Application Note

REV	AN#	ORIGINATOR	DESCRIPTION	DATE
A	AN3218108112012	Dio Yang	CM32181 Application Note	8/Nov/12
B	AN3218107012013	Dio Yang	CM32181 Application Note	7/Jan/13
C	AN3218120052013	Dio Yang	CM32181 Application Note	20/May/13
D	AN3218113062013	Dio Yang	CM32181 Application Note	13/June/13

PART NAME: CM32181 Ambient Light Sensor

APPROVED REVISION: D

CUSTOMER NAME:

Documents:

1) Application note, CM32181 Design Notice

CUSTOMER SIGN-OFF:

TITLE	NAME	SIGNATURE	DATE

1. Introduction

CM32181 is an advanced ambient light sensor with I2C protocol interface and designed by the CMOS process. It is easily operated via a simple I2C command. The active interruption feature within the threshold windows setting offers the benefit of eliminating loading of the controller monitor. CM32181 incorporates a photodiode, amplifiers and analog circuits into a single chip. The best spectral sensitivity is used to closely capture real human eye responses.

CM32181 has excellent temperature compensation. Its robust refresh rate setting does not need an external RC low pass filter. Software shutdown mode is provided which reduces power consumption to be less than 0.5μA. CM32181's operating voltage ranges from 2.5V to 3.6V and consumes only 1uA. The maximum detective light strength is over 100K Lux.

2. Reference Circuit

CM32181 is a cost effective solution for an ambient light sensor with I^2C interface. The standard serial digital interface easily accesses "light intensity" without using complex calculations and programming by an external controller.

The additional capacitor near the V_{DD} pin in the circuit is used for power supply noise rejection. The value is recommended at 0.1uF. The pull-high resistors for the I^2C bus design are recommended to be 2.2KΩ. Pin ADDR is for address Select. Pull high to select address 0x48 or low to select address 0x10. An example of the circuit diagram is shown in Figure 1.

