ISL29177: Android Driver Integration Guide

Rev: A1

30th Jul 2014

VVDN Contact:

Name: Murali Mohan

Email: murali.m@vvdntech.com

Revision History:

Date	Rev No.	Description	By
July 30, 2014	A1	Android driver integration guide , initial version	VVDN

Table of Contents

1 Introduction

This document describes the steps involved in integrating the ISL29177 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	ISL29177
Software	Android driver for sensor
Driver Version	1.0
Released files	1. isl29177.c 2. isl29177.h 3. patch-isl29177-linux-kernel.patch
Features	1. Supports data monitoring of sensor in user friendly format 2. Supports configuring major sensor parameters via debug interface 3. Supports one time calibration to compensate for crosstalk from IR transmitter 4. Simple user interface to dump and configure any required device register

3 Driver integration

STEP 1:

Copy the isl29177.c file to the following directory in the android kernel source code

<kernel top directory>/drivers/input/misc

STEP 2:

Copy the isl29177.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-isl29177-linux-kernel.patch**) to change the board file to add support for ISL29177 device detection during booting of the target platform.

STEP 4:

Edit the Makefile in the following directory to add build support for isl29177 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 isl29177 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_ISL29177 = 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 ISL29177 sensor wired to the target platform. Use the below commands

```
root@android # cd /sys/kernel/isl29177

root@android # ls

debug

enable

reg_map
```

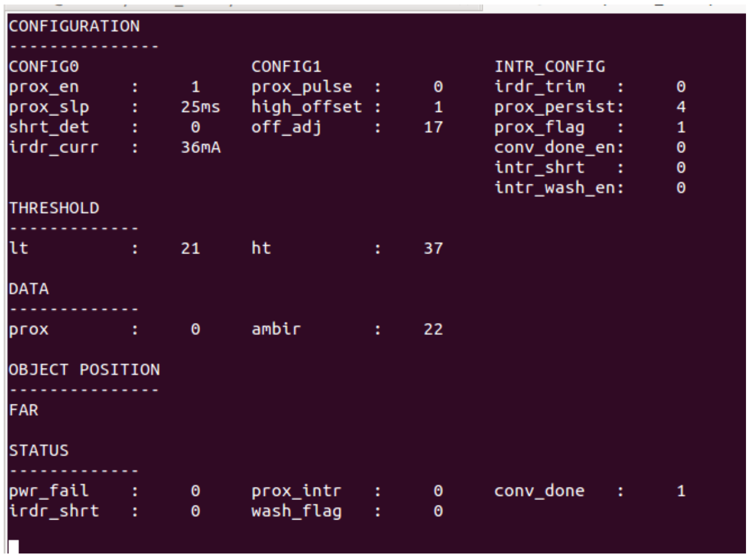
User Interface	Description
debug	This interface enables user to see 1. Configuration data for sensor 2. Prox data 3. Ambient IR 4. Low and high thresholds 5. Object position (NEAR / FAR) 6. Interrupt status
enable	Enable or Disable proximity conversion
reg_map	Enables register dump of sensor and writing to individual registers of device

4.1 Debug Interface

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

```
root@android # cat /sys/kernel/isl29177/debug
```

The output would look like as shown below



```
CONFIGURATION
---------------
CONFIG0                    CONFIG1                    INTR_CONFIG
prox_en     :     1        prox_pulse  :     0        irdr_trim   :     0
prox_slp    :    25ms      high_offset :     1        prox_persist:     4
shrt_det    :     0        off_adj     :    17        prox_flag   :     1
irdr_curr   :    36mA                                 conv_done_en:     0
                                                      intr_shrt   :     0
                                                      intr_wash_en:     0

THRESHOLD
-------------
lt          :    21        ht          :    37

DATA
-------------
prox        :     0        ambir       :    22

OBJECT POSITION
---------------
FAR

STATUS
-------------
pwr_fail    :     0        prox_intr   :     0        conv_done   :     1
irdr_shrt   :     0        wash_flag   :     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/isl29177/debug; sleep 1;
busybox clear;done
```

