General Description

Holtek's human body infrared detector micro modules, the HT7M21xx series, come fully integrated with optical lenses, a passive infrared (PIR) sensor and DSP algorithms. These modules include a wide range of features such as low power consumption, an I²C communication interface and DSP algorithms which improve the reliability of the PIR detector. Their application range includes home security and surveillance system as well as basic industrial safety detection.

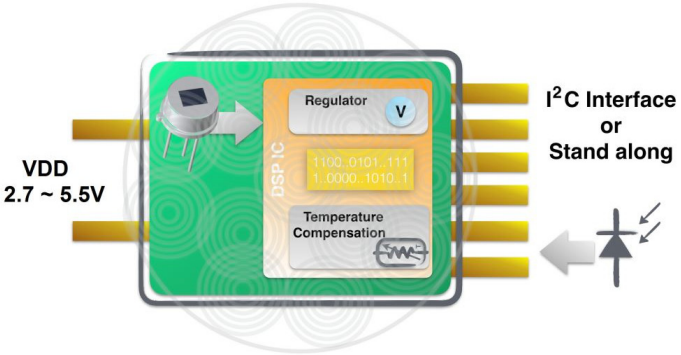
Features

- Operating voltage: 2.7V ~ 5.5V
- Operating temperature: -10 ~ 60°C
- Low power consumption:
  - Operating mode (Moving objects to be detected) < 1.5mA
  - Standby with detecting mode < 50μA (Operating voltage 3.3V)
- Intelligent signal recognition algorithm
- Optional communication interfaces: I²C for Network Mode or I/O for Stand-alone Mode
- Adjustable sensing sensitivity – Network Mode
- Customisable trigger modes: Single/Continuous – Network Mode
- Adjustable trigger output time: 16-bit×100ms – Network Mode
- Low voltage detection: 2.0/2.2/2.4/2.7/3.0/3.3/3.6/4.0V options – Network Mode
- Supports external optical sensors (eg. photo transistors)
- Integrated temperature sensor with temperature compensation
- Quick stabilisation: ready for stable operation within 12 seconds after power on

Applications

- Security Monitoring Systems
- Intelligent Lighting Control
- Home Appliances Energy-saving Control
- Office and Factory Equipment Automation

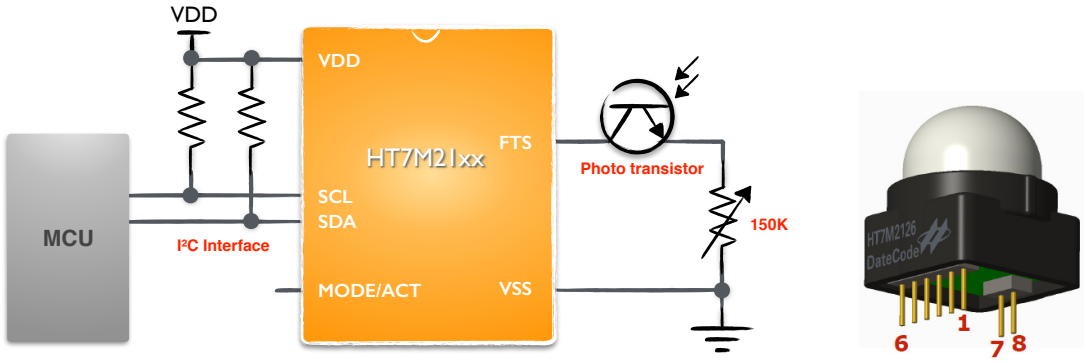
Block Diagram



Mode Selection

The MODE/ACT (MODE/DT) pin is used to select the Network or Stand-alone mode. When a pull-low resistor is externally connected between the MODE/ACT (MODE/DT) pin to the ground, the Stand-alone mode is selected. Otherwise, the Network mode is selected if a pull-high resistor or no resistor is externally on this pin.

