RTK Bootcode

Agenda

- Build Boot loader
- On-Chip Recovery mode
- Upgrade Boot loader
- Rescue System
- Dual Firmware Boot
- Customize kernel boot arguments
- Customize reserved memory for transcoding

Build Boot loader RTD129x SPI

- U-Boot
- Path: SDKRelease/Bootcode/U-Boot64
- Build script: build.sh
- Usage:
  - ./build.sh RTD129x_spi
- Output:
  - Bootcode/U-Boot64/DVRBOOT_OUT/RTD129x_spi
  - (1) Bootloader with burning program
    - $(CHIP_VER) RTD1295_hwsetting_$(DDR)-nas-RTD1295_spi.bin
  - (2) HWsetting
    - hw_setting/$(CHIP_VER) RTD1295_hwsetting_BOOT$(DDR).bin

Build Boot loader RTD129x eMMc

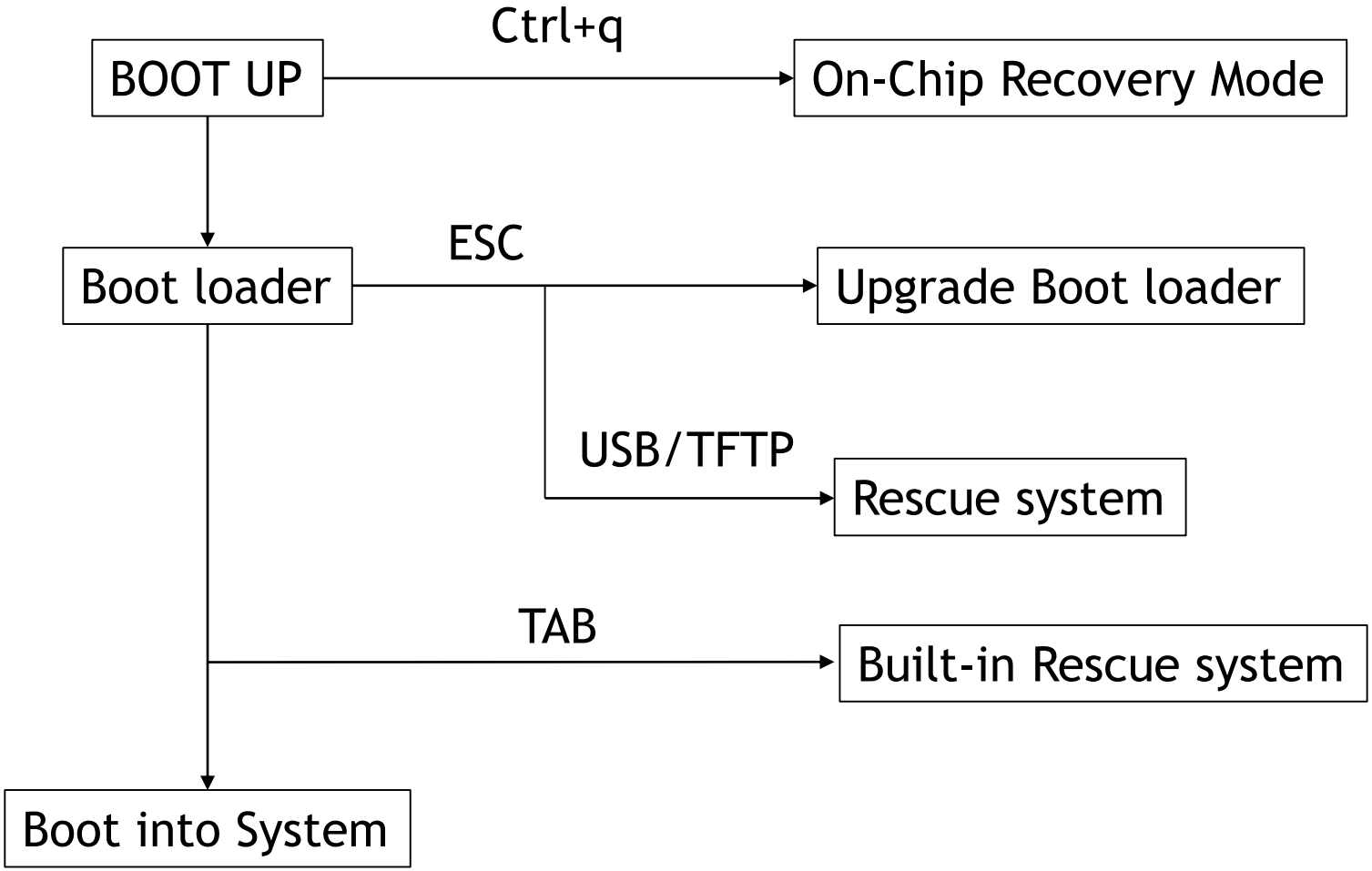
- U-Boot
- Path: SDKRelease/Bootcode/U-Boot64
- Build script: build.sh
- Usage:
  - ./build.sh RTD129x_emmc
- Output:
  - Bootcode/U-Boot64/DVRBOOT_OUT/RTD129x_emmc
  - (1) Bootloader with burning program
    - $(CHIP_VER) _hwsetting_$(DDR)-nas-$(CHIP)_spi.bin
  - (2) HWsetting
    - hw_setting/$(CHIP_VER) _hwsetting_BOOT$(DDR).bin

