DT0037
Design tip

# VL6180X range and ambient light sensor quick setup guide

By Colin Ramrattan

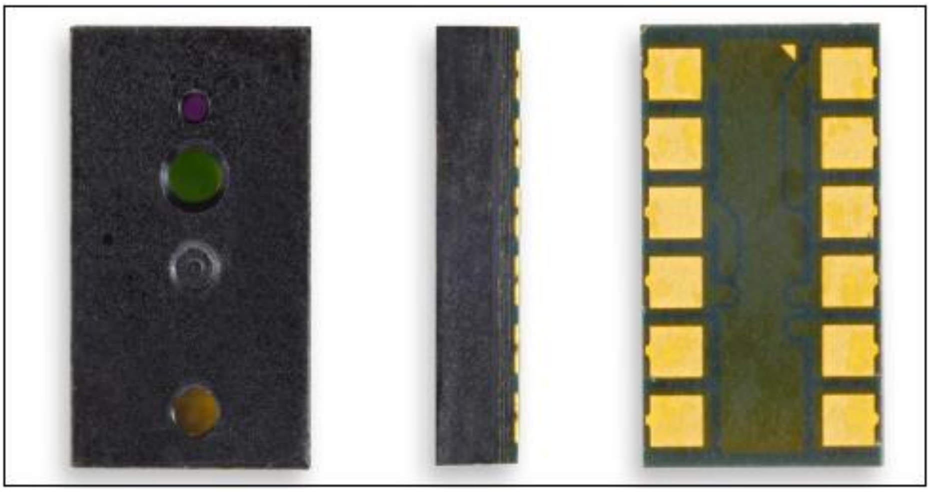
| Main components | |
|---|---|
| VL6180X | Proximity and ambient light sensing (ALS) module |

## Purpose and benefits

The purpose of this document is to guide customers who would like to know the following: (1) how to connect the VL6180X to HOST processor and (2) the minimum settings/register writes required to receive range and ambient light sensor (ALS) measurements out of the VL6180X.

It is assumed that customers who use this document are familiar with coding practices, the $I^2C$ standard and have the technical knowledge to understand the VL6180X datasheet.

**Figure 1. VL6180X device**



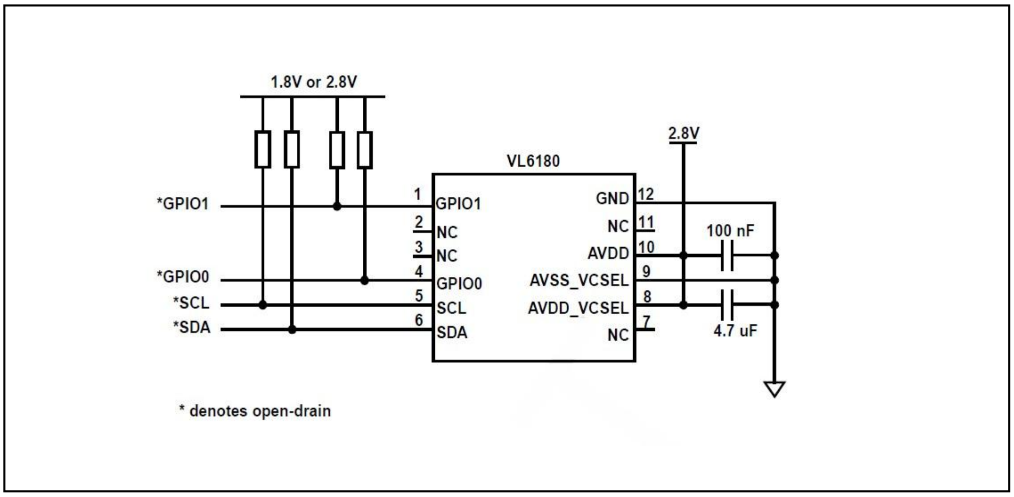
## Overview

The VL6180X contains both an ambient light sensor (ALS) and a proximity measurement sensor. This document will guide the user on how to connect the VL6180X to a suitable HOST processor present the power up sequencing of the VL6180X. Then a simple description of how to setup the VL6180X to perform range and ALS measurements will be given. Appendix 1 provides the required register writes to enable the appropriate tuning for the VL6180X.

