ISL29037: Android Driver Integration Guide

Rev: A1

18th Nov 2014

Contact:

INTERSIL CORPORATION
(USA)

Revision History

Date	Rev No.	Description	By
November 18, 2014	A1	Android driver integration guide	Intersil corp
October 29, 2014	A0	Android driver integration guide	Intersil corp

Table of Contents

1 Introduction

This document describes the steps involved in integrating the ISL29037 Android driver to the Android source code and building images to be tested with the device. **"Pandaboard"** Platform is taken as reference to explain the process.

2 Driver Release

Product	ISL29037
Software	Android driver for sensor
Released files	1. isl29037.c 2. isl29037.h
Features	1. Supports data monitoring of sensor in user friendly format 2. Supports configuring major sensor parameters via debug interface 3. Simple user interface to dump and configure any required device register

3 Driver integration

STEP 1:

Copy the isl29037.c file to the following directory in the android kernel source code

<kernel top directory>/drivers/input/misc

STEP 2:

Copy the isl29037.h file to the following directory in the android kernel source code

<kernel top directory>/include/linux/input

STEP 3:

Edit your board file corresponding to the platform for which you are building the kernel to add the I2C device information. The below file is indicated for Pandaboard

<kernel top directory>/arch/arm/mach-omap2/board-omap4panda.c

STEP 4:

Edit the Makefile in the following directory to add build support for isl29037 driver as per the patch.

<kernel top directory>/drivers/input/misc/Makefile

STEP 5:

Edit the Kconfig in the following directory to add menuconfig support for isl29037 driver as per the patch.

<kernel top directory>/drivers/input/misc/Kconfig

And add the below configuration to the board configuration file (here Pandaboard)

<kernel top directory>/arch/arm/configs/android_omap4_defconfig

CONFIG_INPUT_ISL29037 = y

STEP 6:

Rebuild the kernel source code after making changes as indicated in the above step and replace the kernel image in the boot images.

4 Validating the driver

The driver provides a simplified user interface for verifying the driver operation

STEP1:

The first step after loading the new linux kernel image with driver is to check if the driver really loaded and detected the ISL29037 sensor wired to the target platform. Use the below commands \

```
root@android # cd /sys/kernel/isl29037

root@android # ls

debug

reg_map

enable_als

enable_prox
```

User Interface	Description
Debug	This interface enables user to see 1. Configuration data for sensor 2. IRDR Current 3. Proximity sleep time 4. Range 5. Als/Prox Persistency 6. Ambient/Prox light 7. Low and high Als/prox threshold 8. Prox as well as Current Prox
enable_als	Enable/Disable the Als mode
enable_prox	Enable/Disable the Prox mode
reg_map	Enables register dump of sensor and writing to individual registers of device

4.1 Debug Interface

The debug interface allows the user to monitor the driver configuration, data and thresholds. To use this interface use the below commands

```
root@android # cat /sys/kernel/isl29037/debug
```

The output would look like as shown below

```
------------------------------------------------------------------------
Configration Register:
------------------------------------------------------------------------

CONFIG0                 CONFIG1                 INTR CONFIG
-------                 -------                 -----------
IRDR Current: 31.25mA   ALS Range  : 125  lux   INT conf   : ALS or PROX
PROX Sleep  : 400  ms   ALS en     : Enable     ALS Perst  : 1
PROX En     : Disable   Prox Offset: 0          ALS Flag   : 1
                        INT Algo   : Window     Pwr Fail   : Normal
                                                PROX Perst : 1
                                                PROX Flag  : 0

------------------------------------------------------------------------
Data Register:
------------------------------------------------------------------------

ALS Data  : 34          curr Prox  : 0          Obj Pos    : NEAR
ALS LT    : 192         PROX LT    : 0          Prox Wash  : 0
ALS HT    : 204         PROX HT    : 255        PROX Ambir : 0
ALSIR Comp: 0           Prox       : 0
------------------------------------------------------------------------
```

This command would the print the current information once. In order to get a continuous display of all data use the below command.

```
root@android # while true; do cat /sys/kernel/isl29037/debug; sleep 1;
busybox clear;done
```