Build Boot loader – RTD1619 SPI

- U-Boot
- Path: SDKRelease/Bootcode/U-Boot64
- Build script: build.sh
- Usage: ./build.sh RTD16xx_emmc
- Output:
  - SDKRelease/Bootcode/U-Boot64/DVRBOOT_OUT/RTD16xx_emmc/
  - (1) Bootloader with burning program
    - A01-RTD161x_hwsetting_BOOT_2DDR4_8Gb_s2666-nas-RTD16xx_emmc.bin
  - (2) Bootloader
    - 1. A01-Recovery-uda-RTD161x_hwsetting_BOOT_2DDR4_8Gb_s2666-nas-RTD16xx_emmc.bin
    - 2. A01-Recovery-boot-RTD161x_hwsetting_BOOT_2DDR4_8Gb_s2666-nas-RTD16xx_emmc.bin
  - (3) HWsetting
    - hw_setting/0001-RTD161x_hwsetting_BOOT_2DDR4_8Gb_s2666-nas-RTD16xx_emmc.bin

Build Boot loader – RTD1619 eMMC

- U-Boot
- Path: SDKRelease/Bootcode/U-Boot64
- Build script: build.sh
- Usage: ./build.sh RTD16xx_emmc
- Output:
  - SDKRelease/Bootcode/U-Boot64/DVRBOOT_OUT/RTD16xx_emmc/
  - (1) Bootloader with burning program
    - A01-RTD161x_hwsetting_BOOT_2DDR4_8Gb_s2666-nas-RTD16xx_emmc.bin
  - (2) Bootloader
    - 1. A01-Recovery-uda-RTD161x_hwsetting_BOOT_2DDR4_8Gb_s2666-nas-RTD16xx_emmc.bin
    - 2. A01-Recovery-boot-RTD161x_hwsetting_BOOT_2DDR4_8Gb_s2666-nas-RTD16xx_emmc.bin
  - (3) HWsetting
    - hw_setting/0001-RTD161x_hwsetting_BOOT_2DDR4_8Gb_s2666-nas-RTD16xx_emmc.bin


BOOT UP
Ctrl+q
On-Chip Recovery Mode
Boot loader
ESC
Upgrade Boot loader
USB/TFTP
Rescue system
TAB
Built-in Rescue system
Boot into System

