

# Second Thursdays

- May 14 - Webinar 13: Two Bare-Metal Applications on PolarFire® SoC**
- June 11 - Webinar 14: The PolarFire SoC Icicle Kit Model in Renode**
- July 9 - Webinar 15: Linux® on Renode**
- Aug. 13 - Webinar 16: Building Applications for Linux on PolarFire SoC**
- Sep. 10 - Webinar 17: Real-Time (AMP Mode) on PolarFire SoC**

**Thank you for joining early, our presentation will start on the hour**

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

## Session 13 Two Bare-Metal Applications on PolarFire® SoC



---

A Leading Provider of Smart, Connected and Secure Embedded Control Solutions



SMART | CONNECTED | SECURE

***Hugh Breslin, Design Engineer***

*Thursday May. 14, 2020*

# Supporting Content

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

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

Webinar 11: Handling Binaries

Webinar 12: Simple Peripheral as Software Stimulus

The screenshot shows the Microsemi website's navigation and content. At the top, there's a search bar and a navigation menu with links for 'Ordering', 'Company', 'Partners', and 'Support'. Below this, a banner for 'Libero SoC Design Suite v12.0' is displayed. The main content area is titled 'Mi-V RISC-V Ecosystem' and features a horizontal menu with links: 'Overview', 'Mi-V Partners', 'Tutorials', 'Renode Webinar Series' (highlighted with a red box), and 'Articles and News'. Below the menu, the section 'Getting Started with the RISC-V Based PolarFire™ SoC FPGA Webinar Series' is visible, followed by a paragraph of introductory text and a link to register. Logos for 'Mi-V' and 'antmicro' are shown. At the bottom, the details for 'Webinar 1 (May 2): Discover Renode for PolarFire™ SoC Design and Debug' are listed, including a brief description of the session.

# Agenda

- **Hart Software Services**
- **HSS Boot Image**
- **Creating the HSS Boot Image**
- **Building the HSS**

# Hart Software Services

---

# Hart Software Services

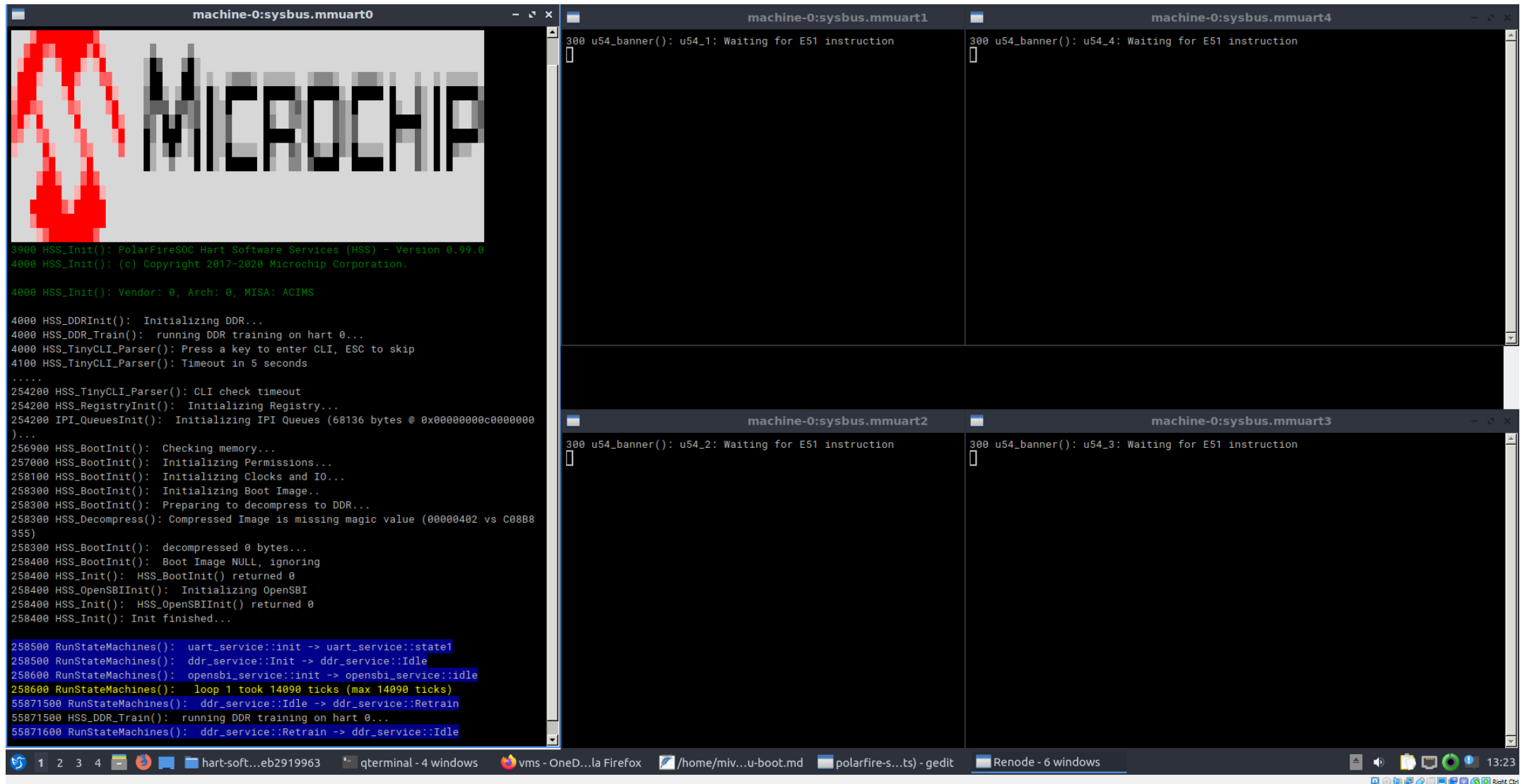
- **HSS is an application that runs on the E51**
- **It uses a superloop monitor to operate**
- **U54s can request the E51 to perform certain tasks / services on their behalf**
- **Features a Machine-Mode soft interrupt handler**
  - Allows the E51 to send messages to the U54s
  - Named “SSMB” Secure Software Message Bus
- **Release scheduled for June through GitHub**

