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

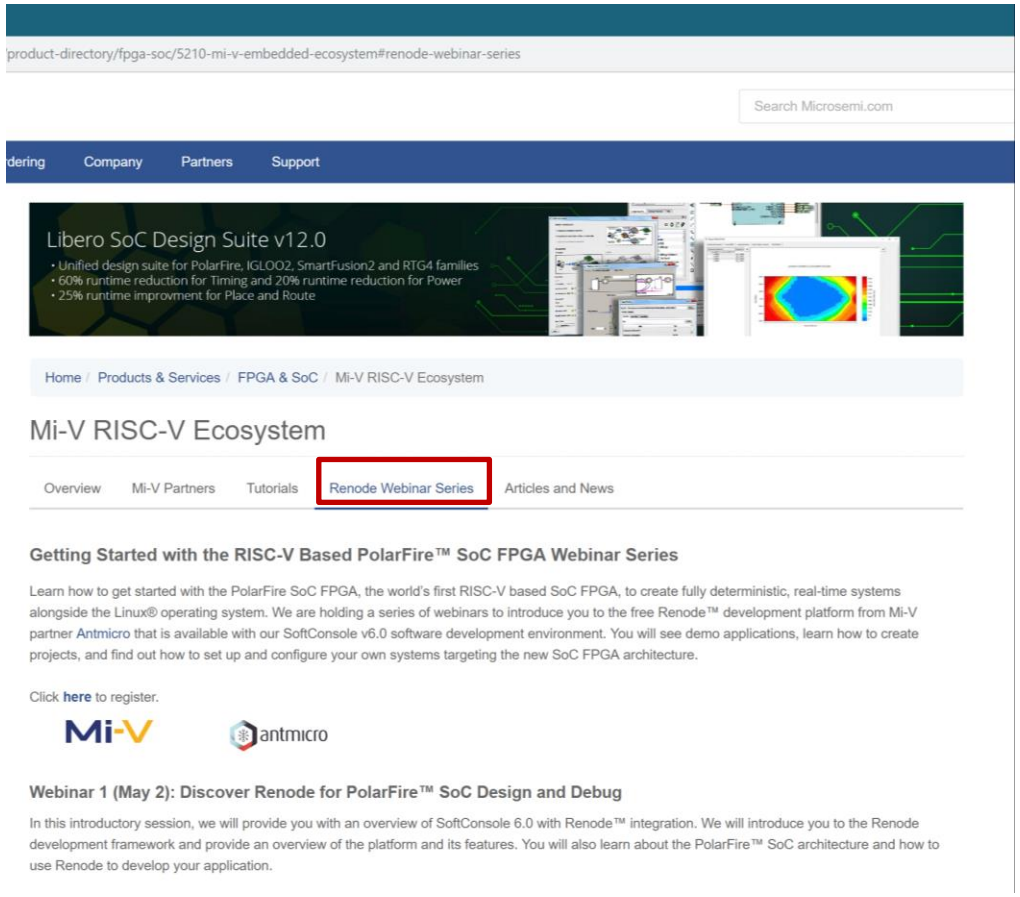
A Leading Provider of Smart, Connected and Secure Embedded Control Solutions

