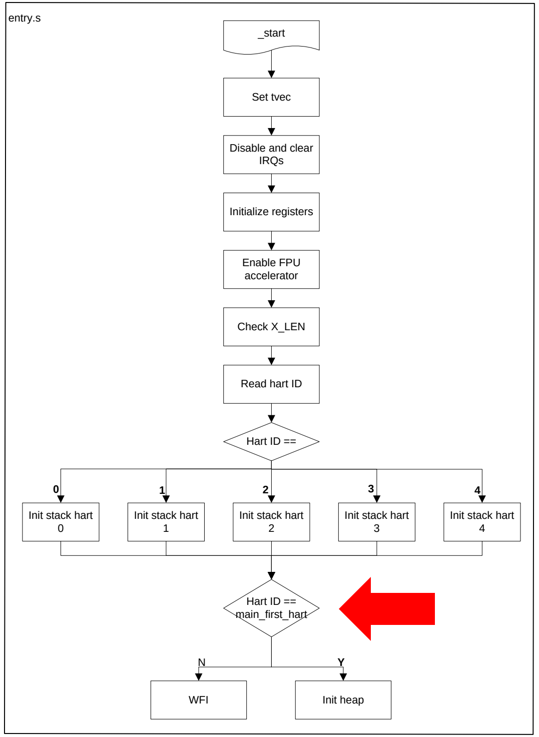
```
<terminated> PolarFire-SoC-Renode-emulation-platform [Program] /usr/bin/mono
10:00:21.1262 [DEBUG] machine-0: Disposing sysbus.i2c0.
10:00:21.1274 [DEBUG] machine-0: Disposing sysbus.i2c1.
10:00:21.1276 [DEBUG] machine-0: Disposing sysbus.envmData.
10:00:21.1280 [DEBUG] machine-0: Disposing sysbus.usb.
10:00:21.1290 [DEBUG] machine-0: Disposing sysbus.l2Lim.
10:00:21.1295 [DEBUG] machine-0: Disposing sysbus.e51DTim.
10:00:21.1296 [DEBUG] machine-0: Disposing sysbus.e51Hart0ITim.
10:00:21.1298 [DEBUG] machine-0: Disposing sysbus.u54Hart1ITim.
10:00:21.1300 [DEBUG] machine-0: Disposing sysbus.u54Hart2ITim.
10:00:21.1301 [DEBUG] machine-0: Disposing sysbus.u54Hart3ITim.
10:00:21.1303 [DEBUG] machine-0: Disposing sysbus.u54Hart4ITim.
10:00:21.1305 [DEBUG] machine-0: Disposing sysbus.ddr.
10:00:21.1306 [DEBUG] machine-0: Disposing sysbus.pcie0.
10:00:21.1316 [DEBUG] machine-0: Disposing sysbus.pcie1.
10:00:21.1325 [INFO] machine-0: Disposed.
```

- ☑ entry.S [line: 168]
- ☑ entry.S [line: 186]
- ☑ system_startup.c [line: 62]
- ☑ system_startup.c [line: 162]
- ☑ system_startup.c [line: 166]
- ☑ system_startup.c [line: 170]
- ☑ system_startup.c [line: 174]
- ☑ system_startup.c [line: 178]

No details to display for the current selection.