## Application Schematic

The VL6180X is a 12-pin module that is contained in a convenient 4.8mm x 2.8mm x 1mm LGA package. A simple application schematic is shown below in Figure 2. This is taken directly from the VL6180X datasheet.

**Figure 2. VL6180X typical application schematic**

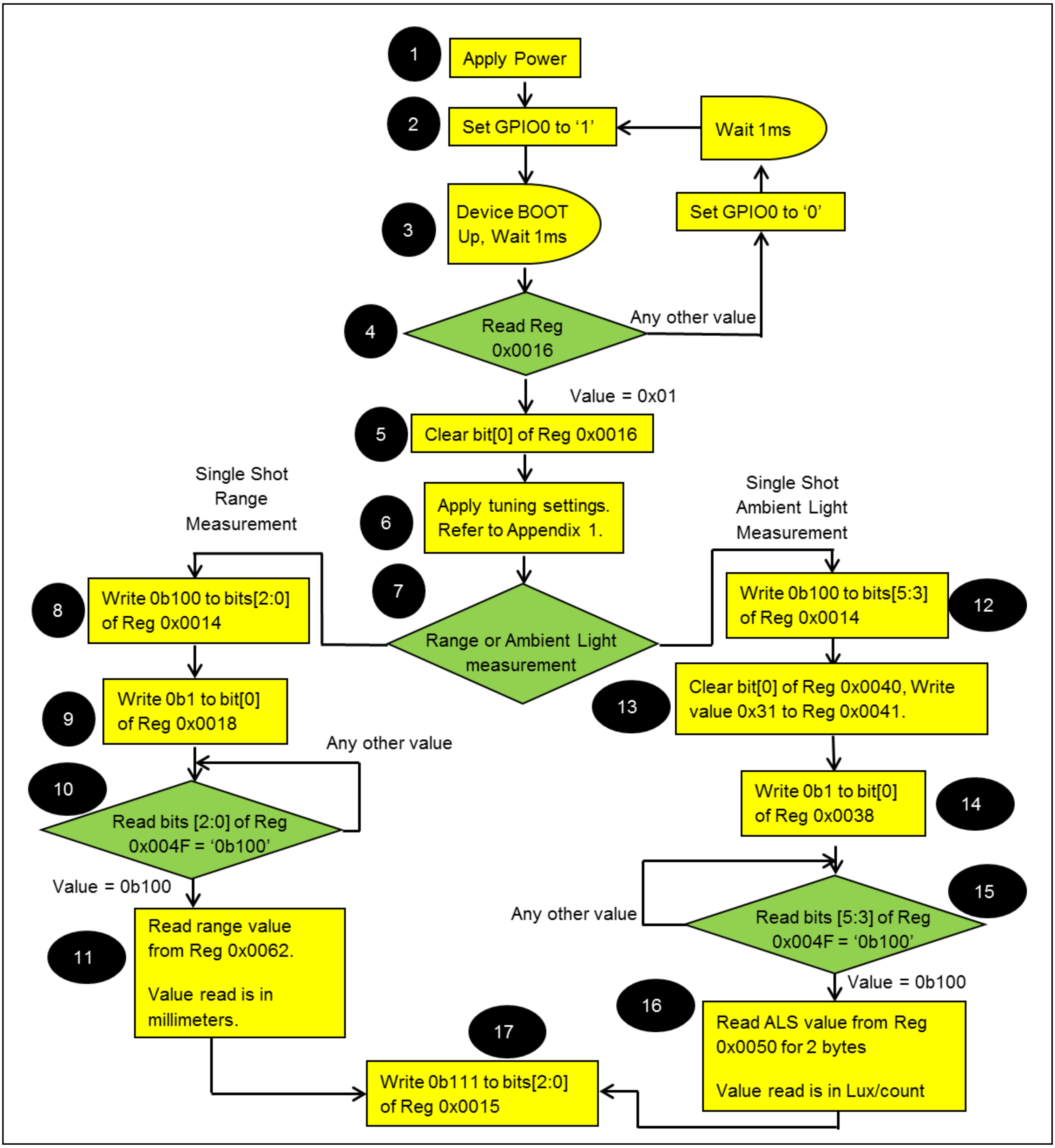



The pull-up resistors on the GPIO pins are important to the VL6180X operation. For example, GPIO0 is considered an enable pin during boot up of the device and if not pulled up will not allow the device to start.

