ISL29125 Android Driver Detailed Description - Driver Notes

Purpose

This is a supplementary document for understanding the isl29125 android driver.

Audience

This document is intended to help understand different pieces of code from the driver that performs specific tasks. Having a fundamental knowledge of Linux driver model would be helpful in completely understanding this document. But certain parts of this document consists of flows which might be understood by all readers.

Driver Registration

The following piece of code does the i2c driver registration for the isl29125 sensor. The structure ***isl29125_driver*** consists of references to the required callbacks supported by the driver which are necessary for the operation of the driver.

```
static struct i2c_driver isl_sensor_driver = {
	.driver = {
		.name = "isl29125",
		.owner = THIS_MODULE,
	},
	.probe    = isl_sensor_probe      ,
	.remove   = isl_sensor_remove     ,
	.id_table = isl_sensor_device_table,
};

static int __init isl29125_init(void)
{
		i2c_register_board_info(3, i2c_devs_info, 1)
		return i2c_add_driver(&isl_sensor_driver);
}


static void __exit isl29125_exit(void)
{

	i2c_del_driver(&isl_sensor_driver);
}
```

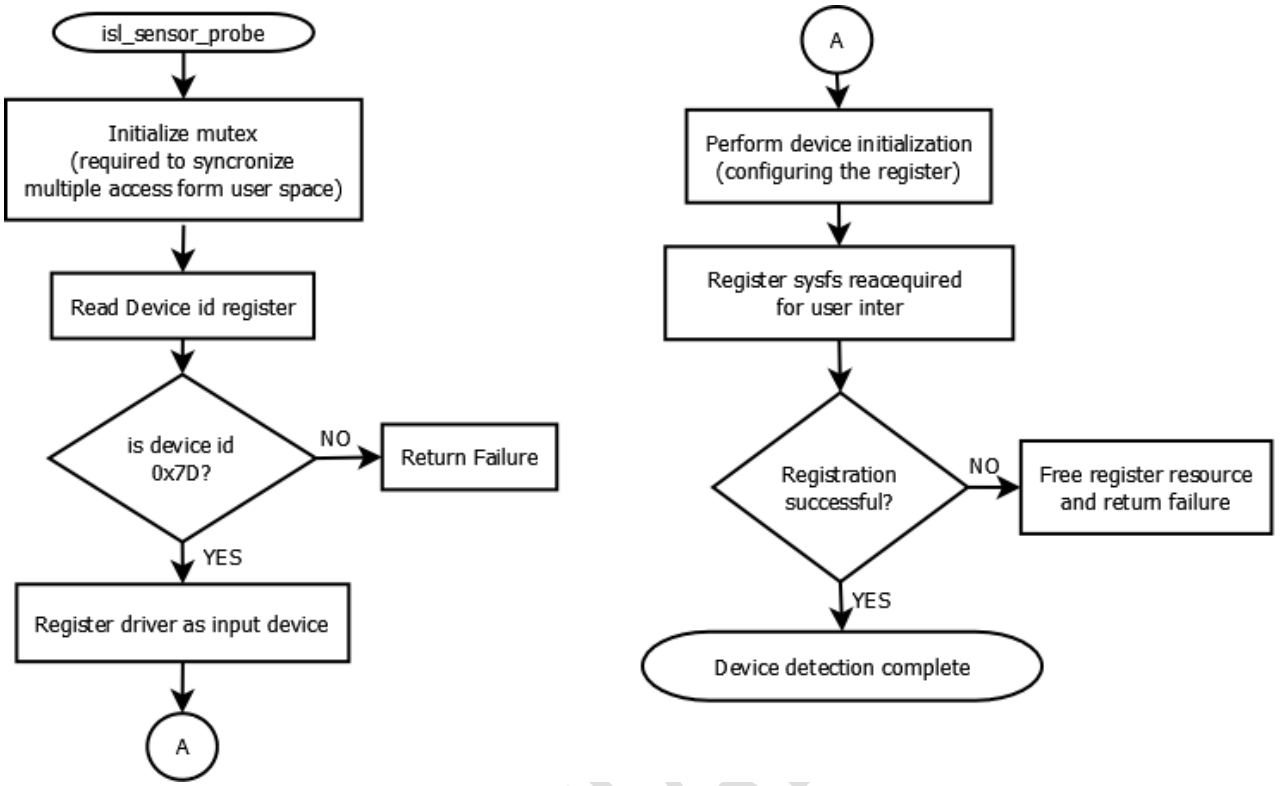
Reference	Description
.name (“isl29125”)	Name by which the driver would be identified in the Linux kernel
.owner (THIS_MODULE)	The name by which the driver is addressed.
isl_sensor_device_table	A table of i2c addresses or devices that would be supported by this driver
Isl_sensor_probe	Called when i2c core finds a matching device
Isl_sensor_remove	Called when this driver is removed from kernel