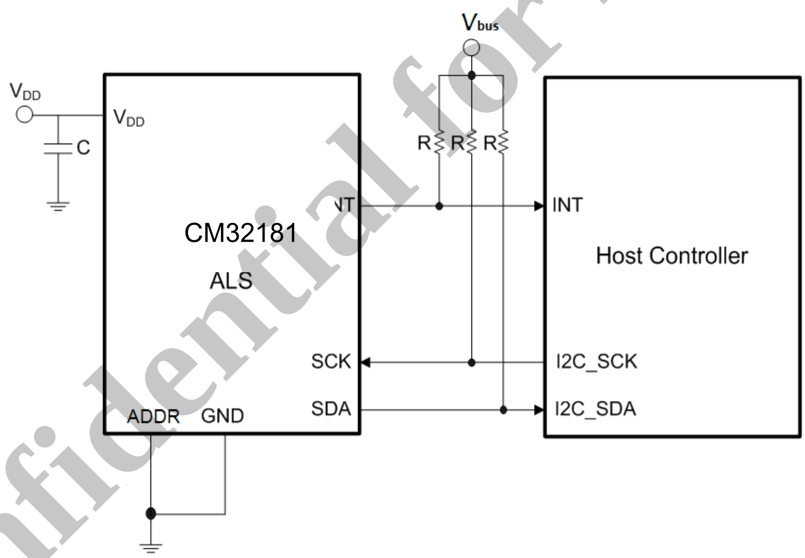


Figure 1. CM32181 Reference Circuit Connection with Host

3. CM32181 Pin PAD Layout

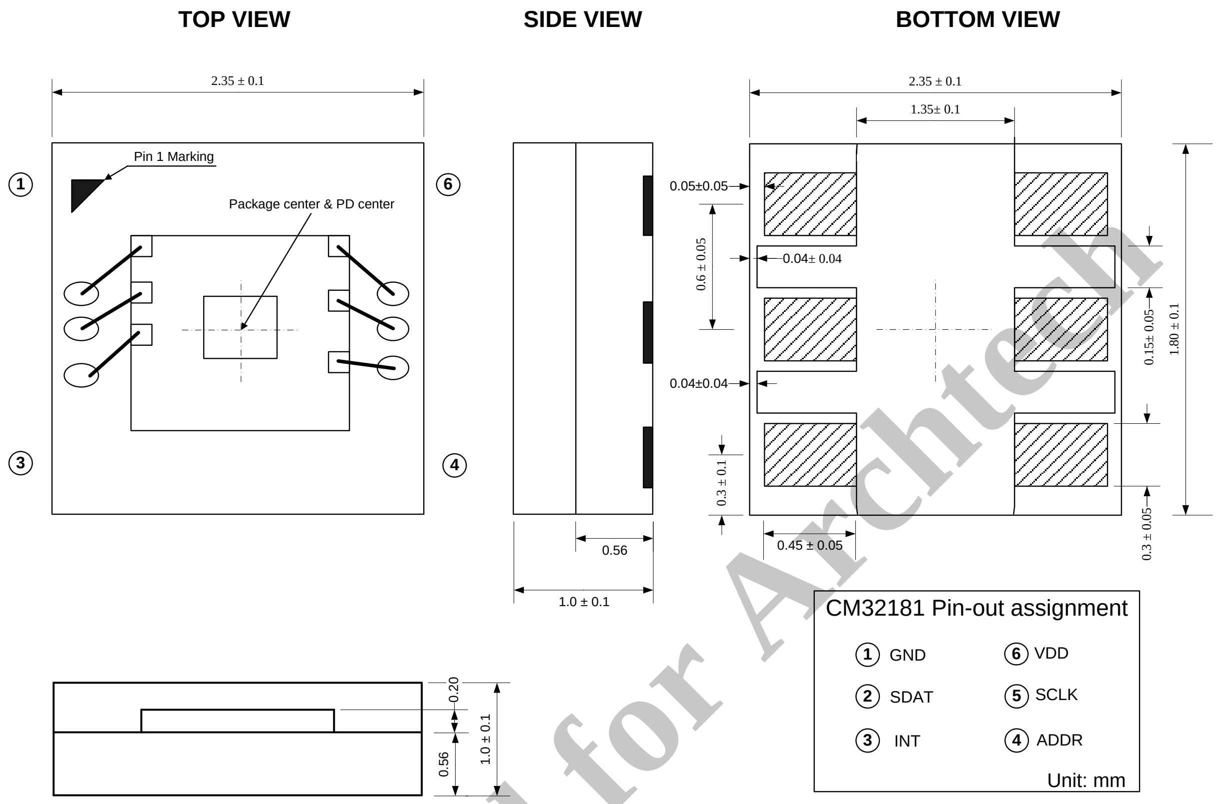


Figure 2-1. CM32181 A3OP Package Dimensions

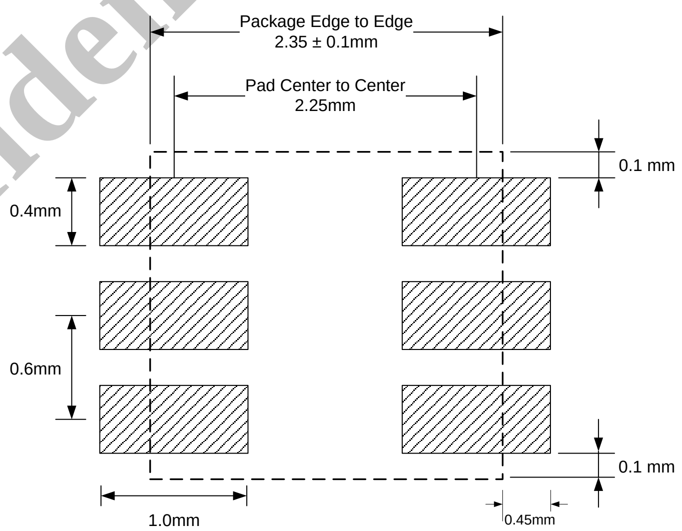
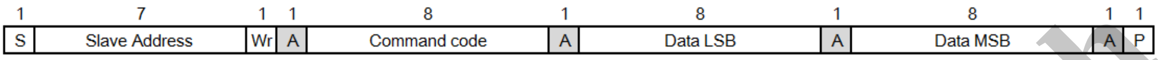


