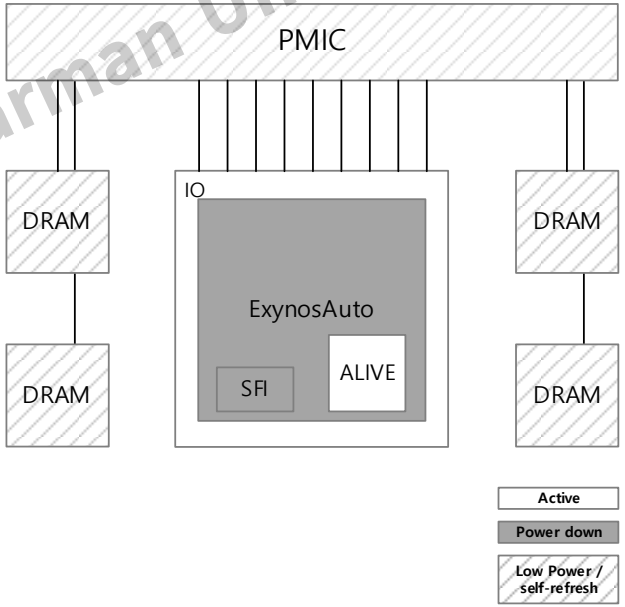
Suspend to RAM Virtualization Architecture ExynosautoV9

2022.04.13 | Samsung

System power mode of ExynosAuto

□ Sleep mode

- ExynosAuto supports system power mode called sleep, which is hardware feature that supports the Suspend to RAM.
- In sleep mode, all clocks except RTC are deactivated.
- In sleep mode, DRAM is in self-refresh mode to retain the contents.
- In sleep mode, the following powers are required.
  - The power required to wake up again.
  - The power that maintains the data stored in DRAM
  - The power that maintains the IO(Input/Output) level



<Power state of hardware components in sleep mode>

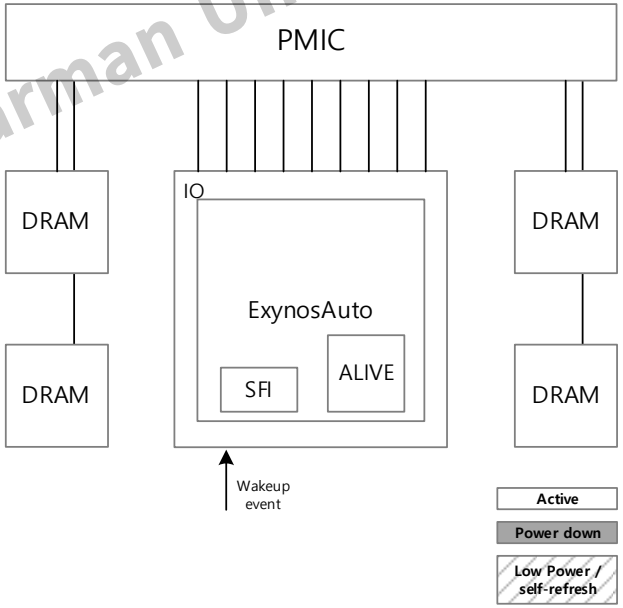
System power mode of ExynosAuto

□ Wake-up from sleep mode

– IOs in ALIVE block can be used as a wake-up source.

□ Wake-up sequence

– Wake-up event on the external IO
– Turns on regulators and clock starts to run
– Hardware logic in ALIVE prepares the SOC to operate
– DRAM switches to normal mode
– ALIVE block releases reset of the primary CPU
– The SOC is restored to the previous state by software sequence.



