ISL29023: Android Driver Integration Guide

Rev: A1

27th Aug 2014

Contact:

INTERSIL CORPORATION
(USA)

Revision History

Date	Rev No.	Description	By
August 27, 2014	A1	Android driver integration guide	Intersil corp

Table of Contents

1 Introduction

This document describes the steps involved in integrating the ISL29023 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	ISL29023
Software	Android driver for sensor
Driver Version	1.0
Released files	1. isl29023.c 2. isl29023.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 isl29023.c file to the following directory in the android kernel source code

<kernel top directory>/drivers/input/misc

STEP 2:

Copy the isl29023.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

Refer to the released patch (**patch-isl29023-linux-kernel.patch**) to change the board file to add support for ISL29023 device detection during booting of the target platform.

STEP 4:

Edit the Makefile in the following directory to add build support for isl29023 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 isl29023 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_ISL29023 = 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 ISL29023 sensor wired to the target platform. Use the below commands

```
root@android # cd /sys/kernel/isl29023

root@android # ls

debug

reg_map
```

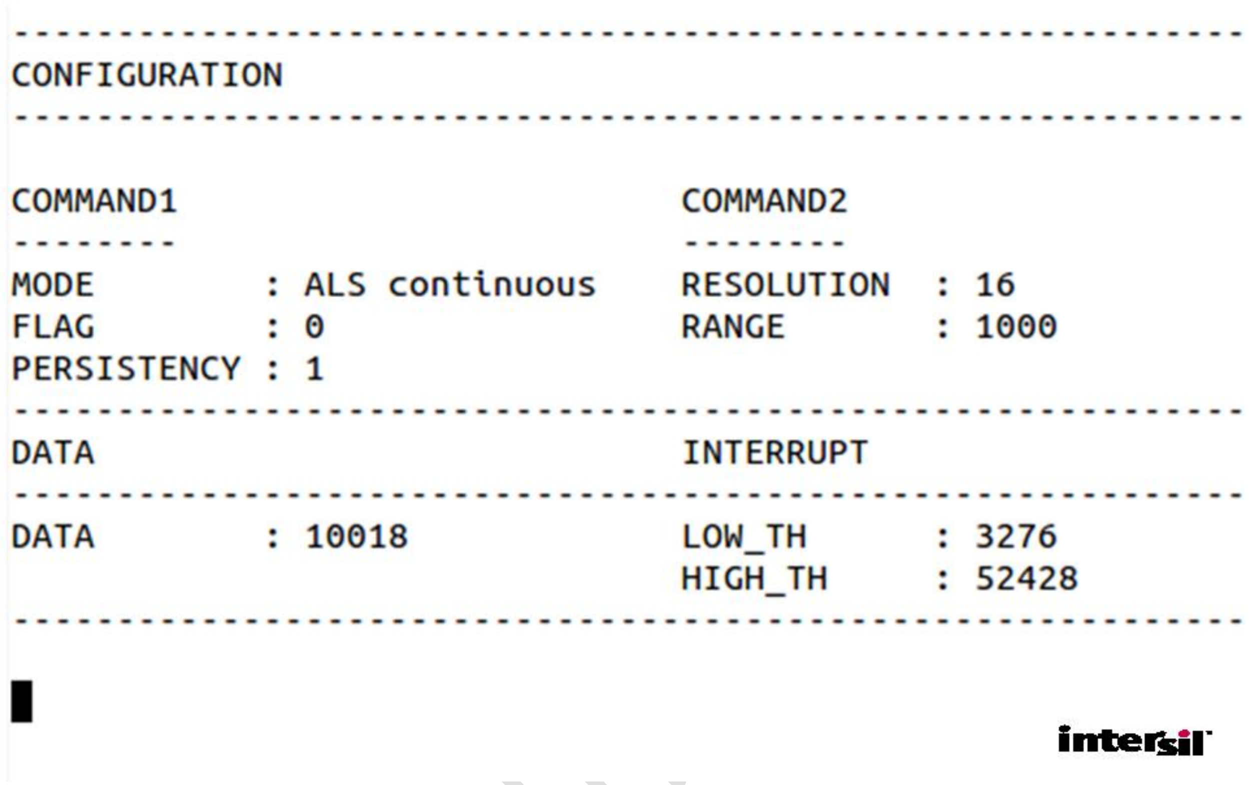
User Interface	Description
Debug	This interface enables user to see 1. Configuration data for sensor 2. Operation mode 3. Resolution 4. Range 5. Persistency 6. Ambient light 7. Low and high threshold
reg_map	Enables register dump of sensor and writing to individual registers of device

