# Kernel driver for lp5562

- TI LP5562 LED Driver

Author: Milo(Woogyom) Kim <milo.kim@ti.com>

## Description

LP5562 can drive up to 4 channels. R/G/B and White. LEDs can be controlled directly via the led class control interface.

All four channels can be also controlled using the engine micro programs. LP5562 has the internal program memory for running various LED patterns. For the details, please refer to 'firmware' section in leds-lp55xx.txt

## Device attribute

engine_mux

3 Engines are allocated in LP5562, but the number of channel is 4. Therefore each channel should be mapped to the engine number.

Value: RGB or W

This attribute is used for programming LED data with the firmware interface. Unlike the LP5521/LP5523/55231, LP5562 has unique feature for the engine mux, so additional sysfs is required

LED Map

| | | |
|---|---|---|
| Red | ... | Engine 1 (fixed) |
| Green | ... | Engine 2 (fixed) |
| Blue | ... | Engine 3 (fixed) |
| White | ... | Engine 1 or 2 or 3 (selective) |

## How to load the program data using engine_mux

Before loading the LP5562 program data, engine_mux should be written between the engine selection and loading the firmware. Engine mux has two different mode, RGB and W. RGB is used for loading RGB program data, W is used for W program data.

For example, run blinking green channel pattern:

```
echo 2 > /sys/bus/i2c/devices/xxxx/select_engine     # 2 is for green channel
echo "RGB" > /sys/bus/i2c/devices/xxxx/engine_mux    # engine mux for RGB
echo 1 > /sys/class/firmware/lp5562/loading
echo "4000600040FF6000" > /sys/class/firmware/lp5562/data
echo 0 > /sys/class/firmware/lp5562/loading
echo 1 > /sys/bus/i2c/devices/xxxx/run_engine
```

To run a blinking white pattern:

```
echo 1 or 2 or 3 > /sys/bus/i2c/devices/xxxx/select_engine
echo "W" > /sys/bus/i2c/devices/xxxx/engine_mux
echo 1 > /sys/class/firmware/lp5562/loading
echo "4000600040FF6000" > /sys/class/firmware/lp5562/data
echo 0 > /sys/class/firmware/lp5562/loading
echo 1 > /sys/bus/i2c/devices/xxxx/run_engine
```

## How to load the predefined patterns

Please refer to 'leds-lp55xx.txt"

## Setting Current of Each Channel

Like LP5521 and LP5523/55231, LP5562 provides LED current settings. The 'led_current' and 'max_current' are used.

## Example of Platform data

```
static struct lp55xx_led_config lp5562_led_config[] = {
        {
                .name           = "R",
                .chan_nr        = 0,
                .led_current    = 20,
                .max_current    = 40,
        },
        {
                .name           = "G",
                .chan_nr        = 1,
                .led_current    = 20,
                .max_current    = 40,
        },
        {
                .name           = "B",
                .chan_nr        = 2,
                .led_current    = 20,
                .max_current    = 40,
        },
        {
                .name           = "W",
                .chan_nr        = 3,
                .led_current    = 20,
                .max_current    = 40,
        },
};

static int lp5562_setup(void)
{
        /* setup HW resources */
}

static void lp5562_release(void)
{
        /* Release HW resources */
}

static void lp5562_enable(bool state)
{
        /* Control of chip enable signal */
}

static struct lp55xx_platform_data lp5562_platform_data = {
        .led_config     = lp5562_led_config,
        .num_channels   = ARRAY_SIZE(lp5562_led_config),
        .setup_resources   = lp5562_setup,
        .release_resources = lp5562_release,
        .enable            = lp5562_enable,
};
```

To configure the platform specific data, lp55xx_platform_data structure is used

If the current is set to 0 in the platform data, that channel is disabled and it is not visible in the sysfs.