TCS34725 Color Sensor User Manual

OVERVIEW

This is a color sensor module based on TCS34725, will output RGB data and light intensity through the I2C interface. Its advantages include high sensitivity, wide dynamic range, accurate measuring, etc.

SPECIFICATION

Working voltage: 3.3V/5V

Controller: TCS34725FN

IO voltage: 3.3V/5V

Interface: I2C

Dimension: 27 x 20(mm)

INTERFACE

PIN	Description
VCC	3.3V/5V
GND	Ground
SDA	I2C Data Input
SCL	I2C Clock Input
INT	Interrupt Output (Open drain output)
LED	LED

CONTENT

Overview....1

Specification....1

Interface....1

Hardware....3

Controller....3

Communication protocol....3

I2C write....3

I2C Read....4

I2C address....4

How to use....5

Download examples....5

examples....5

Raspberry Pi....5

STM32....10

Arduino....12

FAQ....14

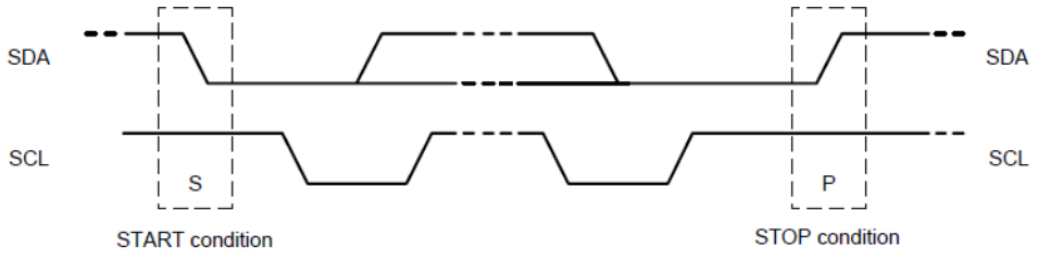
HARDWARE

CONTROLLER

TCS34725 is used for color sensing. TCS34725 is an I2C bus-based color light-to-digital converter with IR filter, provides a digital return of red, green , blue (RGB) and clear light sensing values. The high sensitivity, wide dynamic range and IR blocking filter make the TCS34725 an ideal color sensor solution for use under varying lighting conditions and through attenuating materials.

COMMUNICATION PROTOCOL

I2C bus has two lines, one is data line (SDA) and another is lock line (SDL). There are three kinds of signals when communicating, Start signal, Stop signal and Answer signal.

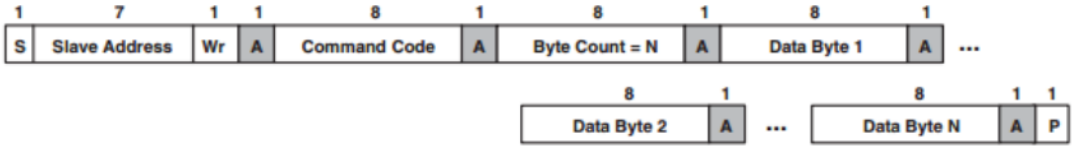


Start signal: When SCL is High, SDA change from High to Low, it start to transmit data

Stop signal: When SCL is High, SDA change from Low to High, it stop transmitting.

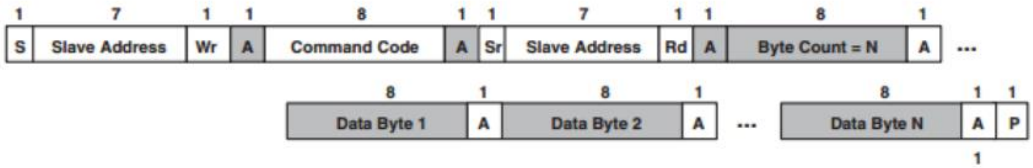
Answer signal: Every time IC send back a certain Low plus to sender after it receives 8 bits data.

I2C WRITE



When working, Raspberry Pi (hereafter named as Master) will first send a Start signal, then send a byte to TCS34725(hereafter named as Slaver), whose first 7bits are address of Slaver and 1 bit write bit. Slave response with Answer signal every time it receives any data. Master send command register address to Slaver, then data of command register. Stop signals is sent to slave to stop communicating.

I2C READ



When working, Master will first send a Start signal, then send a byte to Slaver, whose first 7bits are address of Slaver and 1 bit write bit. Slave response with Answer signal every time it receives any data. Master send command register address to Slave. After that, Mater will send a Start signal again, and then send a byte (7bits address and 1bit read bit) to Slaver. Slaver response and send data of the register to Master, master answer as well. Stop signals will be sent to stop communicating.