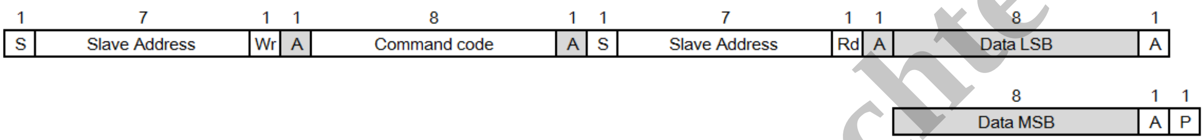
Figure 2-2b. CM32181 OPLGA PCB Layout Footprint

4. Register Implement Description

CM32181 contains a command register written via the I²C bus. All operations can be controlled by the command register. The simple command structure allows the user to easily program the operation setting and latch the light data from CM32181. CM32181's I²C command format description for Read and Write operations between CM32181 and the host is shown in Figure 3. The white areas indicate the host activity and the gray areas indicate CM32181's acknowledgement of the host access activity.



Write Protocol -> Write Command to CM32181



Read Protocol -> Read data from CM32181

S = Start Condition
P = Stop Condition
A = Acknowledge

Figure 3. Send Byte/Receive Byte Protocol

Command Register Format

There are 6 command codes provided by CM32181. Formats of these command code and registers' definition explanations are shown in below Table 1.

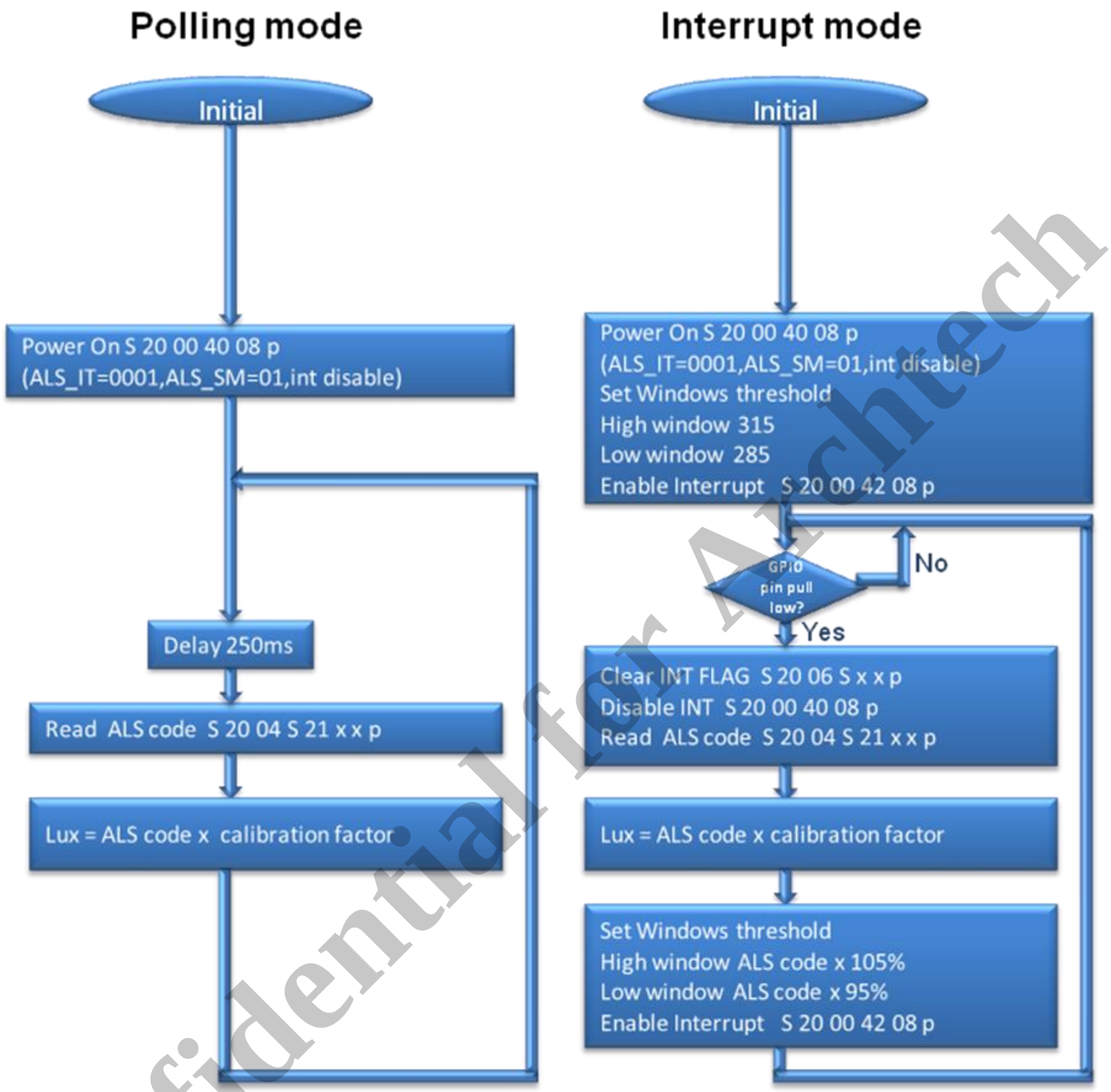
Command code	Register Name	Bit	Function/Description	R/W
00	Reserved	15:13	Set 000b	W
	ALS_SM	12:11	Sensitivity mode selection 00 = ALS Sensitivity x 1 01 = ALS Sensitivity x 2 10 = ALS Sensitivity x (1/8) 11 = ALS Sensitivity x (1/4)	W
	Reserved	10	Set 0b	W
	ALS_IT	9:6	ALS integration time setting 1100 = 25ms 1000 = 50ms 0000 = 100ms 0001 = 200ms 0010 = 400ms 0011 = 800ms	W