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

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

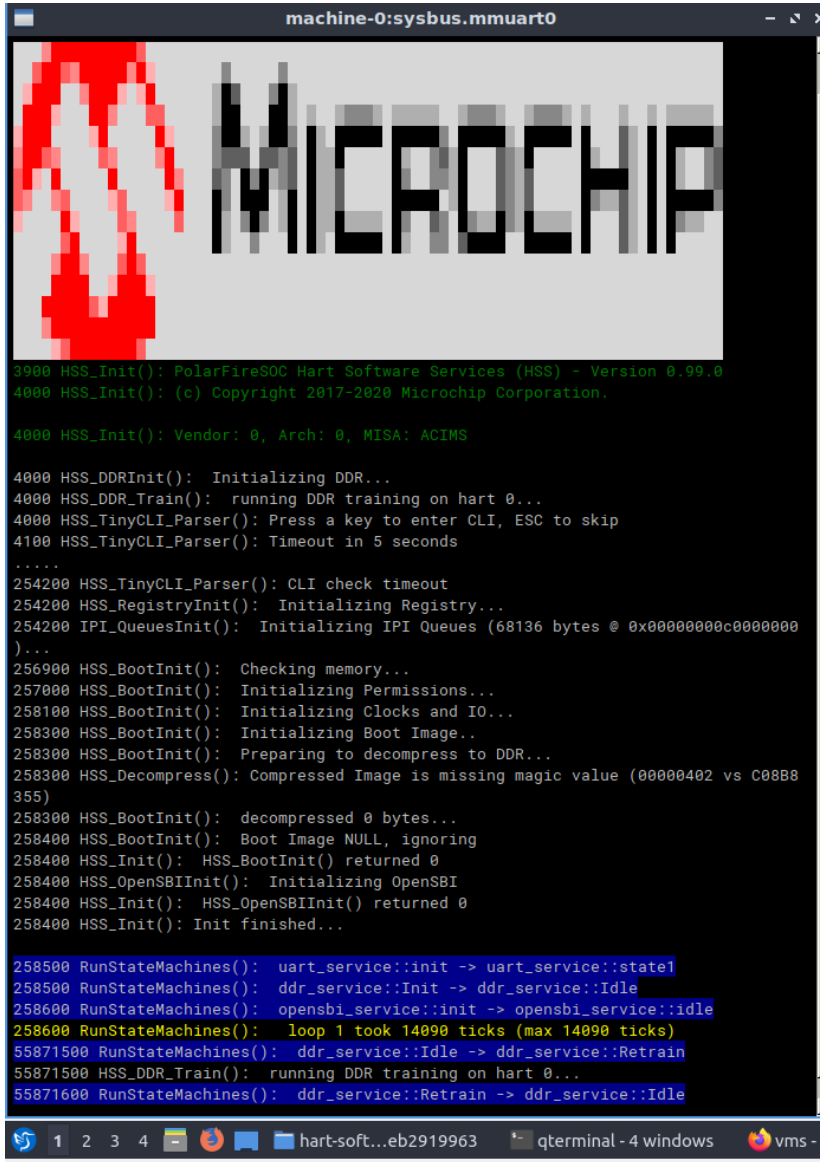
- **Release scheduled for June through GitHub**

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

Hart Software Services



Initialise DDR

Configure system

Initialize and decompress **boot image**

HSS Boot Image

HSS Boot Image

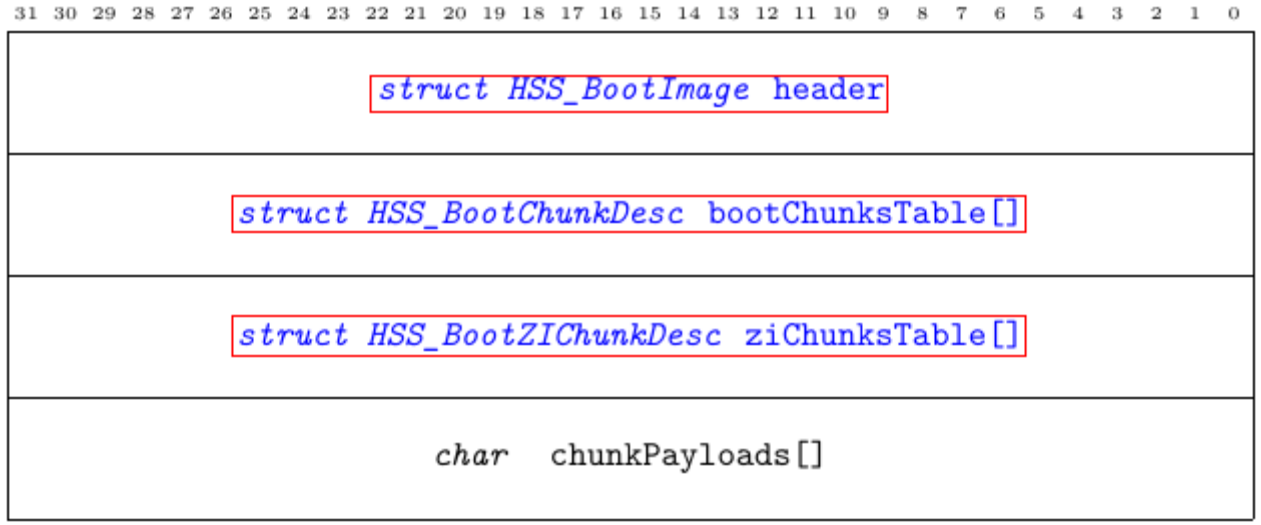
- **The HSS Boot Image is the result of merging two binaries**