Network Application Circuit – Network Mode

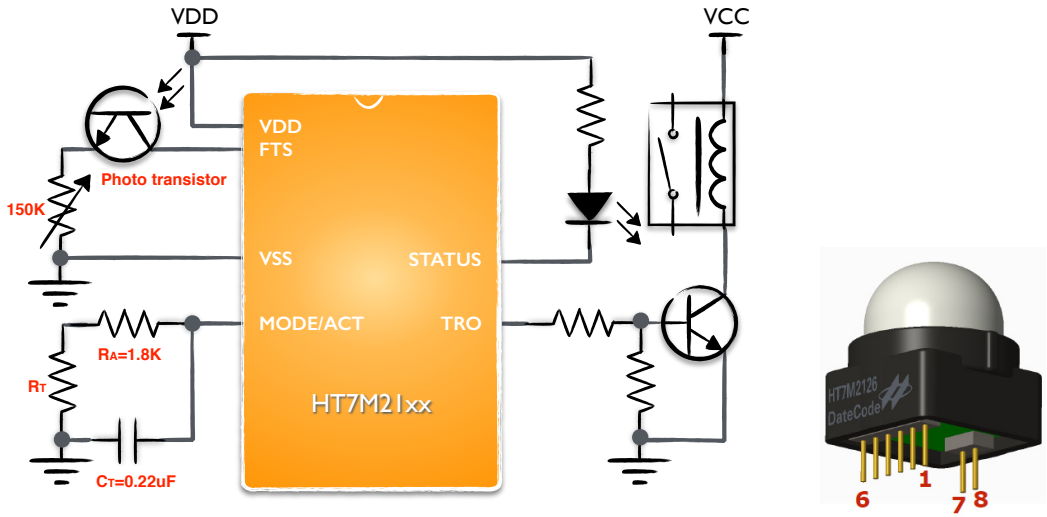


Interface & Pin Assignment – Network Mode

Pin #	Function	Description
1	VSS	Negative power supply, GND
2	VDD	Positive power supply
3	SDA	Serial Data Input/Output for I²C interface
4	SCL	Serial Clock Input for I²C interface
5	FTS	Photo transistor signal
6	VSS	Negative power supply, GND
7	MODE/ACT	Mode Selection/Motion Detection Output
8	TP1	No connection (Test pin)

Note: When the HT7M21xx selects Network mode and the internal enable bit ACTEN is high, the MODE/ACT pin will output a high pulse signal with a width of 30 seconds.

Stand-alone Application Circuit – Stand-alone Mode



Interface & Pin Assignment – Stand-alone Mode

Pin #	Function	Description
1	VSS	Negative power supply, GND
2	VDD	Positive power supply
3	STATUS	Warm-up/Detecting/Low voltage status
4	TRO	PIR trigger output
5	FTS	Photo transistor signal
6	VSS	Negative power supply, GND
7	MODE/DT	Mode & Duration time Selection
8	TP1	No connection (Test pin)

The default configurations for the Stand-alone Mode are as follow:

1. Continuous trigger mode

2. The TRO pin will output a high pulse signal when an available trigger is detected. The high pulse duration is determined by the external pull-low resistance of (R_A+R_T) together with the capacitance of C_T with a fixed value of 0.22μF.

External capacitance C_T = 0.22uF*, V_{DD} = 3.3V

RA+RT Resistance (Ω)	1.8K	2.2K	2.7K	3K	3.3K	3.6K	3.9K
TRO output duration time *	3sec.	10sec.	38sec.	1mins.	3mins.	5mins.	10mins.

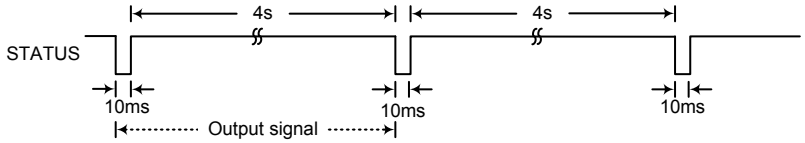
Note: (a) The TRO output high duration is 30 seconds in default if there is no external capacitor C_T.

(b) The value of the resistance, capacitance and TRO output duration time in the above table is used for reference only.

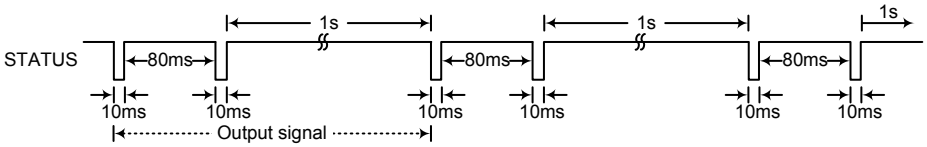
3. The STATUS pin will output various types of signals corresponding to the device in the warm-up mode, standby detection mode or low operating voltage mode where the operating voltage is lower than 2.7V respectively.

(a) Warm-up mode: A toggle output with a frequency of 2.5Hz will be output on the STATUS pin.

(b) Standy detection mode: A signal composed of a low pulse of 10ms and a high pulse of 4s will continuously be output on the STATUS pin.



(c) Low voltage mode: Two low pulses with a width of 10ms will first be output on the STATUS pin and the time duration between these two low pulses is 80ms. Then a high pulse with the width of 1 second will consecutively be output. Such an output signal will continuously be output on the STATUS pin.



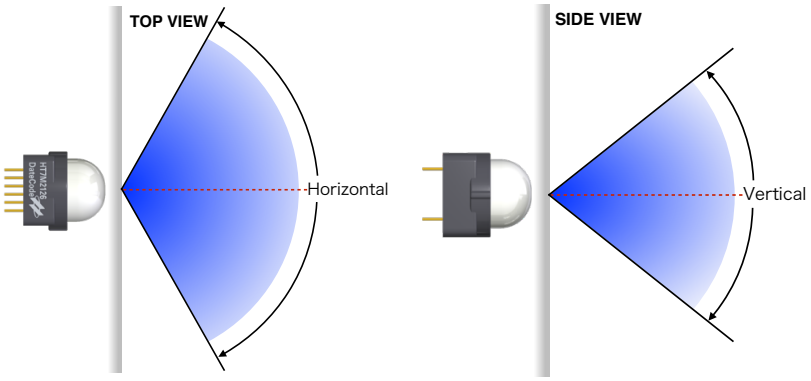
Light Sensing

If there is an external component such as a photo transistor, micro solar cell or CDS device connected to the FTS pin, the PIR module output function will be enabled when the voltage on the FTS pin is greater than $0.24\times V_{DD}$. Otherwise, the PIR module output function will be enabled in default.