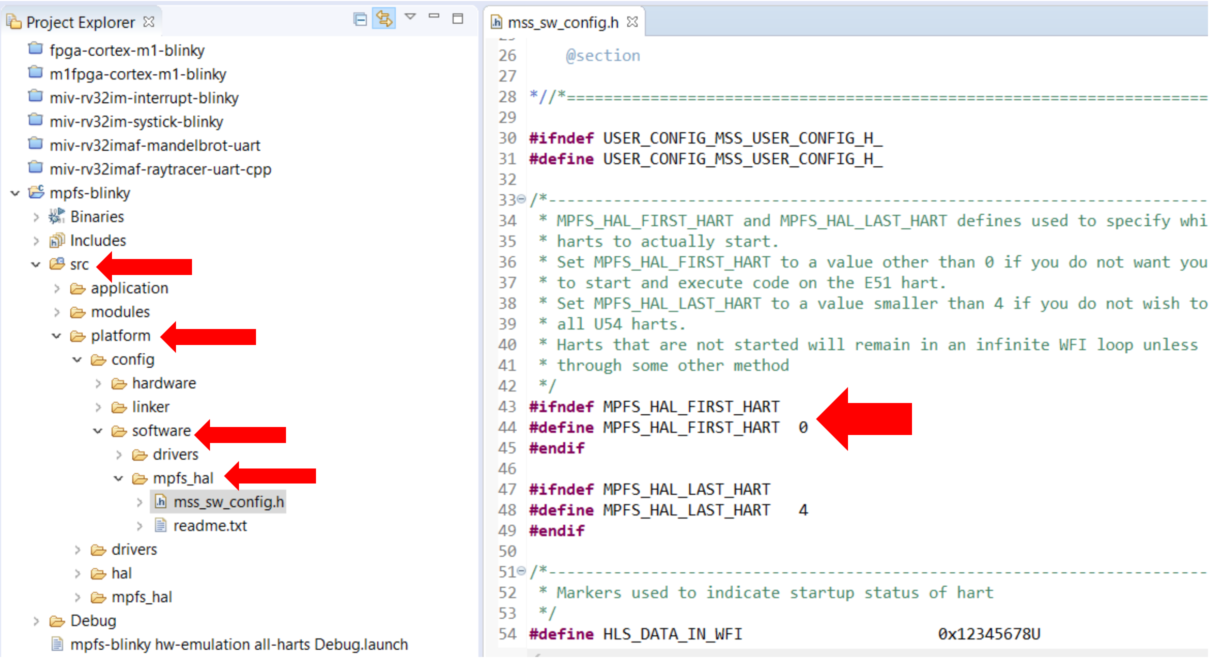
MPFS_HAL_FIRST_HART

MPFS_HAL_FIRST_HART



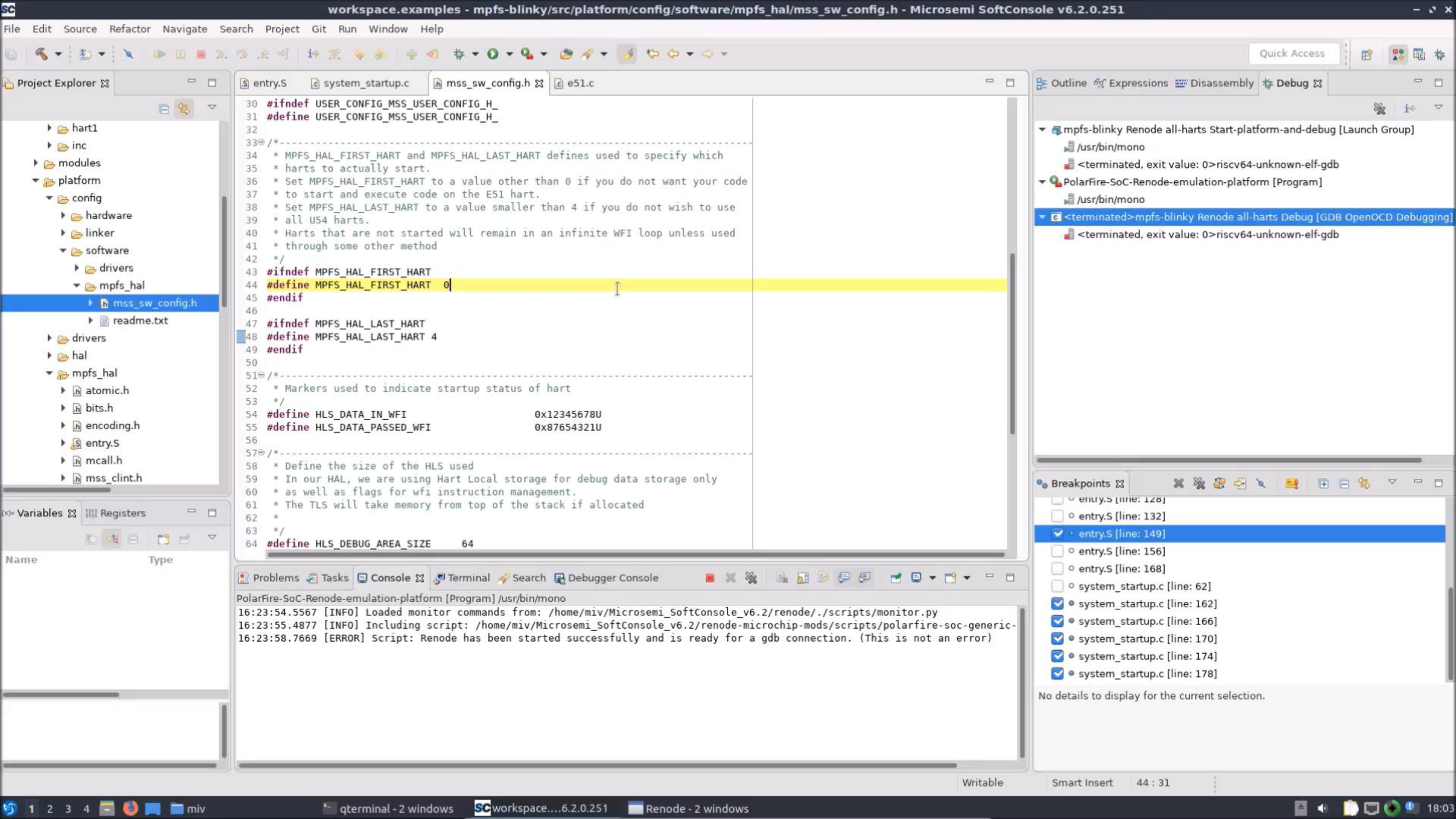
Here the harts will diverge and the main hart of the system will wake harts in use from WFI, the “MPFS_HAL_FIRST_HART” value can be changed to determine which hart is the main hart.