	ALS_PERS	5:4	ALS Persistence protect number setting 00 = 1 01 = 2 10 = 4 11 = 8	W
	Reserved	3:2	Set 00b	W
	ALS_INT_EN	1	ALS Interrupt enable setting 0 = ALS INT Disable 1 = ALS INT Enable	W
	ALS_SD	0	ALS shut down setting 0 = ALS Power On 1 = ALS Shut down	W
01	ALS_WH	15:8	ALS High Threshold Window setting(MSB)	W
	ALS_WH	7:0	ALS High Threshold Window setting(LSB)	W
02	ALS_WL	15:8	ALS Low Threshold Window setting(MSB)	W
	ALS_WL	7:0	ALS Low Threshold Window setting(LSB)	W
03	Reserved	15:3	Set 0000 0000 0000 0b	W
	PSM	2:1	Power Saving Mode 00 = Mode1 01 = Mode2 10 = Mode3 11 = Mode4 Please refer to Table2.	W
	PSM_EN	0	Power Saving Mode enable setting 0 = Disable 1 = Enable	W
04	ALS	15:8	MSB 8bits data of whole ALS 16bits	R
	ALS	7:0	LSB 8bits data of whole ALS 16bits	R
05	Reserved	15:0		R
06	ALS_IF_L	15	ALS crossing Low threshold INT trigger event	R
	ALS_IF_H	14	ALS crossing High threshold INT trigger event	R
	Reserved	13:0		R