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/isl29023/debug
```

The output would look like as shown below



```
------------------------------------------------------------
CONFIGURATION
------------------------------------------------------------

COMMAND1                          COMMAND2
--------                          --------
MODE        : ALS continuous      RESOLUTION  : 16
FLAG        : 0                   RANGE       : 1000
PERSISTENCY : 1
------------------------------------------------------------
DATA                              INTERRUPT
------------------------------------------------------------
DATA        : 10018               LOW_TH      : 3276
                                  HIGH_TH     : 52428
------------------------------------------------------------
```

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/isl29023/debug; sleep 1;
busybox clear;done
```

Parameter configuration using debug interface

Parameter	Command (Example usage)	Value range
Interrupt persistency	echo persist 2 > debug	0, 1, 2, 3
Range	echo range 3 > debug	0, 1, 2, 3
Resolution	echo resolution 1 > debug	0, 1, 2, 3
Operation mode	echo mode 5 > debug	0, 1, 2, 5, 6 (3/4/7 reserved)
Low interrupt threshold	echo lt 350 > debug	0 to 65535
High interrupt threshold	echo ht 2000 > debug	0 to 65535

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

echo lt 350 > debug

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

Ex. For writing the low threshold value as 0x015E. we have to write as

0x04 = 0x015E (Low threshold LSB) it will automatically write in 0x05 register using internal logic

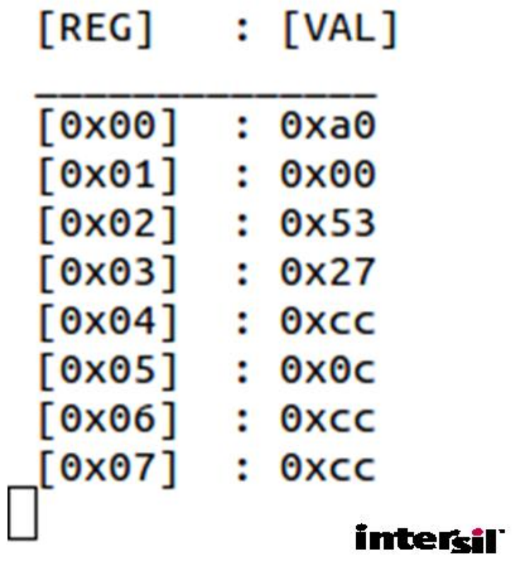
4.2 Reg_map Interface

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 ISL29023 sensor device

```
root@android # cat /sys/kernel/isl29023/reg_map
```

The dumped register map would be as shown below



```
[REG]    : [VAL]
______________
[0x00]   : 0xa0
[0x01]   : 0x00
[0x02]   : 0x53
[0x03]   : 0x27
[0x04]   : 0xcc
[0x05]   : 0x0c
[0x06]   : 0xcc
[0x07]   : 0xcc
```

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
```

IMPORTANT NOTE: In case of write to a 16 bit perticular register (Threshold rester) through regmap we have to follow a particular format for writing data as 8 bit

Ex. For writing the low threshold value as 0x015E. we have to write as

0x04 = 0x5E (Low threshold LSB)

0x05 = 0x01 (Low threshold MSB)

5 Calibration Process

The calibration process would help us to meassure the Ambient light intensity value fall on the sensor device .

Follow the below instructions to

- Automatically adjust the ADC resolution and range on the behalf of ambient light intensity
- To choose appropriate low and high thresholds for end product use case

4.3 Autorange

1. After integration of the driver with linux kernel. autorange sequence will call every interrupt of high resolution timer.
2. the driver would try to compute the range between 10,000 to 16,000 lux from the sensor in any circumstrances.
3. Once the driver loads, it auto computes the range and resolution to be done.
4. Use the debug interface of driver to know what is the current **configuration** by using command

```
$ cat /sys/kernel/isl29023/debug
```

4.4 Threshold Adjustment

The als 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 isl29023.h as

```
e.g

#define ALS_HI_THRESHOLD        3276 (5% of maximum range)

#define ALS_LO_THRESHOLD        52428 (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 lt 200 > debug
```

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