MPFS_HAL_FIRST_HART



```
    @section

*//*========================================================================

#ifndef USER_CONFIG_MSS_USER_CONFIG_H_
#define USER_CONFIG_MSS_USER_CONFIG_H_

/*------------------------------------------------------------------------
 * MPFS_HAL_FIRST_HART and MPFS_HAL_LAST_HART defines used to specify whi
 * harts to actually start.
 * Set MPFS_HAL_FIRST_HART to a value other than 0 if you do not want you
 * to start and execute code on the E51 hart.
 * Set MPFS_HAL_LAST_HART to a value smaller than 4 if you do not wish to
 * all U54 harts.
 * Harts that are not started will remain in an infinite WFI loop unless
 * through some other method
 */
#ifndef MPFS_HAL_FIRST_HART
#define MPFS_HAL_FIRST_HART  0
#endif

#ifndef MPFS_HAL_LAST_HART
#define MPFS_HAL_LAST_HART   4
#endif

/*------------------------------------------------------------------------
 * Markers used to indicate startup status of hart
 */
#define HLS_DATA_IN_WFI                         0x12345678U
```

The “MPFS_HAL_FIRST_HART” value is set to 0 by default to select the E51 as the main hart, this value can be changed to use a U54 hart as the main hart.



workspace.examples - mpfs-blinky/src/platform/config/software/mpfs_hal/mss_sw_config.h - Microsemi SoftConsole v6.2.0.251
File Edit Source Refactor Navigate Search Project Git Run Window Help
Quick Access
Project Explorer
hart1
inc
modules
platform
config
hardware
linker
software
drivers
mpfs_hal
mss_sw_config.h
readme.txt
drivers
hal
mpfs_hal
atomic.h
bits.h
encoding.h
entry.S
mcall.h
mss_clint.h
Variables
Registers
Name
Type
entry.S
system_startup.c
mss_sw_config.h
e51.c
Outline
Expressions
Disassembly
Debug
mpfs-blinky Renode all-harts Start-platform-and-debug [Launch Group]
/usr/bin/mono
<terminated, exit value: 0>riscv64-unknown-elf-gdb
PolarFire-SoC-Renode-emulation-platform [Program]
/usr/bin/mono
<terminated>mpfs-blinky Renode all-harts Debug [GDB OpenOCD Debugging]
<terminated, exit value: 0>riscv64-unknown-elf-gdb
Breakpoints
entry.S [line: 128]
entry.S [line: 132]
entry.S [line: 149]
entry.S [line: 156]
entry.S [line: 168]
system_startup.c [line: 62]
system_startup.c [line: 162]
system_startup.c [line: 166]
system_startup.c [line: 170]
system_startup.c [line: 174]
system_startup.c [line: 178]
No details to display for the current selection.
Problems
Tasks
Console
Terminal
Search
Debugger Console
Writable
Smart Insert
44 : 31
1 2 3 4
miv
qterminal - 2 windows
workspace....6.2.0.251
Renode - 2 windows
18:03