Device Detection

The platform information or board file for target platform or its device tree (used to describe the devices associated with the platform) contains the information on presence of isl29125 sensor with I2C slave address 0x44h being connected.

Once the Linux I2C core driver probes the I2C bus and finds this device, any I2C driver which has registered the device id for isl29125 would be notified about its presence and the driver would now be provided with access to the device.

The function isl29125_probe is called to notify the device presence. The following flow diagram summarizes the activities done in this function.



The following helper functions are called during device detection to perform the specified tasks

Function	Description
setup attributes	Register the user interface support in driver
setup_input_device	Register driver as an input device driver in order to report proximity events to userspace

Device Initialization

The function isl29125_initialize implements the device initialization for the ISL29125 sensor device. It primarily consists of the register writes to the device and a call to offset_adjust to do the initial compensation for the cross-talk.

Below table indicates the register writes and their description in order of initialization.

REG	VAL	Description
0x01	0x0D	Range 10000 Resolution 16-bit ADC start at initb start
0x02	0x5B	Disable IR compensation And default IR compensation value as 0x5B
0x03	0x1D	Interrupt on green Persistency 8 conversion data
0x04, 0x05	0xCC, 0x0C	5% of maximum range respectively
0x06, 0x07	0xCC , 0xCC	80% of maximum range respectively

Register Read / Write API
The following two functions are used in the driver to provide access to a specific set of bits or the complete register in the ISL29125 register space

1. i2c_smbus_read_byte_data(const struct i2c_client *client, u8 command)
2. i2c_smbus_write_byte_data (const struct i2c_client *client , u8 command, u8 value)

Note:

Isl29125_i2c_read_word16(const struct i2c_client *client, u8 command, u16 value)

This function is used to read the 16 bit data from the threshold registers.