Parameter configuration using debug interface

Parameter	Command (Example usage)	Value range
Interrupt persistency	echo prox_persist 2 > debug	0, 1, 2, 3
IRDR current	echo irdr_curr 3 > debug	0, 1, 2, 3,4, 5, 6, 7
Prox sleep	echo prox_slp 1 > debug	0, 1, 2, 3,4, 5, 6, 7
Low interrupt threshold	echo lt 15 > debug	0 to 255
High interrupt threshold	echo ht 25 > debug	0 to 255

4.2 Enable Interface

Use the below command to enable driver operation and state machine (Driver loads in enable mode by default)

```
root@android # echo 1 > /sys/kernel/isl29177/enable
```

Use the below command to disable driver operation and state machine

```
root@android # echo 0 > /sys/kernel/isl29177/enable
```

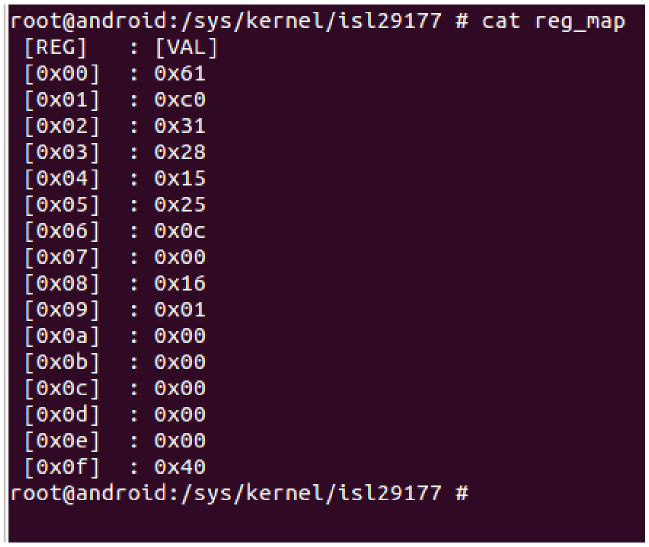
4.3 Reg_map Interface

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 ISL29177 sensor device

```
root@android # cat /sys/kernel/isl29177/reg_map
```

The dumped register map would be as shown below



```
root@android:/sys/kernel/isl29177 # cat reg_map
 [REG]   : [VAL]
 [0x00]  : 0x61
 [0x01]  : 0xc0
 [0x02]  : 0x31
 [0x03]  : 0x28
 [0x04]  : 0x15
 [0x05]  : 0x25
 [0x06]  : 0x0c
 [0x07]  : 0x00
 [0x08]  : 0x16
 [0x09]  : 0x01
 [0x0a]  : 0x00
 [0x0b]  : 0x00
 [0x0c]  : 0x00
 [0x0d]  : 0x00
 [0x0e]  : 0x00
 [0x0f]  : 0x40
root@android:/sys/kernel/isl29177 #
```

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

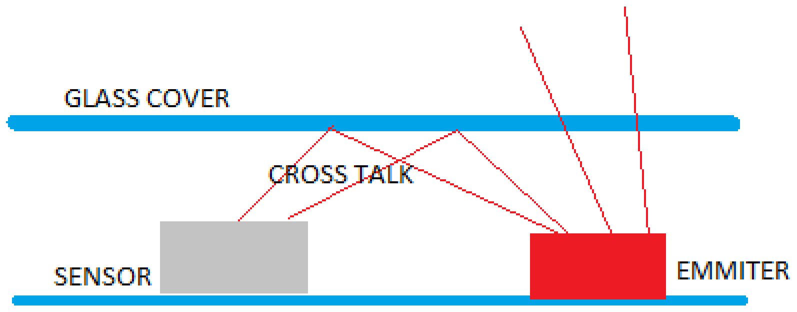
```
root@android # echo <reg> <val> > reg_map

e.g

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

5 Calibration Process

The calibration process would help us to eliminate the non zero proximity caused due to cross talk from the IR transmitter behind the glass cover where the sensor would be placed.



Follow the below instructions to

- Compensate for the cross-talk from the IR transmitter
- To choose appropriate low and high thresholds for end product use case

5.1 Offset Adjustment

1. After integration of the driver with linux kernel. Build the driver by having the following macro defined in the driver file isl29177.c