On-Chip Recovery mode – RTD 129x SPI

- This mode is used for bootloader recovery when the bootloader inside is corrupted
- Hyperterm is recommended to use
- Press ctrl+q when booting up until following console appear
  - d/g/r>
- Press h and then transfer Hwsetting(3) by y-modem
  - hw_setting/$(CHIP)_hwsetting_BOOT_$(DDR).bin
- Press s, then enter
  - 98007058
  - 01500000
- Press d and then transfer bootloader(2) by y-modem
  - $(CHIP_VER)_hwsetting_BOOT_$(DDR)-nas-$(CHIP)_spi.bin
- Press g to start recovery process

On-Chip Recovery mode – RTD129x eMMC

- This mode is used for bootloader recovery when the bootloader inside is corrupted
- Hyperterm is recommended to use
- Press ctrl+q when booting up until following console appear
  - d/g/r>
- Press h and then transfer Hwsetting(3) by y-modem
  - hw_setting/$(CHIP)_hwsetting_BOOT_$(DDR).bin
- Press s, then enter
  - 98007058
  - 01500000
- Press d and then transfer bootloader(2) by y-modem
  - $(CHIP_VER)_hwsetting_BOOT_$(DDR)-nas-$(CHIP)_emmc.bin
- Press g to start recovery process

On-Chip Recovery mode – RTD 16xx SPI

- This mode is used for bootloader recovery when the bootloader inside is corrupted
- Hyperterm is recommended to use
- Press ctrl+q when booting up until following console appear
  - d/g/r>
- Press h and then transfer Hwsetting(3) by y-modem
  - hw_setting/0001-RTD161x_hwsetting_BOOT_2DDR4_8Gb_s2666-nas-spi.bin
- Press d and then transfer bootloader(2) by y-modem
  - A01-Recovery-RTD161x_hwsetting_BOOT_2DDR4_8Gb_s2666-nas-spi.bin
- Press g to start recovery process

On-Chip Recovery mode – RTD16xx eMMC

- This mode is used for bootloader recovery when the bootloader inside is corrupted
- Hyperterm is recommended to use
- Press ctrl+q when booting up until following console appear
  - d/g/r>
- Press h and then transfer Hwsetting(3) by y-modem
  - hw_setting/0001-RTD161x_hwsetting_BOOT_2DDR4_8Gb_s2666-nas-RTD16xx_emmc.bin
- Press d and then transfer bootloader(2) -1 by y-modem
  - A01-Recovery-uda-RTD161x_hwsetting_BOOT_2DDR4_8Gb_s2666-nas-RTD16xx_emmc.bin
- Press g to start recovery process
- Press d and then transfer bootloader(2) -2 by y-modem
  - A01-Recovery-boot-RTD161x_hwsetting_BOOT_2DDR4_8Gb_s2666-nas-RTD16xx_emmc.bin
- Press b1 to start recovery process
- Press b2 to start recovery process

Upgrade Boot loader

- This is for boot loader upgrade when current boot loader is workable
- Press ESC when booting up to enter U-Boot console
  - Realtek>
- Transfer bootloader with burning program(1) to DRAM
  - By USB
    - usb start
    - fatload usb 0:1 0x1500000 A01-RTD161x_hwsetting_BOOT_2DDR4_8Gb_s2666-nas-RTD16xx_spi.bin or RTD161x_hwsetting_BOOT_2DDR4_8Gb_s2666-nas-RTD16xx_emmc.bin
    - go 0x1500000
  - By tftp
    - tftp 0x1500000 A01-RTD161x_hwsetting_BOOT_2DDR4_8Gb_s2666-nas-RTD16xx_spi.bin or RTD161x_hwsetting_BOOT_2DDR4_8Gb_s2666-nas-RTD16xx_emmc.bin
    - go 0x1500000

Rescue system

- This is used to install install.img onto RTD129x/RTD1619
- There are 2 ways to boot into rescue system
  - 1. use built-in rescue system
  - 2. use rescue system from USB or TFTP

- 1. For build-in rescue system, just press TAB when booting up, then the built-in rescue system is used
- Once booting into rescue system, put install.img in USB top directory and plug into RTD129x/RTD1619, the install process will start automatically