I2C ADDRESS

The I2C device address of TCS34725 is 0x29

Device	Address	Package-Leads	Interface Description	Ordering Number
TCS34725	0x29	FN–6	I^2C $V_{BUS} = V_{DD}$ Interface	TCS34725FN
TCS34727	0x29	FN–6	I^2C $V_{BUS} = 1.8$ V Interface	TCS34727FN

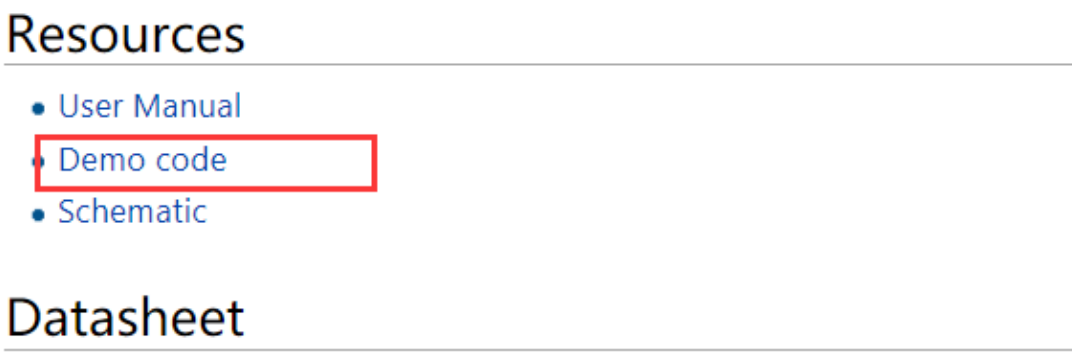
TCS3472 datasheet page 34

Note: 0x29 is 7bit in fact, therefore, when you set the I2C address, you should left-shift one bit, turn it to 0x52

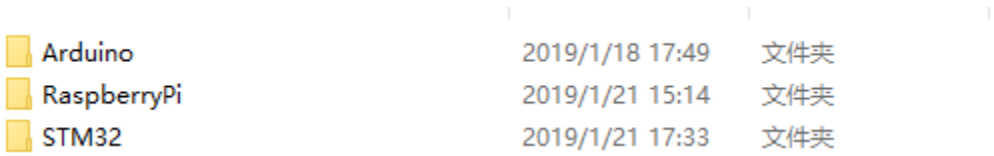
HOW TO USE

DOWNLOAD EXAMPLES

Find and download examples from Waveshare wiki:



Extract the 7z you get:



Arduino: examples for Arduino

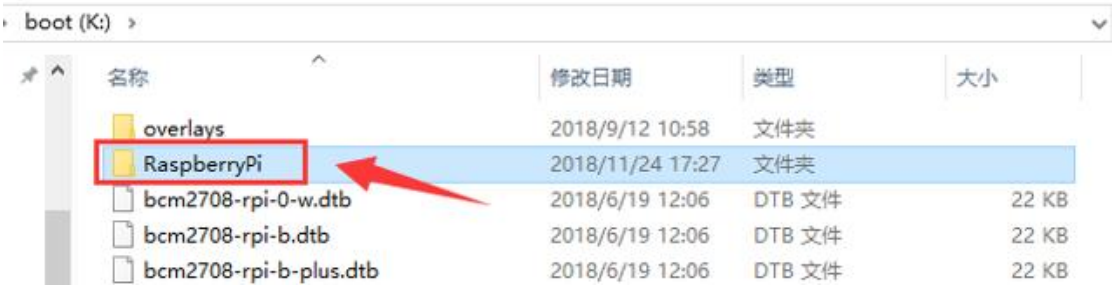
Raspberry Pi: examples for Raspberry Pi(wiringPi, python, bcm2835)

EXAMPLES

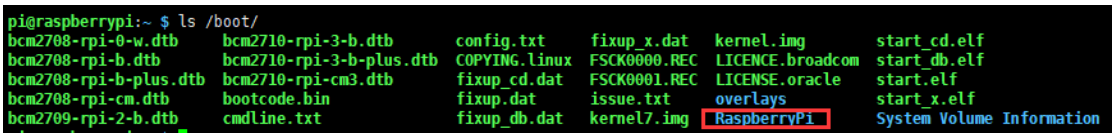
RASPBERRY PI

Insert the SD card (Raspbian installed)