A pull-high resistor with a value of 680kΩ is internally connected to the FTS pin in the PIR module. If the voltage on the FTS pin is greater than $0.24\times V_{DD}$ as the internal pull-high resistor and external component are all taken into account, the PIR module output function will be enabled. The voltage threshold to enable the PIR module output function is fixed as $0.24\times V_{DD}$ in the Stand-alone mode. If the device is in the Network mode, the voltage threshold can be adjusted by configuring the LUMI[6:0] field in the HT7M21xx device using the I^2C interface.

Detection Range

With regard to the HT7M2126, the horizontal view angle is 121°, the Pitch Angle is 77° and the detection distance has about 3.5~6 meters of visual range, as shown below.



HT7M21xx: Lenses FOV (Field of View)

In addition to the HT7M2126, the HT7M21xx series include the HT7M2136, the HT7M2156 and the HT7M2176, etc. Different modules will support different detection distances (far or close) as well as different horizontal and vertical viewing angles (narrow or wide).

Part Number	Viewing Angle H/V	Center Distance	Lens Color
HT7M2126	121° 77°	3.5 ~ 6 meters	Nature
HT7M2127	121° 77°	2.8 ~ 5 meters	Black
HT7M2136	91° 10°	5.5 ~ 8 meters	Nature
HT7M2156	10° 20°	8 ~ 12 meters	Nature
HT7M2176	86° 75°	5 ~ 7.5 meters	Nature

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}-0.3V to V_{SS}+6.0V
Input Voltage ..... V_{SS}-0.3V to V_{DD}+0.3V
Storage Temperature ..... -40°C to 80°C
Operating Temperature ..... -10°C to 60°C
I_{OL} Total 150mA
I_{OH} Total ..... -100mA
Total Power Dissipation 500mW

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to these devices. Functional operation of these devices at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may

affect devices reliability.

Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.7	3.3	5.5	V
I_{DD1}	Operating Current	3.3V	Moving objects to be detected	—	1.2	2.0	mA
I_{DD2}	Operating Current	3.3V	Standby with detection mode, the ACC wake-up time is 4ms.	—	30	50	μA
T_{PIR}	PIR Stabilization Time	3.3V	—	—	12	—	s
V_{IL}	Input Low Voltage (I/O)	—	—	0	—	$0.2V_{DD}$	V
V_{IH}	Input High Voltage (I/O)	—	—	$0.8V_{DD}$	—	V_{DD}	V
I_{OL}	I/O Port Sink Current	3.3V	V_{OL}=0.1V_{DD}	6	12	—	mA
		5V		10	25	—	
I_{OH}	I/O Port Source Current	3.3V	V_{OH}=0.9V_{DD}	-2	-4	—	mA
		5V		-5	-8	—	
R_{PH}	Pull-high Resistance (FTS)	—	—	-5%	680	+5%	kΩ
V_{LVD}	Low Voltage Detector Voltage	—	LVDEN = 1, V_{LVD} = 2.0V	-5%	2.0	+5%	V
			LVDEN = 1, V_{LVD} = 2.2V		2.2		
			LVDEN = 1, V_{LVD} = 2.4V		2.4		
			LVDEN = 1, V_{LVD} = 2.7V		2.7		
			LVDEN = 1, V_{LVD} = 3.0V		3.0		
			LVDEN = 1, V_{LVD} = 3.3V		3.3		
			LVDEN = 1, V_{LVD} = 3.6V		3.6		
			LVDEN = 1, V_{LVD} = 4.0V		4.0		

A.C. Characteristics – I²C Interface

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SCL}	Clock Frequency	—	—	—	—	400	kHz
t_{BUF}	Bus Free Time	—	Time in which the bus must be free before a new transmission can start	1.3	—	—	μs
$t_{HD: STA}$	Start Condition Hold Time	—	After this period, the first clock pulse is generated	0.6	—	—	μs
t_{LOW}	SCL Low Time	—	—	1.3	—	—	μs
t_{HIGH}	SCL High Time	—	—	0.6	—	—	μs
$t_{SU: STA}$	Start Condition Setup Time	—	Time only relevant for repeated START condition	0.6	—	—	μs
$t_{HD: DAT}$	Data Hold Time	—	—	0	—	—	ns
$t_{SU: DAT}$	Data Setup Time	—	—	100	—	—	ns
t_R	SDA and SCL Rise Time	—	Note	—	—	0.3	μs
t_F	SDA and SCL Fall Time	—	Note	—	—	0.3	μs
$t_{SU: STO}$	Stop Condition Set-up time	—	—	0.6	—	—	μs
t_{AA}	Output Valid from Clock	—	—	—	—	0.9	μs
t_{SP}	Input Filter Time Constant (SDA and SCL Pins)	—	Noise suppression time	—	—	50	ns

Note: These parameters are periodically sampled but not 100% tested.