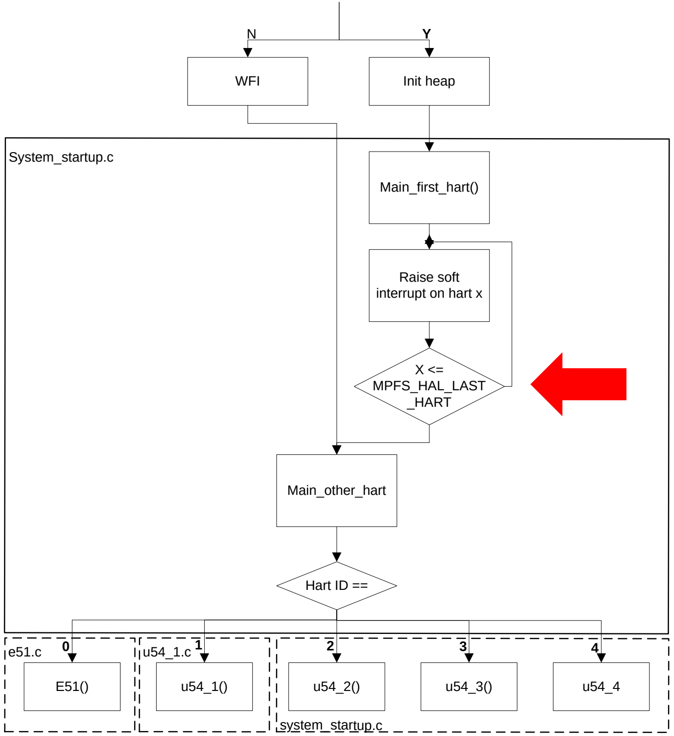
```
#ifndef USER_CONFIG_MSS_USER_CONFIG_H_
#define USER_CONFIG_MSS_USER_CONFIG_H_

/*------------------------------------------------------------------------------
 * MPFS_HAL_FIRST_HART and MPFS_HAL_LAST_HART defines used to specify which
 * harts to actually start.
 * Set MPFS_HAL_FIRST_HART to a value other than 0 if you do not want your code
 * to start and execute code on the E51 hart.
 * Set MPFS_HAL_LAST_HART to a value smaller than 4 if you do not wish to use
 * all U54 harts.
 * Harts that are not started will remain in an infinite WFI loop unless used
 * through some other method
 */
#ifndef MPFS_HAL_FIRST_HART
#define MPFS_HAL_FIRST_HART  0
#endif

#ifndef MPFS_HAL_LAST_HART
#define MPFS_HAL_LAST_HART 4
#endif

/*------------------------------------------------------------------------------
 * Markers used to indicate startup status of hart
 */
#define HLS_DATA_IN_WFI                     0x12345678U
#define HLS_DATA_PASSED_WFI                 0x87654321U

/*------------------------------------------------------------------------------
 * Define the size of the HLS used
 * In our HAL, we are using Hart Local storage for debug data storage only
 * as well as flags for wfi instruction management.
 * The TLS will take memory from top of the stack if allocated
 *
 */
#define HLS_DEBUG_AREA_SIZE     64
```

PolarFire-SoC-Renode-emulation-platform [Program] /usr/bin/mono

```
16:23:54.5567 [INFO] Loaded monitor commands from: /home/miv/Microsemi_SoftConsole_v6.2/renode/./scripts/monitor.py
16:23:55.4877 [INFO] Including script: /home/miv/Microsemi_SoftConsole_v6.2/renode-microchip-mods/scripts/polarfire-soc-generic-
16:23:58.7669 [ERROR] Script: Renode has been started successfully and is ready for a gdb connection. (This is not an error)
```