Rescue system

- 2. For rescue system from USB or TFTP, we should copy following 3 files to USB or TFTP directory

path: SDKRelease/OpenWRT-LEDE/target/linux/realtek/image/rtk-imagefile/workspace/rescue/

  - 1. spi.uImage
  - 2. rescue.spi.dtb
  - 3. rescue.root.spi.cpio.gz_pad.img

- **USB**: press ESC into U-Boot console mode with USB attached and then press 'gor'
- **TFTP**: press ESC and then input following command
  - tftp $kernel_loadaddr spi.uImage;tftp $fdt_loadaddr rescue.spi.dtb;tftp $rootfs_loadaddr rescue.root.spi.cpio.gz_pad.img;go k
- put install.img in USB top directory and plug into RTD129x/RTD1619, the install process will start automatically

TFTP configuration

- TFTP is configurable in boot loader console mode
- Press ESC into console mode and then edit tftp related environment variables

Realtek> env set ipaddr 192.168.0.9

Realtek> env set serverip 192.168.0.56

Realtek> env set gatewayip 192.168.0.254

Realtek> env set netmask 255.255.255.0

Realtek> env save

- ping command can be used to test network

Realtek> ping 192.168.0.56

Dual Firmware Boot

```
      #fw  fwpart_name : actual_size(dec) [(hex)]      /   zone_size(dec) [(hex)]
+-------------------------------------------------------------------------------+ 0x000001000000
|            FREE SPACE :                                 2719744 bytes [0x00298000] |
+-------------------------------------------------------------------------------+ 0x000000d68000
|    2ndfw  linuxKernel :    3698688 bytes [0x00387000] /    4128768 bytes [0x003f0000] |
|                                  Image.lzma.padding                           |
+-------------------------------------------------------------------------------+ 0x000000978000
|    2ndfw  audioKernel :     311296 bytes [0x0004c000] /     393216 bytes [0x00060000] |
|                            bluecore.audio.lzma.padding                        |
+-------------------------------------------------------------------------------+ 0x000000918000
|    2ndfw rescueRootFS :    1048576 bytes [0x00100000] /    1048576 bytes [0x00100000] |
|                          rescue_rootfs.cpio.gz.padding                        |
+-------------------------------------------------------------------------------+ 0x000000818000
|    2ndfw     rescueDT :      49152 bytes [0x0000c000] /      49152 bytes [0x0000c000] |
|                                  rescue.dtb.padding                           |
+-------------------------------------------------------------------------------+ 0x00000080c000
|    2ndfw     kernelDT :      49152 bytes [0x0000c000] /      49152 bytes [0x0000c000] |
|                                  normal.dtb.padding                           |
+-------------------------------------------------------------------------------+ 0x000000800000
|            FREE SPACE :                                  413696 bytes [0x00065000] |
+-------------------------------------------------------------------------------+ 0x00000079b000
|    1stfw  linuxKernel :    3698688 bytes [0x00387000] /    4128768 bytes [0x003f0000] |
|                                  Image.lzma.padding                           |
+-------------------------------------------------------------------------------+ 0x0000003ab000
|    1stfw  audioKernel :     311296 bytes [0x0004c000] /     393216 bytes [0x00060000] |
|                            bluecore.audio.lzma.padding                        |
+-------------------------------------------------------------------------------+ 0x00000034b000
|    1stfw rescueRootFS :    1048576 bytes [0x00100000] /    1048576 bytes [0x00100000] |
|                          rescue_rootfs.cpio.gz.padding                        |
+-------------------------------------------------------------------------------+ 0x00000024b000
|    1stfw     rescueDT :      49152 bytes [0x0000c000] /      49152 bytes [0x0000c000] |
|                                  rescue.dtb.padding                           |
+-------------------------------------------------------------------------------+ 0x00000023f000
|    1stfw     kernelDT :      49152 bytes [0x0000c000] /      49152 bytes [0x0000c000] |
|                                  normal.dtb.padding                           |
+-------------------------------------------------------------------------------+ 0x000000233000
```