Temperature Sensor Characteristics

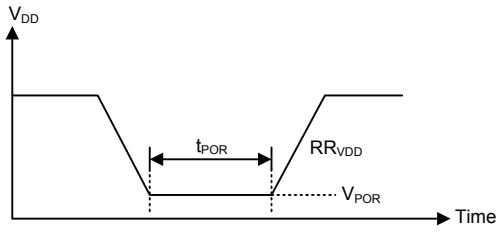
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Analog Voltage	—	—	2.7	—	5.5	V
V_{REFO}	Bandgap Output voltage	3V	No Load	-3%	1.04	+3%	V
V_{TPS}	Temperature Sensor Voltage	—	Bypass pre-buffer	-10%	0.91	+10%	V
T_{slope}	Temperature Sensor Slope	—	Bypass pre-buffer	—	3.12	—	mV/°C

Power-on Reset Characteristics

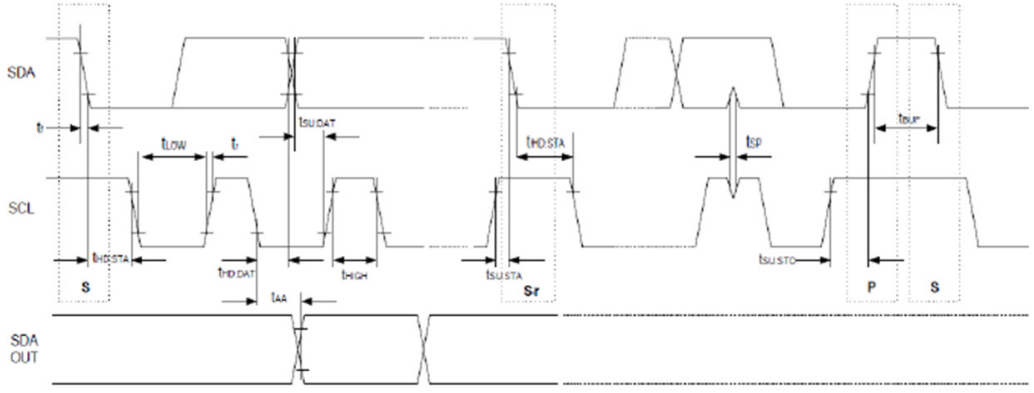
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{POR}	V_{DD} Start Voltage to Ensure Power-on Reset	—	—	—	—	100	mV
RR_{VDD}	V_{DD} Rising Rate to Ensure Power-on Reset	—	—	0.035	—	—	V/ms
t_{POR}	Minimum Time for V_{DD} Stays at V_{POR} to Ensure Power-on Reset	—	—	1	—	—	ms