## Power-Up Sequencing

The VL6180X requires a stable 2.8V power supply for AVDD_VCSEL and AVDD power supply inputs. The VL6180X requires power to be applied to AVDD_VCSEL pin before or at the same time as power is applied to AVDD pin. There is no limit to how early the AVDD_VCSEL power can be applied before AVDD power is applied

Figure 3. VL6180X ranging and ALS software flow



## Software Design

The VL6180X communicates through an I²C interface to the HOST processor. It is assumed that the end customer has experience with the I²C communication standard.

At power up, the VL6180X will be ready to respond to I²C communication 1ms after power is applied. It is assumed that the power applied is within the limits specified in the VL6180X datasheet.

The minimal settings that should be applied to the VL6180X are summarized below in Figure 3.

The software flow shown in Figure3 above applies to a minimal set of settings to allow the VL6180X to perform a single shot range or single shot ambient light measurement.

To describe each step with further detail, the steps in Figure 3 have been numbered. Each corresponding number is described below.

1. Applying power to the VL6180X is as simple as applying 2.8V to the VDDA and VDDA_VCSEL pins. It is important to take note of the power up sequencing described in this document.
2. Changing the GPIO0 to logic '1' allows the device to come out of reset.
3. Device boots in 1ms so the HOST should wait this minimum amount of time before communicating with the VL6180X.
4. Register 0x0016 is described as the SYSTEM__FRESH_OUT_OF_RESET register. If the value in SYSTEM__FRESH_OUT_OF_RESET is 0x01 then the VL6180X is in idle mode and working as expected. If the SYSTEM__FRESH_OUT_OF_RESET register is any other value, reset the device by applying logic '0' to GPIO0. This will return you to step 2.
5. Register 0x0016 is described as the SYSTEM__FRESH_OUT_OF_RESET flag. Writing a '0' to bit[0] allows it to be used for debug purposes.
6. Apply the tuning settings as shown in Appendix 1. This is mandatory after every boot of the VL6180X.
7. The user should choose either the range function and continue to step 8 or the ALS function and continue to step 12.
8. Register 0x0014 is described as the SYSTEM__INTERRUPT_CONFIG_GPIO. By setting bits [2:0] to 0b100, this enables the range function interrupt within the device, not externally on the GPIO1 or GPIO0 pins.
9. Register 0x0018 is described as the SYSRANGE__START register. Writing a '1' to bit[0] starts a single shot range measurement.

10. Register 0x004F is described as the RESULT__INTERRUPT_STATUS_GPIO. This register does not poll the GPIO pins directly. Reading a value of 0b100 from bits[2:0] means that a range measurement is complete and the VL6180X is ready for another command. Reading a value of 0b000 means the device is still busy with the current measurement.

11. Register 0x0062 is described as the RESULT__RANGE_VAL register. The value read from this register is in millimeter units.

12. Register 0x0014 is described as the SYSTEM__INTERRUPT_CONFIG_GPIO. By setting bits [5:3] to 0b100, this enables the ALS function interrupt within the device, not externally on the GPIO1 or GPIO0 pins.

13. Register 0x0040 is described as the SYSALS__INTEGRATION_PERIOD register. This is a two byte register that is set to 50ms in this example. It is important to note that a value of 0x00 in register 0x0040 means 1ms integration time. As a result, the value required in millisecond should be subtracted by a value of 1.

14. Register 0x0038 is described as the SYSALS__START register. Writing a ‘1’ to bit[0] starts a single shot ALS measurement.

15. Register 0x004F is described as the RESULT__INTERRUPT_STATUS_GPIO. This register does not poll the GPIO pins directly. Reading a value of 0b100 from bits[5:3] means that a ALS measurement is complete and the VL6180X is ready for another command.

16. Register 0x0050 is described as the RESULT__ALS_VAL register. This is a 2 byte register and the value read is in Lux/count. The user must then apply the calibration count/lux value and divide by the ALS gain value to achieve the appropriate Lux value. See the examples section for more details.