Dual Firmware Boot

- There are 2 copies of firmware table and images (kernel, dtb, rescue rootfs…etc) installed in RTD129x/RTD1619
- Each firmware table has a sequence number (0-255)
- Boot loader will choose the latest (bigger) firmware table to bootup
- Once the latest firmware table or its content is corrupted, another firmware table will be chosen to bootup

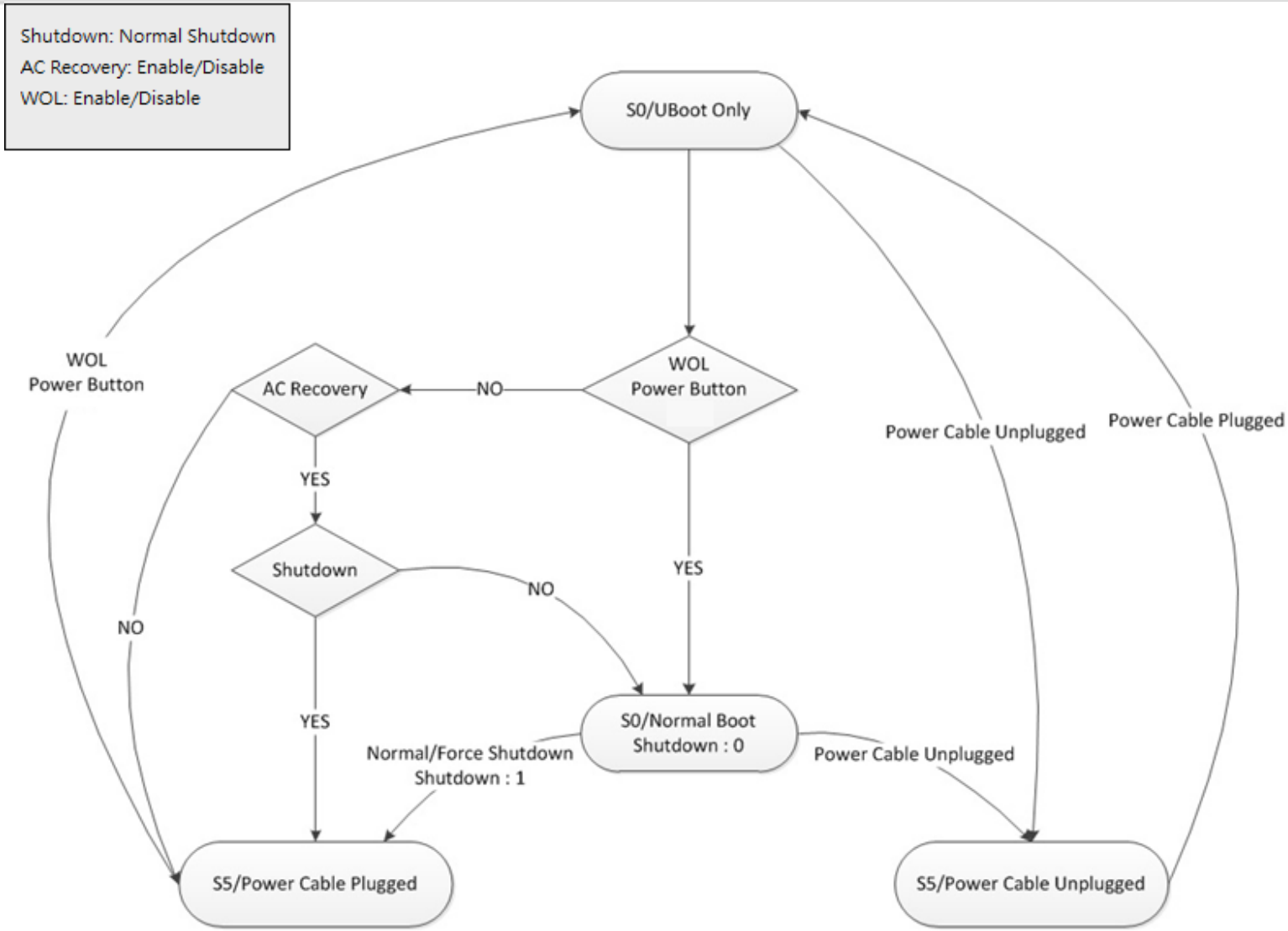
Customize kernel boot arguments

Path: arch/arm/lib/bootm.c

Function: set_custom_boot_args()

