Remote Core Power Management in AMP Systems

2025-08-25

Akashdeep Kaur
Hari Nagalla
Kendall Willis

Texas Instruments

Disclaimers

- This is a technology presentation, not product-readiness or roadmap commitment
- Opinions presented here are that of the speakers and may not reflect that of Texas Instruments Inc.

About us: TI Processors and Open Source

Upstream FIRST!

- ✔ Upstream FIRST mentality!
- ✔ Decades of contribution and collaboration
- ✔ Focus on long term, sustainable and quality products
- ✔ Upstream & opensource ecosystem in device architecture
- ✔ Ingrained culture to give back to the community

Introduction to the Speakers

Akashdeep Kaur

(Software Engineer at Texas Instruments)

- Low Power management on Sitara SOCs
- Security and device management firmwares, U-boot, TF-A

Hari Nagalla

(Software Engineer at Texas Instruments)

- U-boot, TF-A, Linux device driver development on Sitara SoCs

Kendall Willis

(Software Engineer at Texas Instruments)

- Power management on Sitara SoCs
- U-boot, TF-A, Linux device driver development

Agenda

Introduction and Problem Statement

Device Driver Power Management Options

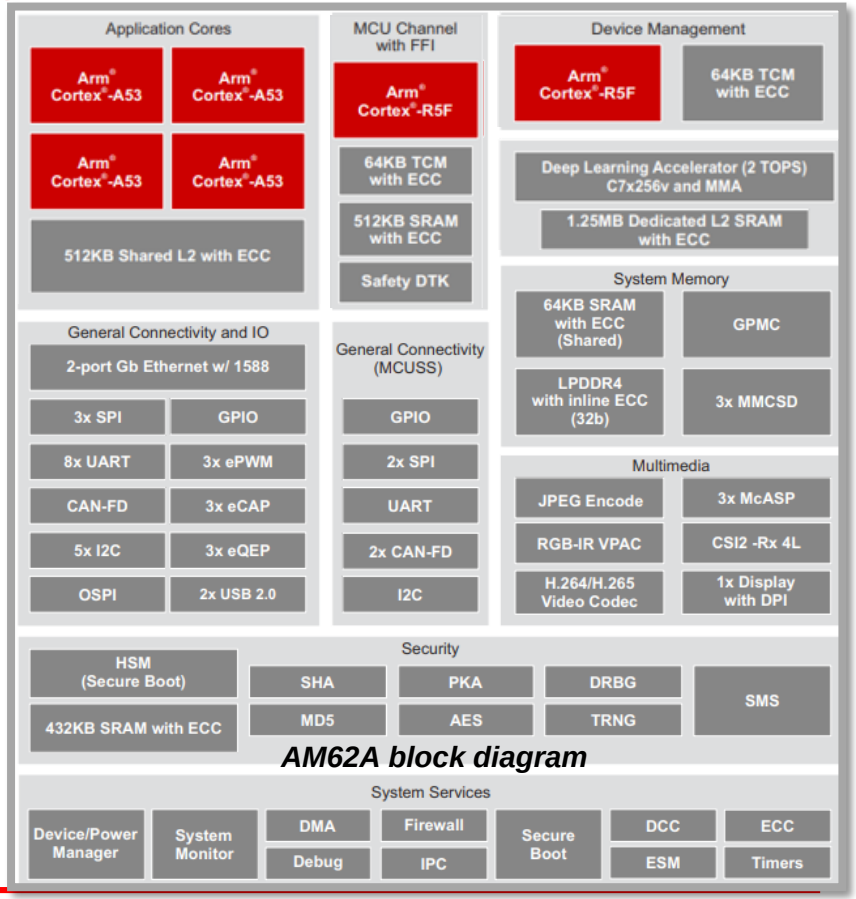
Remoteproc Suspend Resume Implementation

Additional Challenges

Introduction and Problem Statement

Asymmetric Multi Processing (AMP) Systems

- Designed to run diverse tasks, such as real-time operations alongside general-purpose workloads, by employing different core types.
- These systems include:
  - A "host" core typically runs a rich OS, like Linux, and “Remote" cores
- Example TI’s AM62A
  - ARM core cluster running Linux (A53)
  - ARM Realtime Cores (cortex R5F)
    - Generic (Device Management Operations) “DM”
    - Application Specific (e.g. RTOS based application)
  - DSP cores (Signal processing)
- Remote Processor Framework (remoteproc) on Linux manages lifecycles of these remote cores within a SoC including their power management



Problem Statement

How does remoteproc handle multiple Low Power mode requirement in AMP System?