# Hart Software Services

- **Services available include:**

- Boot
- Crypto
- DDR
- FlashFreeze
- Goto
- Ipi\_poll
- Opensbi
- Powermode
- Qspi
- Sgdma
- Spi
- UART
- Watchdog
- Y-Modem

# Hart Software Services



```
machine-0:sysbus.mmuart0
3900 HSS_Init(): PolarFireSOC Hart Software Services (HSS) - Version 0.99.0
4000 HSS_Init(): (c) Copyright 2017-2020 Microchip Corporation.

4000 HSS_Init(): Vendor: 0, Arch: 0, MISA: ACIMS

4000 HSS_DDRInit(): Initializing DDR...
4000 HSS_DDR_Train(): running DDR training on hart 0...
4000 HSS_TinyCLI_Parser(): Press a key to enter CLI, ESC to skip
4100 HSS_TinyCLI_Parser(): Timeout in 5 seconds
.....
254200 HSS_TinyCLI_Parser(): CLI check timeout
254200 HSS_RegistryInit(): Initializing Registry...
254200 IPI_QueueInit(): Initializing IPI Queues (68136 bytes @ 0x00000000c0000000)
...
256900 HSS_BootInit(): Checking memory...
257000 HSS_BootInit(): Initializing Permissions...
258100 HSS_BootInit(): Initializing Clocks and IO...
258300 HSS_BootInit(): Initializing Boot Image..
258300 HSS_BootInit(): Preparing to decompress to DDR...
258300 HSS-Decompress(): Compressed Image is missing magic value (00000402 vs C08B8355)
258300 HSS_BootInit(): decompressed 0 bytes...
258400 HSS_BootInit(): Boot Image NULL, ignoring
258400 HSS_Init(): HSS_BootInit() returned 0
258400 HSS_OpenSBIInit(): Initializing OpenSBI
258400 HSS_Init(): HSS_OpenSBIInit() returned 0
258400 HSS_Init(): Init finished...

258500 RunStateMachines(): uart_service::init -> uart_service::state1
258500 RunStateMachines(): ddr_service::init -> ddr_service::Idle
258600 RunStateMachines(): opensbi_service::init -> opensbi_service::Idle
258600 RunStateMachines(): loop 1 took 14090 ticks (max 14090 ticks)
55871500 RunStateMachines(): ddr_service::Idle -> ddr_service::Retrain
55871500 HSS_DDR_Train(): running DDR training on hart 0...
55871600 RunStateMachines(): ddr_service::Retrain -> ddr_service::Idle

machine-0:sysbus.mmuart1
300 u54_banner(): u54_1: Waiting for E51 instruction
[]

machine-0:sysbus.mmuart4
300 u54_banner(): u54_4: Waiting for E51 instruction
[]

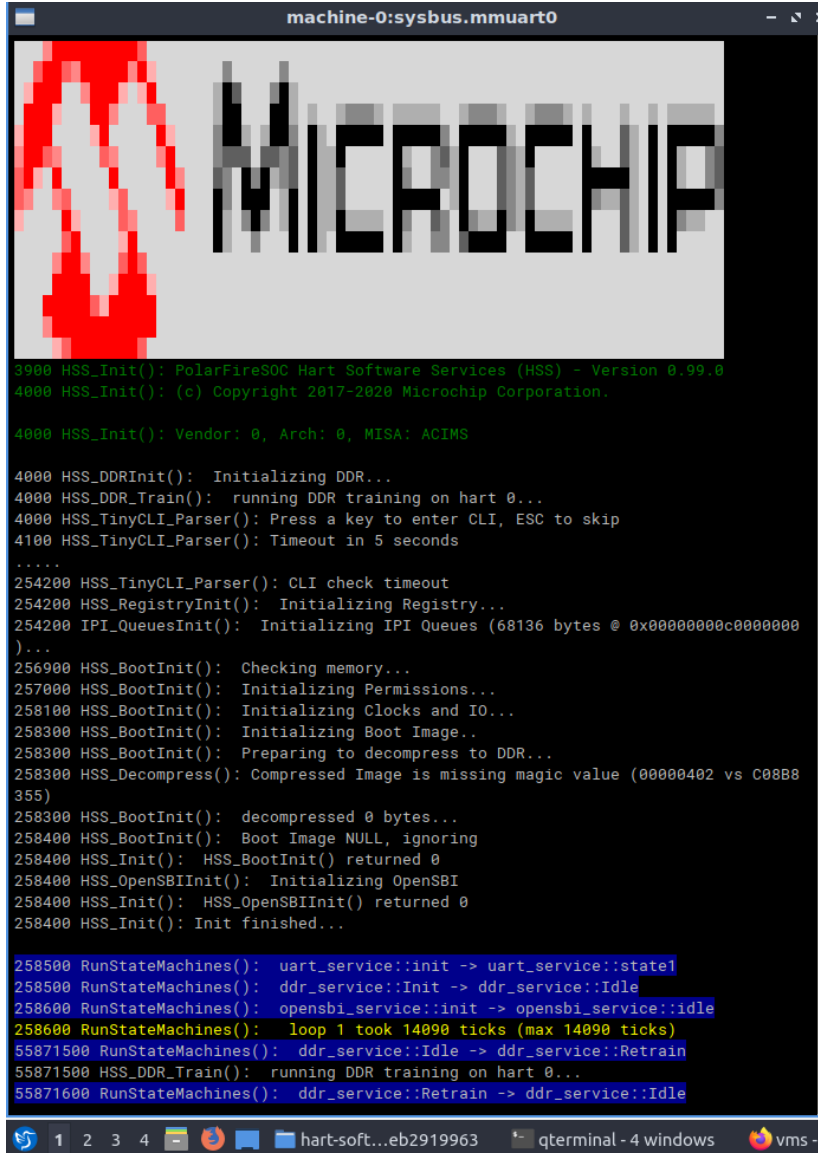
machine-0:sysbus.mmuart2
300 u54_banner(): u54_2: Waiting for E51 instruction
[]

machine-0:sysbus.mmuart3
300 u54_banner(): u54_3: Waiting for E51 instruction
[]
```