MPFS_HAL_LAST_HART

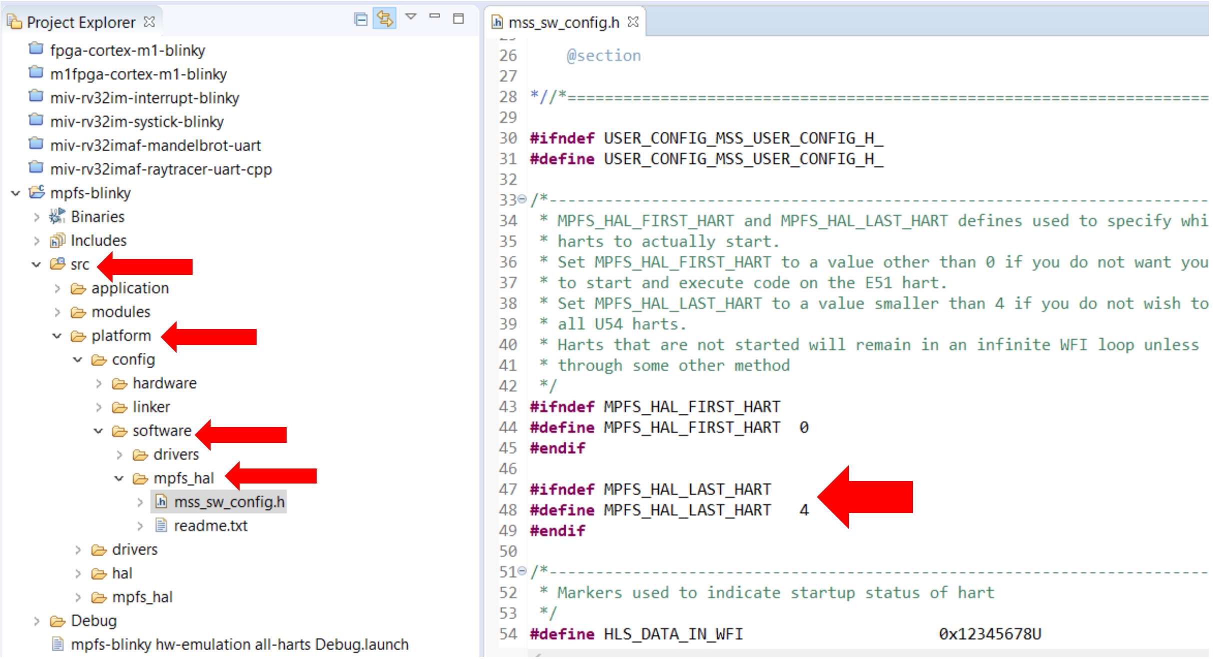
MPFS_HAL_LAST_HART



The main hart will loop here turning on the harts being used in the system.

If harts aren’t being used, the “MPFS_HAL_LAST_HART” value can be changed to leave unused harts in “wfi” mode.

