Data Sheet

Optical Fingerprint Sensor Module

GTS-511E2

Version : V 1.1

www.adh-tech.com.tw

sales@adh-tech.com.tw

Content

■ Overview

GTS-511E2 is the thinnest optical touch fingerprint sensor in the world. It is suitable for a wide range of applications such as Portable Device, Smart Phone, Door Lock, Suitcase and etc. It can thus replace current login ID and key to further enhance the level of security.

■ Features

1. Characteristics

1.1. Just one touch, easy to enroll.
1.2. The thinnest fingerprint optical module.
1.3. Ultra-high DPI without image distortion.
1.4. High C/P ratio

2. Uniqueness

2.1. Special surface design for dry finger.
2.2. Resistant to static electricity and durable.
2.3. Resists 2D fake fingerprint and gives higher safety coefficient.

3. Standardization

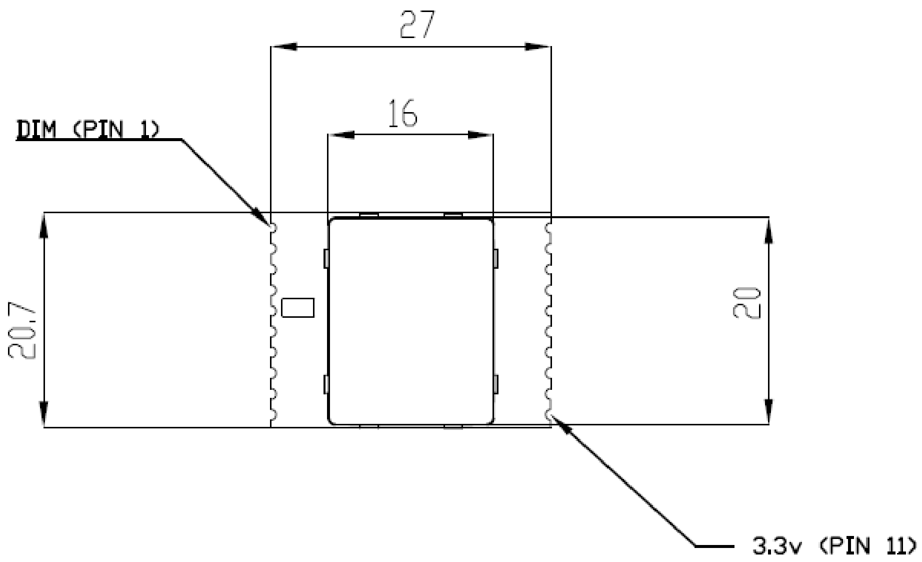
3.1. Mass production with high quality (based on the manufacture procedure of CMOS CAM Module).

■ General Spec

Item	GTS-511E2
Dimension(Plastic Housing)	16 x 20 x (h:6.5±0.1) mm
Sensor	VGA CMOS Image Sensor
Image resolution	430 DPI +/- 2% for QVGA 860 DPI +/- 2% for VGA
Effective sensing area	12 x 14 mm
Image gray scale	8bit/pixel, max 256 gray level
Indicator	Blue LED
Interface	20-Pin Parallel
TV Distortion	< 4%
Operating Temperature	-20℃ ~ 60℃
Operating Humidity	0~80%, Non-condense
Power	DC 3.3V
Power consumption	MCLK = 24MHZ：52mA

■ Hardware Description

1. Pin Assignment



Pin	Define	Pin	Define
1	DIM	20	D4
2	D5	19	PCLK
3	D6	18	D3
4	D7	17	D2
5	SCL	16	D1
6	SDA	15	D0
7	VSYNC	14	LED_CON
8	HSYNC	13	MCLK
9	RSTB	12	DGND
10	PWDN	11	3.3V

2. Pin Description

Pin Define	In/Out	Function Description
DIM	N/A	Reserved, please Float this Pin.
LED_CON	In	LED Light Control S/W, Active High.
RSTB	In	Sensor Reset Pin, Active Low.
D0~D7	Out	Image Data Line, D0 is LSB.
SCL & SDA	In/Out	I2C interface, Pull High Resistance(4.7K)
VSYNC & HSYNC	Out	Setting Image dimension by Clock.
MCLK	In	Input Clock.
PCLK	Out	Output Clock.
PWDN	In	Sensor Enable Pin, Active Low.