Table 1. Register Setting Description

5. Programming flow

5.1



5.1
5.2 Sample code

```
// CM32181 slave address can be 0x20 or 0x90, determined by pin
ADDR configuration
#define CM32181_ADDR_ALS   0x20 // 7-bit: 0x10
//#define CM32181_ADDR_ALS 0x90 // 7-bit: 0x48

// CM32181 registers
#define ALS_CONF 0x00
#define ALS_THDH 0x01
#define ALS_THDL 0x02
#define ALS_PSM  0x03
#define ALS_DATA 0x04
#define ALS_STATUS 0x06

// CM32181 command code 00 register bits
#define CM32181_ALS_CONF_SD                0x0001
#define CM32181_ALS_CONF_INT_EN            0x0002
#define CM32181_ALS_CONF_PERS_MASK         0x0030
#define CM32181_ALS_CONF_PERS_1            0x0000
#define CM32181_ALS_CONF_PERS_2            0x0010
#define CM32181_ALS_CONF_PERS_4            0x0020
#define CM32181_ALS_CONF_PERS_8            0x0030
#define CM32181_ALS_CONF_IT_MASK          0x00C0
#define CM32181_ALS_CONF_IT_100MS          0x0000
#define CM32181_ALS_CONF_IT_200MS          0x0040
#define CM32181_ALS_CONF_IT_400MS          0x0080
#define CM32181_ALS_CONF_IT_800MS          0x00C0
#define CM32181_ALS_CONF_SM_MASK           0x1800
#define CM32181_ALS_CONF_SM_x1             0x0000
#define CM32181_ALS_CONF_SM_x2             0x0800
#define CM32181_ALS_CONF_SM_x1_8           0x1000
#define CM32181_ALS_CONF_SM_x1_4           0x1800
#define CM32181_ALS_CONF_DEFAULT           0x0000

WORD cmd[4] = {CM32181_ALS_CONF_DEFAULT | CM32181_ALS_CONF_PERS_1
| CM32181_ALS_CONF_IT_200MS | CM32181_ALS_CONF_SM_x2, 315, 285, 0};
WORD als_code = 300;
const float change_sensitivity = 5; // in percent
const float calibration_factor = 0.286;
float lux;

struct i2c_msg {
  WORD addr;
  WORD flags;
#define I2C_M_TEN                0x0010
#define I2C_M_RD                 0x0001
#define I2C_M_NOSTART            0x4000
#define I2C_M_REV_DIR_ADDR       0x2000
#define I2C_M_IGNORE_NAK         0x1000
#define I2C_M_NO_RD_ACK          0x0800
#define I2C_M_RECV_LEN           0x0400
  WORD len;
  BYTE *buf;
```

```
};

extern int i2c_transfer(struct i2c_msg *msgs, int num);

//-------------------------------------------------------------------
------------
// C main function
//-------------------------------------------------------------------
------------
void main(void)
{
      initialize_CM32181();

      while (1)
      {
            if (INT_PIN == LOW)
            {
                  clear_interrupt();

                  disable_interrupt();

                  als_code = read_als_data();
                  lux = als_code * calibration_factor;

                  set_als_int_threshold();

                  enable_interrupt();
            }
      }
}

void initialize_CM32181(void)
{
      disable_sensor();

      set_als_int_threshold();

      enable_interrupt();

      enable_sensor();

      // Delay some time after sensor is configured
      delay(250);
}

void clear_interrupt(void)
{
      WORD value;

      // Read ALS_STATUS register to clear interrupt
      CM32181_read_word(CM32181_ADDR_ALS, ALS_STATUS, &value);
}

void enable_sensor(void)
{
      cmd[ALS_CONF] &= ~CM32181_ALS_CONF_SD;
```

```
    CM32181_write_word(CM32181_ADDR_ALS, ALS_CONF, cmd[ALS_CONF]);
}

void disable_sensor(void)
{
     cmd[ALS_CONF] |= CM32181_ALS_CONF_SD;
    CM32181_write_word(CM32181_ADDR_ALS, ALS_CONF, cmd[ALS_CONF]);
}

void enable_interrupt(void)
{
     cmd[ALS_CONF] |= CM32181_ALS_CONF_INT_EN;
    CM32181_write_word(CM32181_ADDR_ALS, ALS_CONF, cmd[ALS_CONF]);
}

void disable_interrupt(void)
{
     cmd[ALS_CONF] &= ~CM32181_ALS_CONF_INT_EN;
    CM32181_write_word(CM32181_ADDR_ALS, ALS_CONF, cmd[ALS_CONF]);
}

void set_als_int_threshold(void)
{
     int threshold_high;

     // Set ALS high threshold
     threshold_high = als_code * (100 + change_sensitivity) / 100;
     if (threshold_high > 65535)
          cmd[ALS_THDH] = 65535;
     else
          cmd[ALS_THDH] = threshold_high;
    CM32181_write_word(CM32181_ADDR_ALS, ALS_THDH, cmd[ALS_THDH]);

     // Set ALS low threshold
     cmd[ALS_THDL] = als_code * (100 - change_sensitivity) / 100;
    CM32181_write_word(CM32181_ADDR_ALS, ALS_THDL, cmd[ALS_THDL]);
}

WORD read_als_data(void)
{
     WORD value;

     CM32181_read_word(CM32181_ADDR_ALS, ALS_DATA, &value);

     return value;
}

int CM32181_read_word(WORD addr, BYTE command, WORD *val)
{
     int err = 0;
     int retry = 3;
     struct i2c_msg msg[2];
     BYTE data[2];

     while (retry--)
     {
```

```
        // Send slave address & command
        msg[0].addr = addr >> 1;
        msg[0].flags = I2C_M_WR;
        msg[0].len = 1;
        msg[0].buf = &command;

        // Read word data
        msg[1].addr = addr >> 1;
        msg[1].flags = I2C_M_RD;
        msg[1].len = 2;
        msg[1].buf = data;

        err = i2c_transfer(msg, 2);

        if (err >= 0)
        {
            *val = ((WORD)data[1] << 8) | (WORD)data[0];
            return err;
        }
    }

    return err;
}

int CM32181_write_word(WORD addr, BYTE command, WORD val)
{
    int err = 0;
    int retry = 3;
    struct i2c_msg msg;
    BYTE data[3];

    while (retry--)
    {
        data[0] = command;
        data[1] = (BYTE)(val & 0xFF);
        data[2] = (BYTE)((val & 0xFF00) >> 8);

        // Send slave address & command
        msg.addr = addr >> 1;
        msg.flags = I2C_M_WR;
        msg.len = 3;
        msg.buf = data;

        err = i2c_transfer(msg, 1);

        if (err >= 0)
            return 0;
    }

    return err;
}
```

