ALL PROGRAMMABLE
ANY MEDIA
5G
4K/8K
ANY STANDARD
ANY MACHINE
ANY NETWORK
5G Wireless • SDN/NFV • Video/Vision • ADAS • Industrial IoT • Cloud Computing

OpenAMP Sync up with Linux Kernel

Linux Kernel Upstream

Linux Kernel Upstream - Remoteproc

- Remoteproc:
  - 4.9:
    - Add ability to boot/shutdown remoteproc with debugfs
    - Introduce autoboot: if it is on, probing remoteproc will not boot the remote
    - fw_rsc_vdev_vring: replace field “reserved” to “pa” in case predefined memory is used for vring. “pa” is not used in 4.9/4.10.
  - 4.10:
    - Introduce ADSP loader
    - Introduce subdevices
    - Anchor vring life cycle in vdev: decouple the vdev resource management from the virtio device management
    - Introduce sysfs to allow user to override firmware, boot/shutdown the remote from userspace
  - 4.11 or above
    - Allow a firmware without resource table

Linux Kernel Upstream - RPMsg

- RPMsg:
  - 4.9:
    - Enable rpmsg driver matching with device tree(DT)
    - Split rpmsg core and virtio backend. This allows using different rpmsg backend
  - 4.10:
    - Introduce a driver override mechanism: provide a mechanism to force a specific driver match on a device
    - rpmsg: Support drivers without primary endpoint: Some types of rpmsg drivers does not have a primary endpoint to tie their existence upon, but wishes to create and destroy endpoints dynamically, e.g. based on user interactions.
  - 4.11 or the above:
    - virtio_rpmsg_cfg to describe shared buffers
    - Char RPMsg device driver

OpenAMP - 2017.04/2017.10

- Resource table update to syncup with Linux kernel
- Decouple Remoteproc from RPMsg
  - Component structure decoupling:
    - Replace hil_proc with remoteproc devices and RPMsg devices
    - Move rpmsg vdev to remoteproc vdev
  - Flow decoupling:
    - Allow to call remoteproc API to do life cycle management without initializing RPMsg resources
    - Allow RPMsg communication with using Remoteproc for life cycle management
- RPMsg user APIs(create, send and callbacks) need to change from RPMsg channel based to RPMsg endpoint based.
- Linux userspace Remoteproc:
  - Allow remoteproc APIs implementation to use Linux kernel remoteproc for life cycle management and notification with remoteproc kernel driver sysfs APIs
- Linux userspace RPMsg:
  - Allow RPMsg APIs implementation to use kernel RPMsg device driver to create endpoings and send/receive messages
- Honor vdev status field of fw_rsc_vdev