- **This is accomplished using the included bin2chunks tool**

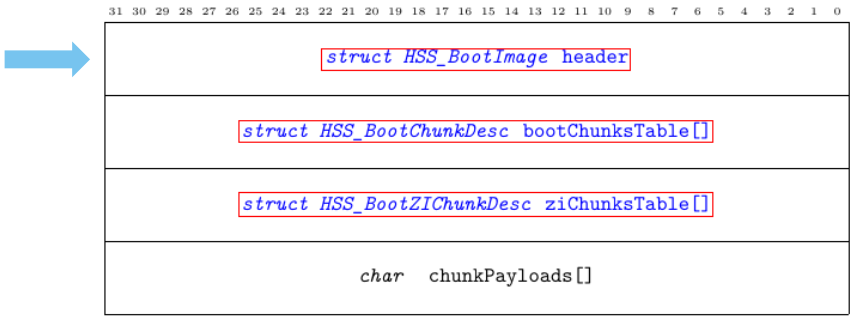
- **bin2chunks will extract and compile the data from both images into one in a fixed structure**

HSS Boot Image

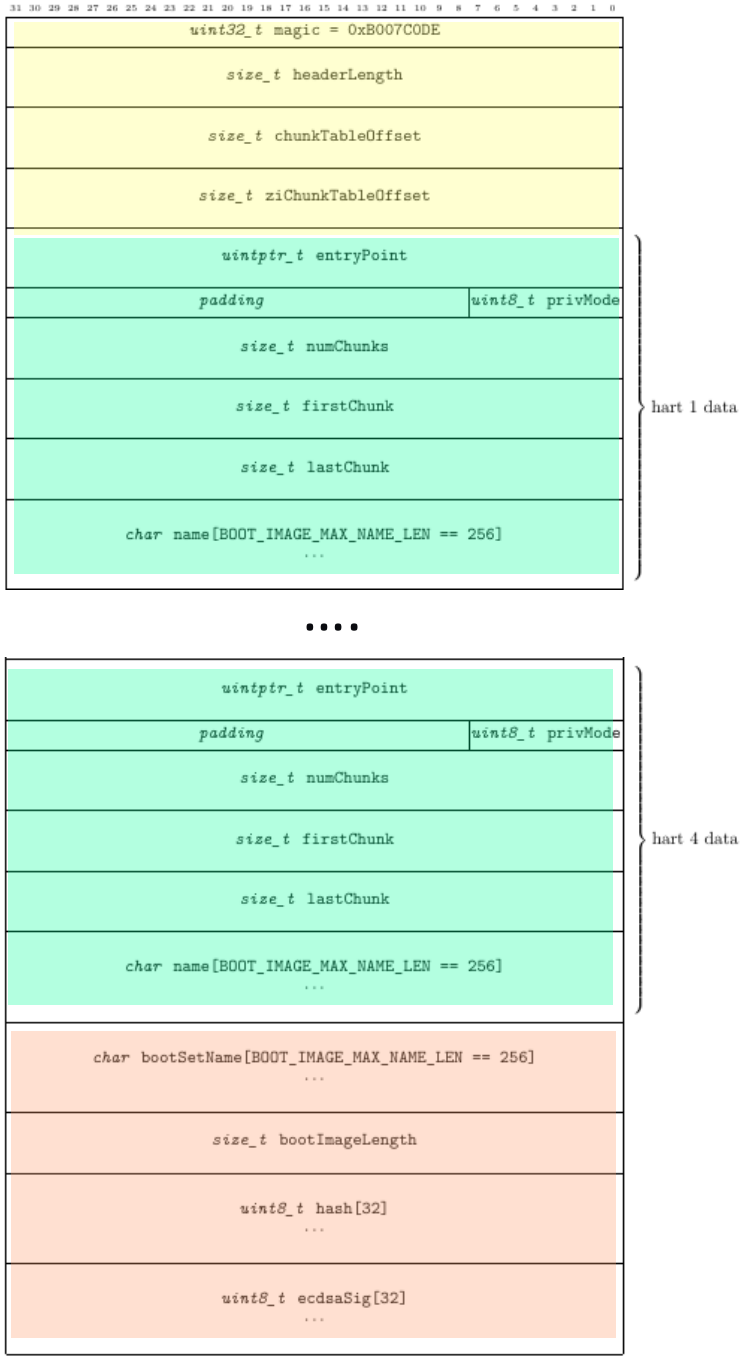
- **The image contains:**
  - A header
  - A table of initialized boot chunks (code and data)
  - A table of BSS and zero-init chunks
  - The data itself