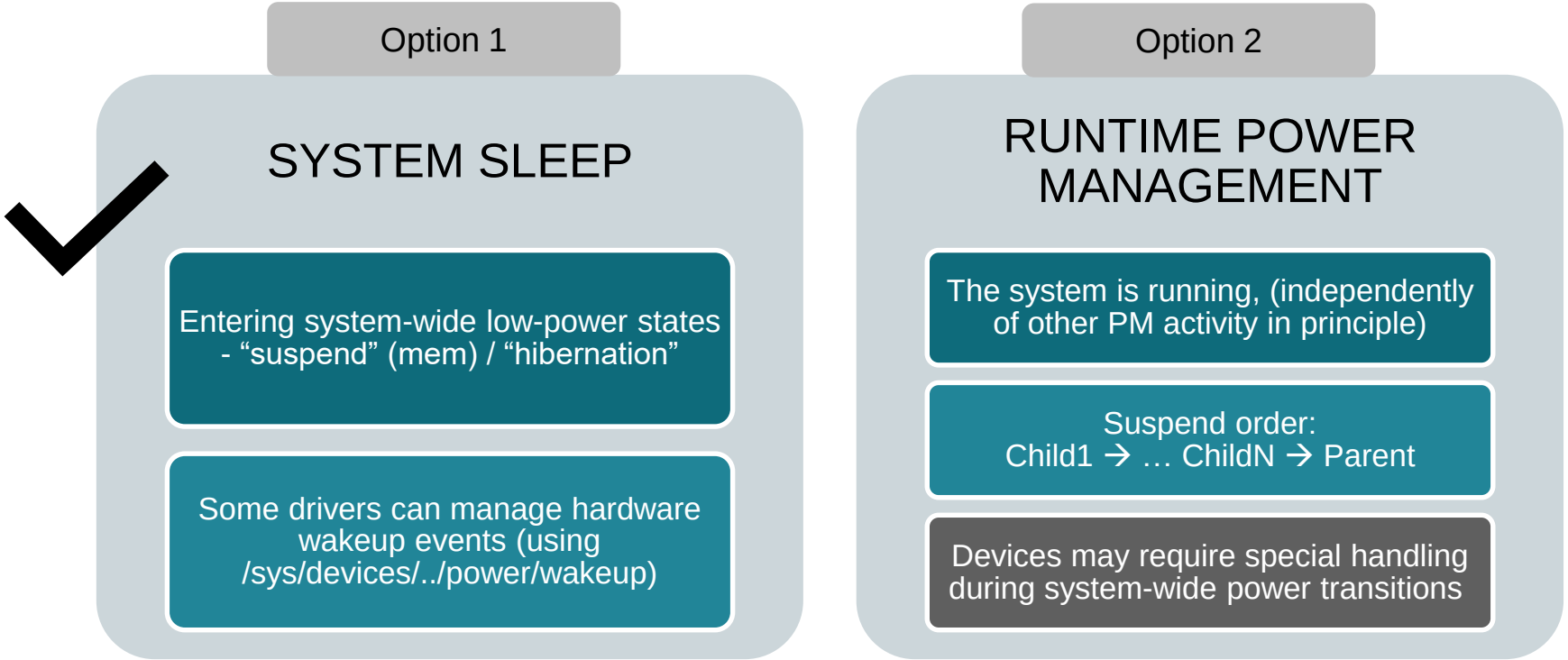
- Linux supports four system sleep states – Hibernation, Suspend-to-idle, Suspend-to-RAM and standby.
- Some SoC(e.g. TI's AM62A) supports multiple flavors of **Suspend-to-RAM** modes.
- How will ***remoteproc*** driver decide on what needs to be done in suspend and resume of each low power mode?