Parameter configuration using debug interface

Parameter	Command (Example usage)	Value range
Als Interrupt persistency	echo als_persist 2 > debug	0, 1, 2, 3
Prox Interrupt persistency	echo prox_persist 2 > debug	0, 1, 2, 3
Range	echo range 3 > debug	0, 1, 2, 3
Irdr current	echo irdr_curr 1 > debug	0, 1, 2, 3
Proximity sleep time	echo prox_slp 5 > debug	0, 1, 2, 3, 4, 5, 6, 7
Als low interrupt threshold	echo als_lt 350 > debug	0 to 4095
Als high interrupt threshold	echo als_ht 2000 > debug	0 to 4095
Prox low interrupt threshold	echo prox_lt 50 > debug	0 to 255
prox low interrupt threshold	echo prox_ht 150 > debug	0 to 255

IMPORTANT NOTE: In case of write threshold through debug we can just write

echo als_lt 350 > debug

this will automatically adjust that particular value in particular low and high threshold MSB and LSB register

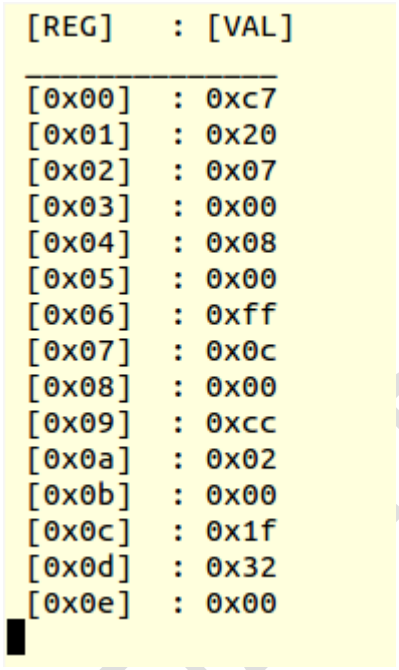
4.2 Reg_map Interface

The reg_map is a convenient interface to do register level debugging of the driver

Use the below command to dump the entire register of ISL29037 sensor device

```
root@android # cat /sys/kernel/isl29037/reg_map
```

The dumped register map would be as shown below



Use the below command to write to a particular register with particular value

```
root@android # echo <reg> <val> > reg_map

e.g

root@android # echo 0x02 0x67 > reg_map
```

5 Calibration Process

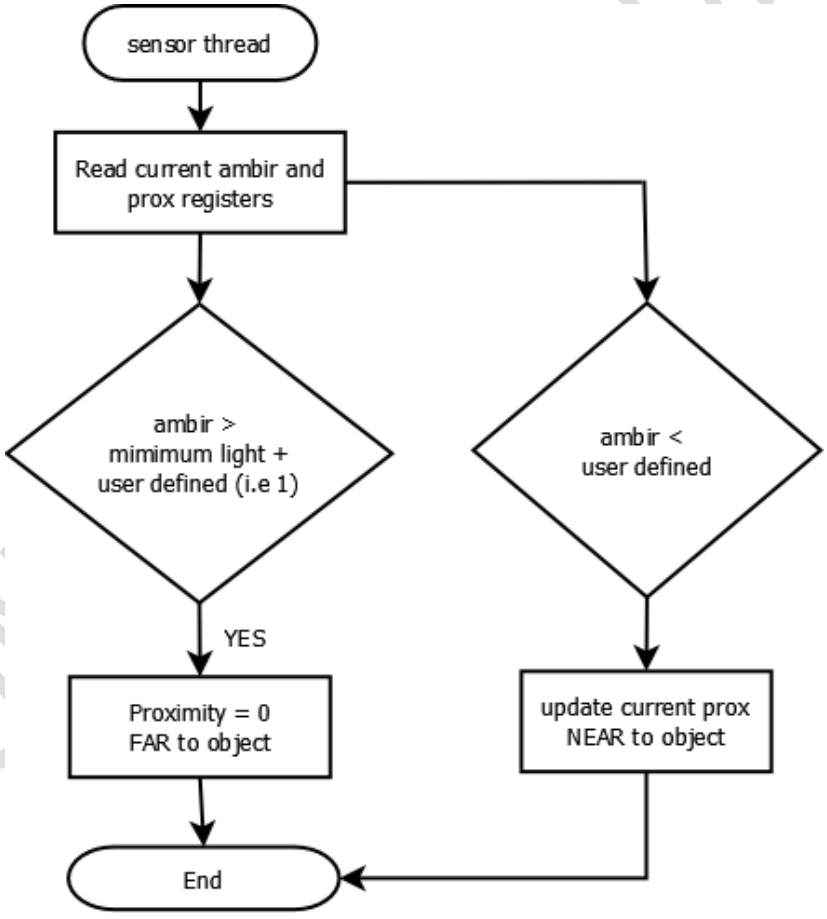
The calibration process would help us to measure the Ambient light intensity as well as Proximity and other parameter value fall on the sensor device.