```
#define ISL29177_CALIBRATION
```

2. When the above macro is defined in the driver , the driver would try to compute the non zero proximity count read from the sensor in absence of any object. This baseline reading is a consequence of cross-talk.

CONDITION: The condition for calibration is that

a. It should be run without any object in front of sensor.

b. It should be run under normal lighting conditions.

3. Once the driver (with calibration enabled) loads, it auto computes the compensation to be done i.e. **proximity offset adjust** value.

4. Use the debug interface of driver to know what is the current **off_adj** by using command

```
$ cat /sys/kernel/isl29177/debug
```

From the output of the above command note the offset adjustment value and fill it in the driver header file isl29177.h

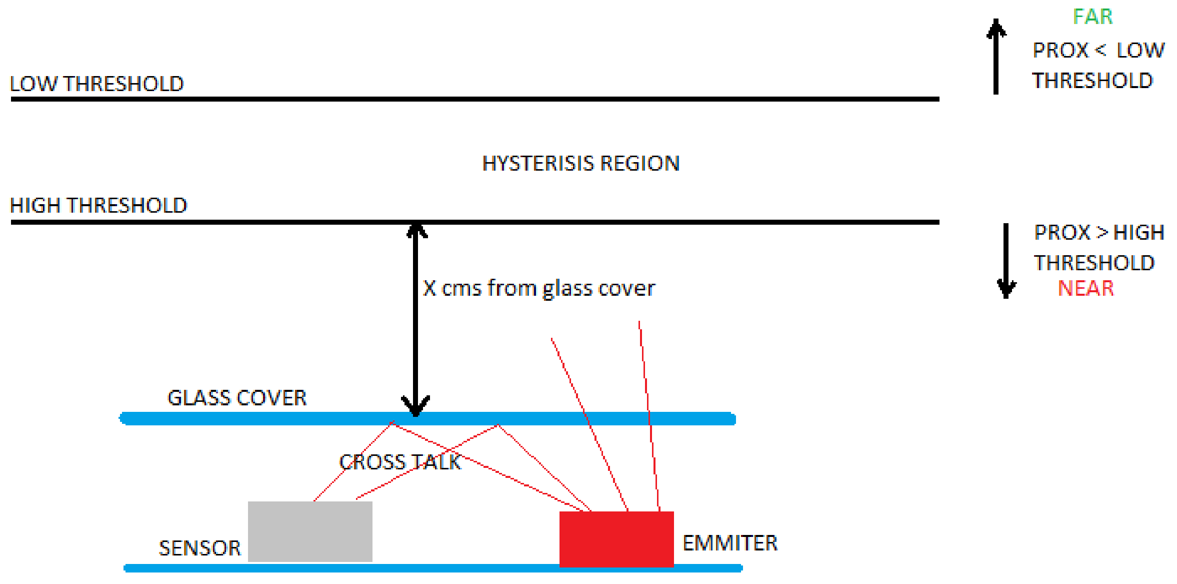
e.g

```
#define OFFSET_ADJUST   17
```

5. Now rebuild the driver by commenting the definition of the macro ISL29177_CALIBRATION. The driver now would directly configure the offset adjust programmed in the header file. At this point the cross-talk from the IR transmitter would be compensated.

5.2 Threshold Adjustment

The proximity high and low interrupt thresholds are the parameters which define the region of interest at a particular distance from the glass cover in front of sensor which would have the ISL29177 sensor behind it.



In order to achieve the required NEAR and FAR regions as per the requirement (distance 'X' cms from glass cover), follow the below steps

1. Keep an object at a fixed distance from the glass cover
2. By default the object distance would be indicated as **FAR** in the debug interface output
3. Now set the threshold to 255 (0xFF) and start decreasing it in steps using debug interface until the debug interface indicates the object as **NEAR**

e.g.

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

4. Once the NEAR threshold (HIGH Threshold) is found, compute the FAR threshold (LOW Threshold) as

   LOW Threshold = High Threshold – 16

5. Program the new HIGH and LOW threshold to the driver header file isl29177.h as

e.g

```
#define PROX_HI_THRESHOLD      46

#define PROX_LO_THRESHOLD      30
```

At this point we have a driver which is compensated for cross-talk and satisfied the required HIGH and LOW thresholds.