AM62A Modes	MCU R5 STATE	DSP STATE	DM* R5 STATE
DeepSleep	OFF	OFF	WFI
MCU Only	RUNNING	OFF	WFI
I/O Only Plus DDR	OFF	OFF	OFF

** – Device Management core*

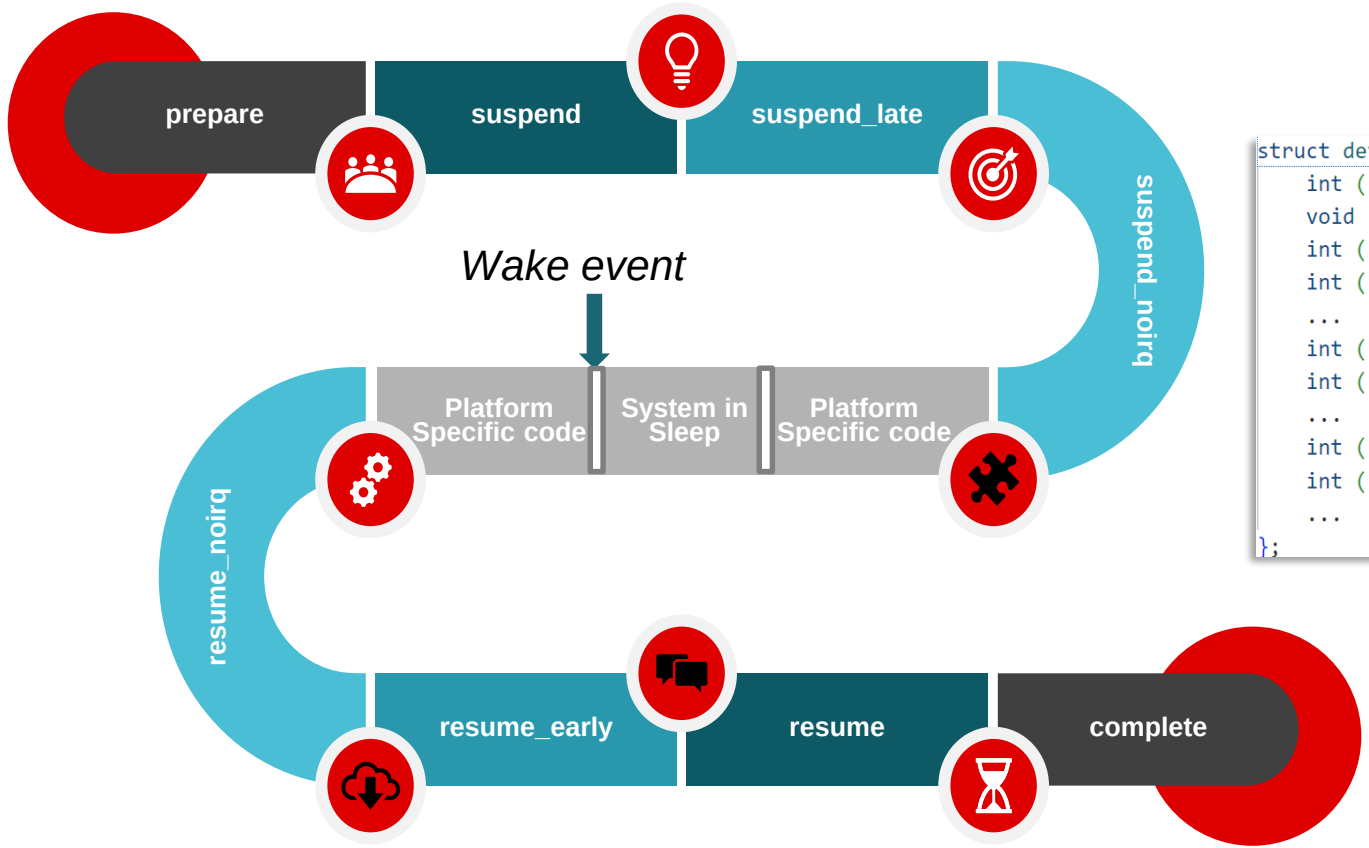
Device Driver Power Management Options

Models to put devices into low-power states



Kernel Power Management , *Runtime Power Management*

System Power Management Phases: PM Domains FW




```
struct dev_pm_ops {
    int (*prepare)(struct device *dev);
    void (*complete)(struct device *dev);
    int (*suspend)(struct device *dev);
    int (*resume)(struct device *dev);
    ...
    int (*suspend_late)(struct device *dev);
    int (*resume_early)(struct device *dev);
    ...
    int (*suspend_noirq)(struct device *dev);
    int (*resume_noirq)(struct device *dev);
    ...
};
```

```
struct platform_driver {
    struct device_driver {
        struct dev_pm_ops *pm;
    }
}
```