6. ALS Window Lens

In most of the applications, there is a window lens on top of the ALS sensor. The window materials greatly reduce the optical input dynamic range of the ALS sensor. To compensate the window optical loss, Fundamental Integration Time setting needs to be readjusted.

Basically, the customer could adopt the optical material, such as PE, Silicon etc,. The light transparency should be over 70% and the light application management will be efficient. The below description is a study case of lens material selection for NB ALS design. It can help the customer to understand how to judge and fine tune Fundamental Integration Time for ALS application:

For ALS lens material selection, three types of GE LEXAN materials had been characterized and their optical effects are presented here

.

1) GY4343T
2) GY2C135T
3) GY3C167T.

Here is the CM32181 window lens optical efficiency.

Lens material	**No Lens**	**GY4343T**	**GY2C135T**	**GY3C167T**	**units**
Average reading @ 100Lux	7000	3756	2646	1430	**step**
	100%	53.67%	37.81%	20.44%	**%**

7. ALS optical window Dimensions

Background

Figure 4 shows the view angle and CM32181 output steps relationship.

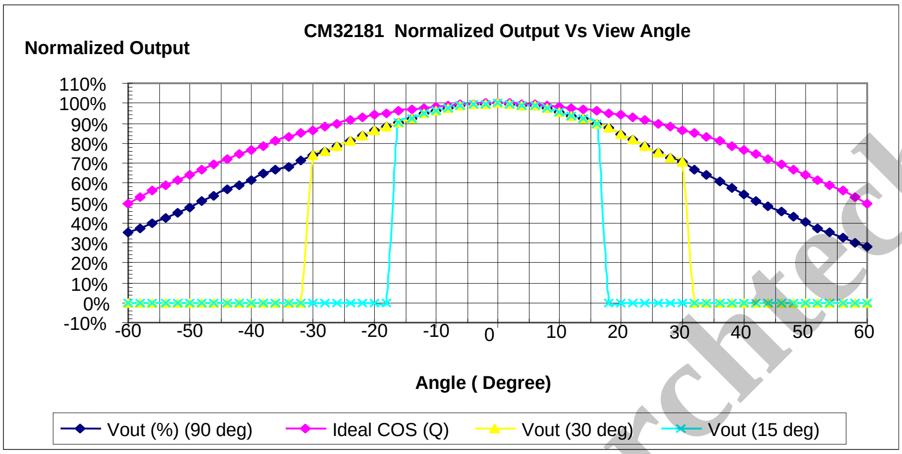


Figure 4. CM32181 Output and Sensitivity with Different Refresh Timing

The dimension design of lens window which is placed in front of CM32181 defines the view angle of CM32181 and affects the angular response of the CM32181 a lot. The minimum dimension recommended here will ensure at lease ±40º light flux reception.