1 2 3 4 hart-soft...eb2919963 qterminal - 4 windows vms - OneD...la Firefox /home/miv...u-boot.md polarfire-s...ts) - gedit Renode - 6 windows 13:23



# Hart Software Services



```
machine-0:sysbus.mmuart0
3900 HSS_Init(): PolarFireSOC Hart Software Services (HSS) - Version 0.99.0
4000 HSS_Init(): (c) Copyright 2017-2020 Microchip Corporation.

4000 HSS_Init(): Vendor: 0, Arch: 0, MISA: ACIMS

4000 HSS_DDRInit(): Initializing DDR...
4000 HSS_DDR_Train(): running DDR training on hart 0...
4000 HSS_TinyCLI_Parser(): Press a key to enter CLI, ESC to skip
4100 HSS_TinyCLI_Parser(): Timeout in 5 seconds
.....
254200 HSS_TinyCLI_Parser(): CLI check timeout
254200 HSS_RegistryInit(): Initializing Registry...
254200 IPI_QueueInit(): Initializing IPI Queues (68136 bytes @ 0x00000000c0000000)
...
256900 HSS_BootInit(): Checking memory...
257000 HSS_BootInit(): Initializing Permissions...
258100 HSS_BootInit(): Initializing Clocks and IO...
258300 HSS_BootInit(): Initializing Boot Image..
258300 HSS_BootInit(): Preparing to decompress to DDR...
258300 HSS-Decompress(): Compressed Image is missing magic value (00000402 vs C08B8355)
258300 HSS_BootInit(): decompressed 0 bytes...
258400 HSS_BootInit(): Boot Image NULL, ignoring
258400 HSS_Init(): HSS_BootInit() returned 0
258400 HSS_OpenSBIInit(): Initializing OpenSBI
258400 HSS_Init(): HSS_OpenSBIInit() returned 0
258400 HSS_Init(): Init finished...

258500 RunStateMachines(): uart_service::init -> uart_service::state1
258500 RunStateMachines(): ddr_service::Init -> ddr_service::Idle
258600 RunStateMachines(): opensbi_service::init -> opensbi_service::idle
258600 RunStateMachines(): loop 1 took 14090 ticks (max 14090 ticks)
55871500 RunStateMachines(): ddr_service::Idle -> ddr_service::Retrain
55871500 HSS_DDR_Train(): running DDR training on hart 0...
55871600 RunStateMachines(): ddr_service::Retrain -> ddr_service::Idle
```

Initialise DDR

Configure system

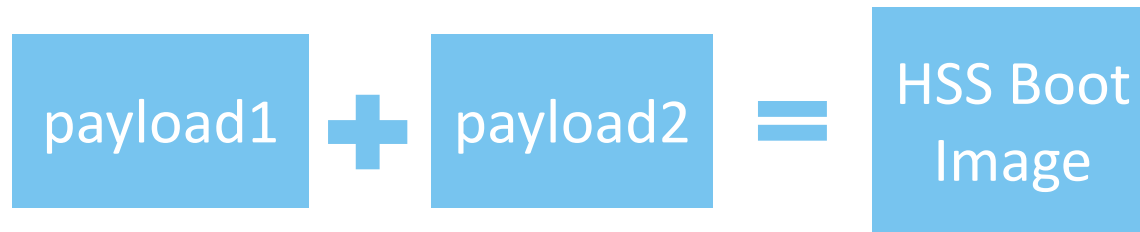
Initialize and decompress **boot image**

