Realtek NAS SDK

Image-Builder

Driving IC Innovation

Image-Builder

- Build Realtek SoC Image file
- Prepare Files for Rescue System
- Customization
  - Specify Firmware Layout in Storage
    - Edit feed.conf to descript storage layout
  - Support Customer's Root Filesystem
    - Edit feed.conf to descript content of root partition
  - Easy to Set
    - Kernel Boot Argument
    - MAC Address
    - IP information for Bootcode
    - Edit file, bootargs.conf.{spi,emmc,nand}

install.img

- Realtek SoC Image file which contains
  - Bluecore.audio
  - Kernel Image
  - DTB for Normal Boot up
  - DTB for Rescue Boot up
  - Root Filesystem for Rescue System
  - Root Filesystem for Normal Boot up (if eMMc or NAND)
  - Layout description file, config.txt
  - Storage Writer, installer

Directories

- build_image.sh
  - Build Script
- arm_bin/
  - arm binary files for running in rescue system
- x86_bin/
  - X86 binary files for creating install.img

Directories

- rescue-rootfs/
  - Rescue Root File System.
  - Use initramfs.sh to compress/decompress rescue root filesystem image.
  - Size limit is 1 MB (1048576 bytes)
    - If exceeding the limit, change CONFIG_ROOTFS_RESCUE_SIZE in Bootcode's U-Boot64/include/configs/rtd161x_qa_{board}.h as in device tree
    - For example, define CONFIG_ROOTFS_RESCUE_SIZE as 0x200000 in rtd161x_qa_spi_64.h when device tree has initrd-start=<0x02200000> and initrd-end=<0x02400000> where initrd start and end are defined in kernel source's include/soc/realtek/memory.h as ROOTFS_RESCUE_START and ROOTFS_RESCUE_END respectively
- feed/
  - Gather Firmware Files
  - Edit feed.conf to descript storage layout
    - Three example files, feeds.conf.emmc, feeds.conf.spi, feeds.conf.nand

feed.conf Content

- storage
  - Define storage type
- storage_size
  - Define size of storage in unit of bytes
- storage_align
  - Define the alignment of storage block in unit of bytes
- storage_eraseblock_size (NAND only)
  - Define the size of erase block in unit of bytes
- storage_start_address
  - The start address of available are for storing firmware. DO NOT CHANGE!
- lzma
  - Use LZMA to compress bluecore.audio and kernel. Recommend for SPI

feed.conf Content

- bootargs
  - Insert U-Boot variables
  - Configuration File
    - bootargs.conf.emmc, bootargs.conf.spi, bootargs.conf.nand
- Content of bootargs.conf.{emmc,spi,nand}
  - ethaddr=00:10:20:30:40:50
  - gatewayip=192.168.100.254
  - ipaddr=192.168.100.1
  - netmask=255.255.255.0
  - SPI Kernel Arguments
    - kernelargs=mtdparts=RtkSFC:1024k(U-Boot)ro,64k(FWtbl)ro,128k(Factory),10944k(FW)ro,4096k(Free),128k(oops) init=/etc/init root=/dev/sda1 rootfstype=ext4 rootwait loglevel=8
  - eMMc Kernel Arguments
    - kernelargs=init=/etc/init root=/dev/mmcblk0p1 rootfstype=squashfs rootwait loglevel=8
  - NAND Kernel Arguments
    - kernelargs=init=/etc/init overlay=/dev/ubi1_0 overlayfs=ubifs rootwait loglevel=8
    - For NAND, build_image.sh script will generate necessary arguments (i.e., mtdparts=, ubi.mtd=, root=, and rootfstype=) according to feeds.conf, and append them to kernelargs

feed.conf Content/bootargs

- Content of bootargs.conf.{emmc,spi,nand}
  - ethaddr
    - Device MAC address
  - gatewayip ➔Bootcode ifconfig
  - ipaddr ➔Bootcode ifconfig
  - netmask ➔Bootcode ifconfig
  - Kernelargs
    - Part of kernel boot arguments
    - Kernel boot arguments is combined with kernelargs and bootargs in DTB files
      - Bootargs = bootargs in dtb + kernelargs
      - Kernelargs could be edited in bootcode and easy to change
        - Init=/et/init or /lib/systemd
        - root=/mmcblk0px
        - rootfstype=squashfs or ext4
        - loglevel