HSS Boot Image



uint32_t magic	A magic value at the start of the boot image to identify the file as a valid image.
size_t headerLength	The length of the header in bytes.
size_t chunkTableOffset	The offset in bytes to the start of the chunk table, starting from the first byte of the magic value.
size_t ziChunkTableOffset	The offset in bytes to the start of the ZI chunk table, starting from the first byte of the magic value.
uintptr_t entryPoint	The entry point address for a particular hart to jump to once boot has completed.
uint8_t privMode	The privilege mode to start that particular hart in.
size_t numChunks	The number of chunks specific to that particular hart.
size_t firstChunk	The location of the first chunk for that particular hart in the chunks table.
size_t lastChunk	The location of the last chunk for that particular hart in the chunks table.
char name[]	A filename for a particular hart's boot image.
char bootSetName[]	A name to describe the collection of all hart boot images.
size_t bootImageLength	The overall length of the boot image, starting at the first byte of the magic value.
uint_8 hash[]	The hash digest of the boot image.
uint_8 ecdsaSig[]	The signature of the boot image.



HSS Boot Image

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
struct HSS_BootImage header
struct HSS_BootChunkDesc bootChunksTable[]
struct HSS_BootZIChunkDesc ziChunksTable[]
char chunkPayloads[]

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
enum HSSHartId owner
uintptr_t loadAddr
uintptr_t execAddr
size_t size
uint32_t crc32

enum HSSHartId owner	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.
uintptr_t loadAddr	The offset from the first byte of magic where the chunked data will be in memory at load time.
uintptr_t execAddr	The address where the chunked lives in memory at execution time.
size_t size	The size of the chunk.
uint32_t crc	A CRC32 of the chunk.

HSS Boot Image

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
struct HSS_BootImage header
struct HSS_BootChunkDesc bootChunksTable[]
struct HSS_BootZIChunkDesc ziChunksTable[]
char chunkPayloads[]