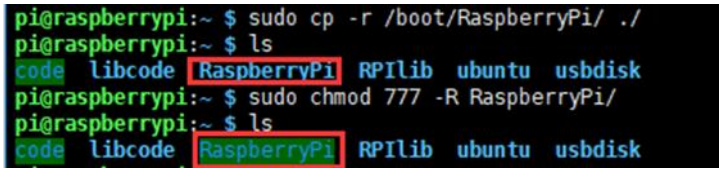
Copy the Raspberry Pi examples to SD card:



Insert SD card to Raspberry Pi and power on, you can find the folder is listed in /boot



Copy the examples to /home/pi and change its permission:



INSTALL LIBRARIES

To run the examples, you need to first install libraries (wiringPi, bcm2835 and python)

and enable I2C interface, otherwise example cannot work properly.

BCM2835

http://www.airspayce.com/mikem/bcm2835/

Download the library from bcm2835 libraries and install:

```
wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.58.tar.gz

sudo tar zxvf bcm2835-1.xx.tar.gz
```

```
cd bcm2835-1.xx

sudo ./configure

make

sudo make check

sudo make install
```

Note: The xx is the version number you download, for example, if the version you download is bcm2835-1.52. then the command you should execute is sudo tar zxvf bcm2835-1.52.tar.gz

wiringPi libraries:

```
sudo apt-get install git

sudo git clone git://git.drogon.net/wiringPi

cd wiringPi

sudo ./build
```

Python libraries:

```
sudo apt-get install python-pip

sudo pip install RPi.GPIO

sudo pip install spidev

sudo apt-get install python-imaging

sudo apt-get install python-smbus
```

Enable I2C interface:

sudo raspi-config

```
┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

  1 Change User Password Change password for the current user
  2 Network Options      Configure network settings
  3 Boot Options         Configure options for start-up
  4 Localisation Options Set up language and regional settings to match your location
  5 Interfacing Options  Configure connections to peripherals
  6 Overclock            Configure overclocking for your Pi
  7 Advanced Options     Configure advanced settings
  8 Update               Update this tool to the latest version
  9 About raspi-config   Information about this configuration tool

                <Select>                        <Finish>
```

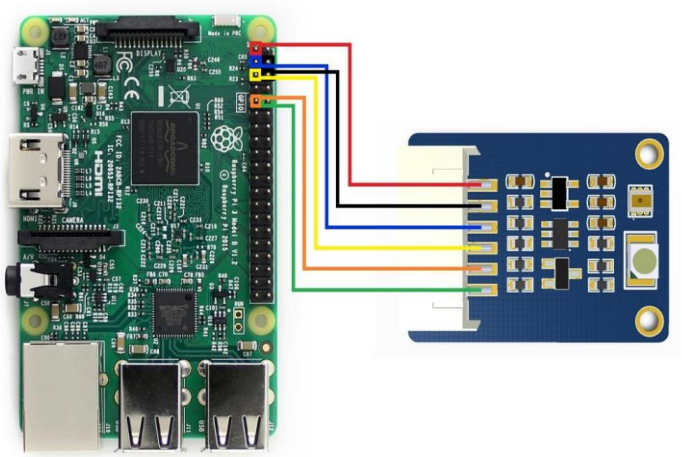
```
┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

  P1 Camera      Enable/Disable connection to the Raspberry Pi Camera
  P2 SSH         Enable/Disable remote command line access to your Pi using SSH
  P3 VNC         Enable/Disable graphical remote access to your Pi using RealVNC
  P4 SPI         Enable/Disable automatic loading of SPI kernel module
  P5 I2C         Enable/Disable automatic loading of I2C kernel module
  P6 Serial      Enable/Disable shell and kernel messages on the serial connection
  P7 1-Wire      Enable/Disable one-wire interface
  P8 Remote GPIO Enable/Disable remote access to GPIO pins
```

Reboot Raspberry Pi and check I2C devices:

```
sudo reboot

i2cdetect -y 1
```

```
pi@raspberrypi:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- 29 -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi:~ $
```

HARDWARE CONNECTION



TCS34725 Color Sensor	Raspberry Pi
VCC	3.3V
GND	GND
SDA	SDA
SCL	SCL
INT	17
LED	18

RUNNING EXAMPLE

BCM2835 example

```
cd bcm2835

sudo ./main
```

WiringPi example

```
cd wiringpi

sudo ./main
```

python example

```
cd python

sudo python main.py
```

Note: If you get error information that files are not exist when running BCM2835 or wiringpi example, please execute make command and try again.