# HSS Boot Image

---

# HSS Boot Image

- The HSS Boot Image is the result of merging two binaries

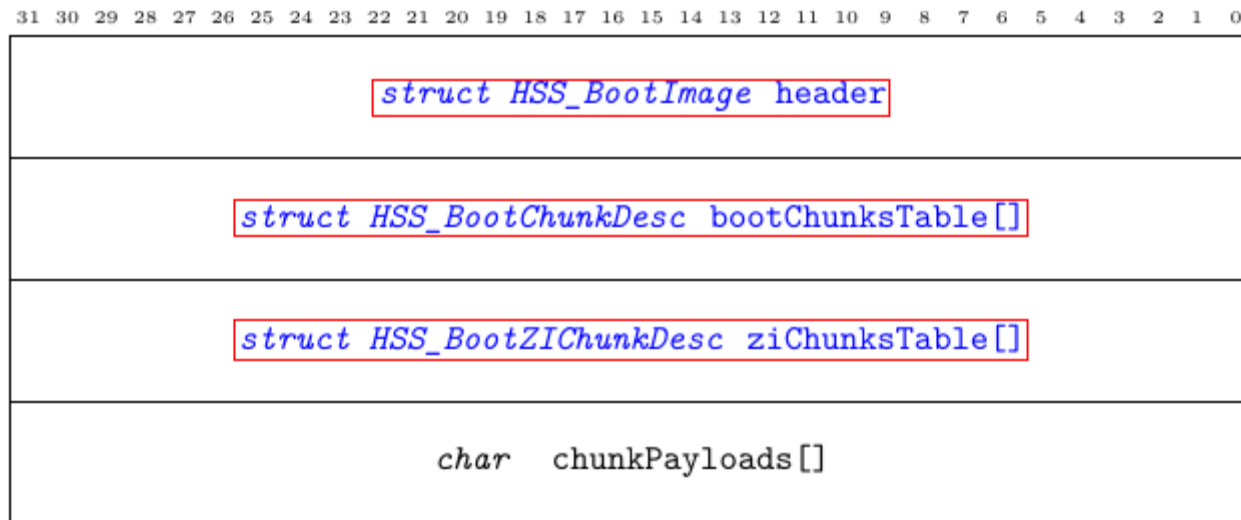


- This is accomplished using the included bin2chunks tool
- bin2chunks will extract and compile the data from both images into one in a fixed structure

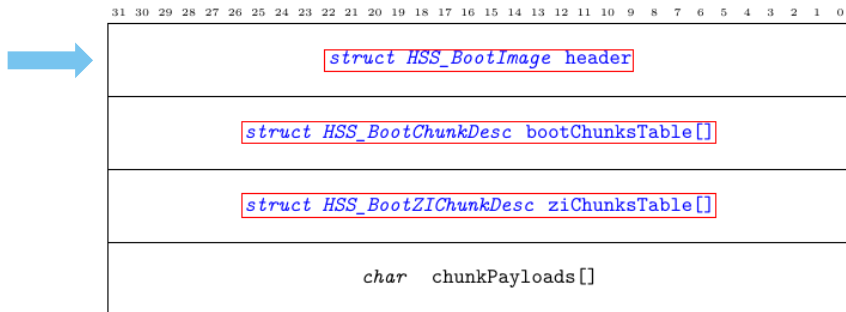
# HSS Boot Image

- **The image contains:**

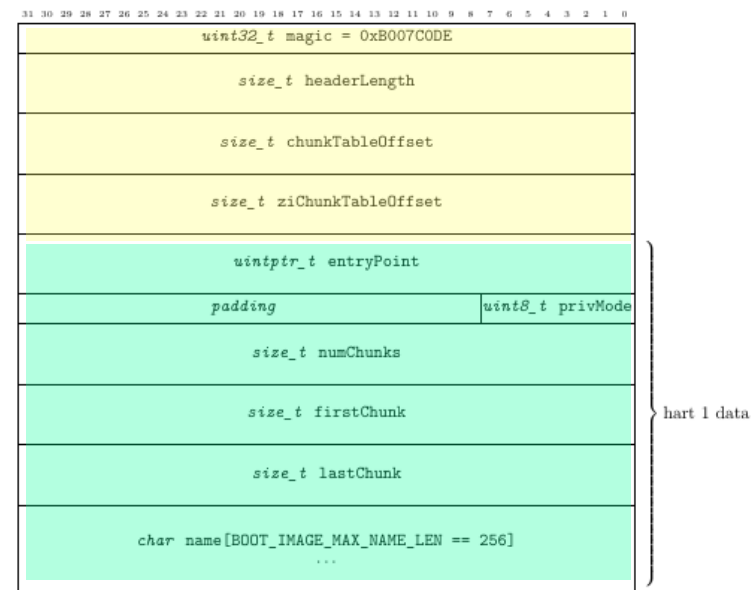
- A header
- A table of initialized boot chunks (code and data)
- A table of BSS and zero-init chunks
- The data itself



