

# ISL29177: Android Driver Integration Guide

---

*Rev: A2*

*25<sup>th</sup> Aug 2014*

## **Contact:**

INTERSIL CORPORATION  
(USA)

**Revision History:**

<b>Date</b>	<b>Rev No.</b>	<b>Description</b>	<b>By</b>
August 25, 2014	A2	Android driver integration guide	Intersil corp
July 30, 2014	A1	Android driver integration guide , initial version	Intersil corp

*Table of Contents*

<b>1</b>	<b>INTRODUCTION .....</b>	<b>4</b>
<b>2</b>	<b>DRIVER RELEASE .....</b>	<b>4</b>
<b>3</b>	<b>DRIVER INTEGRATION.....</b>	<b>4</b>
<b>4</b>	<b>VALIDATING THE DRIVER .....</b>	<b>5</b>
4.1	DEBUG INTERFACE.....	6
4.2	ENABLE INTERFACE .....	8
4.3	REG_MAP INTERFACE.....	8
<b>5</b>	<b>CALIBRATION PROCESS.....</b>	<b>10</b>
5.1	RUNTIME SEQUENCE.....	10
5.2	THRESHOLD ADJUSTMENT.....	11

## 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

<b>Product</b>	ISL29177
<b>Software</b>	Android driver for sensor
<b>Driver Version</b>	1.0.1
<b>Released files</b>	<ol style="list-style-type: none"> <li>1. isl29177.c</li> <li>2. isl29177.h</li> <li>3. patch-isl29177-linux-kernel.patch</li> </ol>
<b>Features</b>	<ol style="list-style-type: none"> <li>1. Supports data monitoring of sensor in user friendly format</li> <li>2. Supports configuring major sensor parameters via debug interface</li> <li>3. Supports one time calibration to compensate for crosstalk from IR transmitter</li> <li>4. Simple user interface to dump and configure any required device register</li> </ol>

## 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	<p>This interface enables user to see</p> <ol style="list-style-type: none"> <li>1. Configuration data for sensor</li> <li>2. Raw Prox data</li> <li>3. Relative prox data (prox data - baseline)</li> <li>4. Baseline</li> <li>5. Ambient IR</li> <li>6. Low and high thresholds</li> <li>7. Object state (NEAR / FAR)</li> <li>8. Light (DIM / BRIGHT)</li> <li>9. Interrupt status</li> </ol>
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 SLEEP : 25 ms	HIGH OFFSET : 1	PROX PERSIST : 4
SHORT DET : 0	OFFSET ADJ : 21	PROX FLAG : 1
IRDR CURR : 12500 uA		CONV DONE EN : 0
		INTR SHORT : 0
		INTR WASH EN : 0
-----		
DATA -----	STATUS -----	
STATE : FAR	POWER FAIL : 0	
LIGHT : DIM	PROX INTERRUPT : 0	
	CONVERSION DONE : 0	
REL PROX : 0	IRDR SHORT : 0	
BASE LINE : 44	WASH FLAG : 0	
RAW PROX : 43		
WASH : 22		
THRES HI : 25		
THRES LOW : 20		
-----		

**intersil**™

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 <b>prox_persist</b> 2 > debug	0, 1, 2, 3
IRDR current	echo <b>irldr_curr</b> 3 > debug	0, 1, 2, 3, 4, 5, 6, 7
Prox sleep	echo <b>prox_slp</b> 1 > debug	0, 1, 2, 3, 4, 5, 6, 7
Low interrupt threshold	echo <b>lt</b> 15 > debug	0 to 255
High interrupt threshold	echo <b>ht</b> 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

[REG]	: [VAL]
[0x00]	: 0x61
[0x01]	: 0xc3
[0x02]	: 0x35
[0x03]	: 0x28
[0x04]	: 0x14
[0x05]	: 0x19
[0x06]	: 0x04
[0x07]	: 0x33
[0x08]	: 0x18
[0x09]	: 0x01
[0x0a]	: 0x01
[0x0b]	: 0x00
[0x0c]	: 0x00
[0x0d]	: 0x00
[0x0e]	: 0x00
[0x0f]	: 0x40



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

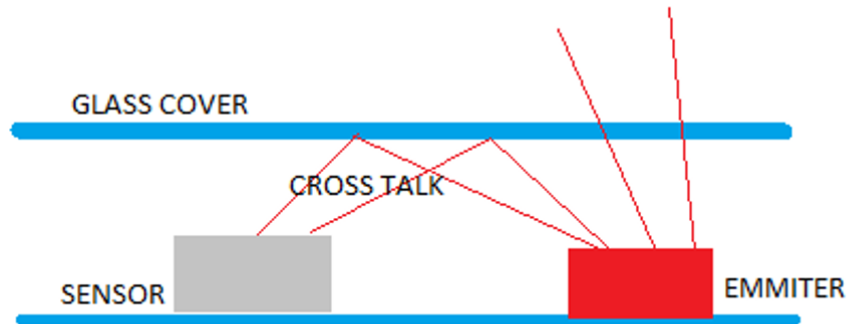
```
root@android # echo <reg> <val> > reg_map
```

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.



**intersil**™

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

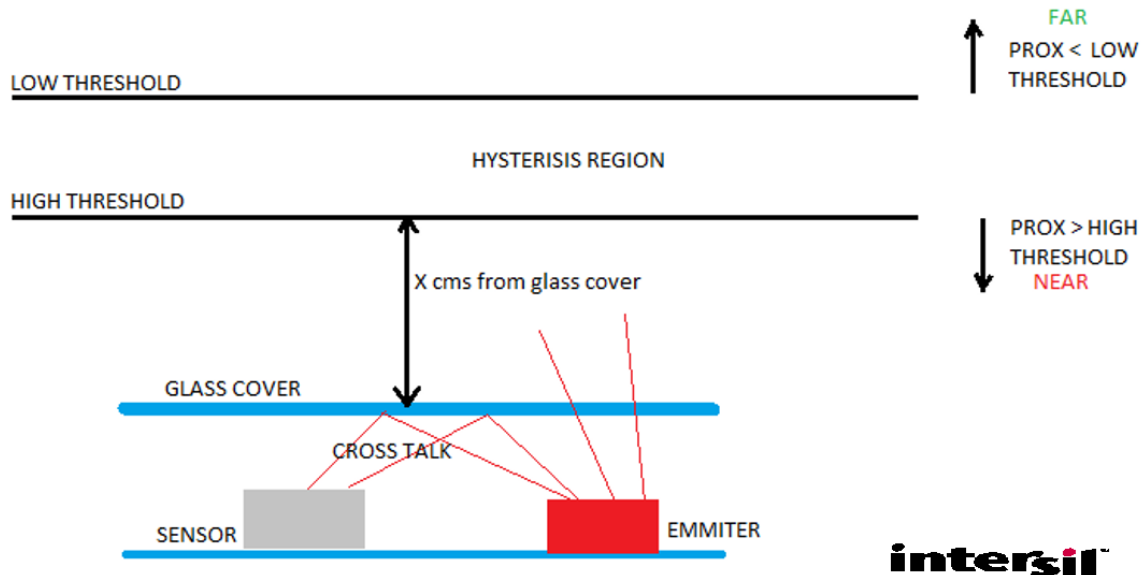
### 5.1 Runtime Sequence

1. After integration of the driver with linux kernel. Runtime sequence will call every interrupt of high resolution timer.
2. the driver would try to compute the proximity count between 10 to 100 from the sensor in any circumstances.
3. Once the driver 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
```

## 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  

$$\text{LOW Threshold} = \text{High Threshold} - 16$$
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.