Designing a Multi-PMIC Linux Driver: *Key Decisions for Mainline Submission*

Shree Ramamoorthy

August 25th, 2025

Speaker Introduction

Shree Ramamoorthy

Software Engineer, Texas Instruments

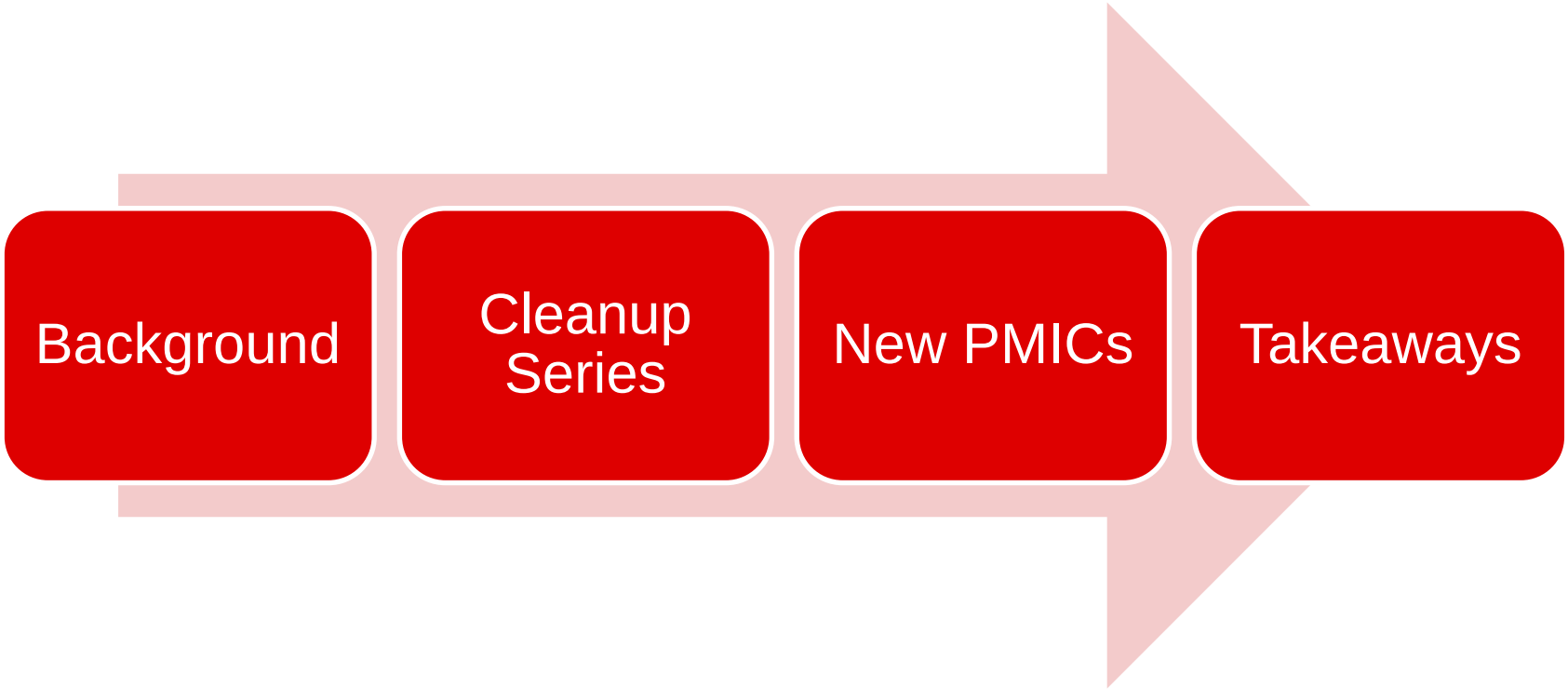
- Dallas, Texas
- Power Management Integrated Circuit Team
- Linux device driver development
- Customer GUIs