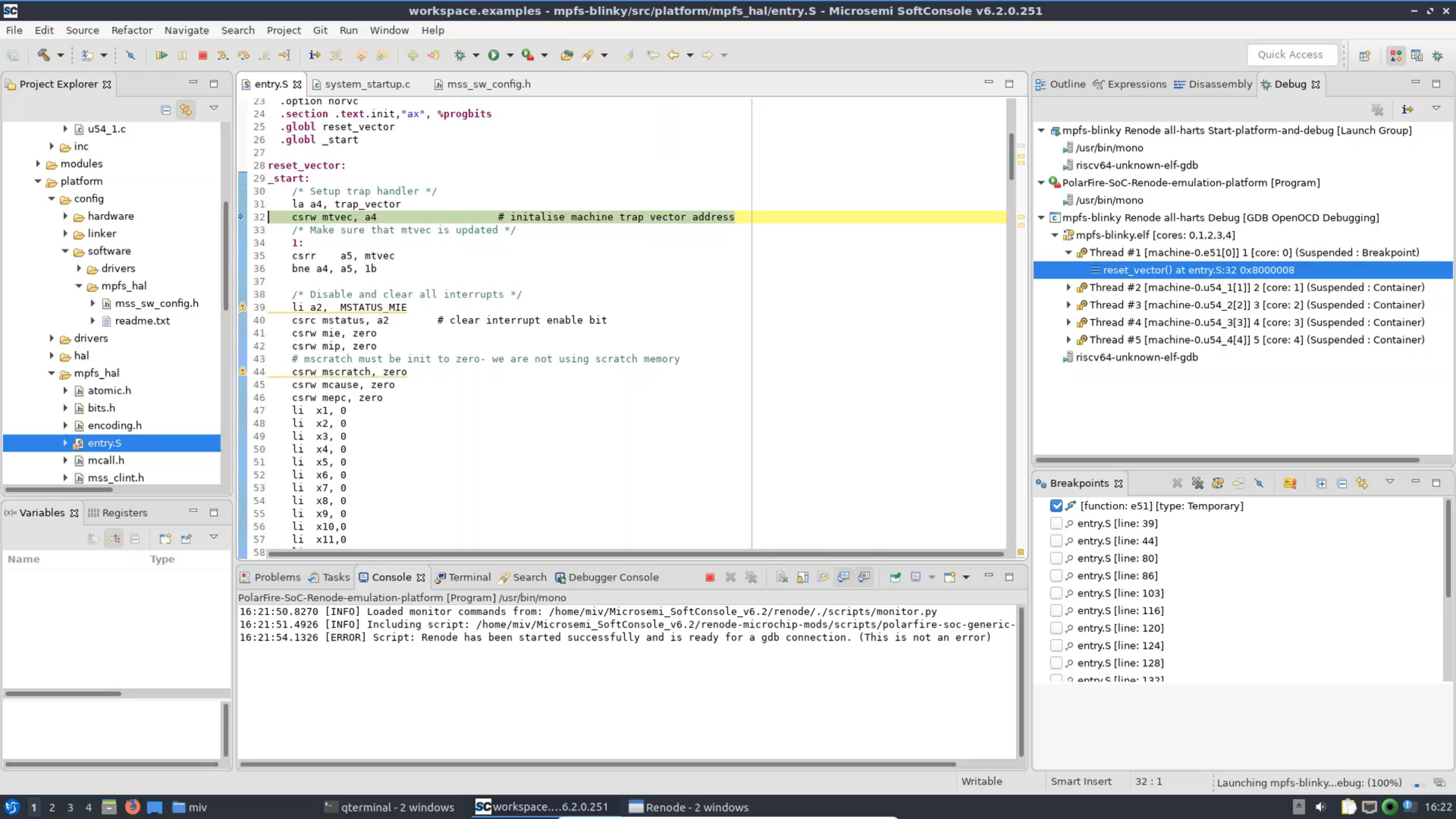
MPFS_HAL_LAST_HART



```
fpga-cortex-m1-blinky
m1fpga-cortex-m1-blinky
miv-rv32im-interrupt-blinky
miv-rv32im-systick-blinky
miv-rv32imaf-mandelbrot-uart
miv-rv32imaf-raytracer-uart-cpp
mpfs-blinky
  Binaries
  Includes
  src
    application
    modules
    platform
      config
        hardware
        linker
        software
          drivers
          mpfs_hal
            mss_sw_config.h
            readme.txt
      drivers
      hal
      mpfs_hal
  Debug
  mpfs-blinky hw-emulation all-harts Debug.launch
```

```
    @section

*//*===========================================================================

#ifndef USER_CONFIG_MSS_USER_CONFIG_H_
#define USER_CONFIG_MSS_USER_CONFIG_H_

/*----------------------------------------------------------------------------
 * MPFS_HAL_FIRST_HART and MPFS_HAL_LAST_HART defines used to specify whi
 * harts to actually start.
 * Set MPFS_HAL_FIRST_HART to a value other than 0 if you do not want you
 * to start and execute code on the E51 hart.
 * Set MPFS_HAL_LAST_HART to a value smaller than 4 if you do not wish to
 * all U54 harts.
 * Harts that are not started will remain in an infinite WFI loop unless
 * through some other method
 */
#ifndef MPFS_HAL_FIRST_HART
#define MPFS_HAL_FIRST_HART  0
#endif

#ifndef MPFS_HAL_LAST_HART
#define MPFS_HAL_LAST_HART   4
#endif

/*----------------------------------------------------------------------------
 * Markers used to indicate startup status of hart
 */
#define HLS_DATA_IN_WFI                    0x12345678U
```