EXPECTED RESULT

The expected result of three examples are similar, here we take python codes as example:

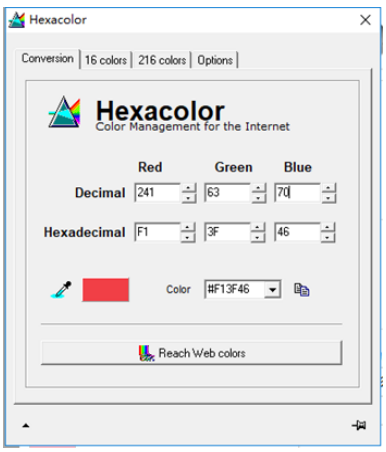
R, G, B value are printed in RGB888 format (DEC), C is light value without processing, RGB565 and RGB888 are HEX data printed in certain format. LUX is light value processed. CT is color temperature. (https://en.wikipedia.org/wiki/Color_temperature)

If you want to measure CT, please turn off LED. INT is interrupt, 1: light value is over threshold.

```
pi@raspberrypi:                                  $ sudo python main.py
TCS34725 initialization success!!
R: 252  G: 70  B: 90  C: 0x7d16  RGB565: 0xfa2b  RGB888: 0xfc465a  LUX: 112  CT: 2339K  INT: 1
R: 252  G: 71  B: 90  C: 0x7d89  RGB565: 0xfa2b  RGB888: 0xfc475a  LUX: 113  CT: 2340K  INT: 0
R: 251  G: 70  B: 90  C: 0x7d23  RGB565: 0xfa2b  RGB888: 0xfb465a  LUX: 113  CT: 2342K  INT: 0
R: 252  G: 71  B: 90  C: 0x7d7b  RGB565: 0xfa2b  RGB888: 0xfc475a  LUX: 113  CT: 2339K  INT: 0
R: 251  G: 70  B: 90  C: 0x7d2a  RGB565: 0xfa2b  RGB888: 0xfb465a  LUX: 113  CT: 2342K  INT: 0
R: 251  G: 70  B: 90  C: 0x7cf6  RGB565: 0xfa2b  RGB888: 0xfb465a  LUX: 113  CT: 2341K  INT: 0
R: 252  G: 70  B: 90  C: 0x7d3c  RGB565: 0xfa2b  RGB888: 0xfc465a  LUX: 113  CT: 2339K  INT: 0
R: 252  G: 70  B: 90  C: 0x7d34  RGB565: 0xfa2b  RGB888: 0xfc465a  LUX: 113  CT: 2341K  INT: 0
R: 252  G: 70  B: 90  C: 0x7d50  RGB565: 0xfa2b  RGB888: 0xfc465a  LUX: 113  CT: 2340K  INT: 0
R: 251  G: 70  B: 90  C: 0x7d5a  RGB565: 0xfa2b  RGB888: 0xfb465a  LUX: 114  CT: 2349K  INT: 0
R: 253  G: 71  B: 91  C: 0x7d23  RGB565: 0xfa2b  RGB888: 0xfd475b  LUX: 113  CT: 2349K  INT: 0
R: 252  G: 71  B: 90  C: 0x7f99  RGB565: 0xfa2b  RGB888: 0xfc475a  LUX: 114  CT: 2332K  INT: 0
R: 253  G: 71  B: 90  C: 0x7d81  RGB565: 0xfa2b  RGB888: 0xfd475a  LUX: 112  CT: 2338K  INT: 0
R: 252  G: 70  B: 90  C: 0x7d01  RGB565: 0xfa2b  RGB888: 0xfc465a  LUX: 112  CT: 2339K  INT: 0
R: 252  G: 70  B: 90  C: 0x7d2d  RGB565: 0xfa2b  RGB888: 0xfc465a  LUX: 113  CT: 2342K  INT: 0
R: 252  G: 71  B: 90  C: 0x7d9c  RGB565: 0xfa2b  RGB888: 0xfc475a  LUX: 113  CT: 2341K  INT: 0
R: 252  G: 70  B: 90  C: 0x7d3d  RGB565: 0xfa2b  RGB888: 0xfc465a  LUX: 112  CT: 2339K  INT: 0
R: 251  G: 70  B: 90  C: 0x7cea  RGB565: 0xfa2b  RGB888: 0xfb465a  LUX: 112  CT: 2339K  INT: 0
R: 252  G: 70  B: 90  C: 0x7d28  RGB565: 0xfa2b  RGB888: 0xfc465a  LUX: 112  CT: 2342K  INT: 0
R: 253  G: 71  B: 90  C: 0x7d91  RGB565: 0xfa2b  RGB888: 0xfd475a  LUX: 112  CT: 2340K  INT: 0
R: 252  G: 71  B: 90  C: 0x7d71  RGB565: 0xfa2b  RGB888: 0xfc475a  LUX: 112  CT: 2339K  INT: 0
R: 251  G: 70  B: 90  C: 0x7cfb  RGB565: 0xfa2b  RGB888: 0xfb465a  LUX: 112  CT: 2340K  INT: 0
```