3. Mechanical Dimensions

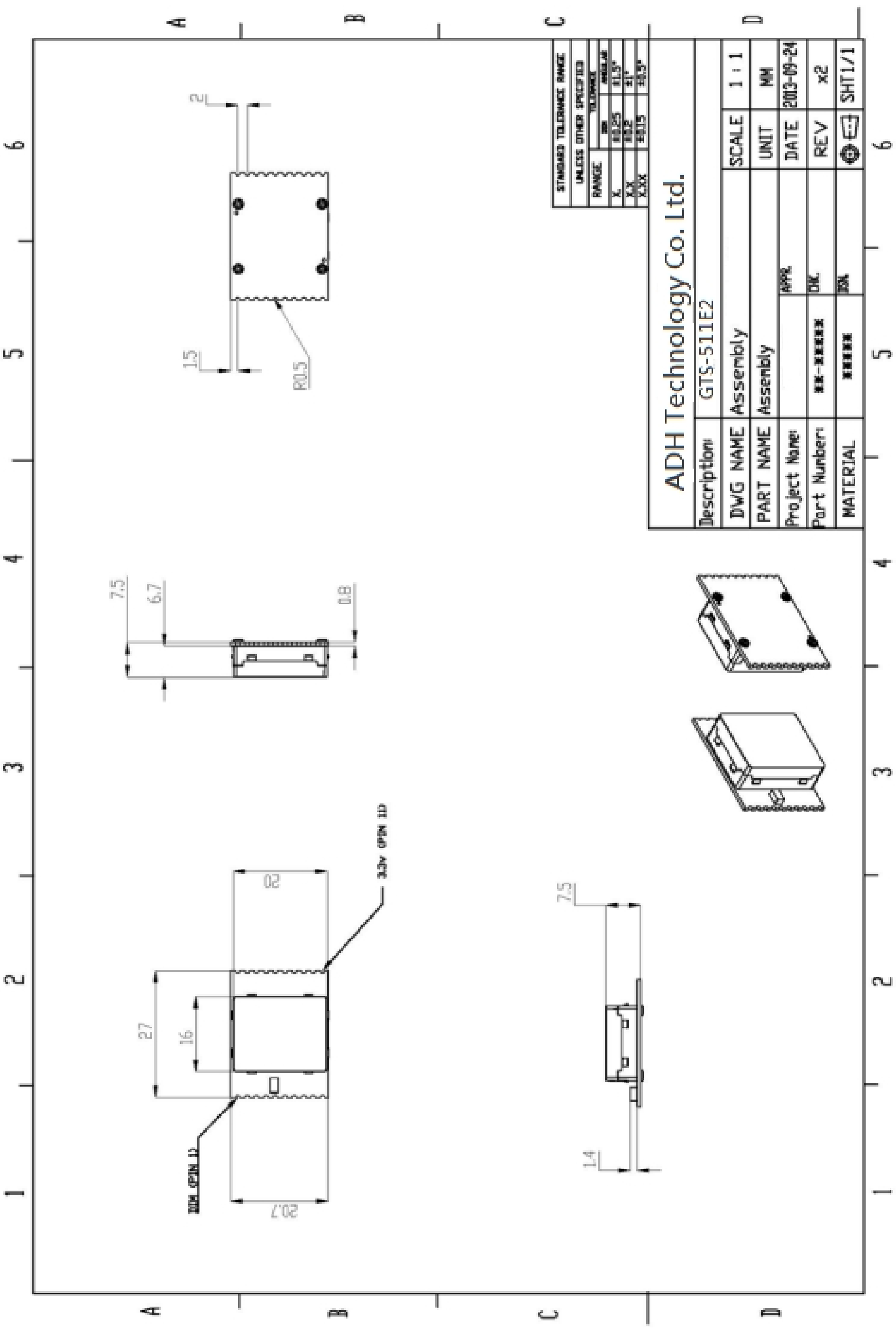


Image Acquisition

1. Default Value

GTS-511E2 provides a set of **16-byte** default register setting to give a unified image quality. These setting value are stored in the EEPROM and can be read through the I^2C interface. ***The address of the EERPOM will vary based on the MCU, please refer to the EERPOM.c and the datasheet of HT2201 for more details.***

