TrueSecure™ GT Series Fingerprint Sensor

Revision: V1.20 Date: June 26, 2016

www.holtek.com

Table of Contents

1 Features

- Single chip fingerprint recognition sensors
- Advanced optical technology
- High accuracy and high recognition speed
- Ultra-thin profile
- Internal I^2C interface for parameter setting
- High C/P Ratio
- CCIR601 Interface
- Special surface design can accommodate dry fingers
- Static electricity resistant durable design
- Resists 2D non-genuine fingerprints for higher safety coefficients

2 Applications

- Notebook computer login
- Household security products
- Vehicle entry systems
- Biometric identification products

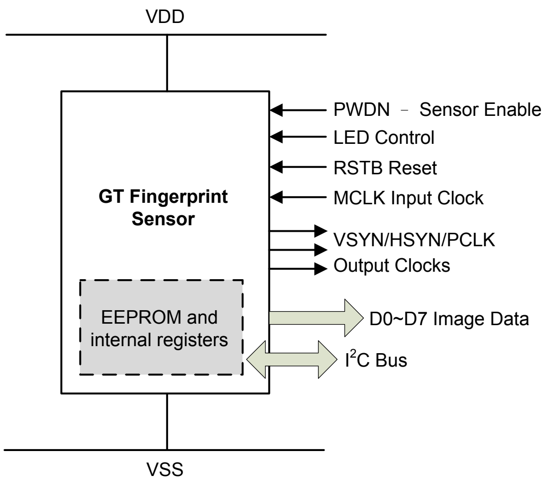
3 Selection Table

The range of devices shown in the selection table are similar in function, but differ mainly in their connector types and dimensions etc.

Part No.	Effective Area (mm^2)	Resolution (DPI)	Image Pixels	Surface Color	Interface	Connector	Dimension (mm^3) (W×D×H)
GT-5110E1-02	14×12.5	450	320×240	White	CCIR601	24-pin FPC	17.0×33.0×7.74
GT-5120E2-02						Stamp Hole	27.0×20.7×7.74

4 Block Diagram

The following shows a simplified version of the internal functions of the fingerprint module.



5 General Description

Fingerprint recognition technology is proving to be an increasingly popular means of secure and accurate means of biometric identification. By eliminating the need to remember multiple passwords this biometric recognition technology will continue to see more prevalent use in everyday products where security features are required. This range of Holtek fingerprint sensors with their advanced optical technology, ultra thin profile, durable structure, high ESD immunity and other features combine to provide designers with a reliable and highly functional sensor for products requiring fingerprint recognition functions.

6 Pin Description

The fingerprint sensors have an 8-bit parallel image data output structure along with several control lines. Note that the pin out description depends upon the part number suffix either E1 or E2.

Pin No.	I/O	Pin Name	Function
1	—	—	Reserved – leave floating
2	I	LED_CON	LED Control – Active high
3	I	RSTB	Reset – Active low
4	O	D0	Data output
5	O	D1	Data output
6	O	D2	Data output
7	O	D3	Data output
8	O	D4	Data output
9	O	D5	Data output
10	O	D6	Data output
11	O	D7	Data output
12	—	3.3V	Power
13	—	3.3V	Power
14	I/O	SCL	I^2C Clock
15	I/O	SDA	I^2C Data
16	—	GND	Ground
17	—	GND	Ground
18	O	VSYNC	Horizontal Synch
19	O	HSYNC	Vertical Synch
20	—	GND	Ground
21	I	MCLK	Input Clock
22	—	GND	Ground
23	O	PCLK	Output Clock
24	I	PWDN	Sensor Enable Pin – active low

GT-5110E1

Pin No.	I/O	Pin Name	Function
1	—	—	Reserved
2	O	D5	Data output
3	O	D6	Data output
4	O	D7	Data output
5	I/O	SCL	I^2C Clock
6	I/O	SDA	I^2C Data
7	O	VSYN	Horizontal Synch
8	O	HSYN	Vertical Synch
9	I	RSTB	Reset – Active low
10	I	PWDN	Sensor Enable Pin – active low
11	—	3.3V	Power
12	—	GND	Ground
13	I	MCLK	Input Clock
14	I	LED_CON	LED Control – Active high
15	O	D0	Data output
16	O	D1	Data output
17	O	D2	Data output
18	O	D3	Data output
19	O	PCLK	Output Clock
20	O	D4	Data output