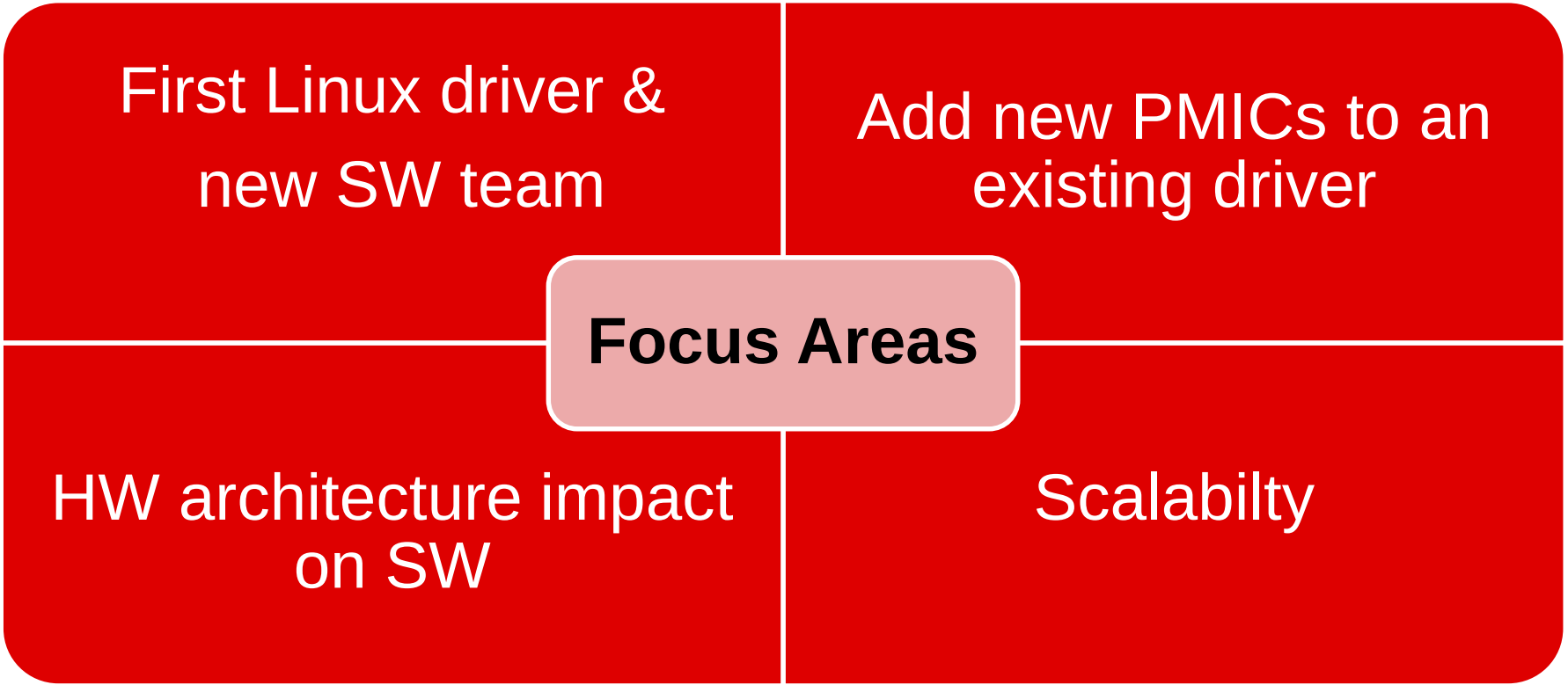
About us: TI and Open Source

Upstream FIRST!

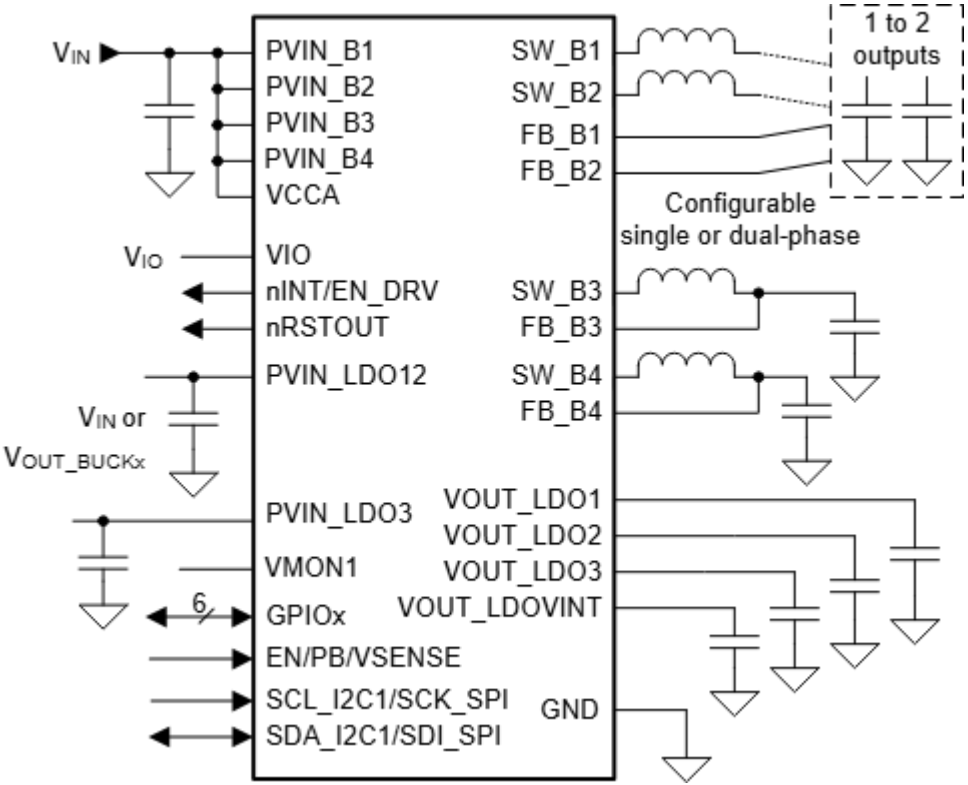
- ✔ Upstream FIRST mentality!
- ✔ Decades of contribution and collaboration
- ✔ Focus on long term, sustainable and quality products
- ✔ Upstream & opensource ecosystem in device architecture
- ✔ Ingrained culture to give back to the community