- This function provides room to append programmable boot arguments.
- For example, the path of boot device might be "/dev/mmcblk0p0" or "/dev/mmcblk0p1", depending on circumstances. This can be done by adding conditional statement in this function to control boot arguments.
- The parameter "custom_boot_args" of this function is combined with kernelargs, and finally become part of kernel boot arguments.

Customize Bootflow: AC-Recovery



Customize Bootflow: AC-Recovery

- Add Config
  - ./Bootcode/U-Boot64/configs/rtd161x_qa_nas_rtk_defconfig

    CONFIG_CUSTOMIZE_FEATURES=y

    CONFIG_CUSTOMIZE_BOOTFLOW_1=y

    CONFIG_CUSTOMIZE_BOOTFLOW_1_PHASE_1=y

  - ./Bootcode/U-Boot64/common/customized_bootflow_rtd161x.c

    #define PWR_KEY_IGPIO 2 (check you board before set)

    #define ACRECOVERY_PATH FACTORY_HEADER_FILE_NAME"ACRECOVERY"

    #define POWER_STAT_PATH FACTORY_HEADER_FILE_NAME"SHUTDOWN"

    #define WAKE_ON_LAN_PATH FACTORY_HEADER_FILE_NAME"WAKEONLAN”

  - ./rtd-1619-mmnas-megingjord-2GB.dts

    rtk_iso_gpio: rtk_iso_gpio@98007100 {

    wakeup-gpio-list = <&rtk_iso_gpio 2 0 1>; /*rtk_gpio gpio# direction value*/

    /*direction: 0=input, 1=output; when direction=0=input, set value is useless */

    wakeup-gpio-enable = <1>; /*0=disable, 1=enable*/

    wakeup-gpio-activity = <0>;/*0=active low, 1=active high*/

Customize Bootflow: AC-Recovery

- Flag name: ACRECOVERY, SHUTDOWN, WAKEONLAN
- User can set AC-recovery, Shutdown & WOL enable/disable by factory tool in user space.
  - Enable: factory flag flag-name
  - Disable: factory unflag flag-name
- As flowchart, shutdown flag will be set to 0(disable) in every normal boot.
- When WOL is enable, be noted that WOL flag of gMAC also needs to set enable before shutdown.
  - echo 1 > /proc/net/eth0/r8169/wol_enable

Customize reserved memory for transcoding

- The following u-boot environment variables can be used to configure the reserved memory size for transcoding.
  - `ion_media_heap0_size`
  - `ion_media_heap1_size`
- Usage
  - Set a positive integer for the size of ion media heap 0 or 1 (unit: MB)
    - If the value exceeds the max. allowable memory size, it will prompt error message and reset the heap size to default value when booting into kernel.

```
Realtek> env set ion_media_heap0_size 199
Realtek> env set ion_media_heap1_size 414
```

  - Set the size of ion media heap 0 & 1 to zero
    - Disable transcode
    - Free the reserved memory for transcoding
    - The board has to restart for this change to take effect.

```
Realtek> env set ion_media_heap0_size 0
Realtek> env set ion_media_heap1_size 0
```