GT-5120E2

7 Electrical Specifications

Symbol	Parameter	Test Conditions	Min	Typ	Max	Units
V_{DD}	Operating Voltage	—	3.3	5.0	6.0	V
I_{DD}	Operating Current	No output pin load	—	—	52	mA
V_{OL}	D0-D7 input low voltage	—	—	—	TBD	V
V_{OH}	D0-D7 input high voltage	—	TBD	—	—	V

8 Other Specifications

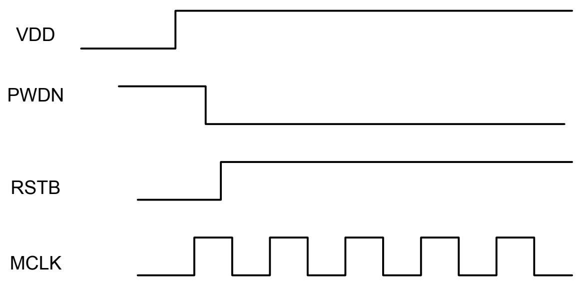
Item	Value
Sensor Interface	CCIR601
Image Gray Scale	8-bit/pixel – max 256 gray levels
TV Distortion	< 4%
Operating Environment Temp	-20°C ~ +60°C
Operating Environment Humidity	-0% ~ 80%

9 Functional Description

These fingerprint recognition modules include a CMOS image sensor which can capture a fingerprint image for processing. The image is transmitted out on an 8-bit CCIR601 interface controlled by the usual CMOS image sensor signals. An internal EEPROM stores certain sensor parameters and can be accessed using an internal I^2C interface. Note that pin 1 on these sensors is a reserved pin and must be left floating.

Power On Sequence

The sensors must be powered on in a certain manner to ensure their correct operation. The following timing diagram shows how this is implemented.



Internal EEPROM

An internal EEPROM in each of the sensors provides the devices with 16 locations for default setups. These setting are used to provide a unified image quality. These settings can be read out by external hardware via the device's I²C interface.

The following table provides a description of the 16 EEPROM data contents.

Address	Description
EEPROM[0]	Black circle centre X-coordinate X+120 = Center X on the side of 320
EEPROM[1]	Black circle centre Y-coordinate Y+108 = Center Y on the side of 240
EEPROM[2]	Reserved
EEPROM[3]	Reserved
EEPROM[4]	Reserved
EEPROM[5]	Reserved
EEPROM[6]	Reserved
EEPROM[7]	Exposure Time High Byte
EEPROM[8]	Exposure Time Low Byte
EEPROM[9]	Reserved
EEPROM[10]	Reserved
EEPROM[11]	Reserved
EEPROM[12]	Reserved
EEPROM[13]	Reserved
EEPROM[14]	Must both be "G" for valid EEPROM data
EEPROM[15]	

Internal Registers

All EEPROM registers are setup via the I^{2}C interface. Ensure that the correct page is selected before setting the value of each register. To select an EEPROM page the following instruction should be used:

i2c_send_data(0xfe, Page)

PAGE 0x00

Address	Default	Description
0x03	EEPROM[7]	Exposure Time High Byte
0x04	EEPROM[8]	Exposure Time Low Byte
0x09	0x00	Windows mode 640×480
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 320×240
0x52	0x00	
0x53	0x00	
0x54	0x00	
0x55	0x00	
0x56	0xF0	
0x57	0x10	
0x58	0x40	
0x70	0x75	Global Gain
0xD3	0x80	Contrast