<Power state of hardware components in normal mode>

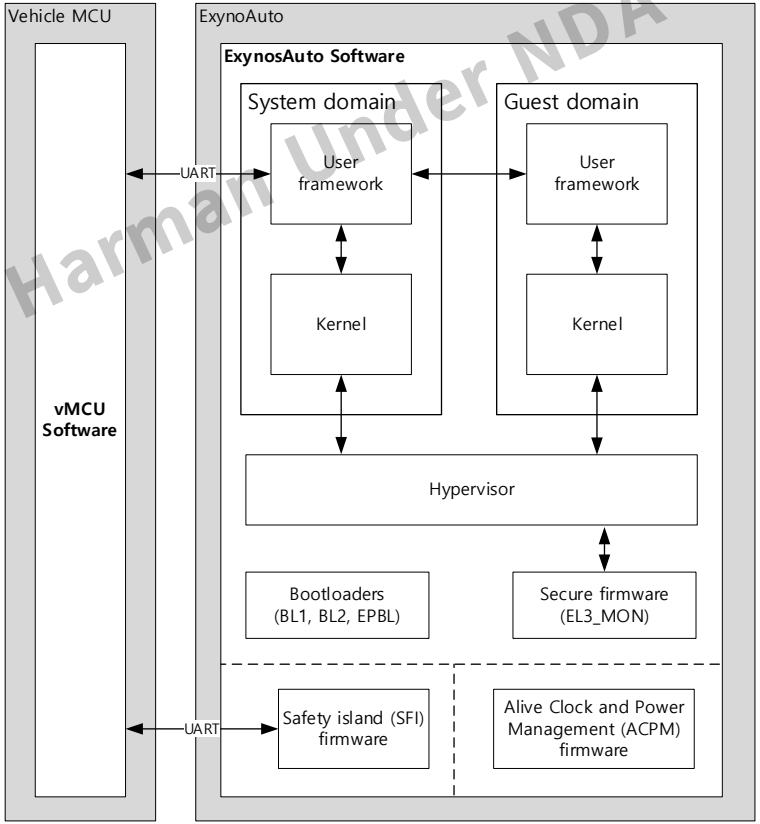
Software Architecture

□ Software components

Software component	Role
Vehicle MCU S/W	Manage power state of Exynosauto. Emulated by PC on the reference board.
User framework	Handshake between each VM. Triggers S2R on linux kernel or VHAL of Android framework
Kernel	PM framework of kernel manages power state of OS
Secure firmware	PSCI (Power state Coordination Interface)
ACPM firmware	Hardware control (PMU/CMU)