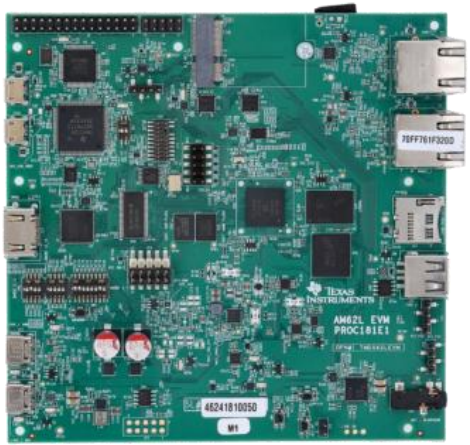
Overview

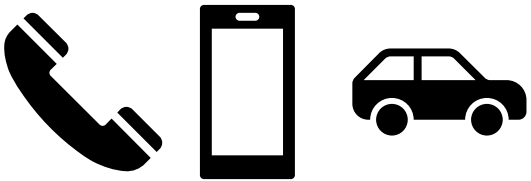



First Linux driver &
new SW team
Add new PMICs to an
existing driver
Focus Areas
HW architecture impact
on SW
Scalabilty

What is a PMIC?







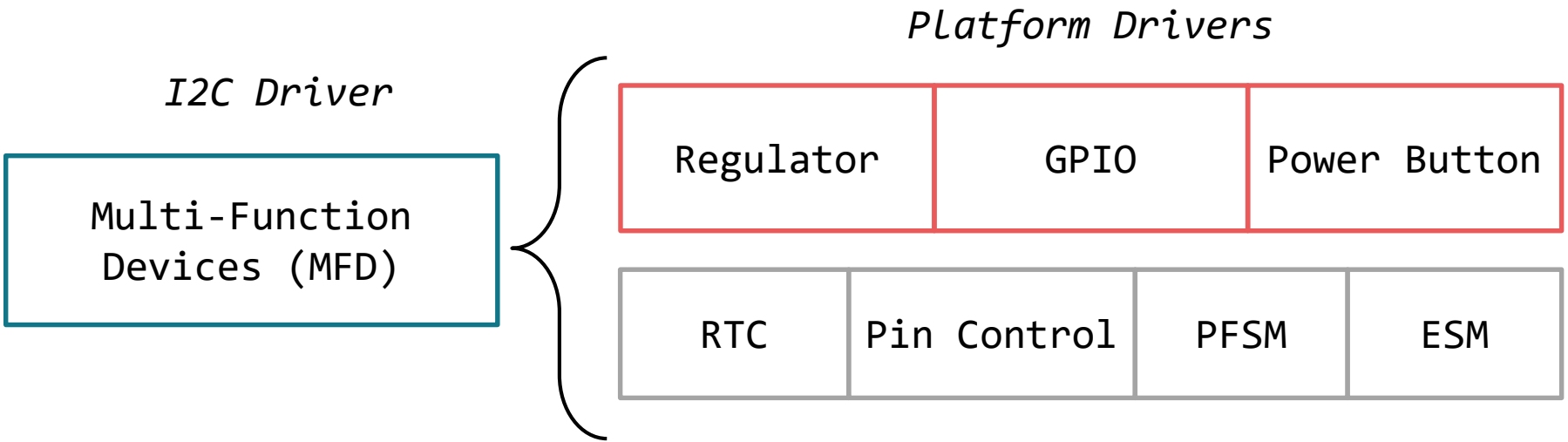
Texas Instruments

Why do PMICs need a Linux Driver?