CMOS Sensor Initialisation

The following instructions use the I²C interface to read and write to the internal ram registers. If the following initialisation is used and the input MCLK clock is 24MHz then PCLK output clock will be 12MHz. To initialise the fingerprint module sensor the following sequence should be used.

```
void Sensor_Init(void)
{
  u8 rtn;
  I2C_WriteSensorRegister(0xfc , 0x16);
  rtn = I2C_ReadSensorRegister(0x00);
  I2C_WriteSensorRegister(0xfe , 0x80);
  I2C_WriteSensorRegister(0xfe , 0x00);
  I2C_WriteSensorRegister(0xfc , 0x16);
  I2C_WriteSensorRegister(0xf1 , 0x01);
  I2C_WriteSensorRegister(0xf0 , 0x07);
  I2C_WriteSensorRegister(0xfa , 0x00);
  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(0x33 , 0x20);
  I2C_WriteSensorRegister(0x34 , 0x20);
  I2C_WriteSensorRegister(0x35 , 0xFF);
  I2C_WriteSensorRegister(0x36 , 0xFF);
  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[7]);
    I2C_WriteSensorRegister(0x04 , EEPROM[8]);
    I2C_WriteSensorRegister(0x44 , 0x02);
    I2C_WriteSensorRegister(0x11 , 0x2A);
    I2C_WriteSensorRegister(0x40 , 0xBf);
    I2C_WriteSensorRegister(0x41 , 0x02);
    I2C_WriteSensorRegister(0x70 , 0x75);
    I2C_WriteSensorRegister(0x7A , 0xFF);
    I2C_WriteSensorRegister(0x7B , 0x70);
    I2C_WriteSensorRegister(0x7C , 0xFF);
    I2C_WriteSensorRegister(0xD3 , 0x80);
    I2C_WriteSensorRegister(0x63 , 0x00);
    I2C_WriteSensorRegister(0x64 , 0x0B);
    I2C_WriteSensorRegister(0x65 , 0x19);
    I2C_WriteSensorRegister(0x66 , 0x2B);
    I2C_WriteSensorRegister(0x67 , 0x37);
    I2C_WriteSensorRegister(0x68 , 0x48);
    I2C_WriteSensorRegister(0x69 , 0x51);
    I2C_WriteSensorRegister(0x6A , 0x5E);
    I2C_WriteSensorRegister(0x6B , 0x97);
    I2C_WriteSensorRegister(0x6C , 0xB2);
    I2C_WriteSensorRegister(0x6D , 0xC9);
    I2C_WriteSensorRegister(0x6E , 0xDF);
    I2C_WriteSensorRegister(0x6F , 0xFF);
}
```