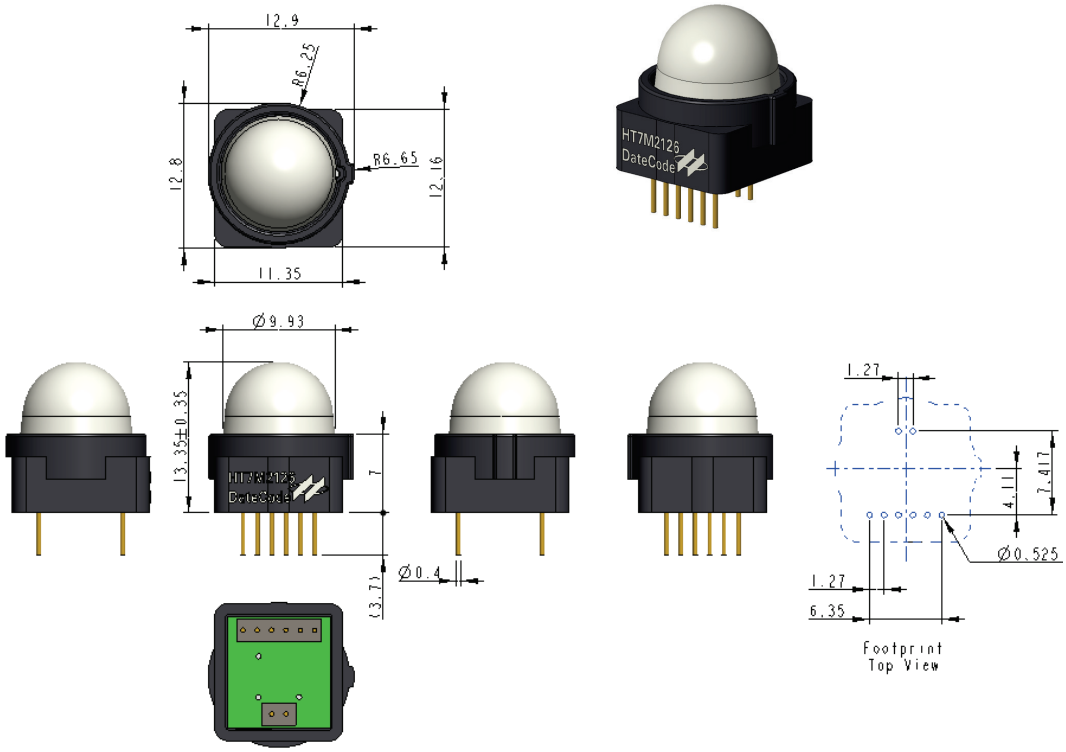
Timing Diagrams

I²C Timing

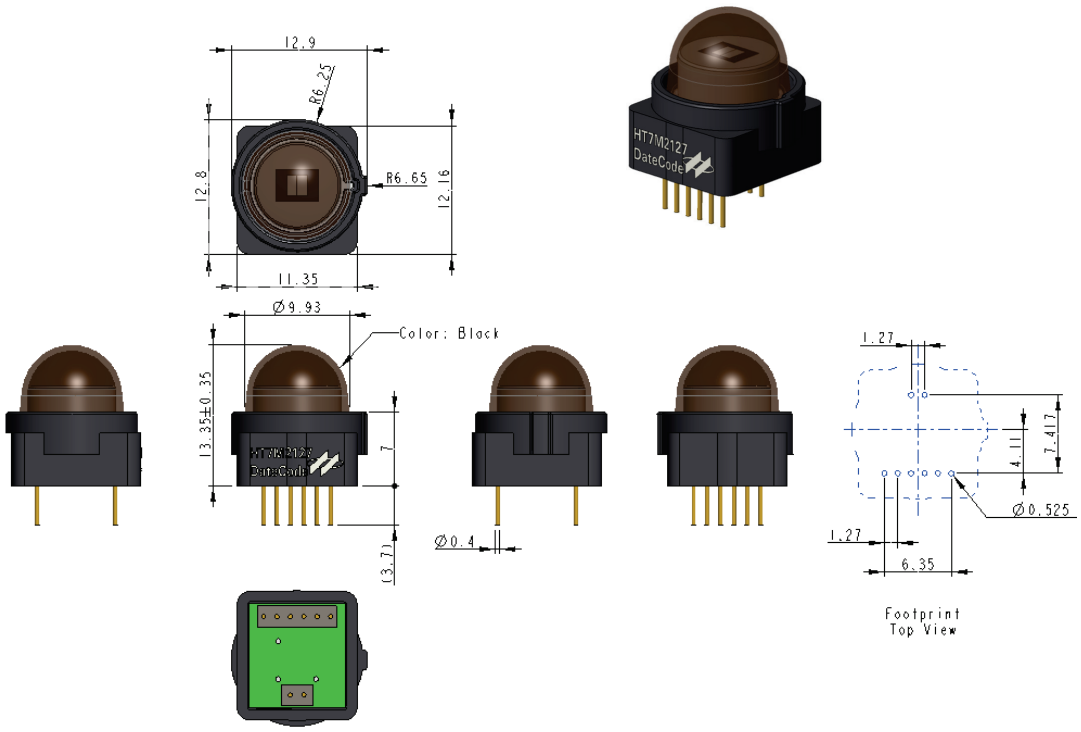


PIR Module Outline Dimensions

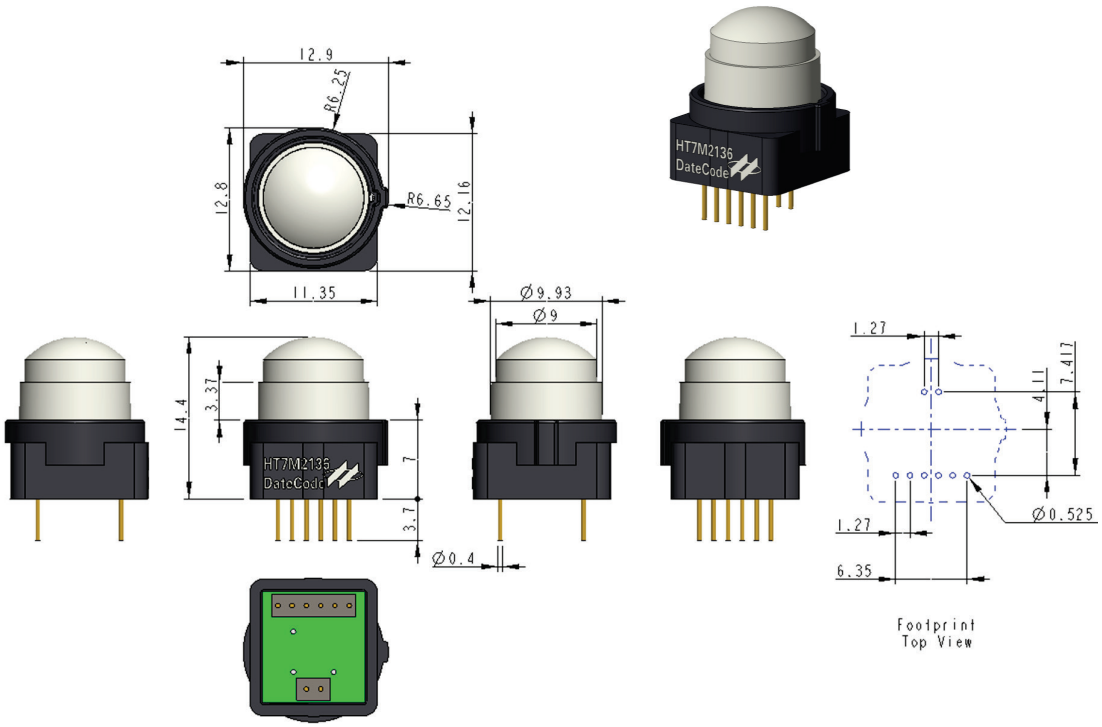
HT7M2126 – Dimensions in mm (Typical Value)