Texas Instruments

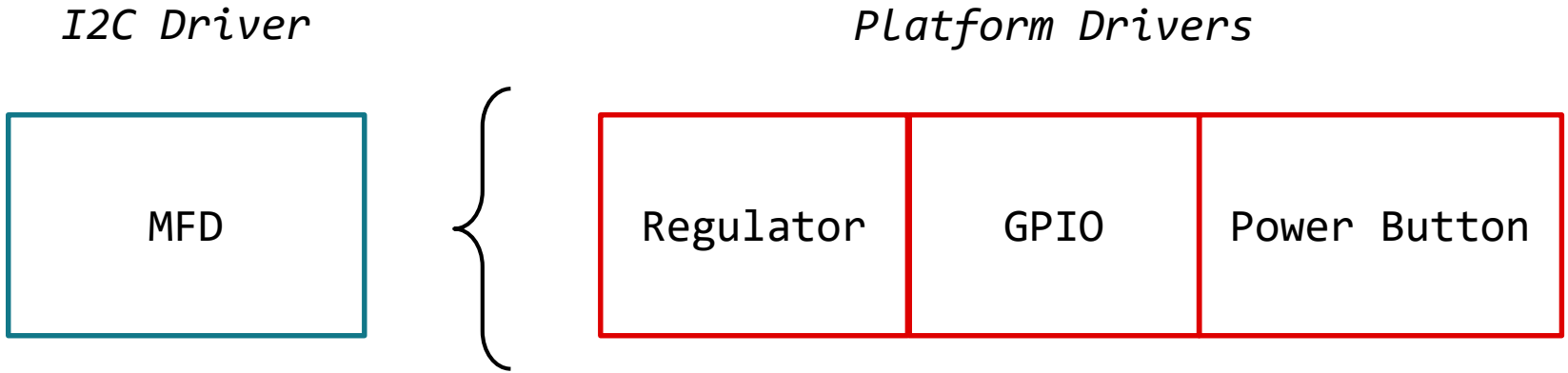
PMIC Linux Driver Structure

Legend
MFD Driver: always included
Subsystems typically included
PMIC feature-specific



Texas Instruments

Existing TPS65219 Driver Structure



Texas Instruments

PMIC Feature Comparison

TPS65214	TPS65215	TPS65219
3 Buck Regulators		
2 LDOs		4 LDOs
1 GPO		2 GPOs
1 GPIO with configurable output during device operation	Same GPIO features	

Texas Instruments

Goal

- Prioritize code re-use *without* sacrificing readability.
- Simplify the integration of additional PMICs.
- Reduce future development effort.

Applied to Kernel Version #	Patch Series
6.14	Cleanup: MFD
	Cleanup: Regulator
	Cleanup: GPIO
6.15	MFD
6.16	Regulator
6.17	GPIO

Cleanup Series

MFD, Regulator, and GPIO

Texas Instruments

MFD

- Migrated to `MFD_CELL` macros

```
static const struct mfd_cell tps65219_cells[] = {
-       {
-               .name = "tps65219-regulator",
-               .resources = tps65219_regulator_resources,
-               .num_resources = ARRAY_SIZE(tps65219_regulator_resources),
-       },
-       { .name = "tps65219-gpio", },
+       MFD_CELL_RES("tps65219-regulator", tps65219_regulator_resources),
+       MFD_CELL_NAME("tps65219-gpio"),
 };
```


Texas Instruments

MFD

- Removed unused, custom-defined macros
- Replaced direct struct definitions with header file include

```
+#include <linux/regmap.h>
 #include <linux/regulator/driver.h>

-struct regmap;
-struct regmap_irq_chip_data;
-
-#define TPS65219_1V35                                   1350000
-#define TPS65219_1V8                                    1800000
```


Texas Instruments

Regulator

- Updated error handling to `dev_err_probe()` instead `dev_err()`

```
-   if (IS_ERR(rdev)) {
-         dev_err(tps->dev, "failed to register %s regulator\n", regulators[i].name);
-         return PTR_ERR(rdev);
-   }
+   if (IS_ERR(rdev))
+         return dev_err_probe(tps->dev, PTR_ERR(rdev),
+                              "Failed to register %s regulator\n",
+                              regulators[i].name);
```


Texas Instruments

GPIO

- Replaced custom GPIO direction macros with existing kernel-defined macros

```
-#define TPS65219_GPIO_DIR_IN                    1
-#define TPS65219_GPIO_DIR_OUT                   0

-       if (tps65219_gpio_get_direction(gc, offset) == TPS65219_GPIO_DIR_OUT)
+       if (tps65219_gpio_get_direction(gc, offset) == GPIO_LINE_DIRECTION_OUT)

-       return tps65219_gpio_change_direction(gc, offset, TPS65219_GPIO_DIR_IN);
+       return tps65219_gpio_change_direction(gc, offset, GPIO_LINE_DIRECTION_IN);
```

