Android Integration Guide for VCNL40x0 Proximity and Ambient Light Sensor

TOC

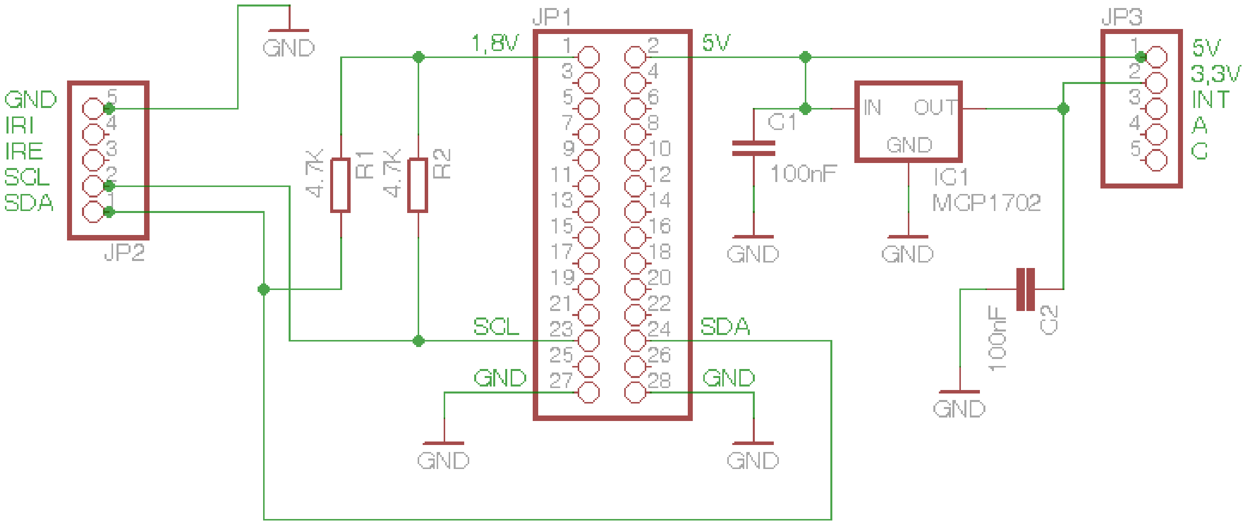
1. Introduction

This document describes the integration of a VNCL4020 sensor into a BeagleBorad.
The device driver for the VNCL4020 sensor consists of two parts. The kernel modules and the userspace module (HAL). Both are delivered as a patch file for BeagleBoard as reference platform.

Specification of reference hardware:

- Sensor: VCNL4020
- BeagleBoard xM (Rev. C)

Connection diagram between BeagleBoard and sensor:



2. Kernel Module

The Kernel Patch is the main device driver part. It contains configuration files that are device specific (for beagle board in this case) and the main device driver implementation.

2.1. *Sources Overview*

The following describes the files/functions and changes.

arch/arm/configs/omap3_beagle_android_vcnl4020_defconfig

Kernel config for Beagle Board → e.g. generated by „make menuconfig“ command
Copy of omap3_beagle_android_defconfig with enabled support for VCNL4020 kernel module

arch/arm/mach-omap2/board-omap3beagle.c

Contains I²C bus address of the VCNL4020 sensor

```
static struct i2c_board_info __initdata beagle_i2c_vcnl4020[] = {
       {
              I2C_BOARD_INFO("vcnl4020", 0x13), //0x13 → i²c bus address
       },
};


static int __init omap3_beagle_i2c_init(void)
...
omap_register_i2c_bus(2, 400, beagle_i2c_vcnl4020,
ARRAY_SIZE(beagle_i2c_vcnl4020)); //2 → bus address, 400 → clock frequency
...
```

drivers/misc/Kconfig

```
“config VCNL4020
tristate "VISHAY VCNL4020 proximity and ambient light sensor"
depends on I2C
help
  If you say yes here you get support for the VISHAY VCNL4020
  proximity and ambient light sensor.
  This driver can also be built as a module.  If so, the module
  will be called vcnl4020a.”
```