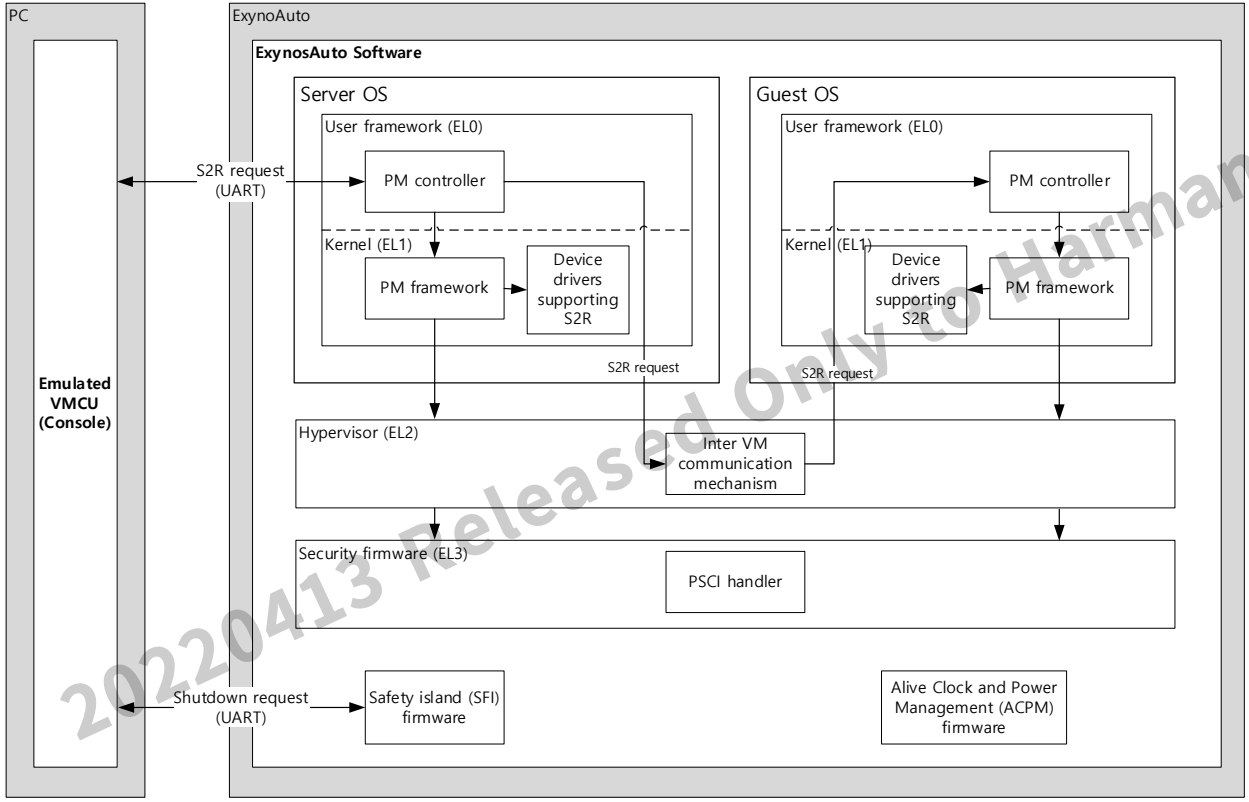
□ Condition

- Guest domain suspends and resumes independently if System domain is alive.
- System domain should start to suspend after All guest domain completes to suspend.
- Safety Island should be in shutdown state before System domain starts to suspend.