Power Management Notifiers - Suspend

- Some operations have to be carried out too late or too early, so the PM callbacks can't be used
- They have to be carried out only after the system is fully functional
- The operation requires user space to be available, or at least won't interfere with user space

Subsystems or drivers having such needs – Can register suspend notifiers that will be called upon the following events by the PM core:

PM_SUSPEND_PREPARE

The system is preparing for suspend

PM_POST_SUSPEND

The system has just resumed or an error occurred during suspend.

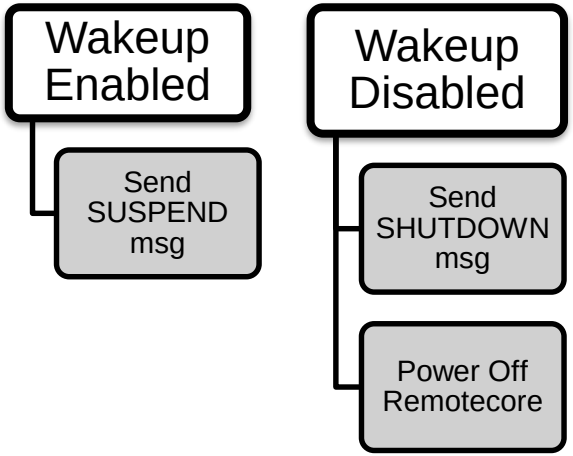
CALLBACK REGISTRATION: *Need to unregister - {register_pm_notifier, unregister_pm_notifier}* *else* *pm_notifier*

Remoteproc Suspend Resume Implementation

Texas Instruments

Option1 – Is remotecore's wakeup enabled?

echo enabled > /sys/bus/platform/devices//power/wakeup*



– Challenges:
  - **Linux Dictated**
  - Remotecore only able to differentiate between two low power states