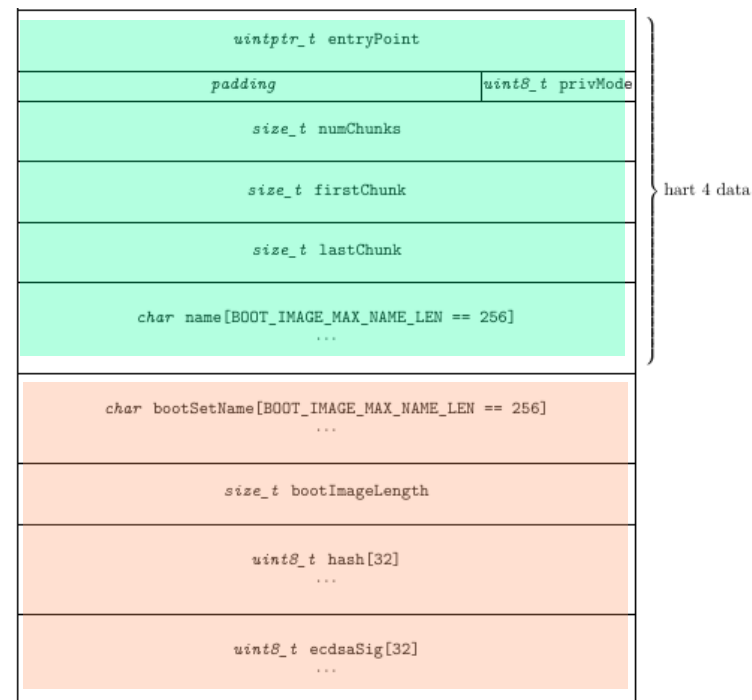
# HSS Boot Image



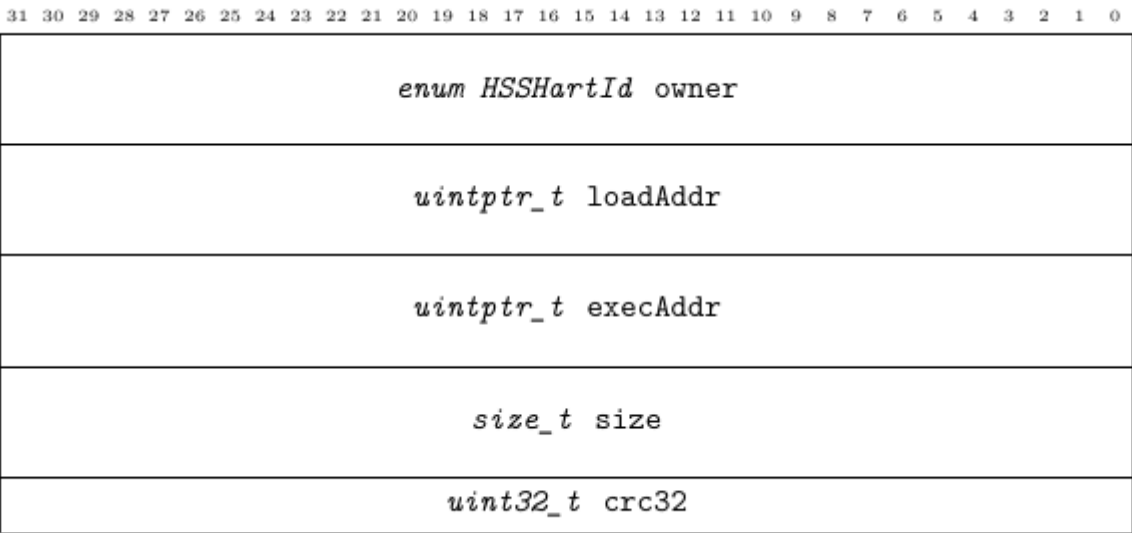
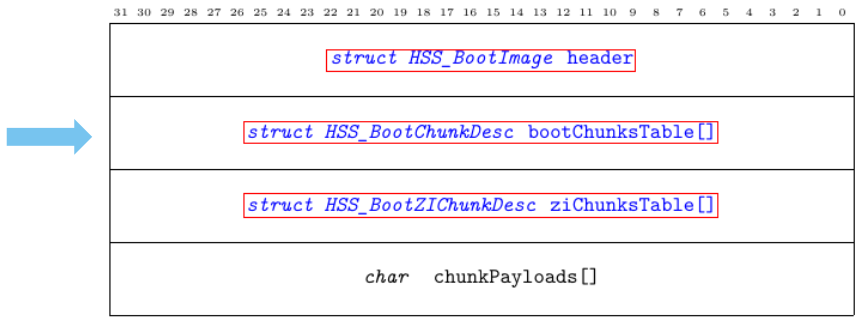
<code>uint32_t magic</code>	A magic value at the start of the boot image to identify the file as a valid image.
<code>size_t headerLength</code>	The length of the header in bytes.
<code>size_t chunkTableOffset</code>	The offset in bytes to the start of the chunk table, starting from the first byte of the magic value.
<code>size_t ziChunkTableOffset</code>	The offset in bytes to the start of the ZI chunk table, starting from the first byte of the magic value.
<code>uintptr_t entryPoint</code>	The entry point address for a particular hart to jump to once boot has completed.
<code>uint8_t privMode</code>	The privilege mode to start that particular hart in.
<code>size_t numChunks</code>	The number of chunks specific to that particular hart.
<code>size_t firstChunk</code>	The location of the first chunk for that particular hart in the chunks table.
<code>size_t lastChunk</code>	The location of the last chunk for that particular hart in the chunks table.
<code>char name[]</code>	A filename for a particular hart's boot image.
<code>char bootSetName[]</code>	A name to describe the collection of all hart boot images.
<code>size_t bootImageLength</code>	The overall length of the boot image, starting at the first byte of the magic value.
<code>uint8_t hash[]</code>	The hash digest of the boot image.
<code>uint8_t ecdsaSig[]</code>	The signature of the boot image.



....