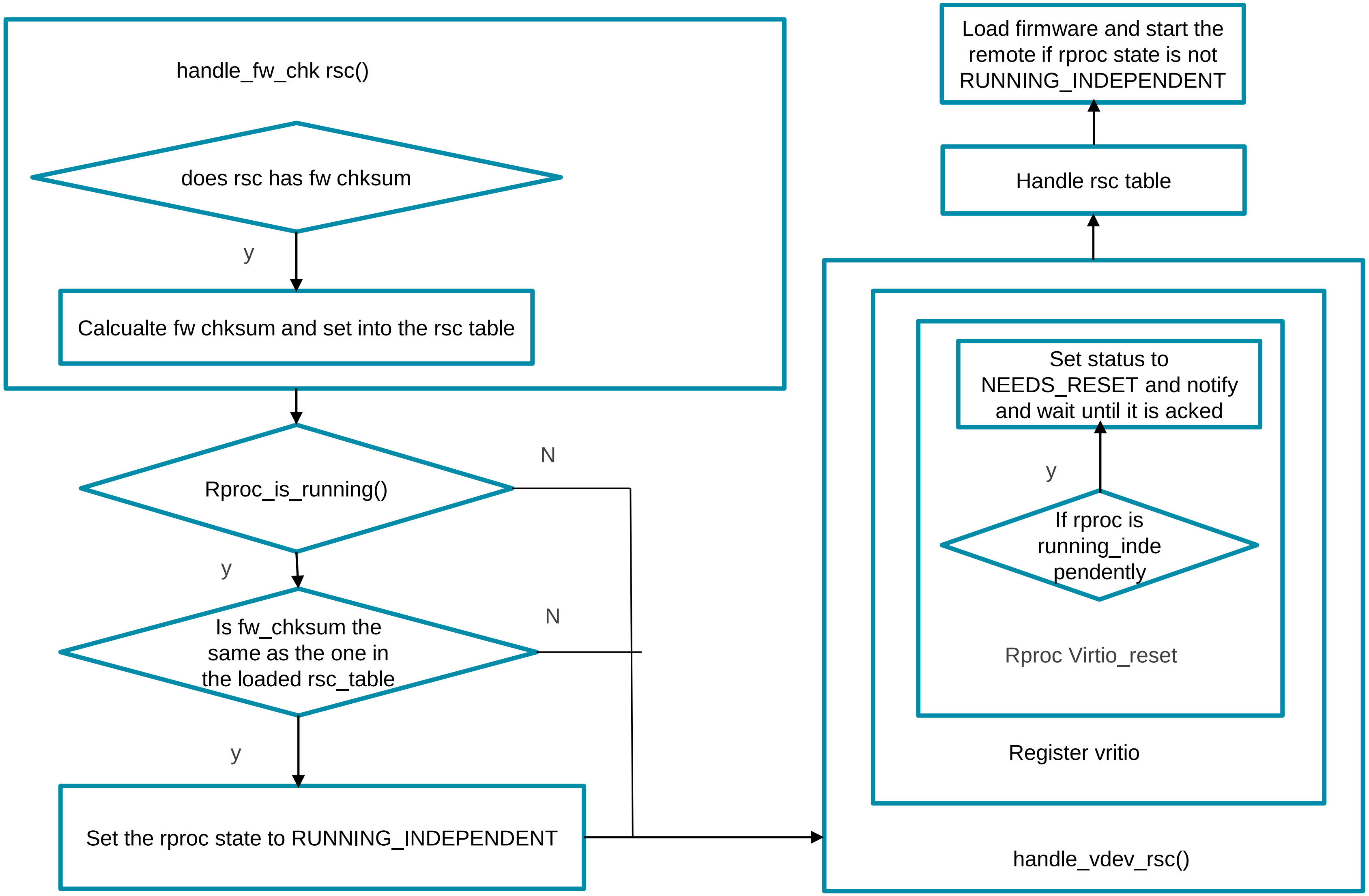
Open Questions

- How to describe predefined shared memory (e.g. for vrings and buffers:
  - Introduce a new resource type:
    - RSC_PROC_MEM
      - In the kernel side, it will declare it as DMA coherent memory. So that it can use the predefined memory when it calls dma_allocate_coherent() to allocate memory for vrings and shared buffers

Open Questions

- Reset, reattach, and recovery:
  - E.g. The remote is still rerunning while the master has restarted.
  - Use vdev reset
    - Remoteproc virtio will set the status to NEEDS_RESET and notify the remote in reset config op if the remote is already running before remoteproc restarts it. And it will wait for the remote to ack
    - OpenAMP will notify user app if it got kicked because it NEEDS_RESET.
    - OpenAMP will set the status to “0” and kick the master back if it sees NEEDS_RESET is set in the virtio status.
  - OpenAMP should honor vdev status changes:
    - OpenAMP should block initialization if vdev status is not driver ok
    - OpenAMP rpmsg operations should not fails if vdev status is not driver ok (auto cleanup?)
  - config ops need kick/wait synchronization
  - Check if the remote is running on the expected firmware on the master side
    - New resource type is introduced: RSC_FW_CHKSUM is used to store the firmware loadable sections checksum. This needs to increase the RSC version
    - New remoteproc driver ops is introduced: is_running()
    - Before if boots the remoteproc with the specified firmware, the remoteproc driver will check:
      - If the remote is running
      - If the checksum in the loaded firmware matches the checksum calculated
      - If RSC_FW_CHKSUM doesn’t exist in the resource table of the specified file, it will always reboot the remote

Open Questions- Reset, Reattach, and Recovery -- Master


handle_fw_chk rsc()
does rsc has fw chksum
y
Calcualte fw chksum and set into the rsc table
Rproc_is_running()
N
y
Is fw_chksum the same as the one in the loaded rsc_table
N
y
Set the rproc state to RUNNING_INDEPENDENT
Load firmware and start the remote if rproc state is not RUNNING_INDEPENDENT
Handle rsc table
Set status to NEEDS_RESET and notify and wait until it is acked
y
If rproc is running_independently
Rproc Virtio_reset
Register vritio
handle_vdev_rsc()

Open Questions

- Virtio config in resource table
  - – E.g. rpmsg: virtio_rpmsg_bus: fix sg_set_buf() when addr is not a valid kernel address
    - This is will be useful for the remote to know where the shared buffers are and memory mapped the region before it can be used if the shared buffers are from master's memory

Open Questions

- How to detect errors:
  - Virtio needs master/slave
  - We can introduce a heartbeat to resource table for master to monitor slave
  - If slave failed to update the heartbeat, rproc reports error.
  - But how about slave to monitor master?
    - User defined?
    - Another heartbeat in the resource table?
    - Or it doesn’t matter?
      - If shared memory are predefined, master can reset vdev when it initialize vdev.
      - What about shared memory is dynamically allocated by master?
  - Other Options:
    - The other side also raise virtio reset
      - Update in the virtio specification? Need a new status flag or field?
    - User defined solution on top of RPMsg/remtoeproc

Open Questions

- How to expose vdev to userspace
  - This is to enable userspace direct access to vdev devices for performance.
  - Each vdev is a vfio device
  - Map vdev resource and vrings to vfio regions
  - Map vdev notification and vrings notifications to vfios irqs
  - Write to the vfio regions will trigger notifications

Open Questions

- How to have communications between the two slaves
  - Shared memory is allocated form the master (.e.g Linux)
  - How can the two slaves talk to each other?
    - Any existing framework?
    - Use existing remoteproc framework?
    - Create resource table on the master side and pass it to the slaves?
    - What is used between the two slaves? Any peer-to-peer solution?

Open Questions

- How to authenticate firmware from Linux?