You can turn the RGB value to color with tools below:

https://www.waveshare.com/w/upload/5/53/Infrared-Temperature-Sensor-Code.7z



Open STM32 project with Keil uVision5. The example is based on HAL libraries.

Development board used is Waveshare XNUCLEO-F103RB, the chip is

STM32F103RBT6. Example uses UART2 (PA2, PA3) to print data, 115200, 8N1.

HARDWARE CONNECTION

TCS34725 Color Sensor	STM32
VCC	3.3V
GND	GND
SDA	SDA/D14/PB9
SCL	SCL/D15/PB8
INT	D8/PA9
LED	PWM1/D9/PC7

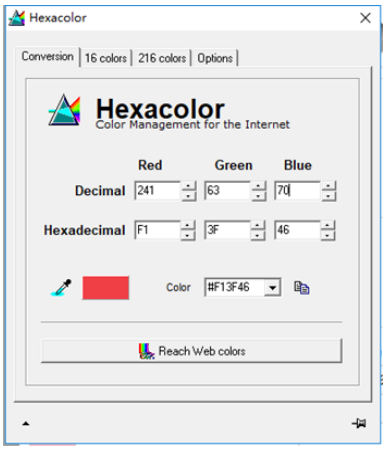
EXPECTED RESULT

This is the output when testing red

```
RGB888 :R=242   G=63  B=71
RGB888=0XF23F47  RGB565=0X97E7
Lux_Interrupt = 0

RGB888 :R=241   G=62  B=70
RGB888=0XF13E46  RGB565=0X8FC6
Lux_Interrupt = 0

RGB888 :R=243   G=63  B=71
RGB888=0XF33F47  RGB565=0X9FE7
Lux_Interrupt = 0

RGB888 :R=243   G=63  B=71
RGB888=0XF33F47  RGB565=0X9FE7
Lux_Interrupt = 0
```

You can turn the RGB value to color with tools below:

https://www.waveshare.com/w/upload/5/53/Infrared-Temperature-Sensor-Code.7z



ARDUINO

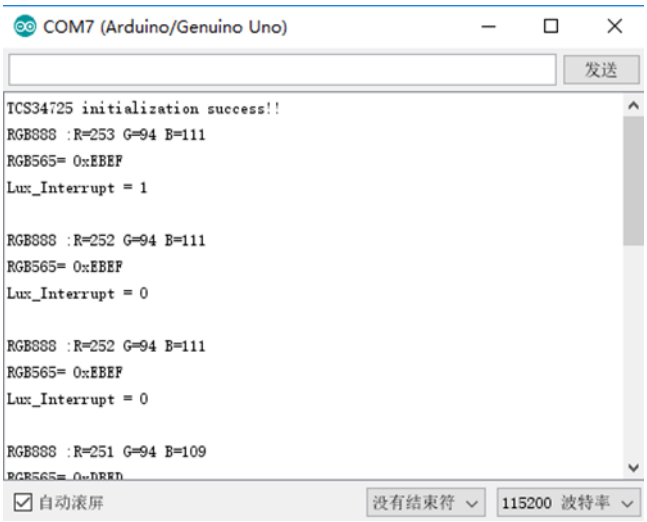
The development board used is Waveshare UNO PLUS(Compatible with Arduino UNO), set serial monitor to 115200