The "MPFS_HAL_LAST_HART" value is set to 4 by default as harts without a defined function will be put into WFI mode from the weakly linked function.



```
 .option norvc
 .section .text.init,"ax", %progbits
 .globl reset_vector
 .globl _start

reset_vector:
_start:
   /* Setup trap handler */
   la a4, trap_vector
   csrw mtvec, a4                  # initalise machine trap vector address
   /* Make sure that mtvec is updated */
   1:
   csrr    a5, mtvec
   bne a4, a5, 1b

   /* Disable and clear all interrupts */
   li a2,  MSTATUS_MIE
   csrc mstatus, a2        # clear interrupt enable bit
   csrw mie, zero
   csrw mip, zero
   # mscratch must be init to zero- we are not using scratch memory
   csrw mscratch, zero
   csrw mcause, zero
   csrw mepc, zero
   li  x1, 0
   li  x2, 0
   li  x3, 0
   li  x4, 0
   li  x5, 0
   li  x6, 0
   li  x7, 0
   li  x8, 0
   li  x9, 0
   li  x10,0
   li  x11,0
```

```
mpfs-blinky Renode all-harts Start-platform-and-debug [Launch Group]
    /usr/bin/mono
    riscv64-unknown-elf-gdb
PolarFire-SoC-Renode-emulation-platform [Program]
    /usr/bin/mono
mpfs-blinky Renode all-harts Debug [GDB OpenOCD Debugging]
    mpfs-blinky.elf [cores: 0,1,2,3,4]
        Thread #1 [machine-0.e51[0]] 1 [core: 0] (Suspended : Breakpoint)
            reset_vector() at entry.S:32 0x8000008
        Thread #2 [machine-0.u54_1[1]] 2 [core: 1] (Suspended : Container)
        Thread #3 [machine-0.u54_2[2]] 3 [core: 2] (Suspended : Container)
        Thread #4 [machine-0.u54_3[3]] 4 [core: 3] (Suspended : Container)
        Thread #5 [machine-0.u54_4[4]] 5 [core: 4] (Suspended : Container)
    riscv64-unknown-elf-gdb
```

☑ [function: e51] [type: Temporary]
☐ entry.S [line: 39]
☐ entry.S [line: 44]
☐ entry.S [line: 80]
☐ entry.S [line: 86]
☐ entry.S [line: 103]
☐ entry.S [line: 116]
☐ entry.S [line: 120]
☐ entry.S [line: 124]
☐ entry.S [line: 128]
☐ entry.S [line: 132]

PolarFire-SoC-Renode-emulation-platform [Program] /usr/bin/mono

```
16:21:50.8270 [INFO] Loaded monitor commands from: /home/miv/Microsemi_SoftConsole_v6.2/renode/./scripts/monitor.py
16:21:51.4926 [INFO] Including script: /home/miv/Microsemi_SoftConsole_v6.2/renode-microchip-mods/scripts/polarfire-soc-generic-
16:21:54.1326 [ERROR] Script: Renode has been started successfully and is ready for a gdb connection. (This is not an error)
```

Agenda

- The PolarFire® SoC Start Up Sequence
- Setting Breakpoints on Specific Harts
- Demo PolarFire ® SoC Start Up Sequence
- MPFS_HAL_FIRST_HART
- MPFS_HAL_LAST_HART

Second Thursdays

Mar. 12	-	Webinar 11: Handling Binaries
April 9	-	Webinar 12: Two Bare-Metal Applications on PolarFire® SoC
May 14	-	Webinar 13: Linux® on Renode
June 11	-	Webinar 14: Building Applications for Linux on PolarFire SoC
July 9	-	Webinar 15: Real-Time (AMP Mode) on PolarFire SoC

MICROCHIP

Thank You

Any Questions?