Customize reserved memory for transcoding

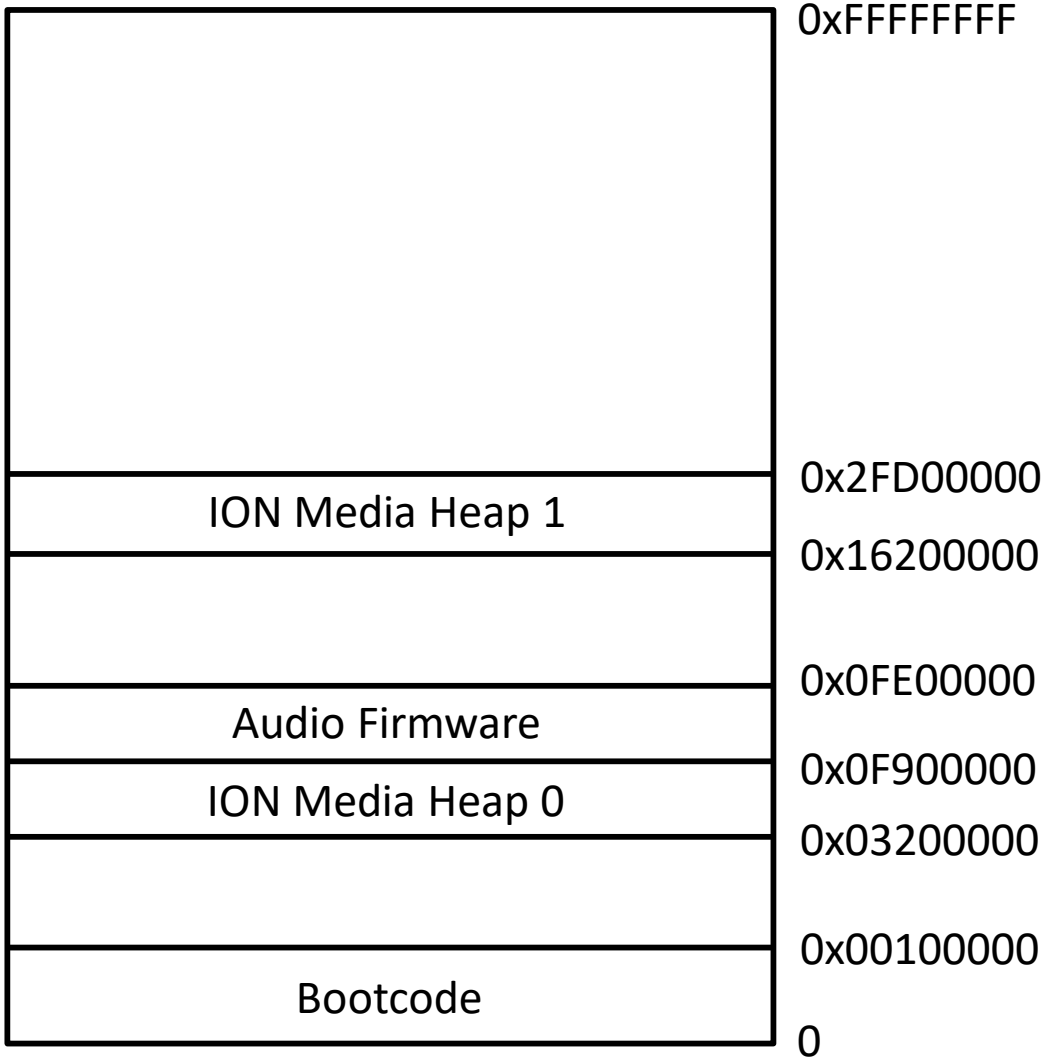
- Usage
  - If the board is configured as pure NAS function (that is, there is no reserved memory for transcoding), attempting to set `ion_media_heap0_size` or `ion_media_heap1_size` will prompt error message and abort.

```
Realtek> env set ion_media_heap0_size 10

Error: env ion_media_heap0_size is reserved for
adjusting the memory size of ion media heap
memreserve for ion media heap is not found.
```

  - To enable the transcode function again, delete the environment variable `ion_media_heap0_size` & `ion_media_heap1_size`.

Customize reserved memory for transcoding



Memory Layout