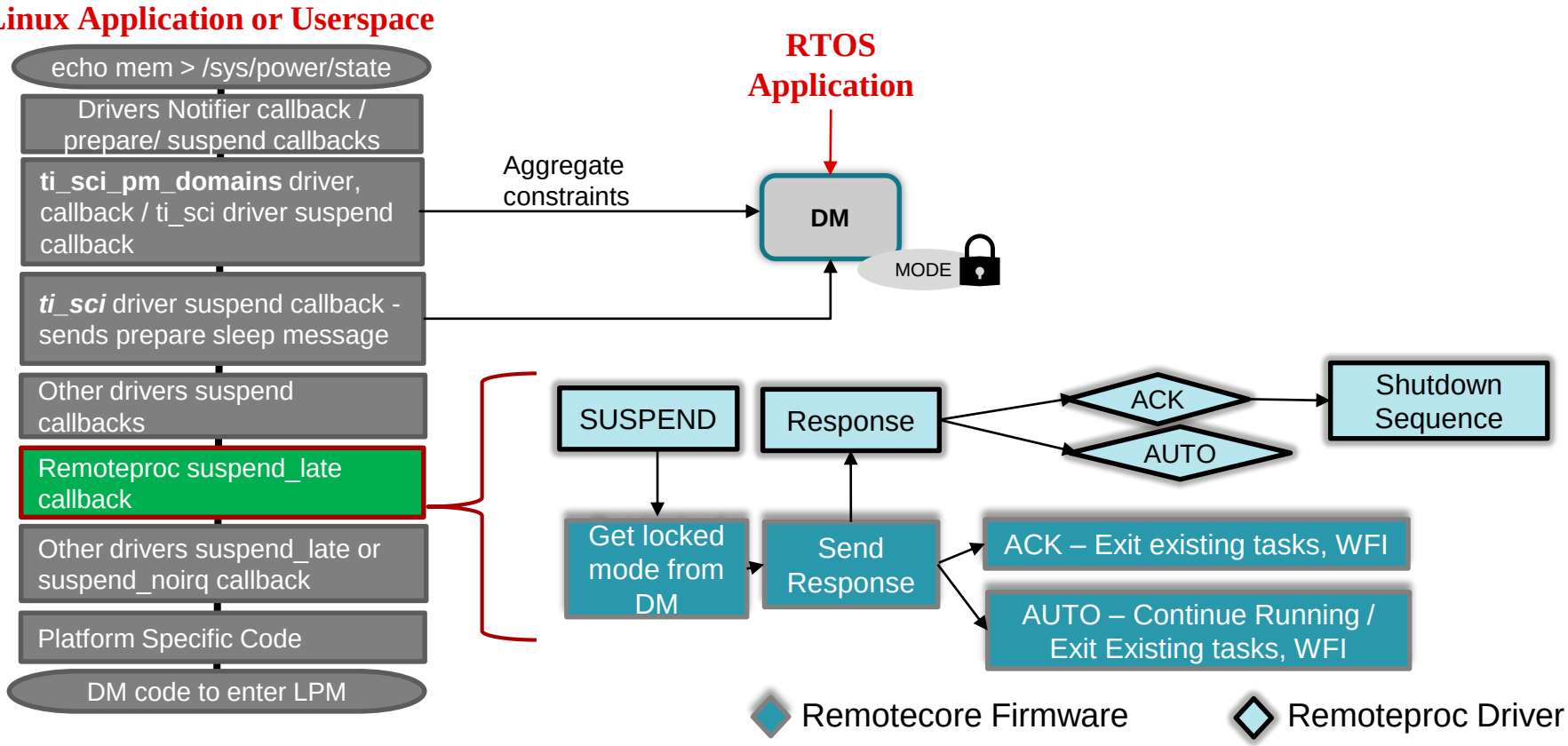
```
case PM_SUSPEND_PREPARE:
    if (!device_may_wakeup(kproc->dev)) {
        rproc_shutdown(kproc->rproc);
        ret = kproc->core->ti_sci->ops.dev_ops.put_device(kproc->core->ti_sci,
                                                          kproc->core->ti_sci_id);
        if (ret) {
            dev_err(kproc->core->dev, "module-reset assert failed, ret = %d\n",
                    ret);
            if (reset_control_deassert(kproc->core->reset))
                dev_warn(kproc->core->dev, "local-reset deassert back failed\n");
        }

    } else {
        reinit_completion(&kproc->suspend_comp);
        ret = mbox_send_message(kproc->mbox, (void *)msg);
        if (ret < 0) {
            dev_err(kproc->dev, "PM mbox_send_message failed: %d\n", ret);
            return ret;
        }
        ret = wait_for_completion_timeout(&kproc->suspend_comp, to);
        if (ret == 0) {
            dev_err(kproc->dev,
                "%s: timedout waiting for rproc completion event\n", __func__);
            return -EBUSY;
        };
    }
    kproc->rproc->state = RPROC_SUSPENDED;
    break;
```