1). The angular response of ALS sensor output is mainly affected by the lens window size. Luminous flux with angular path greater than the ALS viewing angle is blocked by the lens window.

2). To ensure that the ALS response maintains ±40 degree off-axis light sensing without affecting much the angular response, the lens windows dimension A and the ALS vertical position dimension B should follow the guidelines below: (Fig 5)

$$\frac{DIMA/2}{DIMB} = \tan(40^\circ)$$

3). Table 2 shows example of maximum DIM B and minimum DIM A which fulfills the ±40º view angle condition.

Minimum DIMA (mm)	Maximum DIMB (mm)	ALS viewing angle (degree)
4	**2**	**45**
3.5	**2**	**41.1**
3	2	36.8
2.5	2	32
1.5	2	20.5
1	2	14
0.5	2	7

Table 2 Windows Dimension and View Angle Design example

Comments:

1) To ensure a minimum of ±40º degree off-axis light into the ALS sensor, minimum DIM A should not less than 3.4mm in above example. (manufacturing tolerance for mechanical dimensions in the system is not included). Fig 6 shows the design example.
2) If luminous flux is coming from angle larger than the specified viewing angle (±40 degree), the ALS sensor will not response to the luminous flux. The photodiode of the ALS sensor will not receive any luminous flux except the stray light reflected within the LCD bezel and the converter board.
3) The recommended ALS view angle should be maintained in the range of ±40 to ±45 degree viewing angle.

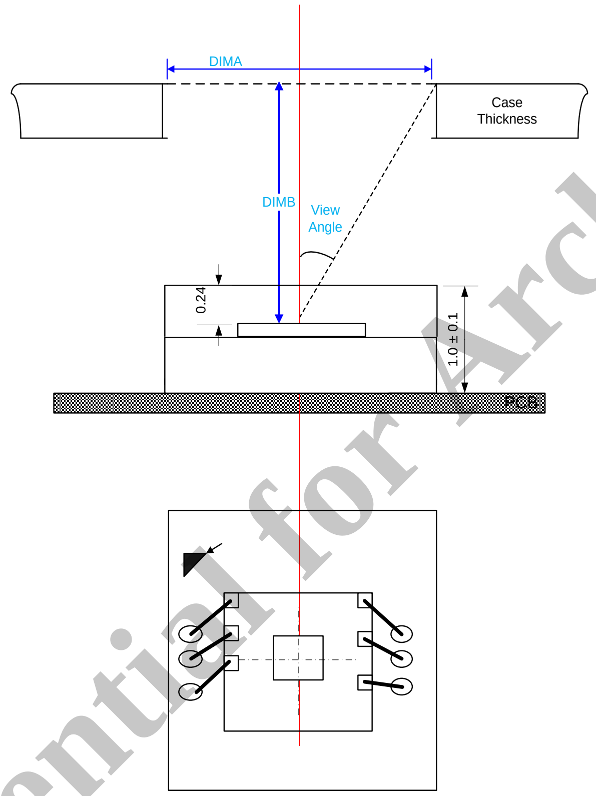


Figure 5. Optical window design for CM32181

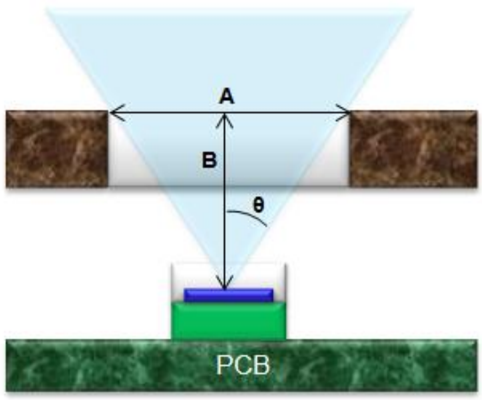


Figure 6. Optical window design example

Alternative solution: Lens with light pipe to guide luminous flux to ALS sensor

1) Using polycarbonate molded compound as lens and light pipe for CM32181.
2) The light pipe center should be located right above the ALS Package center as Figure 7.