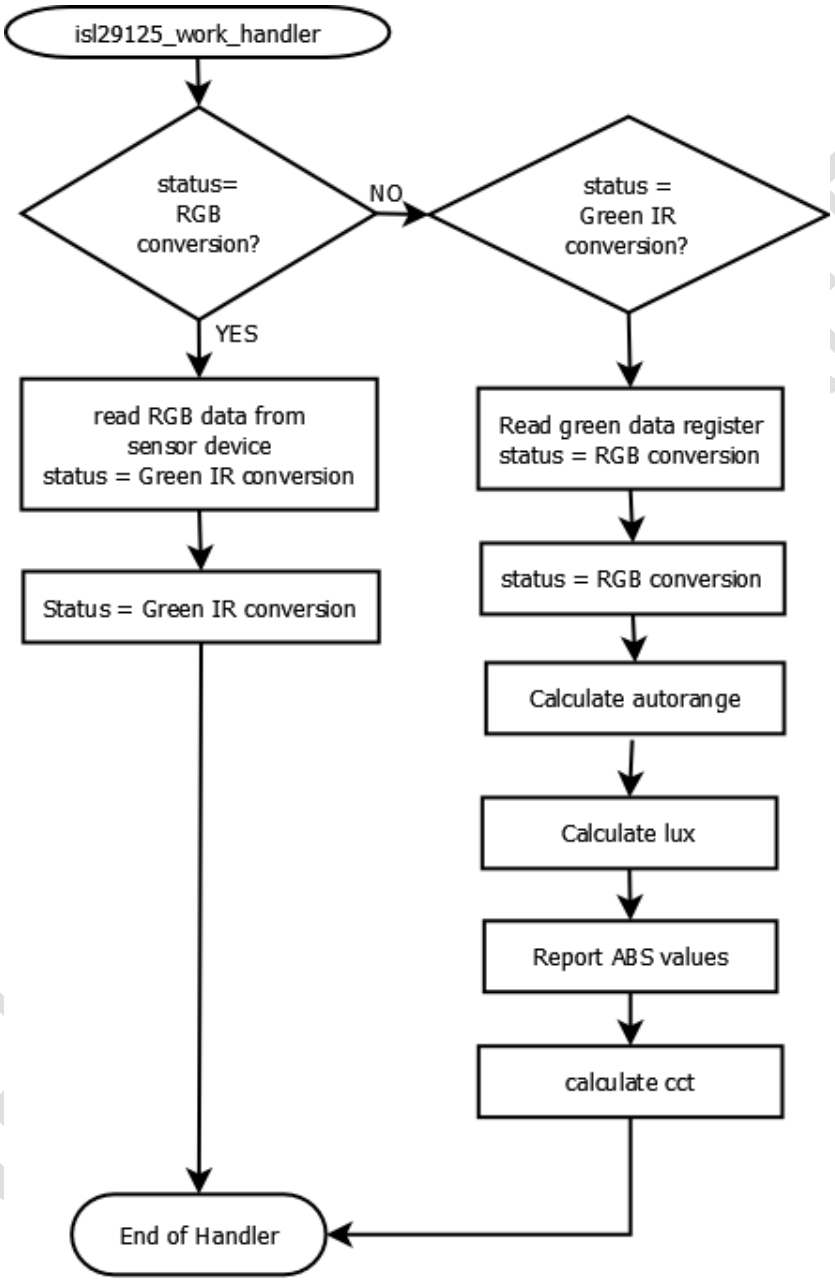
isl29125_work_handler

Currently the driver read the RED, GREEN and BLUE data register after every update (conversion time).

The below flowchart represents the task done in the function **isl29125_work_handler**