drivers/misc/Makefile

```
obj-$(CONFIG_VCNL4020)            += vcnl4020.o
```

drivers/misc/vcnl4020.c

Device driver implementation for light and proximity sensors.
The driver uses a worker queue (thread) to schedule the reading of sensor data. There is one combined worker queue for both sensors (proximity and light).
If sensor data changes an event is generated which is written to the corresponding device files.
The device files can be found in:

```
/sys/class/input/eventX/device/vcnl4020 → proximity sensor files
  prox_delay → time between measurements
  prox_enable → enable/disable sensor
  prox_input → contains last value read from sensor


/sys/class/input/eventY/device/vcnl4020 → ambient light sensor files
  lux_delay → time between measurements
  lux_enable → enable/disable sensor
  lux_input → contains last value read from sensor

For debug purpose all the device files are also available in folder
/sys/devices/platform/omap/omap_i2c.2/i2c-2/2-0013/vcnl4020
  lux_delay
  lux_enable
  lux_input
  prox_delay
  prox_enable
  prox_input
  status → additional file contains the raw data from both sensors
```

Special note: It is not possible to generate an event for the same sensor value twice. To ensure the correct value is delivered when re-enabling the sensor the driver set the value to “-1” when it is disabled. HAL Layer will filter this value later. If Sensor is re-enabled the actual value is sent to HAL layer.

The following struct shows the entry point / device driver definition. The red marked Functions are described below.

```
static struct i2c_driver vcnl4020_driver = {
    .driver = {
        .name = DRIVER_NAME,
        .owner = THIS_MODULE,
        .pm = &vcnl4020_pm_ops
    },
    .id_table = vcnl4020_id,
    .probe = vcnl4020_probe,
    .remove = vcnl4020_remove,
};

static const struct dev_pm_ops vcnl4020_pm_ops = {
    .suspend = vcnl4020_suspend,
    .resume = vcnl4020_resume
};
```

Functions:

```
vcnl4020_id → returns driver name
vcnl4020_probe → function called to detect vcnl4020 sensor on the i²c bus.
Creates an instance of the driver for the detected sensor.
vcnl4020_remove → destroys the instance of the driver.

vcnl4020_pm_ops (Power Management)
  vcnl4020_suspend → Cancel all running worker queues and disable sensor
  vcnl4020_resume → Restart all worker queues and enable sensor
```

2.2. Integration of Sources

Provided are two patch files:

- kernel.patch: Only contains the sources which can be directly applied to all kernel repositories (kernel configuration files and device driver)
- kernel-beagleboard.patch: Contains all required changes to build the kernel for the reference platform

To apply a patch, execute the following command inside the kernel repository (e.g. *board/beagleboard/kernel*):

```
$git am < kernel-beagleboard.patch
```

3. HAL Module

The Android part is represents a proxy between the Android App and the Linux Kernel in the device. The HAL part reads input from device files generated by the kernel module. It also writes control commands to corresponding device files in /sys/class/input/… to configure the sensors.

3.1. Sources Overview

The following describes the files/functions and changes.

libsensors/Android.mk

Makefile for Android

libsensors/sensors.cpp

libsensors/sensors.h

Configuration of the available sensors on the device (name,type,manufacturer)
This configuration is device manufacturer specific.

libsensors/SensorBase.cpp
libsensors/SensorBase.h

Generic base class for both sensors.
Responsible for:

- Sensor state
- Measurement rate
- Event file containing measurement value

It contains the following public Functions called by the Android Framework

```
int activate(bool en) //Activates/deactivates the sensor
int setDelay(int64_t ns); //Set time between measurements (default 200ms for
Beagle Board)
int poll(sensors_event_t *data, int count); //fetch current sensor data
```

libsensors/LightSensor.cpp
libsensors/LightSensor.h

Light sensor specific implementation. Responsible for data transformation (LUX Calculation).

```
int LightSensor::poll //Implementation to return current LUX value of light
sensor
```

libsensors/ProximitySensor.cpp
libsensors/ProximitySensor.h

Proximity sensor specific implementation.

```
int ProximitySensor::poll //Implementation to return current sensor data 0 or 5
for near/far
```

3.2. Integration of Sources

For the device layer are also two patch files provided:

- device.patch: Only contains the sources which can be directly applied to all device repositories (Sensor implementations)
- device-beagleboard.patch: Contains all required changes to integrated the HAL layer into the reference repository

To apply a patch, execute the following command inside the device repository (e.g. *device/ti/beagleboard_xm*):

```
$git am < device-beagleboard.patch
```