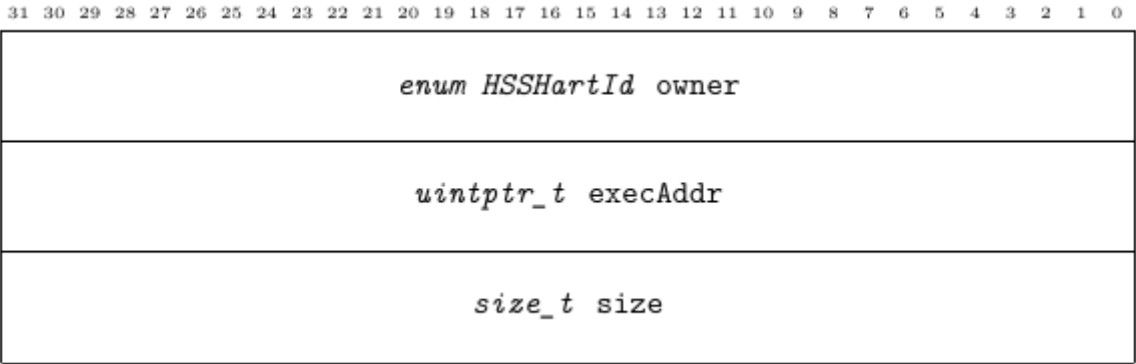
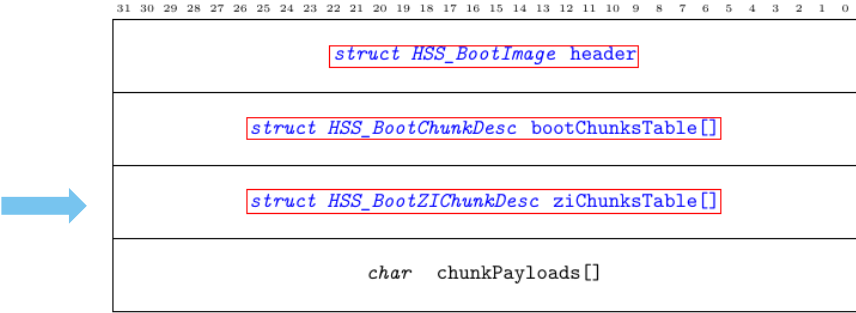


# HSS Boot Image



<code>enum HSSHartId owner</code>	Indicates which hart owns this chunk. This is used to ensure that chunks for a particular hart are only downloaded to memory regions permitted by PMP settings for that hart.
<code>uintptr_t loadAddr</code>	The offset from the first byte of magic where the chunked data will be in memory at load time.
<code>uintptr_t execAddr</code>	The address where the chunked lives in memory at execution time.
<code>size_t size</code>	The size of the chunk.
<code>uint32_t crc</code>	A CRC32 of the chunk.

# HSS Boot Image



<code>enum HSSHartId owner</code>	Indicates which hart owns this chunk. This is used to ensure that chunks for a particular hart are only downloaded to memory regions permitted by PMP settings for that hart.
<code>uintptr_t execAddr</code>	The address where the chunked lives in memory at execution time.
<code>size_t size</code>	The size of the chunk.