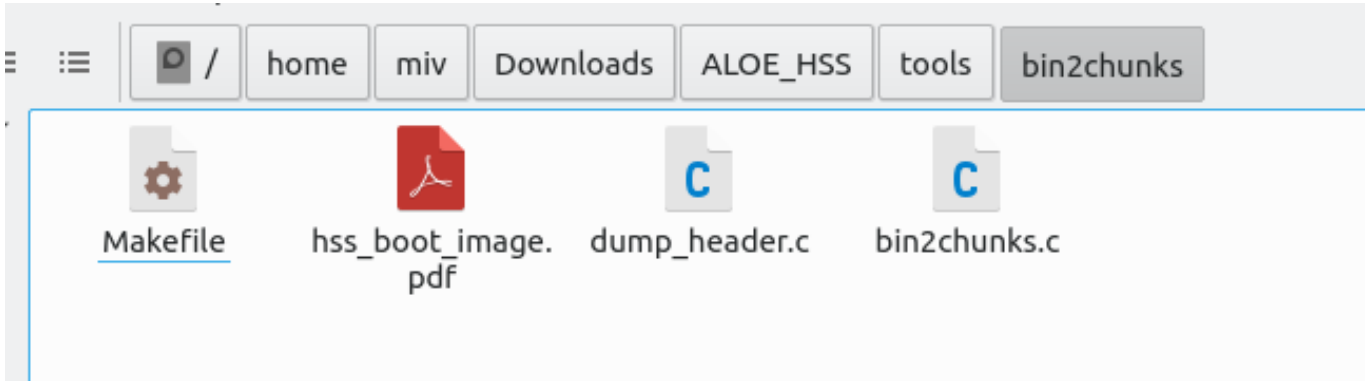
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
enum HSSHartId owner
uintptr_t execAddr
size_t size

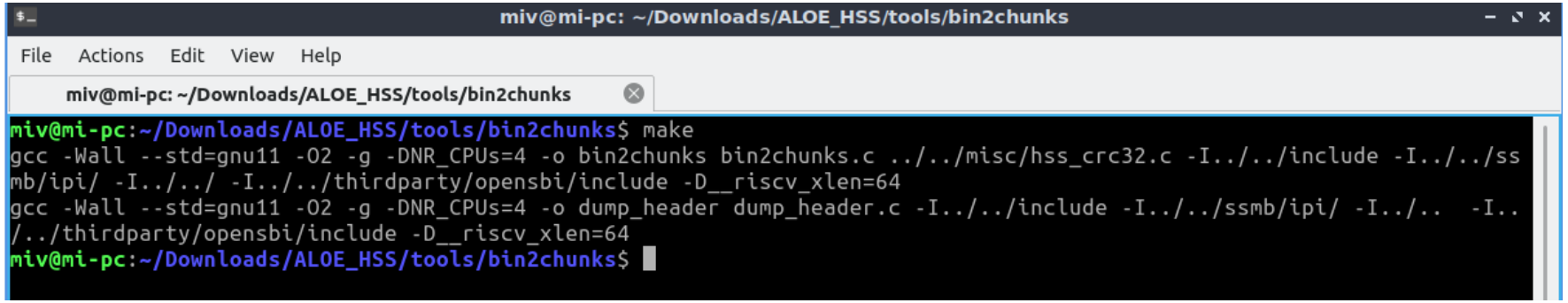
enum HSSHartId owner	Indicates which hart owns this chunk. This is used to ensure that chunks for a particular hart are only downloaded to memory regions permitted by PMP settings for that hart.
uintptr_t execAddr	The address where the chunked lives in memory at execution time.
size_t size	The size of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