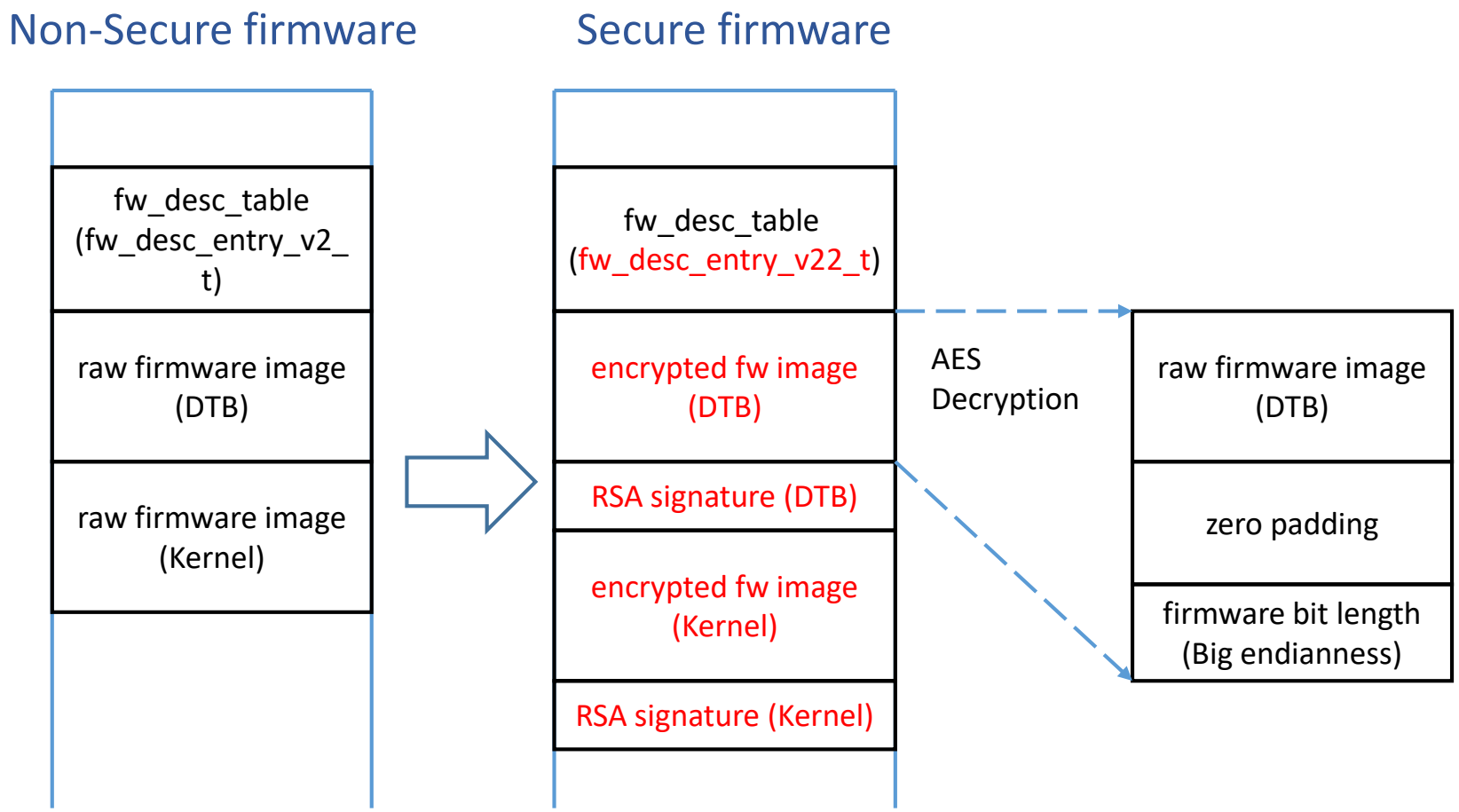
feed.conf Content

- secure_image=[y/n] (default: n)
  - Build secure image, only support EMMC now
- aes_key=[key file] (default: aes_128bit_key_dead.bin)
- rsa_private_key=[key file] (default: rsa_key_2048.fw.pem)
  - Key files are put in the feed/ directory