# Creating the HSS Boot Image

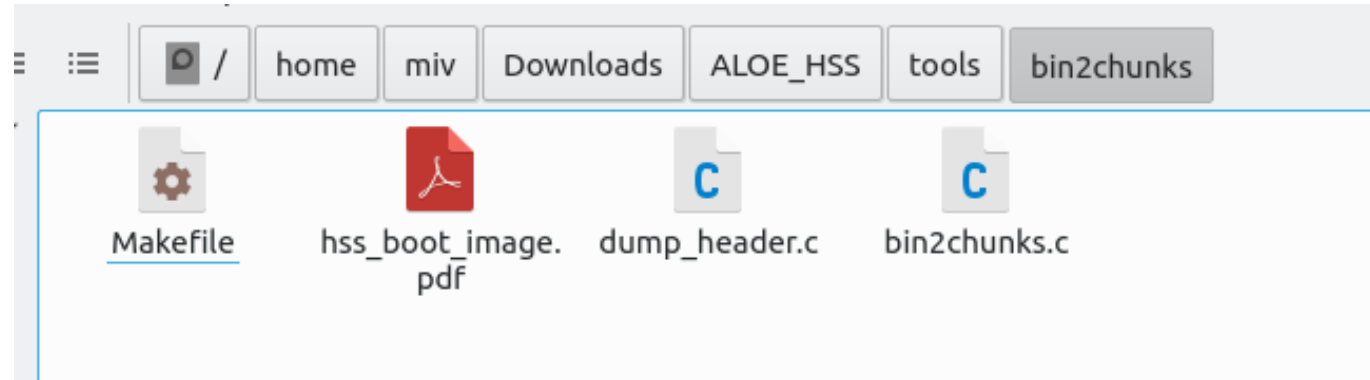
---



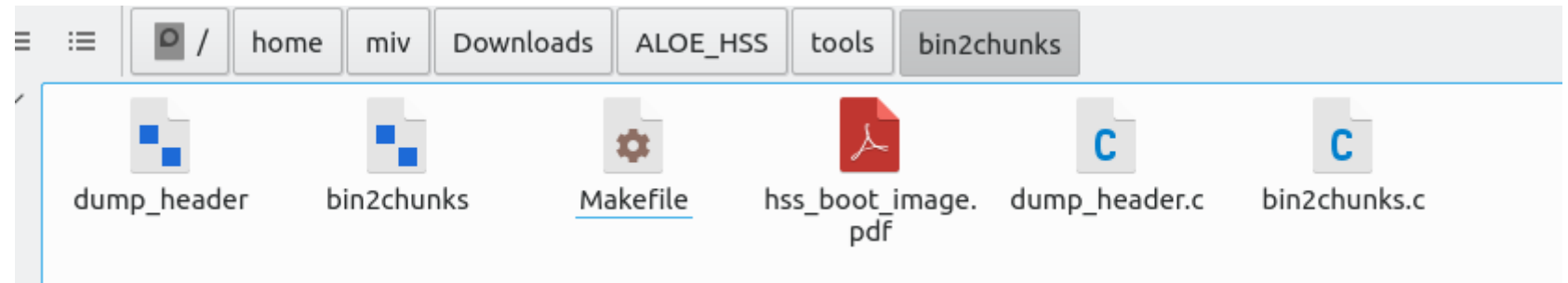
# Creating the HSS Boot Image

hart-software-services

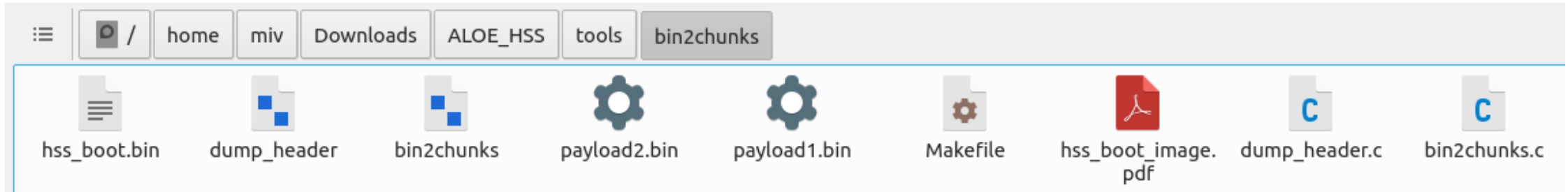
- └ baremetal
  - └ drivers (legacy)
  - └ polarfire-soc-bare-metal-library (subtree)
- └ compression
- └ debug (helper routines for function profiling)
- └ include
- └ init (system initialization)
- └ misc
- └ services (software service state machines)
  - └ boot
  - └ crypto
  - └ ddr
  - └ flashfreeze
  - └ goto
  - └ ipi\_poll
  - └ opensbi
  - └ powermode
  - └ qspi
  - └ sgdma
  - └ spi
  - └ uart
  - └ wdog
  - └ ymodem
- └ ssmb (secure software message bus)
  - └ ipi
- └ thirdparty
  - └ fastlz (fast lossless compression library)
  - └ opensbi (RISC-V OpenSBI)
  - └ riscv-pk (RISC-V Proxy Kernel - legacy)
- └ tools
  - └ bin2chunks



```
miv@mi-pc: ~/Downloads/ALOE_HSS/tools/bin2chunks
File Actions Edit View Help
miv@mi-pc: ~/Downloads/ALOE_HSS/tools/bin2chunks
miv@mi-pc:~/Downloads/ALOE_HSS/tools/bin2chunks$ make
gcc -Wall --std=gnu11 -O2 -g -DNR_CPUs=4 -o bin2chunks bin2chunks.c ../../misc/hss_crc32.c -I../../include -I../../ssmb/ipi/ -I../../thirdparty/opensbi/include -D__riscv_xlen=64
gcc -Wall --std=gnu11 -O2 -g -DNR_CPUs=4 -o dump_header dump_header.c -I../../include -I../../ssmb/ipi/ -I../../thirdparty/opensbi/include -D__riscv_xlen=64
miv@mi-pc:~/Downloads/ALOE_HSS/tools/bin2chunks$
```



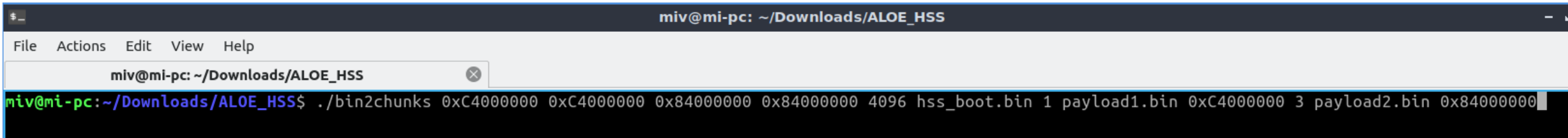
# Creating the HSS Boot Image



Bin2chunks takes 12 arguments:

1. U54\_1 entry point
2. U54\_2 entry point
3. U54\_3 entry point
4. U54\_4 entry point
5. Chunk limit (4096)
6. Output file name
7. Owner of payload 1
8. Payload 1 name
9. Memory address to place payload 1
10. Owner of payload 2
11. Payload 2 name
12. Memory address to place payload 2

# Creating the HSS Boot Image



The screenshot shows a terminal window titled "miv@mi-pc: ~/Downloads/Aloe\_HSS". The terminal has a menu bar with "File", "Actions", "Edit", "View", and "Help". Below the menu bar is a tab labeled "miv@mi-pc: ~/Downloads/Aloe\_HSS". The terminal text shows the command: `miv@mi-pc:~/Downloads/Aloe_HSS$ ./bin2chunks 0xC4000000 0xC4000000 0x84000000 0x84000000 4096 hss_boot.bin 1 payload1.bin 0xC4000000 3 payload2.bin 0x84000000`

1. U54\_1 entry point is 0xC4000000
2. U54\_2 entry point is 0xC4000000
3. U54\_3 entry point is 0x84000000
4. U54\_4 entry point is 0x84000000
5. Chunk limit (4096)
6. hss\_bootImage.bin is the output file name
7. 1 => U54\_1 is the owner of the following binary
8. payload1.bin is the binary name
9. 0xC4000000 is where payload1 will be placed
10. 3 => U54\_3 is the owner of the following binary
11. payload2.bin is the binary name
12. 0x84000000 is where payload2 will be placed