Software Architecture

□ Overall structure of software component for S2R

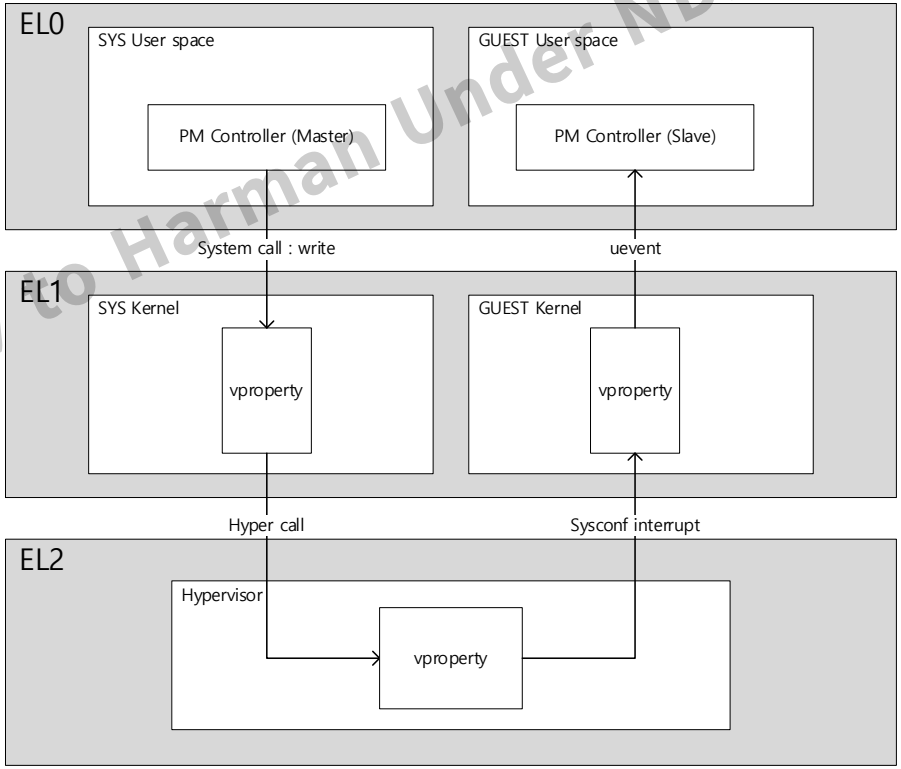


Software Architecture

□ vproperty Mechanism for inter-VM communication

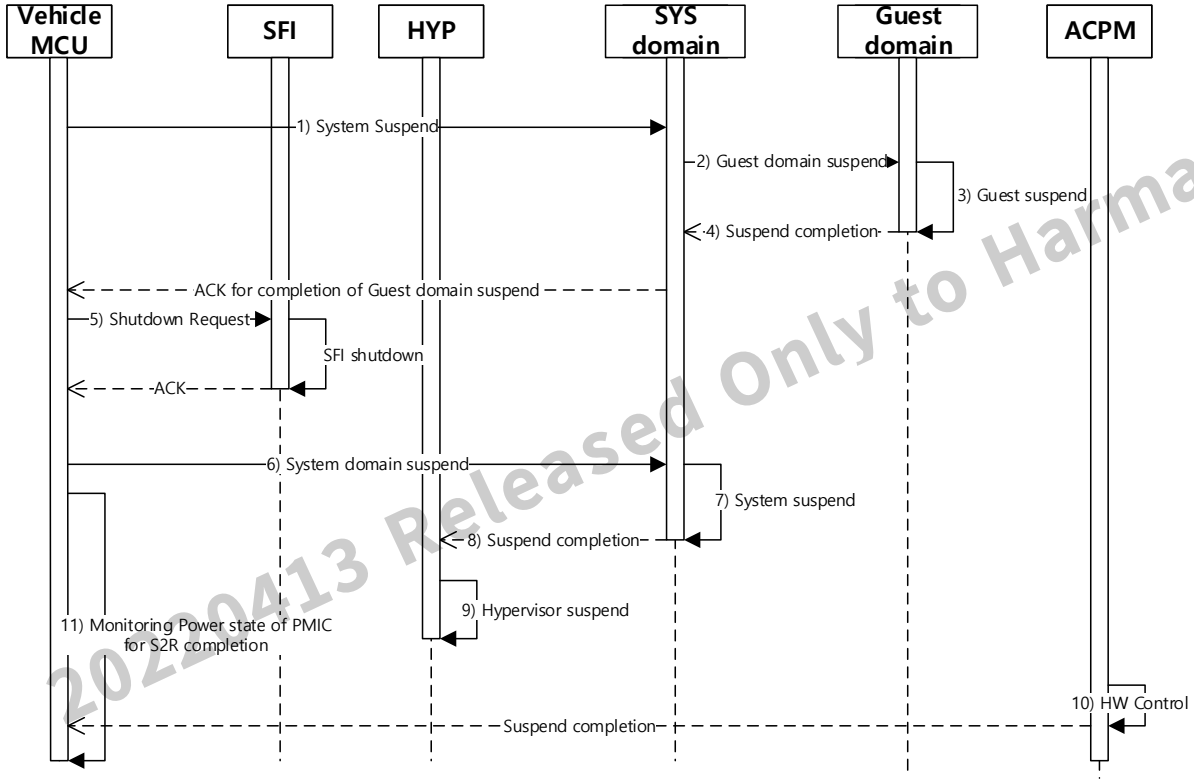
– s2r property of guest VM
  - /sys/nk/prop/nk.vm.<id>.s2r
  - When the guest VM suspends
    - The system VM writes this node and the guest VM gets the uevent
  - When the guest VM resumes
    - The system VM writes this node and the hypervisor wakes the guest VM up.

– Online property
  - /sys/nk/prop/nk.vms.online
  - When the guest VM suspends
    - Hypervisor update this value to 0 and the system domain checks if all guest VMs are offline and it makes itself suspend.