feed.conf Content/secure image

- All firmwares are AES-128-bit encrypted and appended with RSA-signed signature during build stage
- All firmwares are decrypted and check signature validation during booting stage
- AES 128-bit key:
  - should be burned into OTP in advance
  - Encrypt firmware using AES-128-ECB
- RSA private key:
  - 2048-bit, and the public key is burned in storage media with uboot and loaded to memory by FSBL before loading firmwares

feed.conf Content/secure image



feed.conf Content

- update_1stfw, update_2ndfw
  - Image-Builder Supports Dual Firmware Updating.
  - Set to ‘y’ to update the set of firmwares.
  - Set to ‘n’ to ignore the set of firmwares while burning the storage.
- seqnum_1stfw, seqnum_2ndfw
  - Firmware version
  - Bootcode compares sequnum to decide the latest firmware set. Bigger number is the latest firmware.

feed.conf Content

- kerneldtb_file
  - Filename of DTB for normal booting
- kerneldtb_zone
  - The size of a storage area for storing the kerneldtb_file
- kerneldtb_1stfw_addr, kerneldtb_2ndfw_addr
  - The start add of first/second kerneldtb_zone

ALL values MUST aligned by storage_align, in unit of bytes.

feed.conf Content

- rescuedtb_file
  - Filename of DTB for rescue system booting
- rescuedtb_zone
  - The size of a storage area for storing the rescuedtb_file
- rescuedtb_1stfw_addr, rescuedtb_2ndfw_addr
  - The start add of first/second rescuedtb_zone

ALL values MUST aligned by storage_align, in unit of bytes.

feed.conf Content

- rescuefs_file
  - Filename of initramfs image for rescue system booting
  - The example file is rescue-rootfs/rescue_rootfs.cpio.gz
- rescuefs_zone
  - The size of a storage area for storing the rescuedtb_file
- rescuefs_1stfw_addr, rescuefs_2ndfw_addr
  - The start add of first/second rescuefs_zone

ALL values MUST aligned by storage_align, in unit of bytes.

feed.conf Content

- bluecore_file
  - Filename of bluecore.audio
  - The example file is Packages/fw/bluecore.audio/bluecore.audio.zip
  - Unzip it first
- bluecore_zone
  - The size of a storage area for storing the bluecore_file
- bluecore_1stfw_addr , bluecore_2ndfw_addr
  - The start add of first/second bluecore_zone

ALL values MUST aligned by storage_align, in unit of bytes.

feed.conf Content

- kernel_file
  - Filename of kernel image
  - The file is kernel/arch/arm64/boot/Image
- kernel_zone
  - The size of a storage area for storing the kernel_file
- kernel_1stfw_addr , kernel_2ndfw_addr
  - The start add of first/second kernel_zone

ALL values MUST aligned by storage_align, in unit of bytes.

feed.conf Content

- bootlogo_file
  - Filename of boot logo image
  - For videoplayback configuration only
- bootlogo_zone
  - The size of a storage area for storing the kernel_file
- bootlogo_1stfw_addr , bootlogo_2ndfw_addr
  - The start address of bootlogo_zone
  - By default, there is only one bootlogo file will be placed in the storage. Therefore, addr. of bootlogo in two fw. entries are the same.
- For more information about customizing bootlogo, please read the readme file for boot logo.(readme.bootlogo.pdf)

ALL values MUST aligned by storage_align, in unit of bytes.

feed.conf Content

- bootpart_dir
  - Directory name of root filesystem
  - For eMMc and NAND only
  - Customer can build rootfs by themselves
- bootpart_type
  - For eMMc, support squashfs only; for NAND, support squashfs and ubifs
- bootpart_zone
  - The size of a storage area for storing the boot partition
- bootpart_addr
  - The start address of bootpart_zone
- bootpart_name and bootpart_ini (NAND only)
  - MTD partition name and configuration ini-file for ubinize, respectively

ALL values MUST aligned by storage_align, in unit of bytes.

feed.conf Content (EMMC and NAND only)