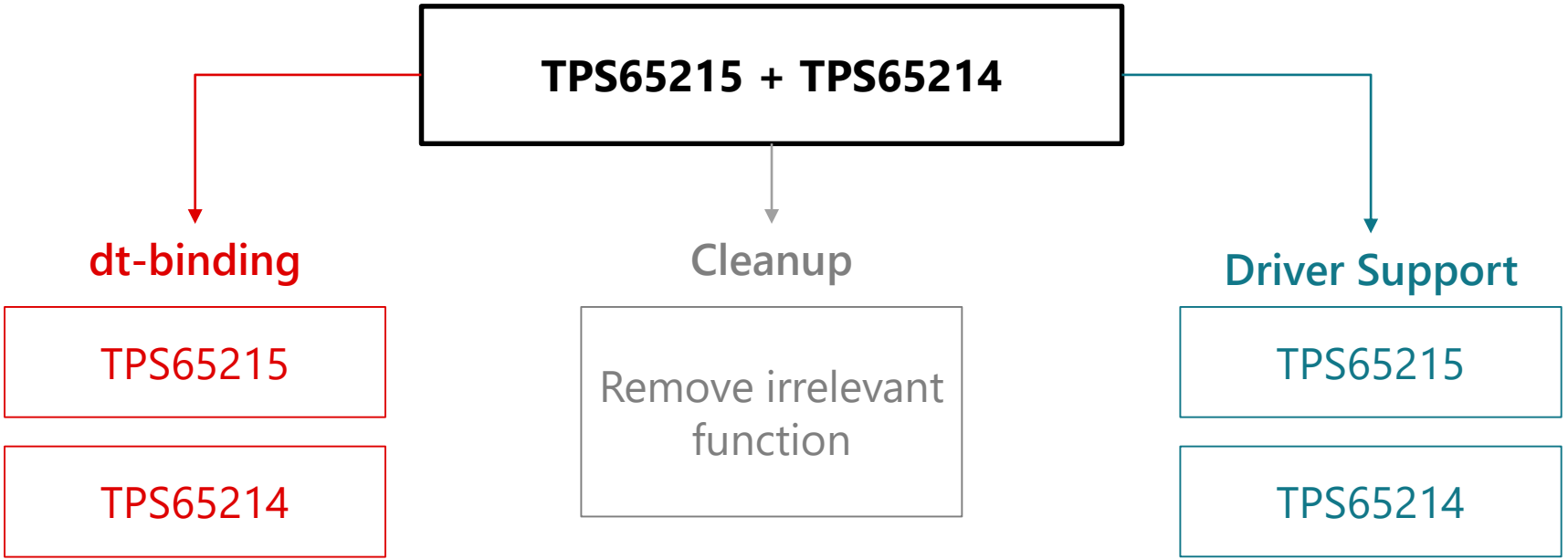

Texas Instruments

Add new PMICs' Linux driver support

TPS65214 and TPS65215

Texas Instruments

MFD Series



Texas Instruments

HW Architecture Impact #1

- TPS65215 and TPS65219: different `status_base` and `ack_base` registers.
- Importance of cross-functional collaboration.

```
+static const struct regmap_irq_chip tps65215_irq_chip = {
+         .name = "tps65215_irq",
+         .main_status = TPS65219_REG_INT_SOURCE,
+         .num_main_regs = 1,
+         .num_main_status_bits = 8,
+         .irqs = tps65215_irqs,
+         .num_irqs = ARRAY_SIZE(tps65215_irqs),
+         .status_base = TPS65215_REG_INT_LDO_2,
+         .ack_base = TPS65215_REG_INT_LDO_2,
+         .clear_ack = 1,
+         .num_regs = 8,
+         .sub_reg_offsets = tps65215_sub_irq_offsets,
+};
```

HW Architecture Impact #2

- TPS65215 and TPS65219: different bits for `regmap_irq_sub_irq_map` (bits 5 & 6).

```
 static unsigned int bit5_offsets[] = {TPS65219_REG_INT_LDO_1_2_POS};
 static unsigned int bit6_offsets[] = {TPS65219_REG_INT_LDO_3_4_POS};
+static unsigned int tps65215_bit5_offsets[] = {TPS65215_REG_INT_LDO_1_POS};
+static unsigned int tps65215_bit6_offsets[] = {TPS65215_REG_INT_LDO_2_POS};

+static struct regmap_irq_sub_irq_map tps65215_sub_irq_offsets[] = {
+         REGMAP_IRQ_MAIN_REG_OFFSET(bit0_offsets),
+         REGMAP_IRQ_MAIN_REG_OFFSET(bit1_offsets),
+         REGMAP_IRQ_MAIN_REG_OFFSET(bit2_offsets),
+         REGMAP_IRQ_MAIN_REG_OFFSET(bit3_offsets),
+         REGMAP_IRQ_MAIN_REG_OFFSET(bit4_offsets),
+         REGMAP_IRQ_MAIN_REG_OFFSET(tps65215_bit5_offsets),
+         REGMAP_IRQ_MAIN_REG_OFFSET(tps65215_bit6_offsets),
+         REGMAP_IRQ_MAIN_REG_OFFSET(bit7_offsets),
+};
```

HW Architecture Impact #3

- TPS65219's LDO3 mapped to TPS65215's LDO2.
- Result: multiple PMIC-specific structs added to MFD driver.

```
+static const struct regmap_irq tps65215_irqs[] = {
+        TPS65219_REGMAP_IRQ_REG(TPS65215_INT_LDO1_SCG, TPS65215_REG_INT_LDO_1_POS),
+        TPS65219_REGMAP_IRQ_REG(TPS65215_INT_LDO1_OC, TPS65215_REG_INT_LDO_1_POS),
+        TPS65219_REGMAP_IRQ_REG(TPS65215_INT_LDO1_UV, TPS65215_REG_INT_LDO_1_POS),
+        TPS65219_REGMAP_IRQ_REG(TPS65215_INT_LDO2_SCG, TPS65215_REG_INT_LDO_2_POS),
+        TPS65219_REGMAP_IRQ_REG(TPS65215_INT_LDO2_OC, TPS65215_REG_INT_LDO_2_POS),
+        TPS65219_REGMAP_IRQ_REG(TPS65215_INT_LDO2_UV, TPS65215_REG_INT_LDO_2_POS),
         // 27 lines removed for clarity
+        TPS65219_REGMAP_IRQ_REG(TPS65219_INT_LDO1_RV_SD, TPS65219_REG_INT_TO_RV_POS),
+        TPS65219_REGMAP_IRQ_REG(TPS65215_INT_LDO2_RV_SD, TPS65219_REG_INT_TO_RV_POS),
+};
```

Why did each PMIC add similar code churn?

