Software Design Description (SDD)

Rev: A1

17 Oct 2013

Android device driver for ISL29035 sensor
For
Intersil Corporation

VVDN Contact:

Bhupender Saharan

VVDN Technologies

+1 408 807 3951

bhupi@vvdntech.com

Revision History:

Date	Rev No.	Description	By
17 Oct 2013	A1	VVDN_ISLU_SNSR_SDD_ISL29035_A1.pdf	VVDN

Table of Contents

Table of Figures

1 Introduction

This document describes software components of ISL29035 device driver for Android Jellybean (version 4.2) and Android Ice cream sandwich (version 4.0.4) which is to be developed by VVDN for Intersil Corporation.

This SDD is made for the reference of

1. Product managers and QAD at VVDN & Intersil to understand the software development phases.
2. Engineering Team at VVDN for System Architecture, Design and development of device driver.
3. System Integration and Verification teams at VVDN / Intersil for SW validation.

2 Scope of the document

This document describes the software design specification of ISL29035 device driver in detail.

3 Reference documents

S.NO	Description	Revision	Date
1	VVDN_ISLU_SNSR_SDD_ISL29035_A1.pdf	A1	17 Oct 2013
2			
3			

4 Overview of project

Intersil manufactures **ISL29035**, an Ambient light sensor device. It is a low power, high sensitivity Ambient light sensor with an I2C (SM Bus compatible) interface. Its advanced self-calibrated photodiode array emulates human eye response with excellent IR rejection. The scope of this project is to develop and test ISL29035 device driver on Android Operating system.

5 Software Requirement

5.1.1 Operating system requirement

The sensor device driver should be supported on following operating systems

1. Android Jelly Bean (version 4.2)
2. Android Ice cream sandwich (version 4.0.4)

5.1.2 Sensor Operations

The sensor device driver should support the following operations

1. Configure the ADC resolution
   a. 4 – Bit
   b. 8 – Bit
   c. 12 – Bit
   d. 16 – Bit

2. Configure device operating modes
   a. Power down
   b. Continuous IR
   c. Continuous ALS
   d. IR once
   e. ALS once

3. Configure the 16-bit High and 16-bit Low interrupt thresholds for ALS/IR.

4. Configure the Interrupt persistency
   a. 1 Integration Cycle
   b. 4 Integration Cycles
   c. 8 Integration Cycles
   d. 16 Integration Cycles

5. Configure the ALS / IR range
   a. 1000 Lux
   b. 4000 Lux
   c. 16000 Lux
   d. 64000 Lux

6 System Block Diagram

The Intersil ISL29035 sensor would be interfaced to a TI's OMAP4460 based Panda-board via an I2C and Interrupt line.

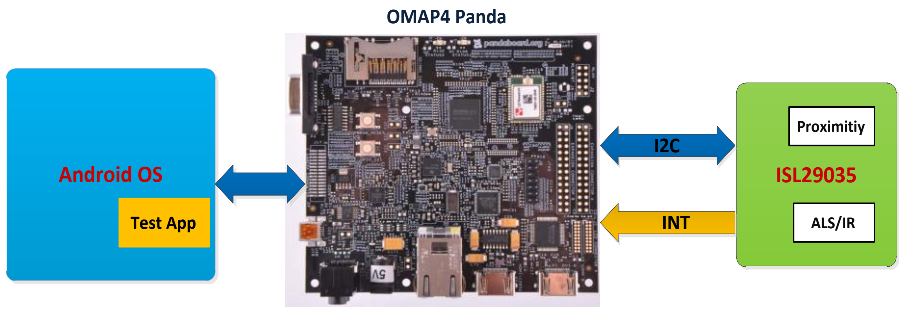

Figure 1 System block diagram

I2C interface would be used to:

1. Configure the sensor device
2. Read the sensor device Configurations and Parameters
3. Read the Ambient Light luminance and IR Digital counts

Sensor interrupt pin would be connected to an interruptible GPIO pin on Panda-board.

7 Software Architecture

The system software architecture diagram is shown below.