- normalpart_count
  - Count of normal partitions which decides how many normal partitions will be created (MAX=3)
- normalpartX_type (X=1 or 2 or 3)
  - normal partition could be treated as swap, overlay or general purpose partition according to this value (swap, overlay, or ext4 for eMMC; ubifs for NAND)
  - overlay: specify this overlay partition in kernelargs
- normalpartX_file
  - if normalpartX_type is ext4, this will be the filename of ext4 image which will be write on storage
- normalpartX_zone
  - Partition size of normalpartX
- normalpartX_addr
  - Starting address of normalpartX
- normalpartX_name, normalpartX_dir, normalpart1_ini (NAND only)
  - MTD partition name, the directory for UBIFS filesystem, and configuration ini-file for ubinize, respectively

ALL values MUST aligned by storage_align, in unit of bytes.

feed.conf Content (spi only)

- initramfs_file
  - File name of initramfs cpio compressed image
- initramfs_zone
  - The size of storage area for storing initramfs_file
- initramfs_addr
  - The starting address of initramfs_zone

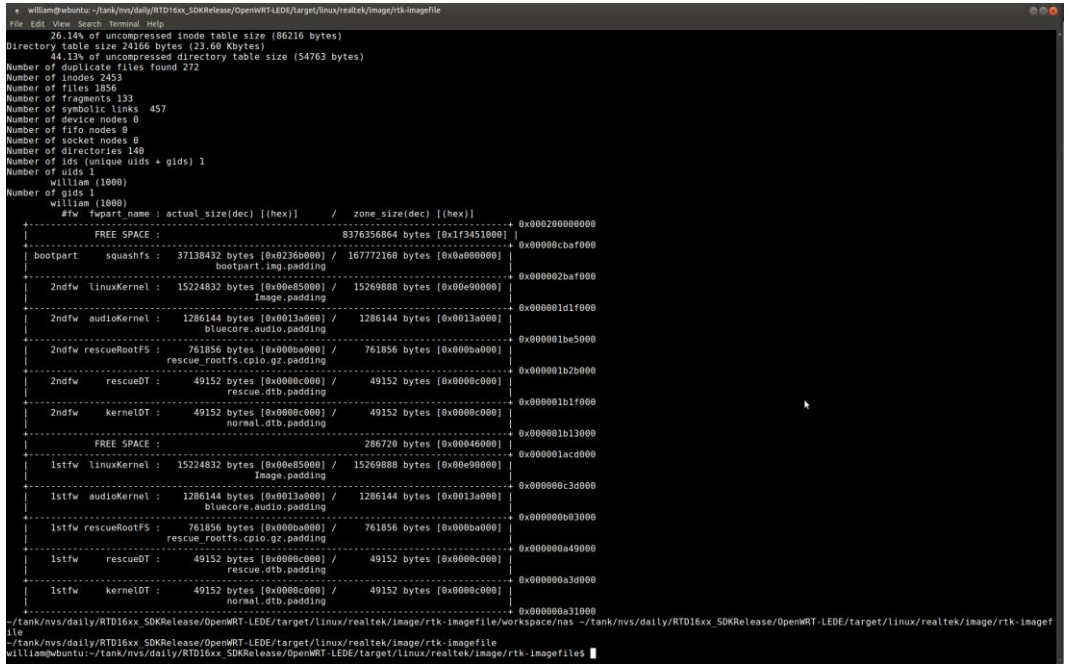
ALL values MUST aligned by storage_align, in unit of bytes.

How to Use Image-Builder

- Prepare Firmwares
  - Kernel
    - arch/arm64/boot/Image
  - Rescue System DTB
    - arch/arm64/boot/dts/realtek/rtd16xx/rtd-1619-nas-qa-rescue.dtb
  - Normal Opernation DTB
    - arch/arm64/boot/dts/realtek/rtd16xx/rtd-1619-nas-mjolnir-2GB.dtb
  - Bluecore.audio
    - Packages/fw/bluecore.audio/bluecore.audio.zip, unzip it
  - Rescue System Root Filesystem
    - Image-Builder/rescue-rootfs/rescue_rootfs.cpio.gz
  - Root Filesystem
    - OpenWRT-LEDE/build_dir/target-aarch64_cortex-a55_glibc/root-realtek
- Copy Firmwares to Image-Builder/feed

How to Use Image-Builder

- Edit feed/feed.conf
  - Reference files, feed.conf.{spi,emmc, nand}
- In Image-Builder
  - Run command./build-image.sh feed
  - X86/storage_layout/ layout-checker helps to check the position of each firmware zone.