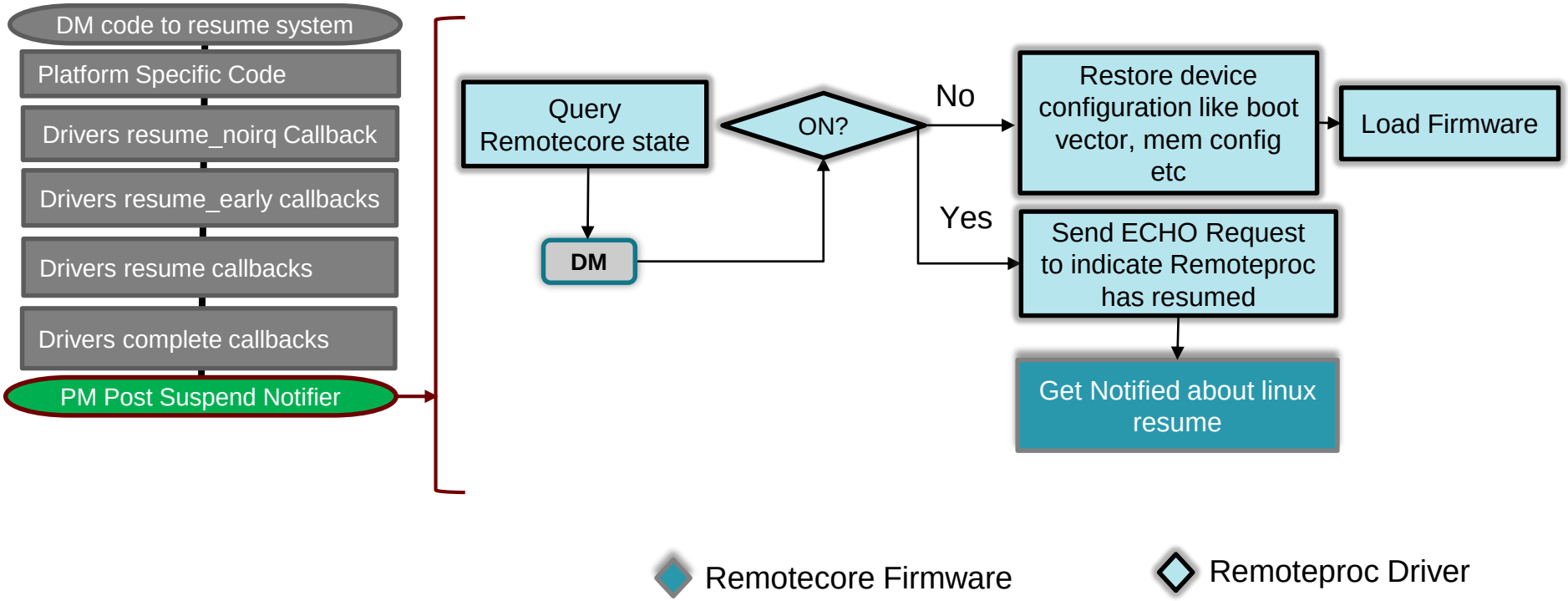
Abbreviations

- TISCI (Texas Instruments System Control Interface)
  - Message-passing protocol to interact with device manager or security manager firmware
- DM
  - Device management core responsible to perform low power handling
- Constraints
  - Restrict low power mode entry e.g.
  - For **Device drivers**, these can be:
    - **PM QoS request -** The maximum resume latency
    - **Device wakeup**
  - User constraints can be set through existing CPU QoS interface

Remoteproc Suspend Flow in TI SOCs - suspend_late



Remoteproc Resume Flow in TI SOCs – PM Notifier



TI's R5 remoteproc driver – PM Operation Callback

Register Device Power Management operation hooks

```
static const struct dev_pm_ops k3_r5_pm_ops = {
    LATE_SYSTEM_SLEEP_PM_OPS(k3_r5_suspend_late, NULL)
};

static struct platform_driver k3_r5_rproc_driver = {
    .probe = k3_r5_probe,
    .driver = {
        .name = "k3_r5_rproc",
        .pm = &k3_r5_pm_ops,
        .of_match_table = k3_r5_of_match,
    },
};
```

Suspend late callback Implementation

```
static int k3_r5_suspend(struct rproc *rproc)
{
    unsigned long msg = RP_MBOX_SUSPEND_SYSTEM;
    ...
    if (rproc->state != RPROC_RUNNING) {
        dev_err(dev, "remoteproc state not suspendable! State=%d\n", rproc->state);
        return ret;
    }

    kproc->suspend_status = 0;
    reinit_completion(&kproc->suspend_comp);

    ret = mbox_send_message(kproc->mbox, (void *)msg);
    if (ret < 0) {
        dev_err(dev, "PM mbox_send_message failed: %d\n", ret);
        return ret;
    }

    ret = wait_for_completion_timeout(&kproc->suspend_comp, to);
    if (ret == 0) {
        dev_err(dev, "%s: timedout waiting for rproc completion event\n", __func__);
```