Follow the below instructions to

- Automatically adjust the wash out on behalf of proximity value of the sensor
- To choose appropriate low and high thresholds for end product use case

4.3 Wash-out

The wash-out functionality is explained in the below diagram, this function is called every time when the driver thread will execute.



1. After integration of the driver with linux kernel. washout sequence will call every polling of high resolution timer.
2. The driver would try to compute the prox value always on the condition Prox Ambir > “minimum light” + user defined (user defined = 1).
3. Once the driver loads, it auto computes the prox value.
4. Use the debug interface of driver to know what is the current **configuration** by using command

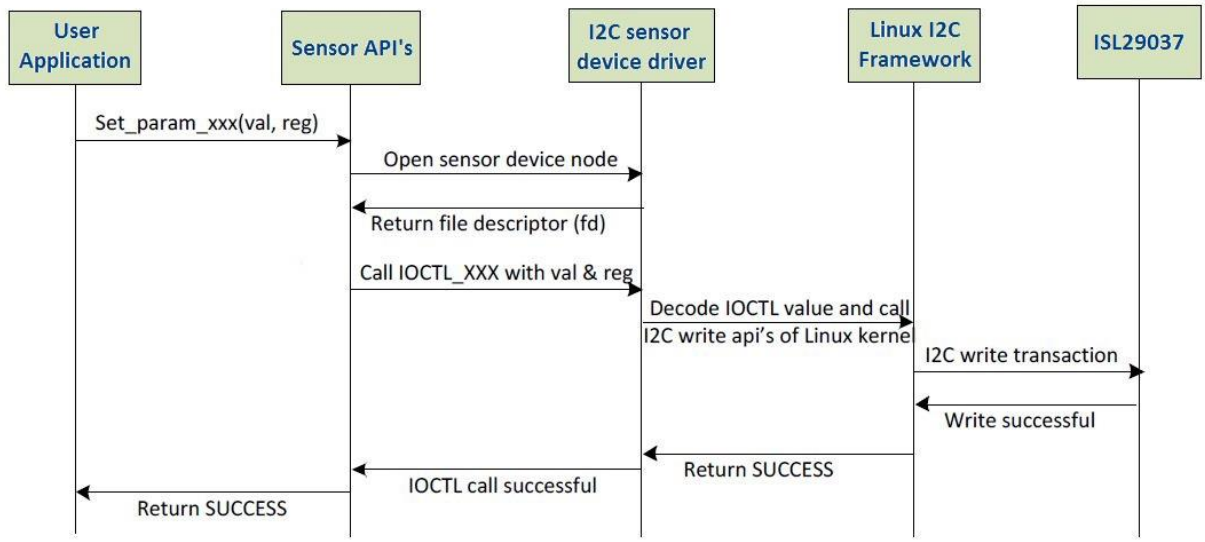
```
$ cat /sys/kernel/isl29037/debug
```

NOTE:

In the debug interface the variables curr prox and prox, shows the actual proximity data from the sensor device and wash-out prox data (using algorithm) respectively.

4.4 Programming Device Parameters

The programming of device parameters is taken care by IOCTL implementation of the driver. Each SYSFS can Read/Write certain parameters to program the device. The general sequence diagram for programming a device parameter from a test application is shown below.



4.5 Threshold Adjustment

The als/prox high and low interrupt thresholds are the parameters which define the region of interest at a particular light from the sensor.

These threshold high and low level decides the intrrupt occurancy on the sensor device. On every particular clock cycle it will generate an interrupt if it crosses the high threshold range and below the low threshold range.that range could be decide by the user.

1. Program the new HIGH and LOW threshold to the driver header file isl29037.h as

e.g

#define ALS_HI_THRESHOLD 0x0C0 (5% of maximum range)

#define ALS_LO_THRESHOLD 0x0CC (80% of maximum range)

2. We can change the low and high threshold values as the user requirement using below command.

e.g.

```
root@android # echo als_lt 200 > debug

root@android # echo prox_lt 200 > debug
```

At this point we have a driver which is calibrated for HIGH and LOW thresholds.