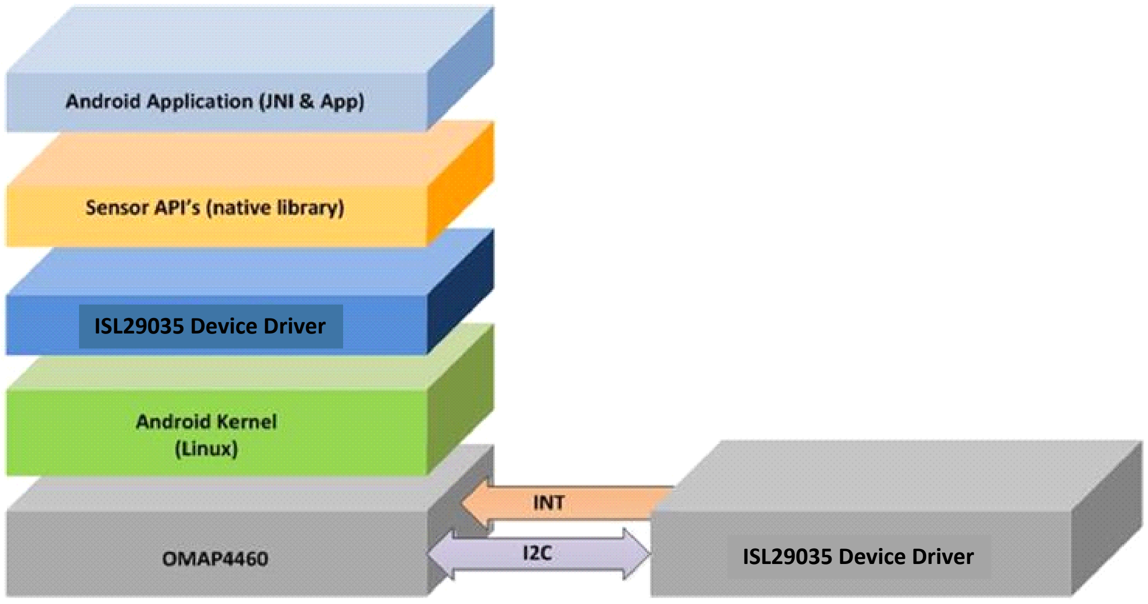

Figure 2 Software architecture diagram

Omap4460 Panda-board is the hardware platform for initial deployment and testing of the driver. The Android ICS Operating system with android kernel (Version 3.2.0) is running to support the Device driver functionality. An android demo test application is provided to test the parameters and configurations of the sensor. This test application will be running on top of the android operating system.

The test application is designed to read the sysfs device files exported by ISL29035 driver. These sysfs files will be updated on every frequent read or write operation.

Note: The Test application is just a Demo application to demonstrate the basic functionality of ISL29035 driver. For the best view of application use HDMI LCD Screen of resolution 17'.

8 Device driver use case model

The use case model for ISL29035 device driver is shown below. The functionalities provided by the device driver are as follows. Omap4460 System on Chip (panda-board) is used to host the driver for ISL29035 sensor service.