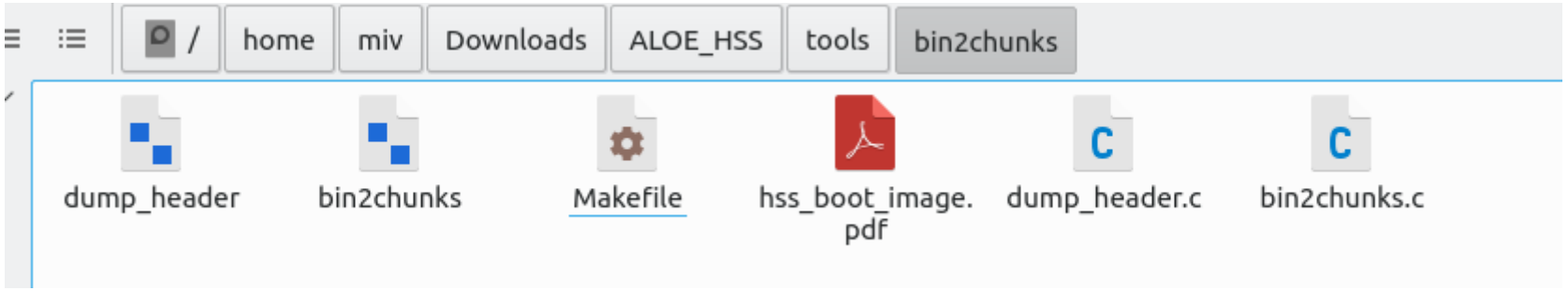
Creating the HSS Boot Image

Creating the HSS Boot Image

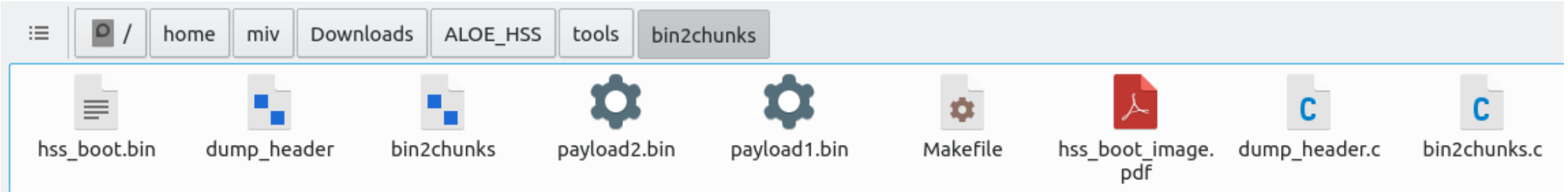
```
hart-software-services
├── baremetal
│   ├── drivers (legacy)
│   └── polarfire-soc-bare-metal-library (subtree)
├── compression
├── debug (helper routines for function profiling)
├── include
├── init (system initialization)
├── misc
├── services (software service state machines)
│   ├── boot
│   ├── 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
```







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

```
miv@mi-pc: ~/Downloads/ALOE_HSS
File  Actions  Edit  View  Help
miv@mi-pc: ~/Downloads/ALOE_HSS
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

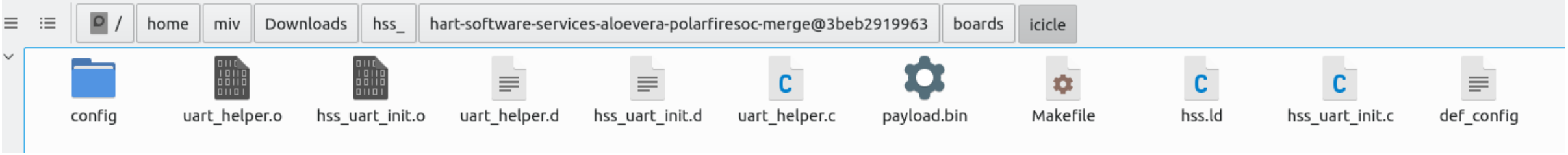
miv@mi-pc: ~/Downloads/ALOE_HSS/tools/bin2chunks

```
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_sgmii.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/utils/libfdt/fdt_rw.o
 CC        build/lib/utils/libfdt/fdt_sw.o
 CC        build/lib/utils/libfdt/fdt_strerror.o
 CC        build/lib/utils/libfdt/fdt_empty_tree.o
 CC        build/lib/utils/irqchip/plic.o
 AR        build/lib/libsbiutils.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® on Renode

Aug. 13 - Webinar 16: Building Applications for Linux on PolarFire SoC

Sep. 10 - Webinar 17: Real-Time (AMP Mode) on PolarFire SoC