17. Register 0x0015 is described as the SYSTEM__INTERRUPT_CLEAR register and writing 0b111 to bits[2:0] will clear the interrupt status. This will ensure that the correct range or ALS measurement is read back to the HOST processor.

## Range Measurement Example

If the device is setup as shown in Figure 3 the following example below gives an output example of what should be read from the device.

Assuming there is a piece of white or gray material located at 50mm above the device and a single shot measurement is started, the example below shows how to interpret the value read.

*When single shot measurement is complete, the value is read from the RESULT__RANGE_VAL register (0x0062) = 0x32. Converting this hex value to decimal, the resulting value is 50mm.*

## ALS Measurement Example

If the device is setup as shown in Figure 3 the following example below gives an output example of what should be read from the device.

Assuming there the device is in a 200lux environment, by starting a single shot ambient light measurement, the example below shows how to interpret the value read.

*When ambient light measurement is complete, the value is read from the RESULT__ALS_VAL register (0x0050) = 0x01 (high byte) and read from 0x0051 = 0x38 (low byte).The values read should be concatenated together giving a values of 0x0138. By converting this hex value to decimal, the resulting value is 312. Then this value should be multiplied by the ALS calibration value, 0.32 count/lux is the default value, multiplied by 100 and then divided by the gain and then divided by the ALS integration time. The equation is shown below:*

*312 * [ALS calibration value] * 100/ ([ALS gain] * [ALS Integration time]) = 312 * 0.32 * 100 / (1 * 50) = 199.68 Lux.*

The example above uses the default calibration value of the VL6180X from ST Microelectronics. If there is cover glass used on top of the VL6180X, the ALS calibration value shall be recalculated by the end user to ensure the proper lux measurement.

## Appendix 1: VL6180X Tuning Settings

The list of commands below are the appropriate tuning settings that should be applied to the VL6180X module. These settings should be applied in this order with no substitutions. The commands below are applied top to bottom and then the next column over to the right, top to bottom. The commands are given in the following format:

`Writebyte(Register_Address, Value to Write)`

```
WriteByte(0x0207, 0x01);
WriteByte(0x0208, 0x01);
WriteByte(0x0133, 0x01);
WriteByte(0x0096, 0x00);
WriteByte(0x0097, 0xFD);
WriteByte(0x00e3, 0x00);
WriteByte(0x00e4, 0x04);
WriteByte(0x00e5, 0x02);
WriteByte(0x00e6, 0x01);
WriteByte(0x00e7, 0x03);
WriteByte(0x00f5, 0x02);
WriteByte(0x00D9, 0x05);
WriteByte(0x00DB, 0xCE);
WriteByte(0x00DC, 0x03);
WriteByte(0x00DD, 0xF8);
WriteByte(0x009f, 0x00);
WriteByte(0x00a3, 0x3c);
WriteByte(0x00b7, 0x00);
WriteByte(0x00bb, 0x3c);
WriteByte(0x00b2, 0x09);
WriteByte(0x00ca, 0x09);
WriteByte(0x0198, 0x01);
WriteByte(0x01b0, 0x17);
WriteByte(0x01ad, 0x00);
WriteByte(0x00FF, 0x05);
WriteByte(0x0100, 0x05);
WriteByte(0x0199, 0x05);
WriteByte(0x0109, 0x07);
WriteByte(0x010a, 0x30);
WriteByte(0x003f, 0x46);
WriteByte(0x01a6, 0x1b);
WriteByte(0x01ac, 0x3e);
WriteByte(0x01a7, 0x1f);
WriteByte(0x0103, 0x01);
WriteByte(0x0030, 0x00);
WriteByte(0x001b, 0x0A);
WriteByte(0x003e, 0x0A);
WriteByte(0x0131, 0x04);
WriteByte(0x0011, 0x10);
WriteByte(0x0014, 0x24);
WriteByte(0x0031, 0xFF);
WriteByte(0x00d2, 0x01);
WriteByte(0x00f2, 0x01);
```

## Support material

| Related design support material | |
|---|---|
| MOB-EK2-180-01/1 | Product/ system evaluation board |
| **Documentation** | |
| Datasheet: VL6180X - Proximity and ambient light sensing (ALS) module | |

## Revision history

| Date | Version | Changes |
|---|---|---|
| 9-July-2014 | 1 | Initial release |