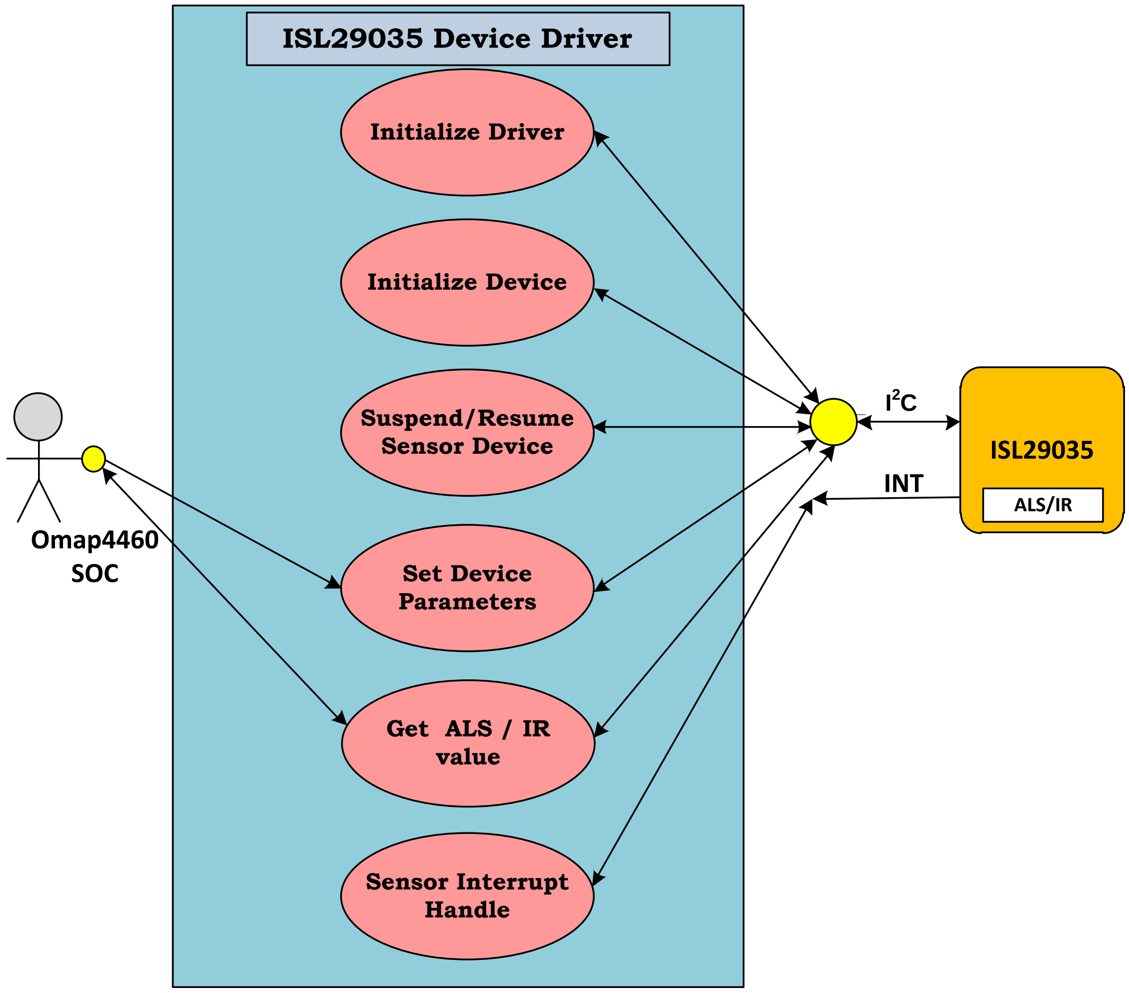

Figure 3 Sensor device driver use case diagram

8.1.1 Initialize driver

The driver initialization code does the following

1. Registers the I2C device driver with the Linux I2C core framework.
2. Requests and set up the GPIO line for handling the sensor interrupts.
3. Registers the Interrupt handlers for the IRQ number corresponding to the sensor interrupt line.

8.1.2 Initialize the device

Once the I2C core framework recognizes the ISL29035 (slave address 1000100b or 0x44h) on the I2C bus. It calls the probe function of the device driver (that supports/services the ISL29035 device) which verifies the ALS sensor device, and writes the initial configuration parameters to the device. At this stage the sensor device is ready to operate in the default operating conditions.

8.1.3 Get ALS Lux and IR counts

The Ambient light intensity is read (in the form of ADC code) from the ISL29035 sensor device registers shown in the below table (Values are stored in two 8-bit registers as MSB and LSB). The values are read when the user application sends an IOCTL request as per sysfs.

REG	**DATA**
0x03h	**MSB data**
0x02h	**LSB data**

The Data in MSB and LSB together called as ADC code.

Conversion between Lux and ADC code is:

Lux = (Range x ADC code) / 2 ^ n ; where n is ADC resolution

ADC code is represented in Hex code.

8.1.4 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. Please refer the Application Programmer's Guide for valid read/write parameters.

The general sequence diagram for programming a device parameter from a test application is shown below.