Texas Instruments

Patch Diff Analysis

TPS65215	TPS65214
2 files changed	2 files changed
209 insertions(+)	184 insertions(+)
12 deletions(-)	7 deletions(-)

Driving forces behind code churn

1. PMIC-specific structs added to promote readability
2. Introduced new `tps65219_chip_data` struct

```
struct tps65219_chip_data {
    const struct regmap_irq_chip *irq_chip;
    const struct mfd_cell *cells;
    int n_cells;
};
static const struct tps65219_chip_data chip_info_table[] = {
    [TPS65219] = {
          .irq_chip = &tps65219_irq_chip,
          .cells = tps65219_cells,
          .n_cells = ARRAY_SIZE(tps65219_cells),
         },
};
```

Driving forces behind code churn

1. PMIC-specific structs added to promote readability
2. Introduced new `tps65219_chip_data` struct

```
struct tps65219_chip_data {
    const struct regmap_irq_chip *irq_chip;
    const struct mfd_cell *cells;
    int n_cells;
};
static const struct tps65219_chip_data chip_info_table[] = {
    [TPS65219] = {
          .irq_chip = &tps65219_irq_chip,
          .cells = tps65219_cells,
          .n_cells = ARRAY_SIZE(tps65219_cells),
         },
};
```


Texas Instruments

Driving forces behind code churn

1. PMIC-specific structs added to promote readability
2. Introduced new `tps65219_chip_data` struct

```
struct tps65219_chip_data {
    const struct regmap_irq_chip *irq_chip;
    const struct mfd_cell *cells;
    int n_cells;
};
static const struct tps65219_chip_data chip_info_table[] = {
    [TPS65219] = {
          .irq_chip = &tps65219_irq_chip,
          .cells = tps65219_cells,
          .n_cells = ARRAY_SIZE(tps65219_cells),
        },
};
```