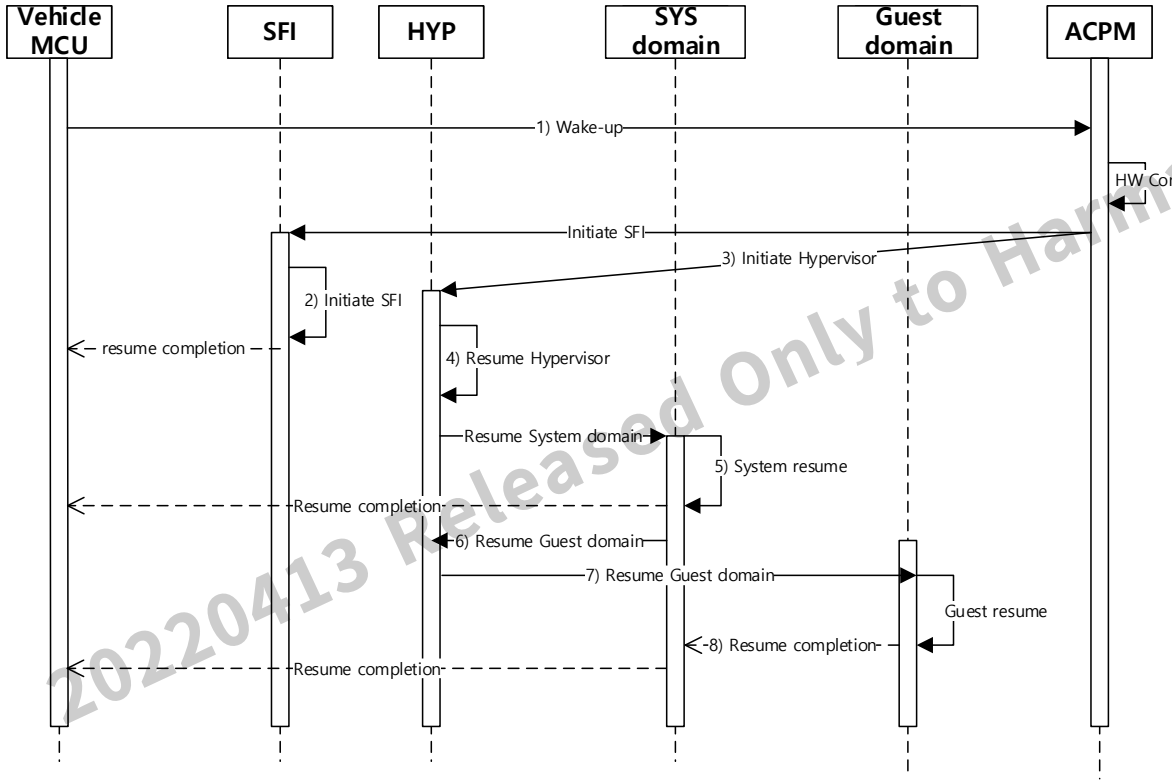
Software Architecture

□ Sequence diagram to suspend



Software Architecture

□ Sequence diagram to resume

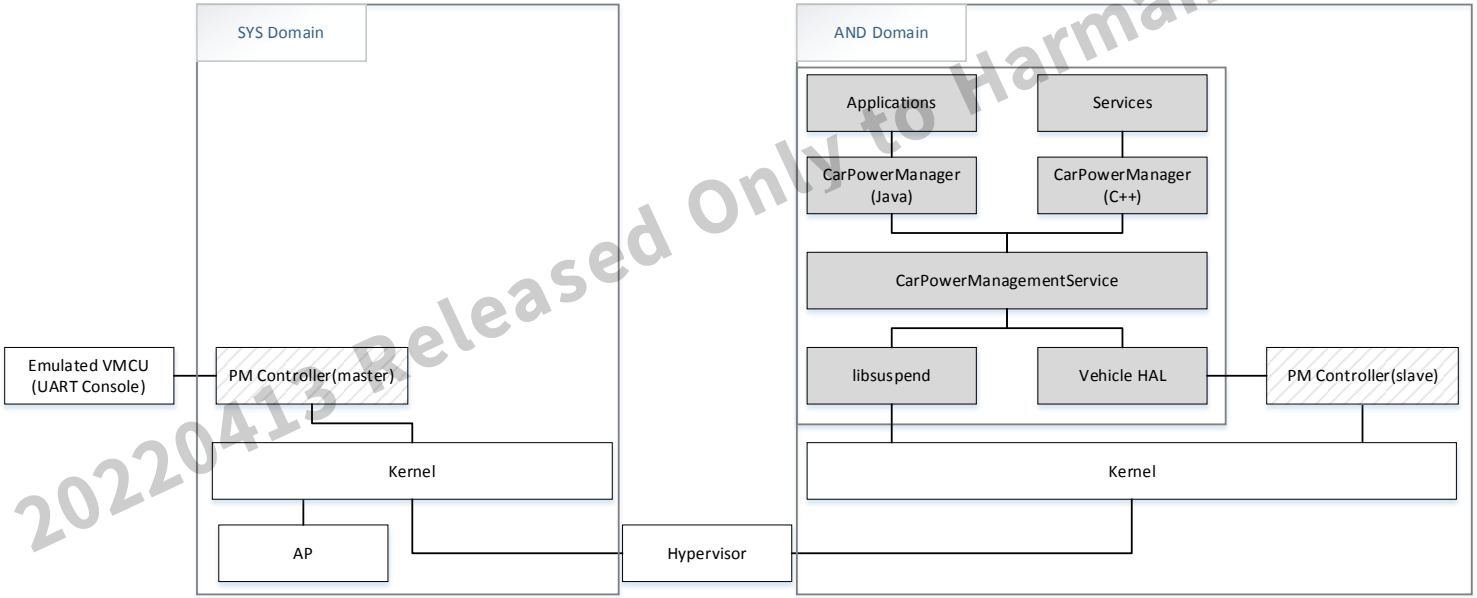


Software Architecture

□ Suspend to RAM in Android Automotive

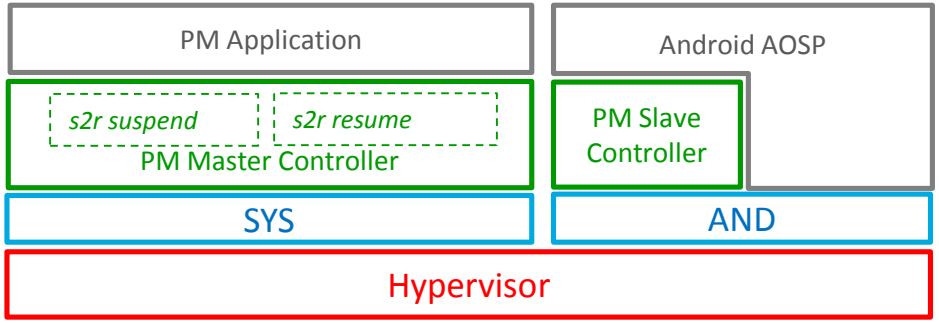
– PM controller in Android VM does not request S2R to kernel directly but it just send “SHUTDOWN_PREPARE” to Vehicle HAL.

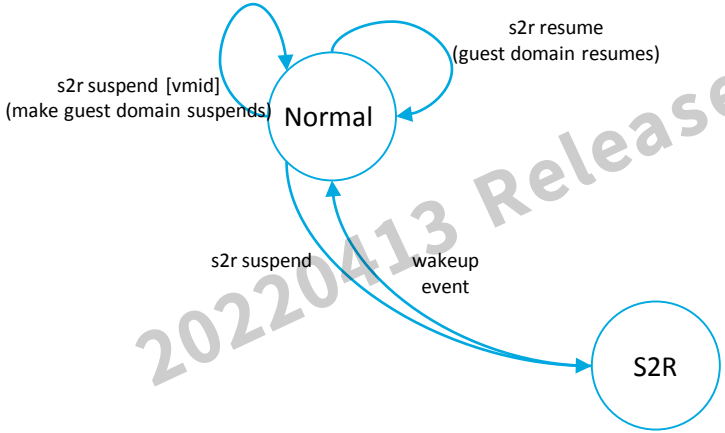
– libsuspend requests kernel to suspend after Android framework completes to prepare S2R.