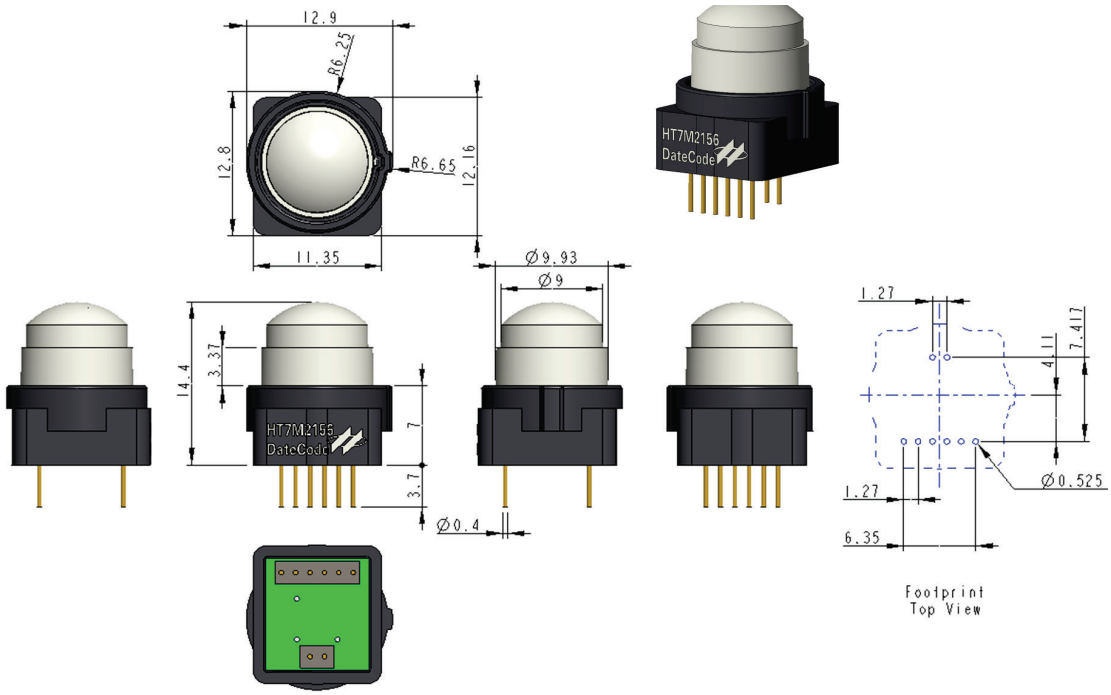
HT7M2127 – Dimensions in mm (Typical Value)