HARDWARE CONNECTION

TCS34725 Color Sensor	Arduino
VCC	3.3V/5V
GND	GND
SDA	SDA
SCL	SCL
INT	D8
LED	D9

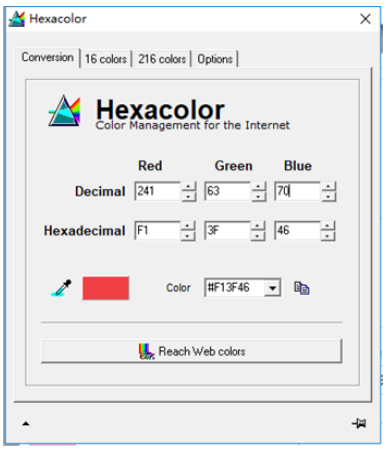
EXPECTED RESULT

This is the output when testing red



You can turn the RGB value to color with tools below:

https://www.waveshare.com/w/upload/5/53/Infrared-Temperature-Sensor-Code.7z



FAQ

1. Q: Raspberry Pi example initializing failed?

```
bcm2835 init success !!!
TCS34725 initialization error!!
```

```
Traceback (most recent call last):
  File "main.py", line 28, in <module>
    GPIO.cleanup()
NameError: name 'GPIO' is not defined
```

A: Please check if you connect sensor correctly, and check i2C device with

command i2cdetect -y 1

```
pi@raspberrypi:~                                $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- 29 -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Please correct connecting and restart

2. Q: What happened when running example by mistake?

A: If you find that python or bcm2835 examples cannot work properly after

running wiringpi codes, please just restart Raspberry Pi can test again

3. Q: Data output are incorrect when using STM32 and Arduino examples?

A: Please check if you choose the correct COM port (according to device manager).

If all the setting are correct, please exchange RXD and TXD and try again.

4. Q: Why the RGB data outputted are all 0

```
RGB888 :R=0   G=0  B=0
RGB888=0X0  RGB565=0X0
Lux_Interrupt = 0

RGB888 :R=0   G=0  B=0
RGB888=0X0  RGB565=0X0
Lux_Interrupt = 0

RGB888 :R=0   G=0  B=0
RGB888=0X0  RGB565=0X0
Lux_Interrupt = 0
```

A: Please check if you connect device correctly then press reset button

```
TCS34725 initialization error!!
TCS34725 initialization error!!
TCS34725 initialization error!!
TCS34725 initialization error!!
TCS34725 initialization error!!
TCS34725 initialization error!!
TCS34725 initialization error!!
TCS34725 initialization error!!
```

5. Q: The RGB data output are all 253?

```
R: 253  G: 253  B: 253  C: 0xffff  RGB565: 0xffff  RGB888: 0xfdfdfd  LUX: 0  CT: 5201K  INT: 1
R: 253  G: 253  B: 253  C: 0xffff  RGB565: 0xffff  RGB888: 0xfdfdfd  LUX: 0  CT: 5201K  INT: 0
R: 253  G: 253  B: 253  C: 0xffff  RGB565: 0xffff  RGB888: 0xfdfdfd  LUX: 0  CT: 5201K  INT: 0
R: 253  G: 253  B: 253  C: 0xffff  RGB565: 0xffff  RGB888: 0xfdfdfd  LUX: 0  CT: 5201K  INT: 1
R: 253  G: 253  B: 253  C: 0xffff  RGB565: 0xffff  RGB888: 0xfdfdfd  LUX: 0  CT: 5201K  INT: 0
R: 253  G: 253  B: 253  C: 0xffff  RGB565: 0xffff  RGB888: 0xfdfdfd  LUX: 0  CT: 5201K  INT: 1
R: 253  G: 253  B: 253  C: 0xffff  RGB565: 0xffff  RGB888: 0xfdfdfd  LUX: 0  CT: 5201K  INT: 0
R: 253  G: 253  B: 253  C: 0xffff  RGB565: 0xffff  RGB888: 0xfdfdfd  LUX: 0  CT: 5201K  INT: 0
R: 253  G: 253  B: 253  C: 0xffff  RGB565: 0xffff  RGB888: 0xfdfdfd  LUX: 0  CT: 5201K  INT: 1
```

A: The light intensity value is over measure range, you can try to modify the gain parameter in initial codes, or add statement TCS34725_Set_Gain(TCS34725_GAIN_16X)） following initialize part.

6. Q: Color detect is abnormal after modifying integrate time

A: The integrate time is relate to maximum data of RGB channels. If the color turns darker or lighter after modification, please try to change the brightness of LED

7. Q: Why interrupt cannot be triggered or be triggered all the time after modifying integrate time

A: Interrupt is relate to data of Clear channel. Data of Clear channel is influenced by integrate time. When gain is 60:

Integrate time	Max value of Channel
2.4ms	1024
24ms	10240
50ms	5400
101ms	21504
154ms	65535
700ms	65535

Therefore, you should modify the threshold value if sample rate is fast. And please increase brightness of LED when you set integrate time to 2.4ms.