We divide the **runtime sequence** function in 2 following subsequences:

1. Read RGB data form the device register.
2. Schedule work queue. (restart the timer)
3. Report input absolute data.

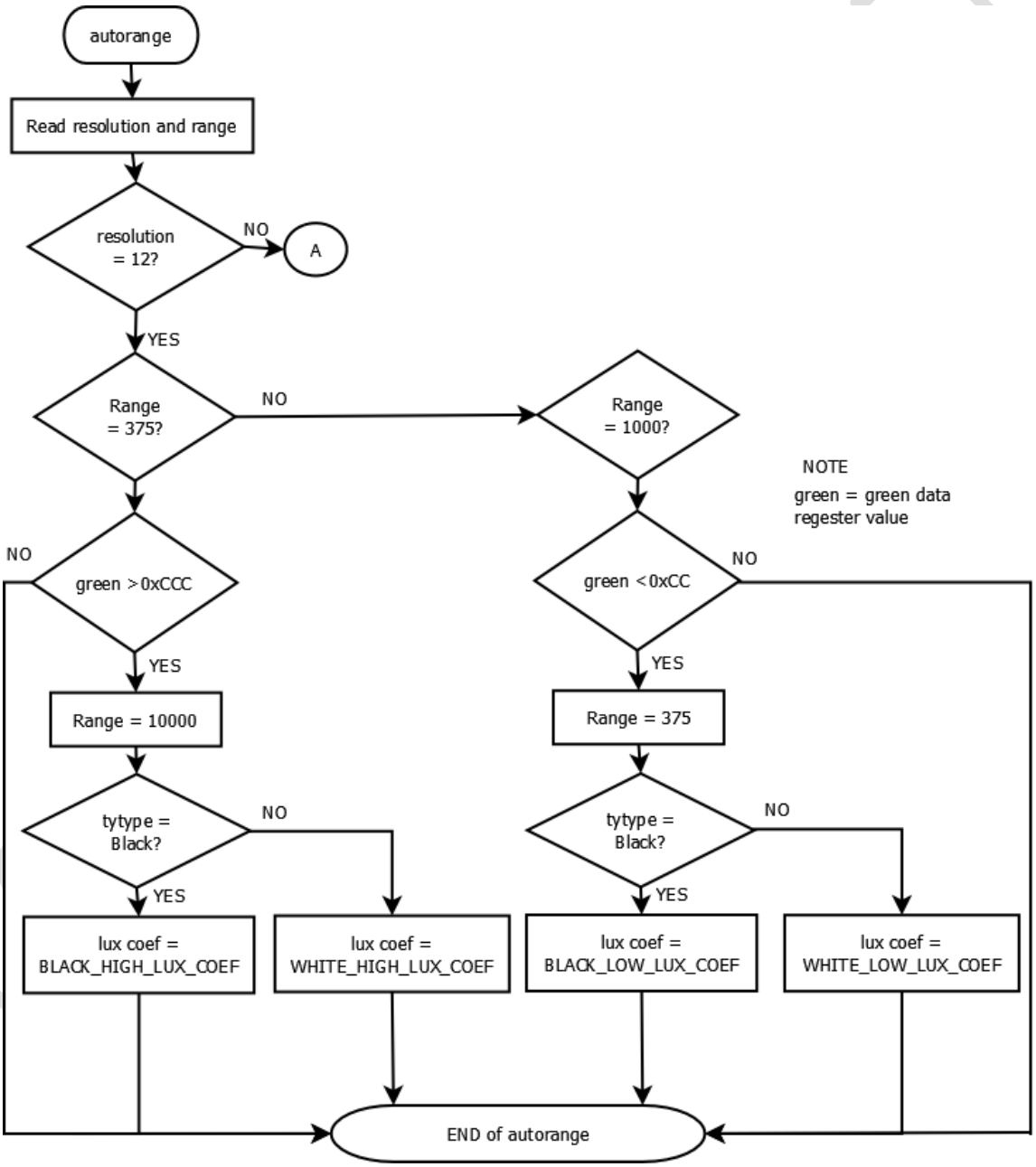
NOTE: This thread calls RGB conversion and Green IR conversion alternately