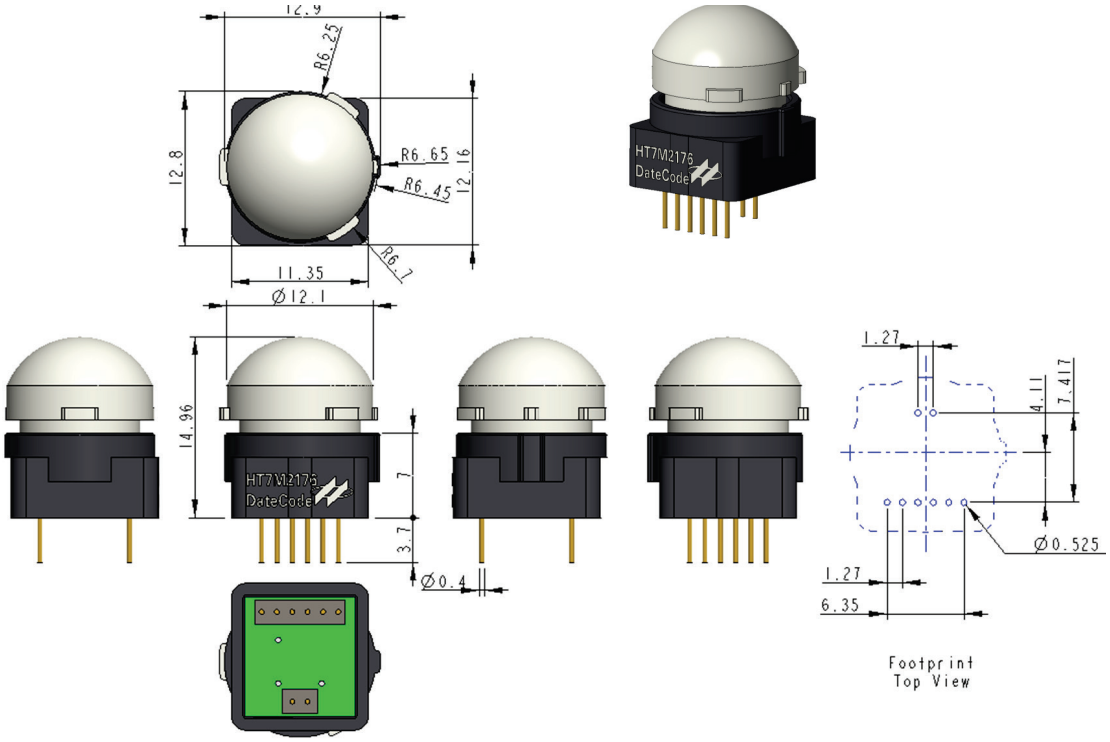
HT7M2136 – Dimensions in mm (Typical Value)



HT7M2156 – Dimensions in mm (Typical Value)



HT7M2176 – Dimensions in mm (Typical Value)



Appendix A – Communication Protocol

Protocol definitions

Module I²C Address Defined

I²C Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	R/W
	1	0	0	1	1	0	0	x

Register Pointer

HOST_CMD	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	D7	D6	D5	D4	Pointer Bit			

Bit 7~4 **D7~D4**: Writable bits, must be fixed at "0"

Bit 7~4 must always be cleared or written to "0". This device has additional registers that are reserved for test and calibration. If these registers are accessed, the device may not perform according to the specification.

Bit 3~0 Pointer Bits:

0000 = Config standby with detecting mode
0001 = Configuration register (CONFIG)
0010 = Config module address
0011 = Config Trig time interval
0100 = EEPROM access
0101 = PIR A/D RAW data
0110 = Optical sensor A/D RAW data
0111 = Temperature sensor A/D RAW data
1000 = Trig register
1001 = Manufacture ID
1010 = Device ID/Revision register
1011 = Test result inquire
1100 = Reset test result
1xxx = RFU (Note)

Note: Some registers contain calibration codes and should not be accessed. Accessing these registers could cause permanent sensor decalibration.

Bit Assignment Summary for all Registers

For User Mode									
Register Pointer (hex)	Msb/ Lsb	Bit Assigment							
		7	6	5	4	3	2	1	0
00H R/W	Msb	Temp7	Temp6	Temp5	Temp4	Temp3	Temp2	Temp1	Temp0
	Lsb	—	—	—	—	—	—	ACC1	ACC0
01H R/W	Msb	VLVD2	VLVD1	VLVD0	LVDEN	PirEN	D10	Trig mode	ACTEN
	Lsb	Threshold2	Threshold1	Threshold0	PGAC4	PGAC3	PGAC2	PGAC1	PGAC0
02H R/W	Msb	Lumi6	Lumi5	Lumi4	Lumi3	Lumi2	Lumi1	Lumi0	LUMIEN
	Lsb	Madd6	Madd5	Madd4	Madd3	Madd2	Madd1	Madd0	D0
03H R/W	Msb	Titv15	Titv14	Titv13	Titv12	Titv11	Titv10	Titv9	Titv8
	Lsb	Titv7	Titv6	Titv5	Titv4	Titv3	Titv2	Titv1	Titv0
04H R/W	Msb	Dbit7	Dbit6	Dbit5	Dbit4	Dbit3	Dbit2	Dbit1	Dbit0
	Lsb	—	EEOK	R/W	—	EEADD3	EEADD2	EEADD1	EEADD0
05H R	Msb	—	—	—	—	PirRAW11	PirRAW10	PirRAW9	PirRAW8
	Lsb	PirRAW7	PirRAW6	PirRAW5	PirRAW4	PirRAW3	PirRAW2	PirRAW1	PirRAW0
06H R	Msb	—	—	—	—	LumiRAW11	LumiRAW10	LumiRAW9	LumiRAW8
	Lsb	LumiRAW7	LumiRAW6	LumiRAW5	LumiRAW4	LumiRAW3	LumiRAW2	LumiRAW1	LumiRAW0
07H R	Msb	—	—	—	—	TsRAW11	TsRAW10	TsRAW9	TsRAW8
	Lsb	TsRAW7	TsRAW6	TsRAW5	TsRAW4	TsRAW3	TsRAW2	TsRAW1	TsRAW0
08H R	Msb	Ini	—	—	—	—	—	—	LVD
	Lsb	BLumi	—	—	—	—	Fnoise	Fagtrg	Ftrg
09H R	Msb	0	0	0	0	0	1	0	0
	Lsb	1	1	0	1	1	0	0	1
0AH R	Msb	Ver15	Ver14	Ver13	Ver12	Ver11	Ver10	Ver9	Ver8
	Lsb	Ver7	Ver6	Ver5	Ver4	Ver3	Ver2	Ver1	Ver0

"—": Unimplemented, read as "0".