APPENDIX

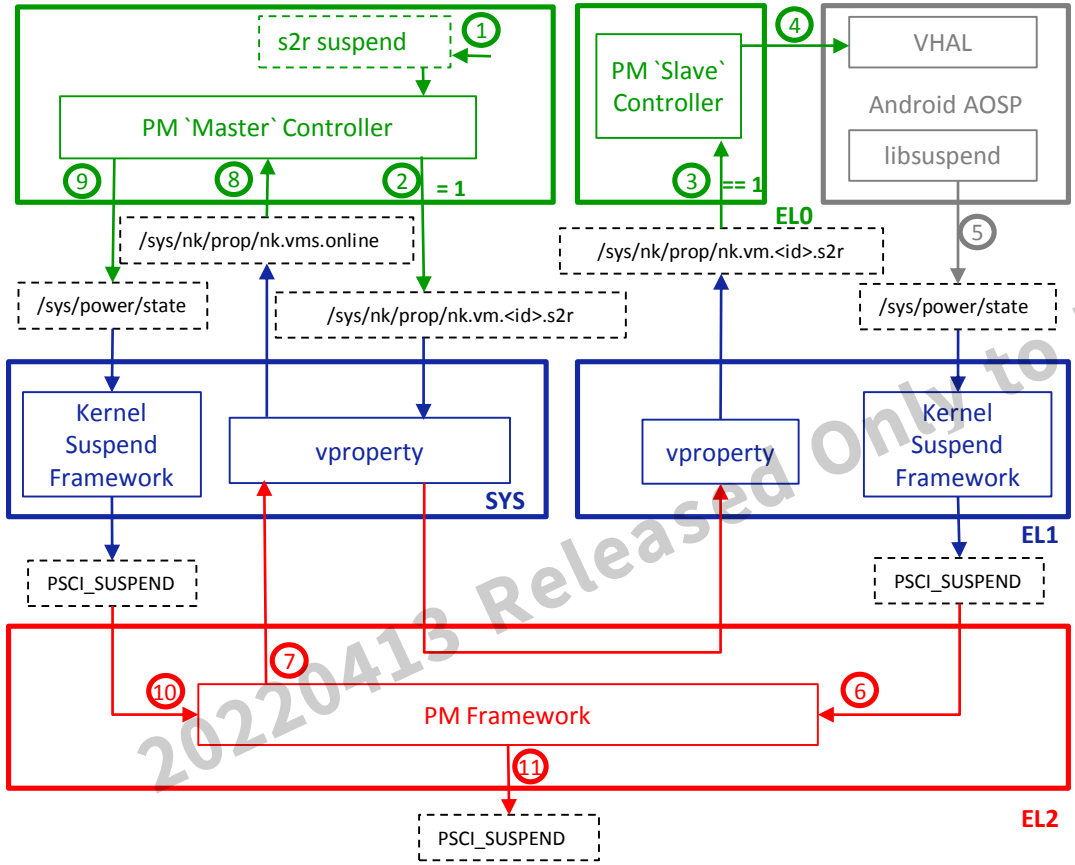
PM States and state transitions





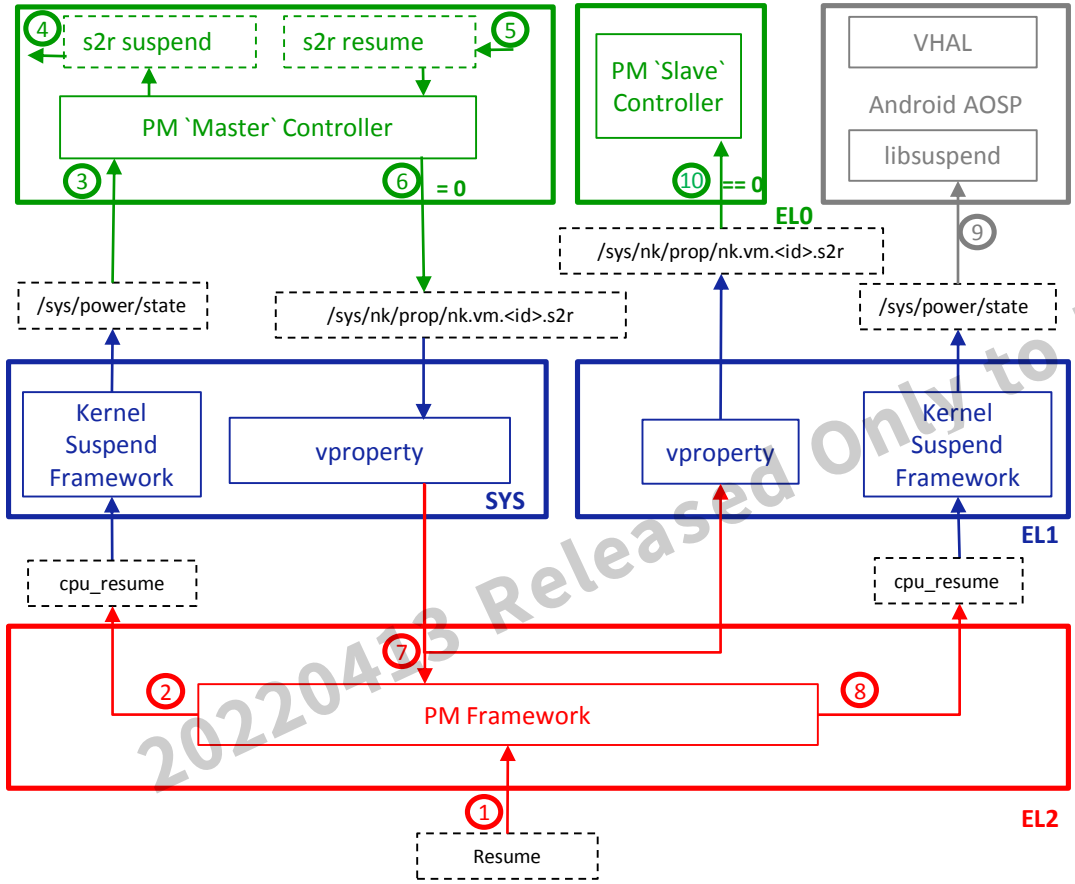
- There are two system-wide PM states: Normal and S2R.
- At any moment of time the system either resides in one of the above PM states or is transitioning from one PM state to another.
- Transitions between PM states are triggered by a SYS domain user-land PM Application (e.g. OOC) and hardware wakeup events.
- The PM Application triggers transitions using PM Commands: `s2r suspend` and `s2r resume`.
- The PM Commands implemented by a distributed PM Controller which the Master component is deployed on the SYS domain and the Slave components on all other domains.
- The PM Controller implements the PM Commands leveraging a set of the GuestOS and the HV PM Features.
- `s2r suspend` command triggers transition from the Normal state to the S2R state.
- The system enters in the S2R state if `s2r suspend` succeeds; otherwise the system remains in the Normal state.
- A transition from the S2R state to the Normal state is triggered by a hardware wakeup event.