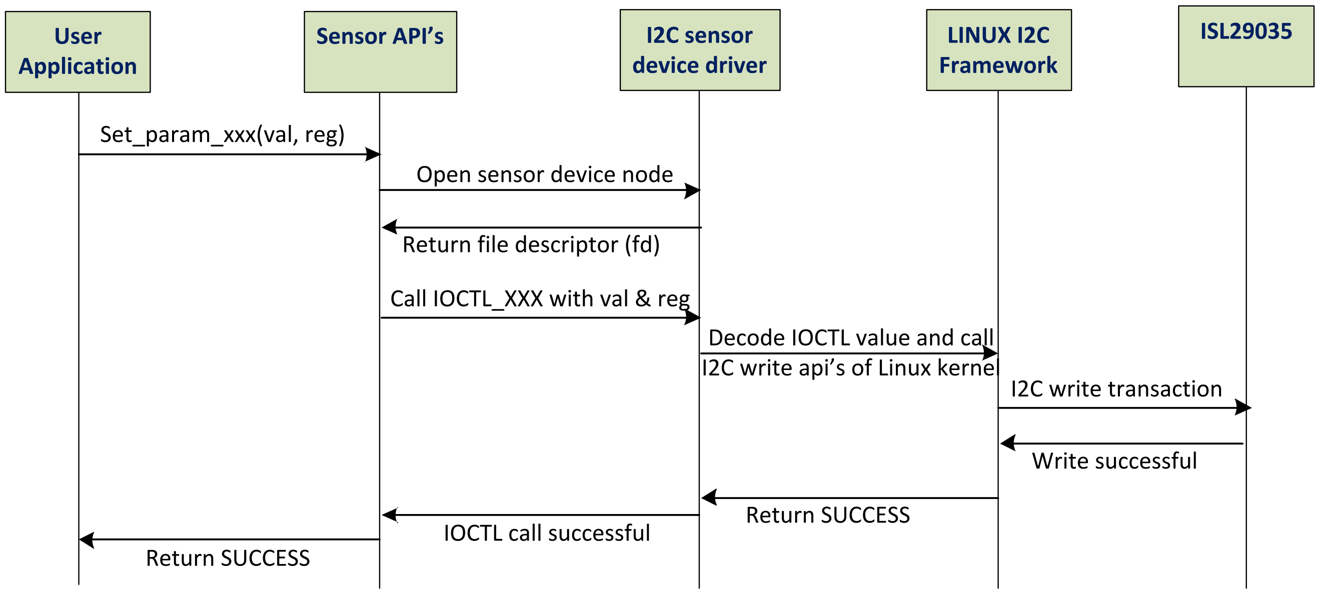

Figure 4 Sequence diagram for programming sensor device parameter

8.1.5 Device Operation mode

The device operation mode indicates the operating mode of the device. Sensor device supports the following operating modes.

1. Power down
2. ALS once
3. IR once
4. ALS continuous
5. IR continuous

The device operation mode indicates the device state and which light's component is being converted by ADC. The device operation mode is programmed by writing bits [D7:D5] in Command Register -1 (0x00h) of sensor device.

Add	Register	D7	D6	D 5	D 4:D3	D2	D1	D0
0x00h	Command I	OP2	OP1	OP0	RESERVED	INT	PRST1	PRST0

8.2 Full scale range (FSR)

FSR defines the sensitivity range of the sensor device. There are four FSR ranges for ALS/IR sensing.

8.2.1 ALS Sensing Ranges

1. 1000 Lux
2. 4000 Lux
3. 16000 Lux
4. 64000 Lux

This parameter is programmed by writing to the **Data sensing range** bits [D1:D0] in Command-II register (0x01h) of sensor device.

Add	Register	D7	D6	D5	D4	D3	D2	D1	D0
0x01h	Command II	RESERVED				RES1	RES0	Range1	Range0

D1	**D0**	**FSR Sensing Range**
0	0	1,000
0	1	4,000
1	0	16,000
1	1	64,000

8.2.2 ADC resolution

The ADC conversion mode can be set to four resolutions

1. 4 - bit
2. 8 - bit
3. 12 - bit
4. 16 - bit

The ADC resolution affects the Lux per count of sensor device for different Lux ranges.

This parameter is programmed by writing to the **ADC resolution** bits [D3:D2] in Command-II (0x01h) register of sensor device.

Add	Register	D7	D6	D5	D4	D3	D2	D1	D0
0x01h	Command II	RESERVED				RES1	RES0	Range1	Range0

D3	D2	ADC Resolution (Bits)
0	0	16
0	1	12
1	0	8
1	1	4

8.2.3 Interrupt threshold

The interrupt threshold value defines a window of Lux values. Whenever the current ADC measurement reaches above or below this threshold window, the sensor device raises an interrupt to the processor.

There are two thresholds for ALS / IR

1. High threshold
2. Low threshold

These parameters are programmed by writing to the following registers

REG	DATA	REG	DATA
0x04h	Interrupt Low threshold-1(LSB)	0x06h	Interrupt High threshold-1 (LSB)
0x05h	Interrupt Low threshold-2 (MSB)	0x07h	Interrupt High threshold-2 (MSB)

8.2.4 Interrupt persistency

Interrupt persistency is used to avoid false interrupt cases due to noise or sudden spikes in the ambient light conditions. This parameter defines the ‘N’ number of consecutive interrupt threshold events after which the interrupt will be raised to processor and interrupt flag will be set.

This parameter is programmed by writing to the **Interrupt persist control bits** [D1:D0] of Command-I register (0x00h) of sensor device.

Add	Register	D7	D6	D 5	D 4	D3	D2	D1	D0
0x00h	Command I	OP2	OP1	OP0	RESERVED		INT	PRS1	PRS0

8.2.5 ID Register

The ID Register has three different type of information. Using the identification bits to identify the device in a system the software should mask the Bit0 - Bit2 and Bit6 – Bit7 to properly identify the device.

The ISL29035 provides **3-bits** to identify the device in a system. These bits are located on register address **0x0F, Bit3 – Bit5**.The device identification bits are read only bits.

The default value of this bit is **BOUT = 1** during the initial power-up, which indicates the device may possibly have gone through a brownout condition. The status bit should be reset to **BOUT = 0** by an I2C write command during the initial configuration of the device.