Sample code to Obtain an Image

The following is the sample code to place a 320×240 image into the buffer. To clip an effective area, EEPROM[0,1] gives the coordinates of the black centre relative to the top left 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
}
```

Image Output Quality

For the above the default values are based on an MCLK frequency of 24 MHz and a PCLK frequency of 12 MHz. If slower clocks are used, then the image will become brighter with the same default values. Therefore the default exposure time setup in EEPROM[7~8] will have to be reduced to obtain the same image quality.

The following equation provides approximately the same brightness for a different PCLK:

Y = (EEPROM[7] * 256 + EEPROM[8]) / (12 / X)

In the above equation X is the new PCLK and Y is the new exposure time

Set Y to the corresponding registers as shown by the following and then fine tune the value to get the final desired image:

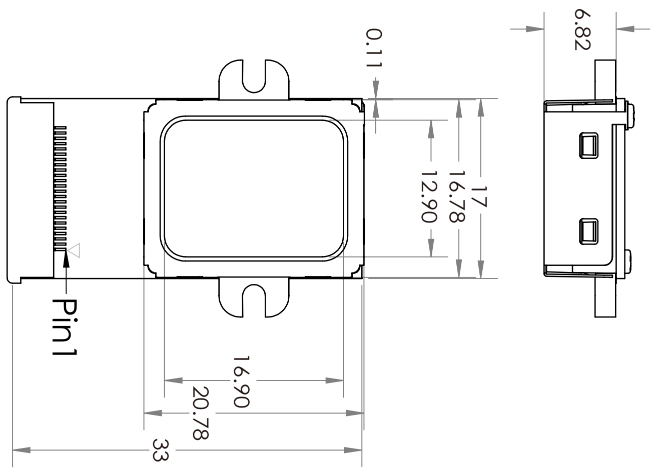
```
i2c_send_data(0xfe, 0x00); // Select PAGE 0x00
i2c_send_data(0x03, Y/256); // High byte
i2c_send_data(0x04, Y%256); // Low byte
```

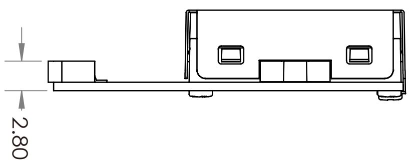
If the final fine-tuned exposure time is Y, save Y to the EERPOM to be used by future sensor initialisation processes.

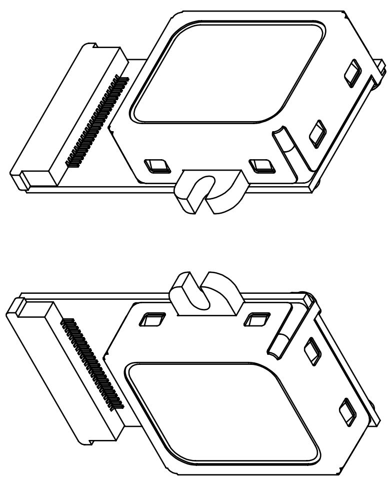
```
EEPROM[7] = Y/256
EEPROM[8] = Y%256
```

10 Mechanical Specifications

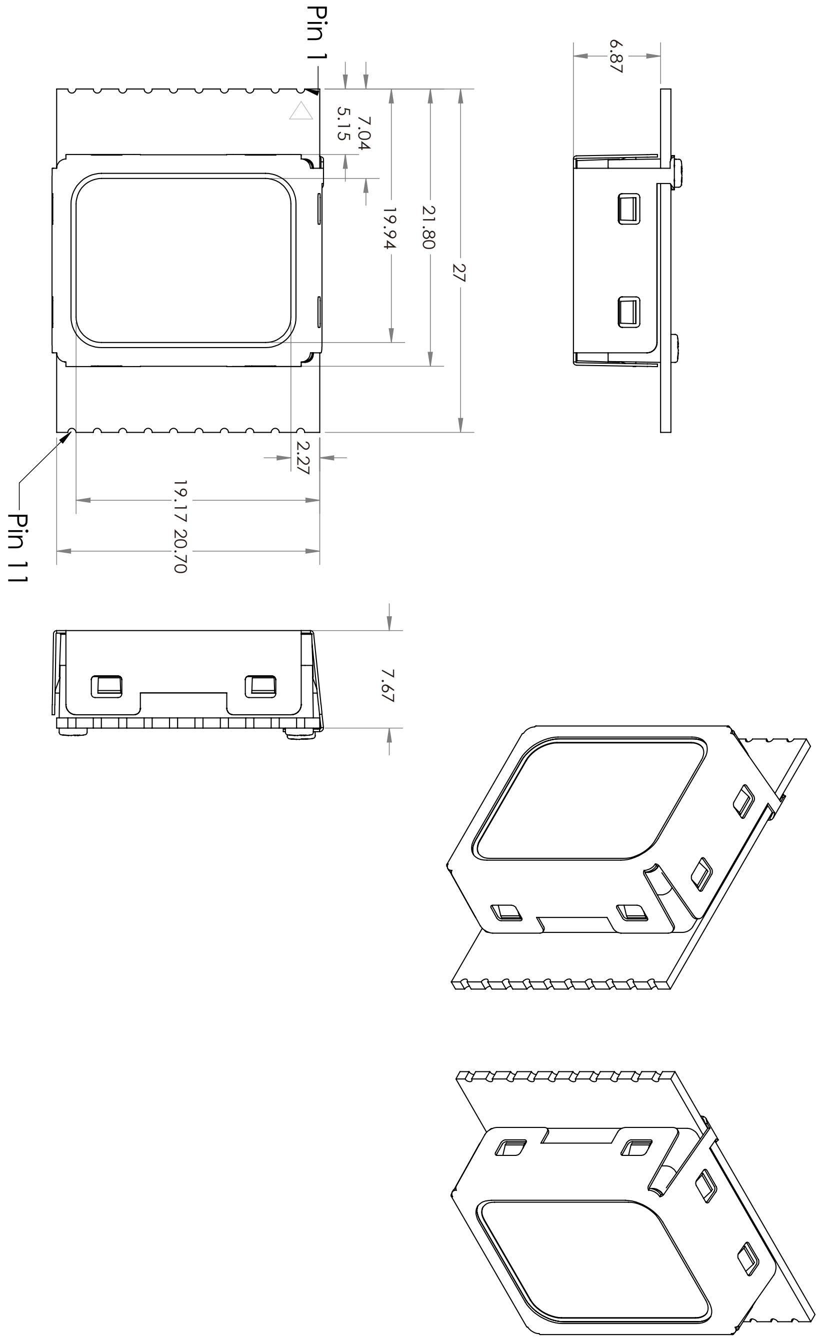
GT-5110E1







GT-5120E2



Note that Holtek's fingerprint recognition products have been designed in conjunction with Gingy Technology.