Entering in the S2R state



1. A PM application calls the `s2r suspend` command implemented by the PM Master Controller
2. The Master Controller starts a transition to the S2R state by writing '1' into all the `nk.vm.<id>.s2r` HV vproperties.
3. A Slave Controller detects the `nk.vm.<id>.s2r` value update after getting uevent.
4. The Slave Controller send a SHUTDOWN_PREPARE message to VHAL of Android AOSP to execute S2R.
5. After preparation of Android AOSP is completed, it triggers its domain transition into S2R state by writing 'mem' in '/sys/power/state'
6. The Linux kernel Suspend Framework executes the PSCI_SUSPEND SMC which is intercepted by the HV PM Framework.
7. HV PM Framework sets the slave VM offline and updates `nk.vms.online` vproperty.
8. The Master Controller detects the 'nk.vms.online' vproperty value update (*)
9. When all slave VMs are offline (suspended) the Master Controller triggers the SYS domain transition in the S2R state writing `mem` in `/sys/power/state`.
10. The Linux kernel Suspend Framework executes the PSCI_SUSPEND SMC which is intercepted by the HV PM Framework.
11. HV PM Framework sets the SYS VM offline and performs the final PSCI_SUSPEND SMC.

Resuming from the S2R state



1. A wakeup event resumes HV.
2. HV PM Framework resumes the SYS VM.
3. The Master Controller returns from `/sys/power/state` write.
4. The `s2r suspend` command returns successfully.
5. The `s2r resume` command is used to resume the other guest OSes.
6. The Master Controller wakeups all Slaves writing `0` in their 'nk.vm.<id>.s2r' HV vproperties.
7. HV notifies all Slaves that the vproperty value has been updated. The notification is a wakeup event.
8. HV PM Framework resumes all Slaves.
9. Kernel PM framework returns to Android libsuspend and Android proceeds its own procedure to resume.
10. The Slave Controller may observe that its own `nk.vm.<id>.s2r` property value is `0`.

Reference Implementation

- **The PM Master Controller is implemented by the command (`s2r`) which are provided as Python scripts provided by the `meta-vl` Yocto layer.**
  - meta-vl/recipes-support/s2r/files/s2r

- **The PM Slave Controller is provided as a service with implementation:**
  - Android VM, as a C++ daemon application provided as part of device specific extensions.

    See device/samsung/exynosauto/s2rd for ExynosAuto9 based boards.