# Creating the HSS Boot Image

```
miv@mi-pc: ~/Downloads/Aloe_HSS/tools/bin2chunks
File Actions Edit View Help
miv@mi-pc: ~/Downloads/Aloe_HSS/tools/bin2chunks
miv@mi-pc:~/Downloads/Aloe_HSS/tools/bin2chunks$ ./bin2chunks 0xC4000000 0xC4000000 0x84000000 0x84000000 4096 hss_boot.bin 1 payload1.bin 0xC4000000 3 payload2.bin 0x84000000
entryPoint[0] set to c4000000
entryPoint[1] set to c4000000
entryPoint[2] set to 84000000
entryPoint[3] set to 84000000
chunkSize set to 4096
output file set to >>hss_boot.bin<<
- processing image 1
- hart owner is >>1<<
- input file is >>payload1.bin<<
execAddr[0] set to c4000000
- binSize[0] is 1616
- processing image 2
- hart owner is >>3<<
- input file is >>payload2.bin<<
execAddr[1] set to 84000000
- binSize[1] is 1616
Boot Image Size:          1552
Boot Image Padded Size:   1552

Number of Chunks:         3
Chunk Table Size:         120
Chunk Table Padded Size:  120

Number of ZI Chunks:      1
ZI Chunk Table Size:      24
ZI Chunk Table Padded Size: 24

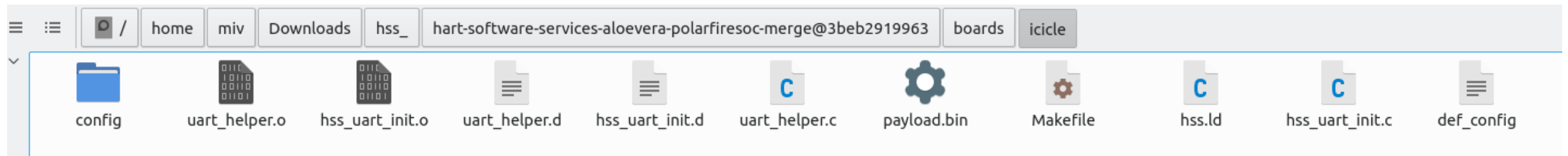
./bin2chunks: 9888 bytes written to >>hss_boot.bin<<
./bin2chunks: headerCrc is 0x3D022642
miv@mi-pc:~/Downloads/Aloe_HSS/tools/bin2chunks$
```

# Building the HSS

---

# Building the HSS

- Create your HSS boot image
- Place the resulting image in your board folder



- Build the HSS

# Building the HSS

```
miv@mi-pc: ~/Downloads/hss_...arfiresoc-merge@3beb2919963
miv@mi-pc:~/Downloads/hss_/hart-software-services-aloevera-polarfiresoc-merge@3beb2919963$ make MACHINE=icicle
Makefile:69: BOARD not specified
ICICLE selected
rules.mk:97: Not enabling -flto as stack protector enabled
MAKEDEP    baremetal/polarfire-soc-bare-metal-library/src/platform/mpfs_hal/mss_mutex.d
MAKEDEP    crt.d
MAKEDEP    misc/stack_guard.d
MAKEDEP    services/qspi/qspi_api.d
MAKEDEP    baremetal/sysreg.d
MAKEDEP    baremetal/polarfire-soc-bare-metal-library/src/platform/mpfs_hal/mss_stubs.d
MAKEDEP    baremetal/polarfire-soc-bare-metal-library/src/platform/mpfs_hal/mss_util.d
MAKEDEP    baremetal/polarfire-soc-bare-metal-library/src/platform/mpfs_hal/nwc/mss_sgmi.d
MAKEDEP    baremetal/polarfire-soc-bare-metal-library/src/platform/mpfs_hal/nwc/mss_pll.d
MAKEDEP    baremetal/polarfire-soc-bare-metal-library/src/platform/mpfs_hal/nwc/mss_io.d
MAKEDEP    baremetal/polarfire-soc-bare-metal-library/src/platform/mpfs_hal/nwc/mss_ddr.d
MAKEDEP    baremetal/polarfire-soc-bare-metal-library/src/platform/mpfs_hal/nwc/mss_nwc_init.d
MAKEDEP    baremetal/drivers/mss_watchdog/mss_watchdog.d
MAKEDEP    baremetal/polarfire-soc-bare-metal-library/src/platform/drivers/mss_mmuart/mss_uart.d
MAKEDEP    baremetal/polarfire-soc-bare-metal-library/src/platform/drivers/mss_qspi/mss_qspi.d
MAKEDEP    baremetal/polarfire-soc-bare-metal-library/src/platform/drivers/mss_pdma/mss_pdma.d
MAKEDEP    thirdparty/opensbi/lib/sbi/sbi_string.d
MAKEDEP    misc/hss_tinycli.d
MAKEDEP    misc/hss_progress.d
MAKEDEP    misc/hss_memcpy_via_pdma.d
MAKEDEP    misc/hss_crc32.d
CC          build/lib/utls/libfdt/fdt_rw.o
CC          build/lib/utls/libfdt/fdt_sw.o
CC          build/lib/utls/libfdt/fdt_strerror.o
CC          build/lib/utls/libfdt/fdt_empty_tree.o
CC          build/lib/utls/irqchip/plic.o
AR          build/lib/libsbutils.a
LD          hss.elf
NM          hss.sym
BIN         hss.bin
HEX         hss.hex
   text    data      bss      dec      hex filename
   76554   1944   107516  186014  2d69e hss.elf
miv@mi-pc:~/Downloads/hss_/hart-software-services-aloevera-polarfiresoc-merge@3beb2919963$
```

# Agenda

- **Hart Software Services**
- **HSS Boot Image**
- **Creating the HSS Boot Image**
- **Building the HSS**



# Thank you!

---

Any questions?

# Second Thursdays

- June 11 - Webinar 14: The PolarFire SoC Icicle Kit Model in Renode**
- July 9 - Webinar 15: Linux<sup>®</sup> on Renode**
- Aug. 13 - Webinar 16: Building Applications for Linux on PolarFire SoC**
- Sep. 10 - Webinar 17: Real-Time (AMP Mode) on PolarFire SoC**