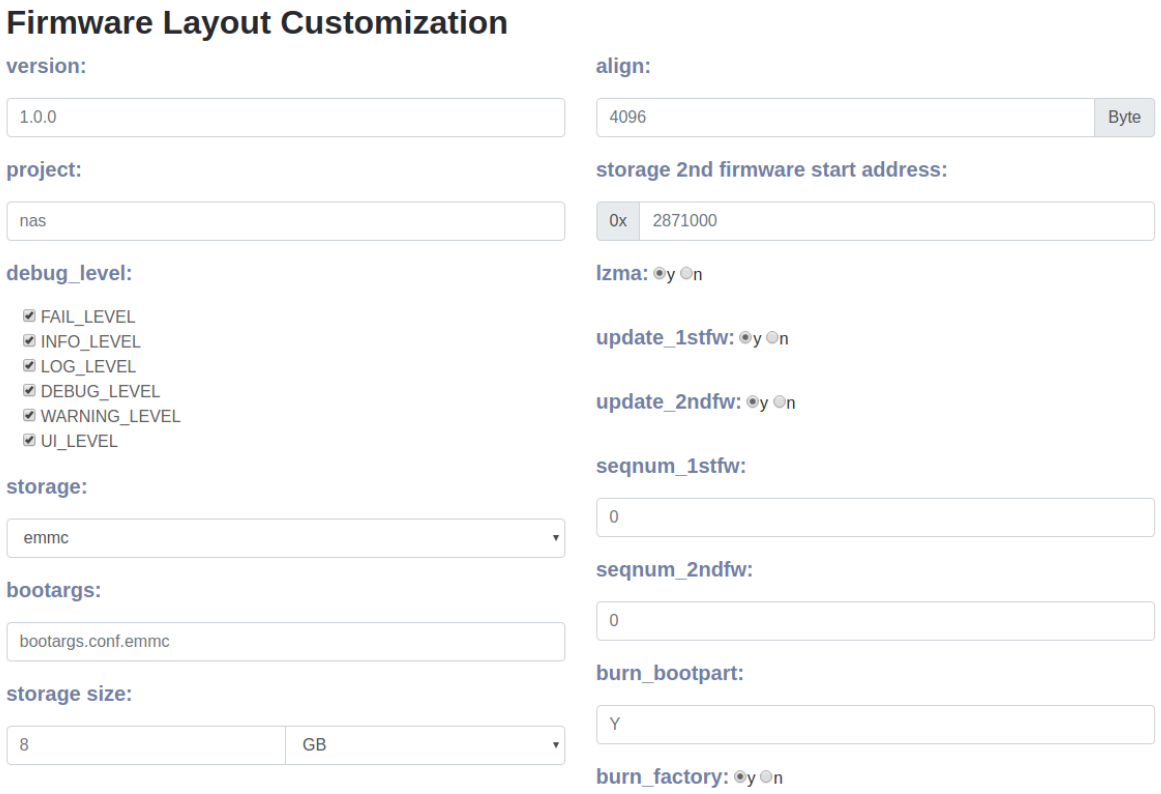


How to Use Image-Builder--Output

- Image-Builder/install.img
  - Rescue system use install.img to upgrade firmwares
- Image-Builder/workspace/rescue
  - Rescue system files
    - {spi,emmc,nand}.uImage
    - rescue.{spi,emmc,nand}.dtb
    - rescue.root.{spi,emmc,nand}.cpio.gz_pad.img