Below are the corresponding descriptions of each value:

Address	Description
EEPROM[0]	Coordinate X of the Center of Black circle X+112 = Center X on the side of 320
EEPROM[1]	Coordinate Y of the Center of Black circle Y+108 = Center Y on the side of 240
EEPROM[2]	Reserved
EEPROM[3]	Reserved
EEPROM[4]	Reserved
EEPROM[5]	Global Gain
EEPROM[6]	High Byte of Exposure Time
EEPROM[7]	Low Byte of Exposure Time
EEPROM[8]	Reserved
EEPROM[9]	Reserved
EEPROM[10]	Reserved
EEPROM[11]	Reserved
EEPROM[12]	Reserved
EEPROM[13]	Reserved
EEPROM[14]	*Must both be 'G' to be a valid EEPROM data*
EEPROM[15]	

2. Register Setting

The register is set through the I^2C interface. Switch to the corresponding page before setting the value of each register.

To switch to a specific page:

i2c_send_data(0xfe, Page); **// Set the register page**

PAGE 0x00

Address	Default	Description
0x03	EEPROM[6]	High Byte of Exposure Time
0x04	EEPROM[7]	Low Byte of Exposure Time
0x09	0x00	Windows mode 640X480
0x0A	0x04	
0x0B	0x00	
0x0C	0x00	
0x0D	0x01	
0x0E	0xE8	
0x0F	0x02	
0x10	0x88	
0x11	0x2A	Sh_Delay
0x44	0xA2	Output format(YCbCr)
0x50	0x01	Crop out window mode
0x51	0x00	Subsample output 320X240
0x52	0x00	
0x53	0x00	
0x54	0x00	
0x55	0x00	
0x56	0xF0	
0x57	0x10	
0x58	0x40	
0x70	EEPROM[5]	Global Gain

Note: Please refer to Datasheet of GC0329 Sensor for more details

3. Initialization of CMOS Sensor

Below are the sequences to initialize the CMOS Sensor:

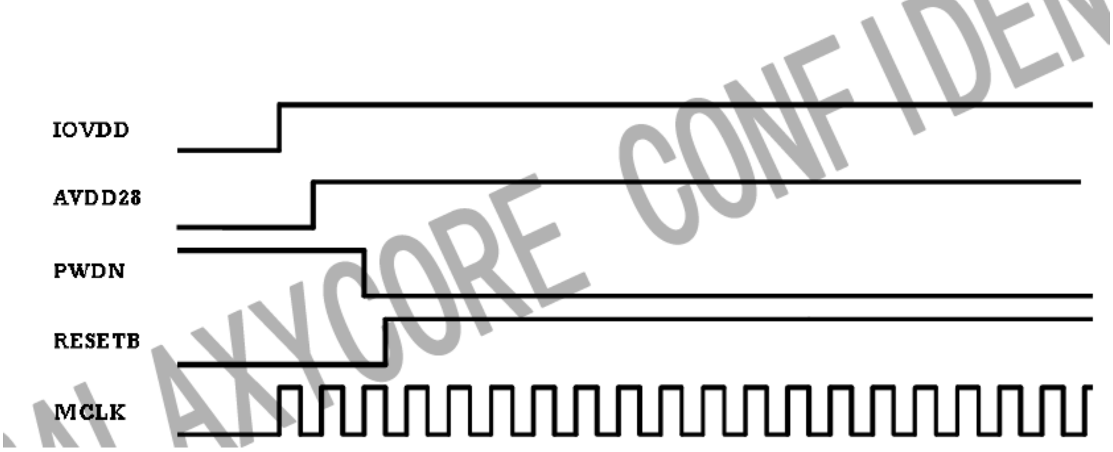
```
void Sensor_Init(void)
{
   u8 rtn;
   /* Reset sensor */
   GPIO_SetOutBits(GPIOA, GPIO_PIN_0); //PWDN High
   GPIO_ClearOutBits(GPIOA, GPIO_PIN_1);  //RESTB Low
   SystickDelayXmS(5);
   GPIO_ClearOutBits(GPIOA, GPIO_PIN_0); //PWDN Low
   SystickDelayXmS(1);
   GPIO_SetOutBits(GPIOA, GPIO_PIN_1);     //RESTB High
   I2C_WriteSensorRegister(0xfc , 0x16);
  // Soft reset
   I2C_WriteSensorRegister(0xfe , 0x80);
   I2C_WriteSensorRegister(0xfe , 0x00);
   I2C_WriteSensorRegister(0xfc , 0x16);
   I2C_WriteSensorRegister(0xf1 , 0x01);
   I2C_WriteSensorRegister(0xf0 , 0x07);
   I2C_WriteSensorRegister(0xfa , 0x11);
   I2C_WriteSensorRegister(0x24 , 0x3f);
   I2C_WriteSensorRegister(0x46 , 0x02);
   I2C_WriteSensorRegister(0x09 , 0x00);
   I2C_WriteSensorRegister(0x0a , 0x04);
   I2C_WriteSensorRegister(0x0b , 0x00);
   I2C_WriteSensorRegister(0x0c , 0x00);
   I2C_WriteSensorRegister(0x0d , 0x01);
   I2C_WriteSensorRegister(0x0e , 0xe8);
   I2C_WriteSensorRegister(0x0f , 0x02);
   I2C_WriteSensorRegister(0x10 , 0x88);
   I2C_WriteSensorRegister(0x17 , 0x14);
   I2C_WriteSensorRegister(0x19 , 0x05);
   I2C_WriteSensorRegister(0x1F , 0xC0);
   I2C_WriteSensorRegister(0x1E , 0x15);
   I2C_WriteSensorRegister(0x20 , 0x00);
```

```
I2C_WriteSensorRegister(0x21 , 0x48);
I2C_WriteSensorRegister(0x22 , 0xDA);
I2C_WriteSensorRegister(0x23 , 0x41);
I2C_WriteSensorRegister(0x24 , 0x16);
I2C_WriteSensorRegister(0x26 , 0xF7);
I2C_WriteSensorRegister(0x33 , 0x20);
I2C_WriteSensorRegister(0x34 , 0x20);
I2C_WriteSensorRegister(0x35 , 0x20);
I2C_WriteSensorRegister(0x36 , 0x20);
I2C_WriteSensorRegister(0x41 , 0x00);
I2C_WriteSensorRegister(0x42 , 0xFE);
I2C_WriteSensorRegister(0x4F , 0x00);
I2C_WriteSensorRegister(0x70 , 0x40);
I2C_WriteSensorRegister(0x76 , 0x8A);
I2C_WriteSensorRegister(0xB0 , 0x00);
I2C_WriteSensorRegister(0xBC , 0x00);
I2C_WriteSensorRegister(0xBD , 0x00);
I2C_WriteSensorRegister(0xBE , 0x00);
I2C_WriteSensorRegister(0x4D , 0x03);
I2C_WriteSensorRegister(0xfe , 0x00);
I2C_WriteSensorRegister(0x4b , 0xCA);
I2C_WriteSensorRegister(0x50 , 0x01);
I2C_WriteSensorRegister(0x51 , 0x00);
I2C_WriteSensorRegister(0x52 , 0x00);
I2C_WriteSensorRegister(0x53 , 0x00);
I2C_WriteSensorRegister(0x54 , 0x00);
I2C_WriteSensorRegister(0x55 , 0x00);
I2C_WriteSensorRegister(0x56 , 0xf0);
I2C_WriteSensorRegister(0x57 , 0x01);
I2C_WriteSensorRegister(0x58 , 0x40);
I2C_WriteSensorRegister(0x59 , 0x22);
I2C_WriteSensorRegister(0x5a , 0x03);
I2C_WriteSensorRegister(0x5b , 0x00);
I2C_WriteSensorRegister(0x5c , 0x00);
I2C_WriteSensorRegister(0x5d , 0x00);
```

```
    I2C_WriteSensorRegister(0x5e , 0x00);
    I2C_WriteSensorRegister(0x5f , 0x00);
    I2C_WriteSensorRegister(0x60 , 0x00);
    I2C_WriteSensorRegister(0x61 , 0x00);
    I2C_WriteSensorRegister(0x62 , 0x00);
    I2C_WriteSensorRegister(0x03 , EEPROM[6]);
    I2C_WriteSensorRegister(0x04 , EEPROM[7]);
    I2C_WriteSensorRegister(0x70 , EEPROM[5]);
    I2C_WriteSensorRegister(0xD3 , 0x50);
    I2C_WriteSensorRegister(0x44 , 0xA2);
    I2C_WriteSensorRegister(0x11 , 0x2A);
    I2C_WriteSensorRegister(0x40 , 0xef);
}
```