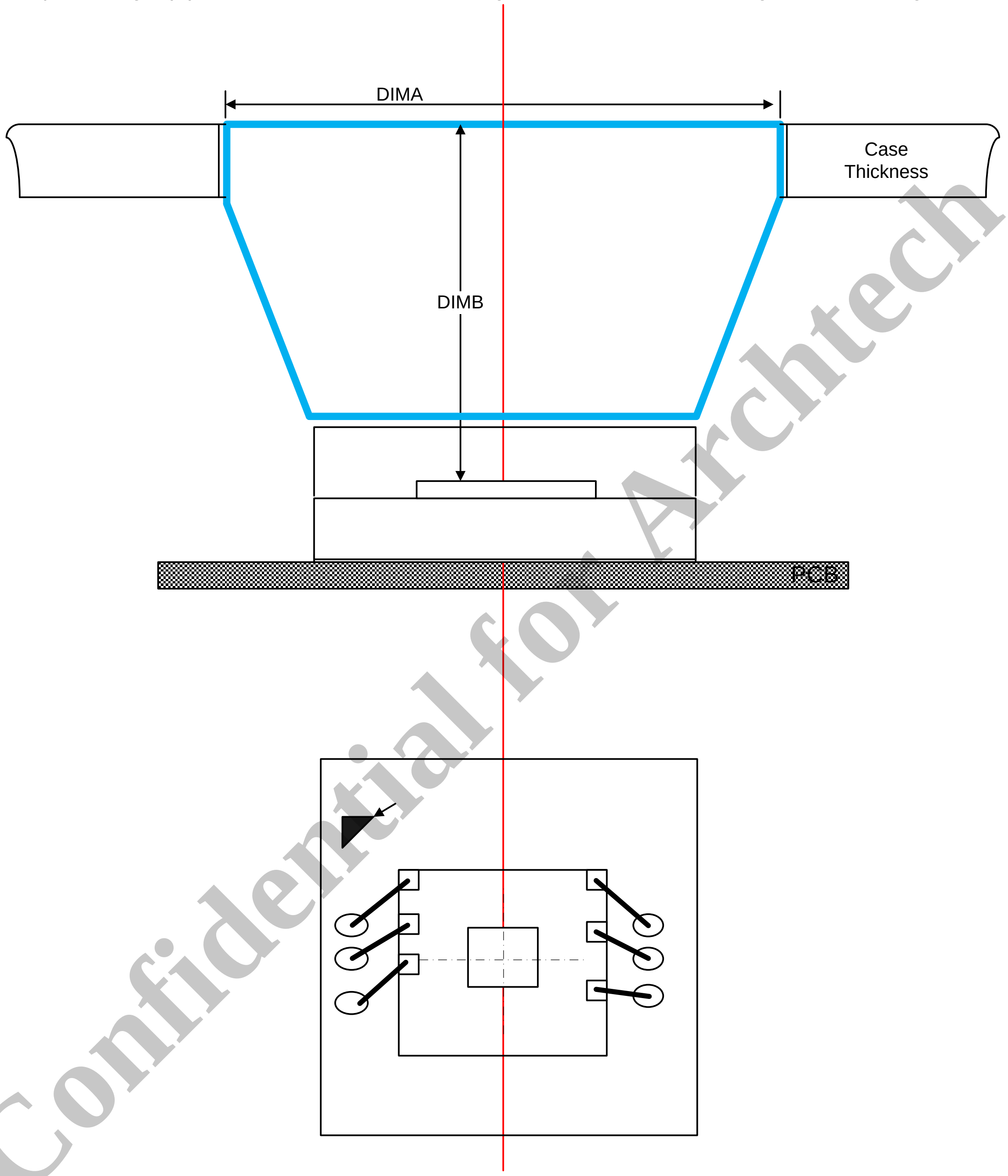


Figure 7. CM32181 Light Pipe Design