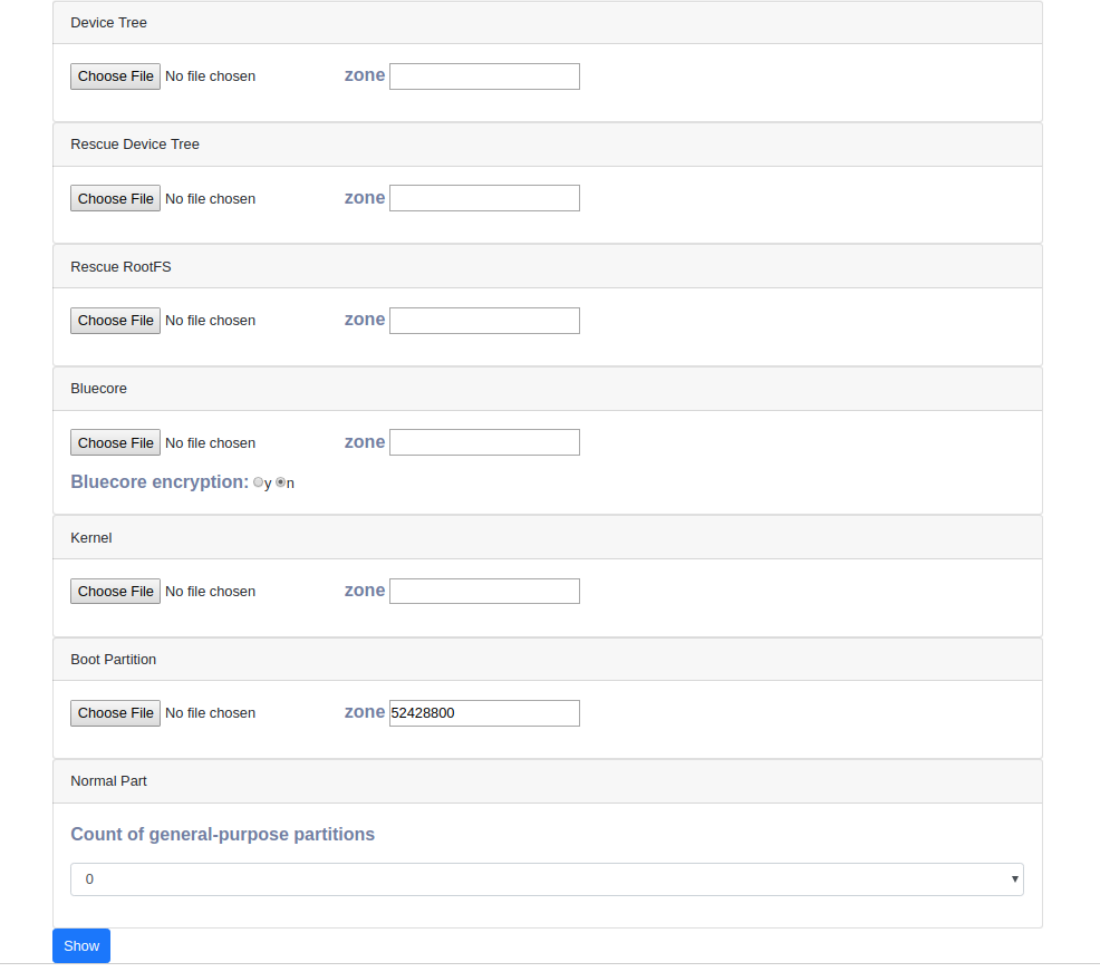
How to use WebUI to build config

- Open Image-Builder/StorageLayoutUI/index.html
- Choose the storage type
  - Emmc
  - Spi
- And corresponding settings
  - Bootargs
  - Storage size
  - Storage align
- Supports Dual Firmware Updating
  - Storage 2nd firmware address
  - Update firmware
  - Seqnum firmware



How to use WebUI to build config

- Choose six files
  - Device Tree
  - Rescue Device Tree
  - Rescue RootFS
  - Bluecore
  - Kernel
  - Boot Partition (only emmc)
- Zone MUST aligned by storage_align
- Customization Normal Part
  - Emmc: 0, 1, 2, 3
  - Spi: none



How to use WebUI to build config

- Click “OK”
  - You can review the layout
- click “submit”
  - It can produce feed.conf.{spi, emmc}
  - In /home/Downloads
- Copy feed.conf.{spi, emmc} to

  RTD16xx_SDK_Develop/Image-Builder/feed