Process Response From Remotecore

```
if (kproc->suspend_status == RP_MBOX_SUSPEND_ACK) {
    /* shutdown the remote core */
    ret = rproc_shutdown(rproc);
    if (ret) {
        dev_err(dev, "rproc_shutdown failed, ret = %d\n", ret);
        return -EBUSY;
    }
    kproc->rproc->state = RPROC_SUSPENDED;
} else if (kproc->suspend_status == RP_MBOX_SUSPEND_CANCEL) {
    return -EBUSY;
} else if (kproc->suspend_status == RP_MBOX_SUSPEND_AUTO) {
    kproc->rproc->state = RPROC_SUSPENDED;
}
```

TI's R5 remoteproc driver – PM Suspend Notifier

Register Callback for PM Suspend notifications

```
} else if (!c_state) {
    dev_info(cdev, "configured R5F for remoteproc mode\n");
    /* add support for suspend/resume */
    kproc->pm_notifier.notifier_call = r5f_pm_notifier_call;
    register_pm_notifier(&kproc->pm_notifier);
    ret = 0;
```

```
static int r5f_pm_notifier_call(struct notifier_block *bl,
                unsigned long state, void *unused)
{
    ...

    switch (state) {
    case PM_POST_SUSPEND:
        if (rproc->state == RPROC_SUSPENDED)
            return k3_r5_resume(rproc);
        break;
    }
```

```
static int k3_r5_resume(struct rproc *rproc)
{
    ...

    if (rproc->state != RPROC_SUSPENDED)
        return ret;

    ret = get_core_status(core, &cstatus);
    if (ret) {
        dev_err(dev, "failed to get core status: %d\n", ret);
        return ret;
    }
```

Read current state to decide resume action

```
if (cstatus) {
    dev_err(dev, "Core is on in resume\n");
    msg = RP_MBOX_ECHO_REQUEST;
    ret = mbox_send_message(kproc->mbox, (void *)msg);
    if (ret < 0) {
        dev_err(dev, "PM mbox_send_message failed: %d\n", ret);
        return ret;
    }
} else {
    dev_info(dev, "Core is off in resume\n");
    /* restore device configuration */
    ret = ti_sci_proc_set_config(core->tsp, core->boot_vec,
                                 core->set_cfg, core->clr_cfg);
    if (ret)
        dev_err(dev, "set config failed: %d\n", ret);

    ret = rproc_boot(rproc);
    if (ret) {
        dev_err(dev, "rproc_boot failed: %d\n", ret);
        return ret;
    }
}
kproc->rproc->state = RPROC_RUNNING;
```

Additional Challenges

Texas Instruments

Challenges to Get Suspend Resume working in Remoteproc Systems

- The remotecores that are booted early (u-boot or spl) may not support lpm!
  - Firmware loaded early by SPL or u-boot may not be available to remoteproc driver
    - Remotecore firmware may save and restore the context in suspend and resume
    - Same remotecore firmware can be loaded in linux filesystem

- H/W used by remoteproc driver not in clean state can prevent system wide suspend e.g.
  - Unread messages in various FIFOs
  - rpmsg channel not properly suspended
  - remotecores does not have ipc support

References

- https://www.kernel.org/doc/html/v4.18/driver-api/pm/index.html
- https://git.ti.com/cgit/ti-linux-kernel/ti-linux-kernel/tree/drivers/remoteproc?h=ti-linux-6.12.y
- https://github.com/TexasInstruments/mcupsdk-rm-pm-hal

Credits and Acknowledgement

- Texas Instruments Inc.
- The Linux Foundation.
- Baylibre

Q&A

- Contact Information:
  - Akashdeep Kaur <a-kaur@ti.com>
  - Hari Nagalla <hnagalla@ti.com>
  - Kendall Willis <k-willis@ti.com>
- Also on IRC @ libera.chat #linux-ti

Learn more about TI products

- https://www.ti.com/linux
- https://www.ti.com/processors
- https://www.ti.com/edgeai

Why choose TI MCUs and processors?

Scalability

Our products offer scalable performance that can adapt and grow as the needs of your customers evolve.

Efficiency

We design products that extend battery life, maximize performance for every watt expended, and unlock the highest levels of system efficiency.

Affordability

We strive to make innovation accessible to all by creating cost-effective products that feature state-of-the-art technology and package designs.

Availability

Our investment in internal manufacturing capacity provides greater assurance of supply, supporting your growth for decades to come.