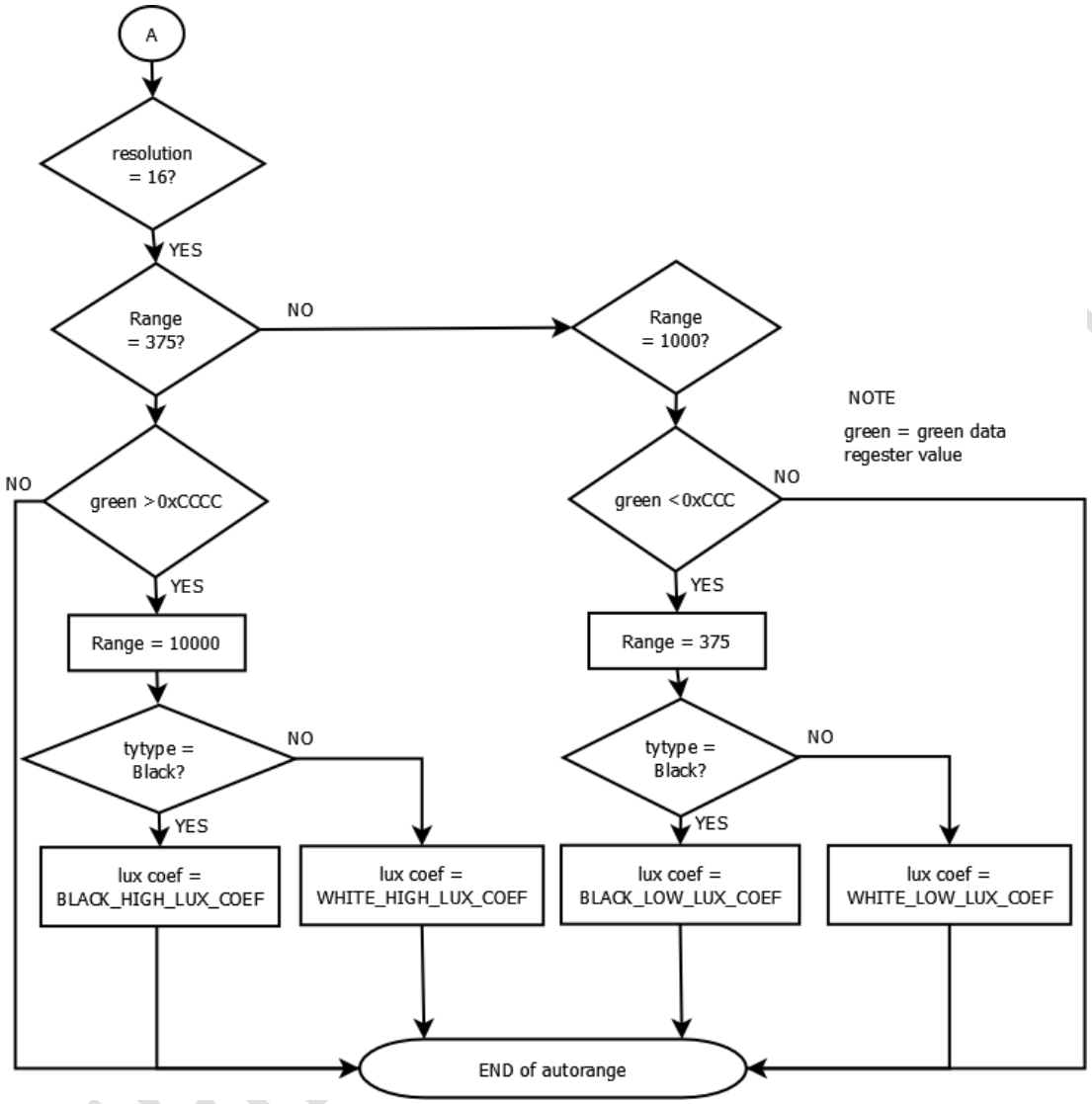


In the driver code below functions are mainly used

1. **Autorange:**

   Autorange automatically adjusts the range of the sensor device as per the current resolution. Following diagram shows the flow of the autorange function





2. **Cal_lux**

This function defines calculate lux value by using the parameter:

- Raw_green_ircomp
- Lux_coef

```
static unsigned long long cal_lux(struct isl29125_data_t *dat)
{
        dat->lux = ((dat->raw_green_ircomp * dat->lux_coef) >> 8) / 100;

        return dat->lux;
}
```

3. **Cal_cct**

To calculate cct follow the below steps:
For detailed description regarding calculate cct refer to this link: http://en.wikipedia.org/wiki/Planckian_locus

Step 1:

Read the raw Red, Green and Blue data from the sensor device register.

Step 2:

Calculate the three co-ordinates defining a color are given by X0, Y0, Z0 as per the tptype range black or white.

Step 3:

This will give the Planckian locus in CIE XYZ color space. If these coordinates are *X0*, *Y0*, *Z0*, then in the CIE chromaticity coordinates will be:

x = X0 / (X0 + Y0 + Z0)

y = Y0 / (X0 + Y0 + Z0)

Step 4:

Calculate cct value by:

cct = n * (n * ((-449 * n) / 1000 + 3525) / 1000 - 6823) / 1000 + 5520;

where n = (x – 3320)/(y - 1858)

NOTE:
All the numeric values are the coefficient, it may vary according to the coefficient.

User Interface
The driver implements the user interface using the “linux sysfs” infrastructure.

The driver provides access to three files from the userspace

I. show_cct
II. show_lux
III. show_xy_value
IV. show_raw_adc
V. show_reg_dump

show_cct
This sysfs file allows user to show the current cct value calculated by the function cal_cct.

show_lux
This sysfs file allows user to show the current lux value calculated by the function cal_lux.

show_xy
This sysfs file allows user to show the current x, y locus values calculated by using the equation.
This will give the Planckian locus in CIE XYZ color space. if the CIE XYZ color space, the three coordinates defining a color are given by *X0*, *Y0*, and *Z0*,

x = X0 / (X0 + Y0 + Z0)

y = Y0 / (X0 + Y0 + Z0)

show_raw_adc
This sysfs file allows user to show the current raw data of red, green and blue registers value read from the sensor device register.

show_xy
This sysfs file allows user to show the current x, y locus values calculated by using the equation.
This will give the Planckian locus in CIE XYZ color space. If the CIE XYZ color space, the three coordinates defining a color are given by *X0*, *Y0*, and *Z0*,

x = X0 / (X0 + Y0 + Z0)

y = Y0 / (X0 + Y0 + Z0)

show_reg_dump
This sysfs file allows user to dump entire register of the ISL29177 sensor device as well as to write to individual registers of the device.