Texas Instruments

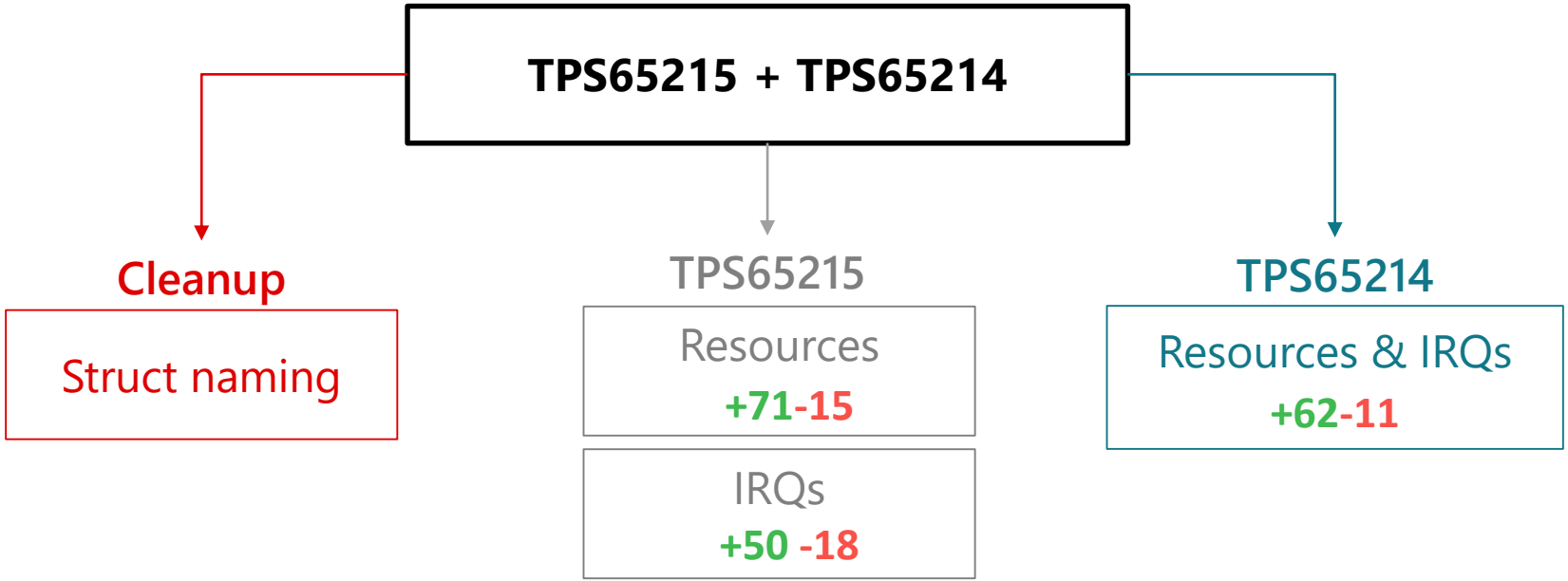
Driving forces behind code churn

1. PMIC-specific structs added to promote readability
2. Introduced new `tps65219_chip_data` struct

```
struct tps65219_chip_data {
    const struct regmap_irq_chip *irq_chip;
    const struct mfd_cell *cells;
    int n_cells;
};
static const struct tps65219_chip_data chip_info_table[] = {
    [TPS65219] = {
          .irq_chip = &tps65219_irq_chip,
          .cells = tps65219_cells,
          .n_cells = ARRAY_SIZE(tps65219_cells),
        },
};
```


Texas Instruments

Regulator Series



Texas Instruments

Restructure regulator driver

- **Goal**: Little to no changes for the `probe()`.
- **Attempt:** abstract registering regulators & IRQ handling into helper functions.
- **Lesson Learned:** resulting diff was hard to read & did not pass kernel build tests.
- **Implementation**
  - Created common structs for overlapping information.
  - Added device-specific structs.
  - Handle common structs first in `probe()`, then device-specific.
  - Add new PMIC to `chip_data` struct.
- **Result**: Easier for reviewers and patch organization.

Texas Instruments

Regulator chip_data struct

```
struct tps65219_chip_data {
    size_t rdesc_size;
    size_t common_rdesc_size;
    size_t dev_irq_size;
    size_t common_irq_size;
    const struct regulator_desc *rdesc;
    const struct regulator_desc *common_rdesc;
    struct tps65219_regulator_irq_type *irq_types;
    struct tps65219_regulator_irq_type *common_irq_types;
};
```


Texas Instruments

Register regulators in probe()

```
struct tps65219_chip_data {
    size_t rdesc_size;
    size_t common_rdesc_size;
    size_t dev_irq_size;
    size_t common_irq_size;
    const struct regulator_desc *rdesc;
    const struct regulator_desc *common_rdesc;
    struct tps65219_regulator_irq_type *irq_types;
    struct tps65219_regulator_irq_type *common_irq_types;
};
```

```
for (i = 0; i <  pmic->common_rdesc_size; i++) {
        rdev = devm_regulator_register(&pdev->dev, &pmic->common_rdesc[i], &config);
        if (IS_ERR(rdev))
            return dev_err_probe(tps->dev, PTR_ERR(rdev), "Failed to register %s regulator\n",
                                                          pmic->common_rdesc[i].name);
 }
 // Then, do the same to register PMIC-specific regulators
```


Texas Instruments

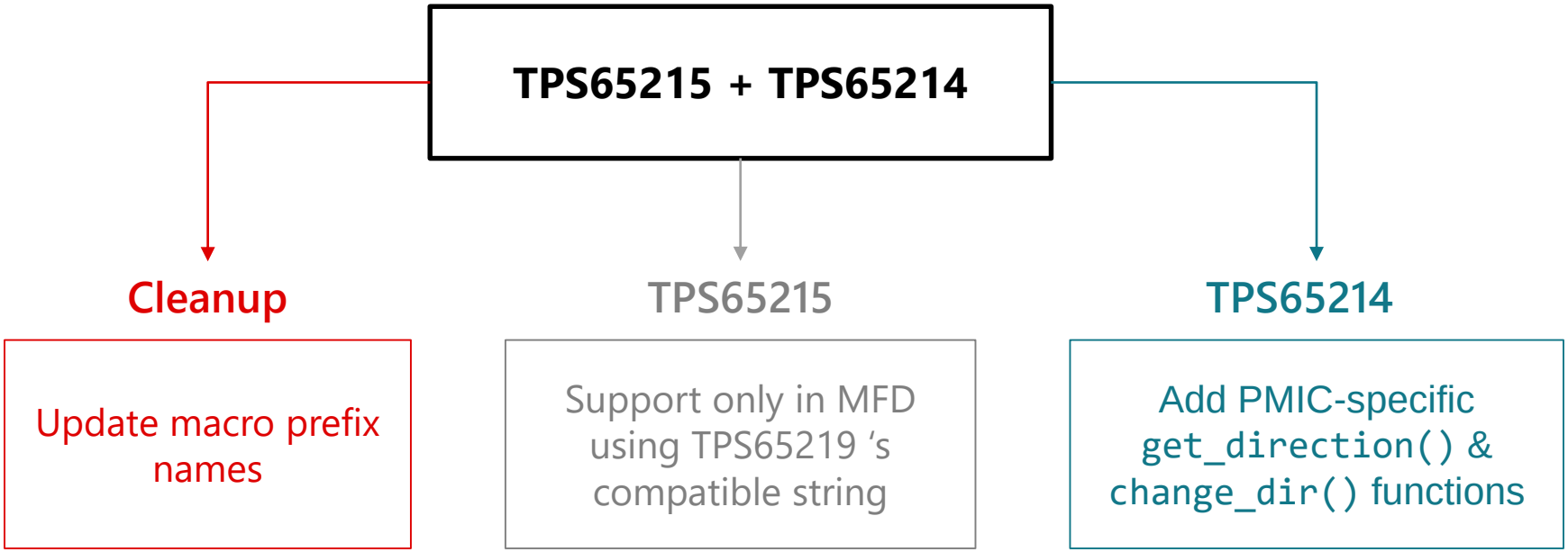
Register regulators in probe( )

```
struct tps65219_chip_data {
    size_t rdesc_size;
    size_t common_rdesc_size;
    size_t dev_irq_size;
    size_t common_irq_size;
    const struct regulator_desc *rdesc;
    const struct regulator_desc *common_rdesc;
    struct tps65219_regulator_irq_type *irq_types;
    struct tps65219_regulator_irq_type *common_irq_types;
};
```

```
for (i = 0; i <  pmic->common_rdesc_size; i++) {
        rdev = devm_regulator_register(&pdev->dev, &pmic->common_rdesc[i], &config);
        if (IS_ERR(rdev))
            return dev_err_probe(tps->dev, PTR_ERR(rdev), "Failed to register %s regulator\n",
                                                          pmic->common_rdesc[i].name);
 }
 // Repeat the same implementation to register PMIC-specific regulators
```