Figure for hardware timing

4.2.1 Power On Sequence



4. Sample code for Getting Image

Below is the sample code to get a 320x240 image to the buffer. To clip an effective area, the **EEPROM[0,1]** gives the coordinate of black center relative to the left top corner

```
void get_image(u8* ptr_img)
{
        u32   i,j;
        u8*   tmp_ptr;
        cmos_led_on(true); //Set pin.2 to Hight Level
        while(ROM_GPIOPinRead(GPIO_PORTA_BASE,CMOS_VSYNC)==0)// pin.18==0
        {
        }
        while(ROM_GPIOPinRead(GPIO_PORTA_BASE,CMOS_VSYNC)!=0)// pin.18==1
        {
        }
        while(ROM_GPIOPinRead(GPIO_PORTA_BASE,CMOS_VSYNC)!=0) // pin.18==1
        {
        }
        for   (j=0;j<240;j++)
        {
          while(GPIOPinRead(GPIO_PORTA_BASE,CMOS_HSYNC)==0)    //    pin.19==0
          {
          }
          tmp_ptr=ptr _img+j;
          for(i=0;i<320;i++)
          {
             *(tmp_ptr+i*240)=*(u8*)(0x400073FC);     //Store image data (1 byte)
             while(GPIOPinRead(GPIO_PORTA_BASE,CMOS_PCLK) !=RESET) ;//pin.23==1
             while(GPIOPinRead(GPIO_PORTA_BASE,CMOS_PCLK) ==RESET) ; // pin.23==0
             while(GPIOPinRead(GPIO_PORTA_BASE,CMOS_PCLK) !=RESET) ; // pin.23==1
          }
          while(GPIOPinRead(GPIO_PORTA_BASE,CMOS_HSYNC) !=RESET);    //pin.18==1
        }
        cmos_led_on(false);//Set pin.2 to Low Level
}
```

5. Output images

- The default values are set based on **MCLK = 24MHz** and **PCLK = 12MHz**.
- For slower clock, the images will become **brighter** with the same values.
- Thus, the default exposure time **EEPROM[6~7]** will need to be **lowered** dependently to get the same image quality.
- The equation below gives roughly the same brightness for different PCLK:

  Y = (EEPROM[6] * 256 + EEPROM[7]) / (12 / X)

  *where **X** is the new PCLK and **Y** is the new exposure time*

- Set **Y** to the corresponding registers as below and then fine tune the value to get the final desired image:

```
I2C_WriteSensorRegister( (0xfe, 0x00);     // Select PAGE 0x00
I2C_WriteSensorRegister( (0x03, Y/256);    // High byte
I2C_WriteSensorRegister( (0x04, Y%256);    // Low byte
```

- If the final fine-tuned exposure time is **Y**, save **Y** to the EERPOM for future initialization of the CMOS sensor :

  EEPROM[6] = Y/256

  EEPROM[7] = Y%256