Add	Register	D7	D6	D 5	D 4	D3	D2	D1	D0
0x0Fh	ID	BOUT	RESERVED	1	0	1	RESERVED		

8.3 Sensor device suspend/resume

The drivers suspend and resume routines does the following operation.

1. Put the sensor in standby mode when android OS sleeps
2. Wake the sensor from standby when android OS wakes up

Both these events are taken care in the device driver by use of driver's suspend and resume callback functions.

8.4 Handle sensor interrupts

The ALS sensor device has the following sources of interrupt

1. ALS value exceeds the lower or upper threshold
2. ADC completed the conversion

The interrupt handling scheme is explained in the below flowchart.

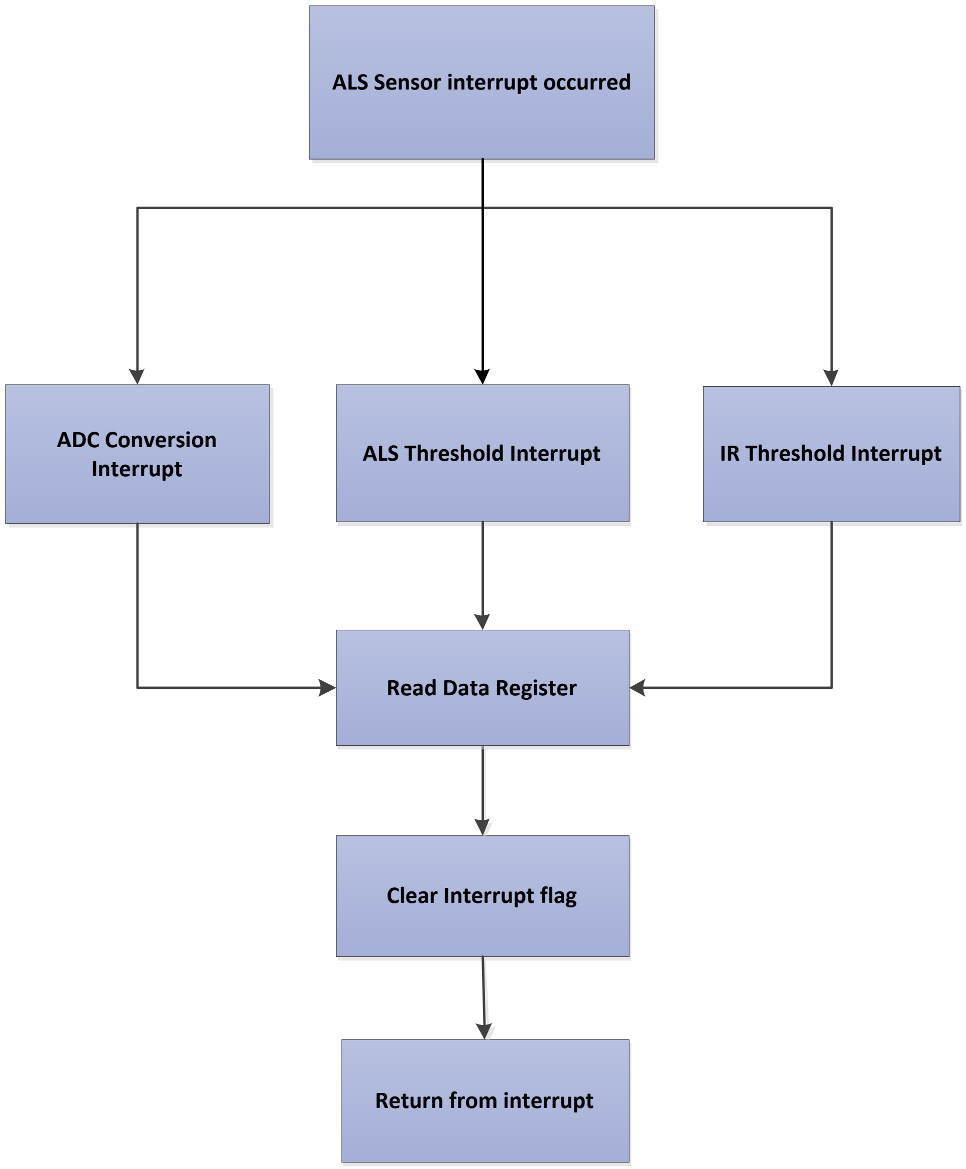

Figure 5 Sensor device interrupt handling

9 ALS sensor Programmer's Guide

Note: A Detailed programmer's Guide for ISL29035 ALS sensor's **sysfs** access would be provided along with the device driver.