Texas Instruments

GPIO Series


TPS65215 + TPS65214
Cleanup
Update macro prefix names
TPS65215
Support only in MFD using TPS65219 's compatible string
TPS65214
Add PMIC-specific get_direction() & change_dir() functions

HW Architecture Impact #1

- The different GPIO direction registers and bit assignments for input and output values.
- Input and output were assigned to opposite values for the PMICs.
- Required PMIC-specific `_get_direction()` functions.

Register Address	Bit	TPS65219 Field	TPS65214 Field
0x1E	1	GPO2_EN	GPIO_CONFIG: 0x0 = input 0x1 = output
0x1F	3	MULTI_DEVICE_EN: 0x0 = GPO 0x1 = GPIO	GPIO_VSEL_CONFIG: 0x0 = GPIO

Texas Instruments

HW Architecture Impact #2

- TPS65214: `GPIO_CONFIG` can be changed during device operation.
- `GPIO_VSEL_CONFIG` is configured as GPIO.
- Required PMIC-specific `_change_direction()` functions.

Register Address	Bit	TPS65219 Field	TPS65214 Field
0x1E	1	GPO2_EN	GPIO_CONFIG: 0x0 = input 0x1 = output
0x1F	3	MULTI_DEVICE_EN: 0x0 = GPO 0x1 = GPIO	GPIO_VSEL_CONFIG: 0x0 = GPIO

Texas Instruments

GPIO Mapping from Linux to PMIC

- TPS65214 has 2 pins vs. TPS65215 & TPS65219 have 3 pins.
- **Consideration**
  - Maintain mapping to support backwards compatibility.
  - Existing driver mapped Linux GPIO offset 0 to the PMIC GPIO pin.
- **Highlight**
  - Reached out to the original author to gain context on mapping.
  - 1:1 patch reviews before submitting the next upstream version
- **Result**
  - Existing code calculation supports all PMICs.
  - Add comments explaining the GPIO mapping for future developers/users.
  - Corrected understanding after gaining context from the original author

Takeaways

- Understand the driver precedent and possible feature history.
- Think about the patch from a reviewer's POV.
- Consider requesting to merge interdependent subsystem patch series.
- MFD Driver Idea: Organize all `regulator_resources` and `regmap_irq` structs into common & device-specific structs.

Texas Instruments

Acknowledgments

- Michael Leonard
- Praneeth Bajjuri
- Vishal Mahaveer
- Andrew Davis
- Jonathon Cormier
- Dhruva Gole
- Bryan Brattlof
- Maintainers!
- + Many more ☺

Q&A

s-ramamoorthy@ti.com

Learn more about TI

TI

TI PMICs

Why choose TI PMICs?

Broad portfolio of functional safety-compliant devices

We have over a decade of functional safety expertise, resulting in a broad and scalable portfolio from quality-managed to ASIL D-compliant PMICs.

Software drivers and development tools

Complete your design with Linux® and RTOS software drivers designed for easy integration into your software environment. We offer GUIs to develop and evaluate your PMIC configuration settings.

Demo boards for embedded systems

Start your design with our growing catalog of SoC reference designs, which range from connection block diagrams to comprehensive and validated hardware, software and design files.