0. Config standby with detecting mode

- **Config Register → ADDRESS: 00H**

Bit	15	14	13	12	11	10	9	8
Name	Temp7	Temp6	Temp5	Temp4	Temp3	Temp2	Temp1	Temp0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	ACC1	ACC0
R/W	—	—	—	—	—	—	R/W	R/W

Bit 15~8 **Temp7~ Temp0**: Current Temperature (°C)

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **ACC1~ACC0**: Define the preiod of standby with detecting mode

00: 4ms (Default)

01: 8ms

10: 16ms

11: 32ms

1. Sensor Config Register – CONFIG

- **Sensor Config Register → ADDRESS: 01H**

Bit	15	14	13	12	11	10	9	8
Name	VLVD2	VLVD1	VLVD0	LVDEN	PirEN	D10	Trig mode	ACTEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Name	Threshold2	Threshold1	Threshold0	PGAC4	PGAC3	PGAC2	PGAC1	PGAC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 15~13 **VLVD2~VLVD0**: Select LVD Voltage
000: 2.0V
001: 2.2V
010: 2.4V
011: 2.7V (Default)
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.0V

Bit 12 **LVDEN**
1: Turn on the low voltage detection function (Default)
0: Turn off the low voltage detection function

Bit 11 **PirEN**
1: Enable PIR detect (Default)
0: Disable PIR detect

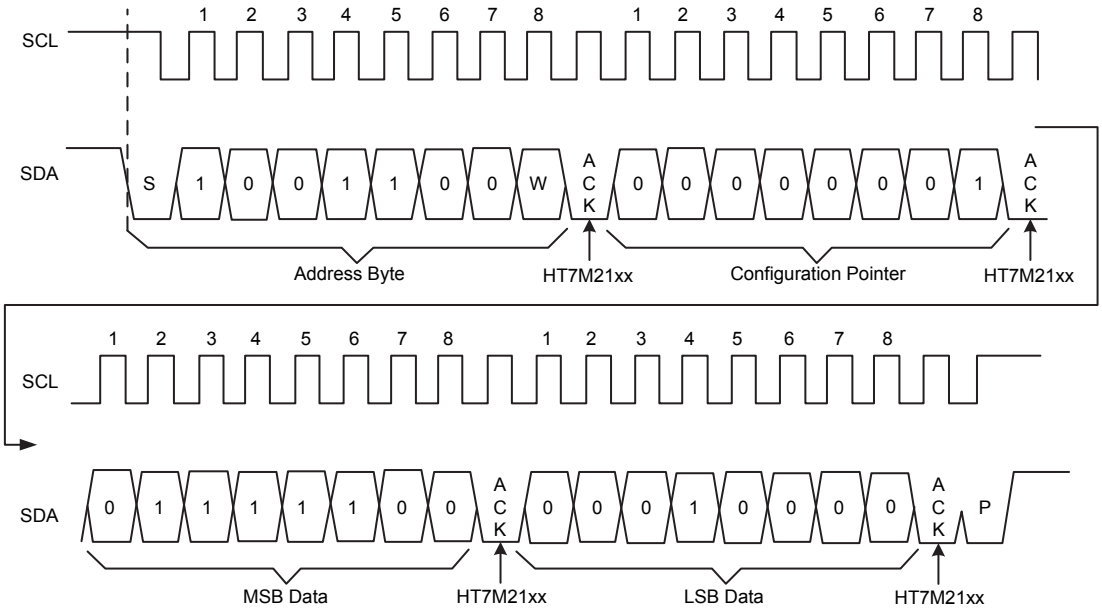
Bit 10 **D10**: Reserved bit
1: Not use, reserve for test mode
0: Not use, reserve for test mode (Default)

Bit 9 **Trig mode**
1: Continuous trigger (Default)
0: Single trigger

Bit 8 **ACTEN**
1: Enable ACT pin function (Default)
0: Disable ACT pin function

Bit 7~5 **Threshold2~Threshold0**
000: Threshold Trigger 1 (offset ± 0.2), (Default)
001: Threshold Trigger 2 (offset ± 0.3)
010: Threshold Trigger 3 (offset ± 0.4)
011: Threshold Trigger 4 (offset ± 0.5)
100: Threshold Trigger 5 (offset ± 0.6)
101: Threshold Trigger 6 (offset ± 0.7)
110: Threshold Trigger 7 (offset ± 0.8)
111: Threshold Trigger 8 (offset ± 0.9)
Note: lower sensitivity when at high threshold trigger.

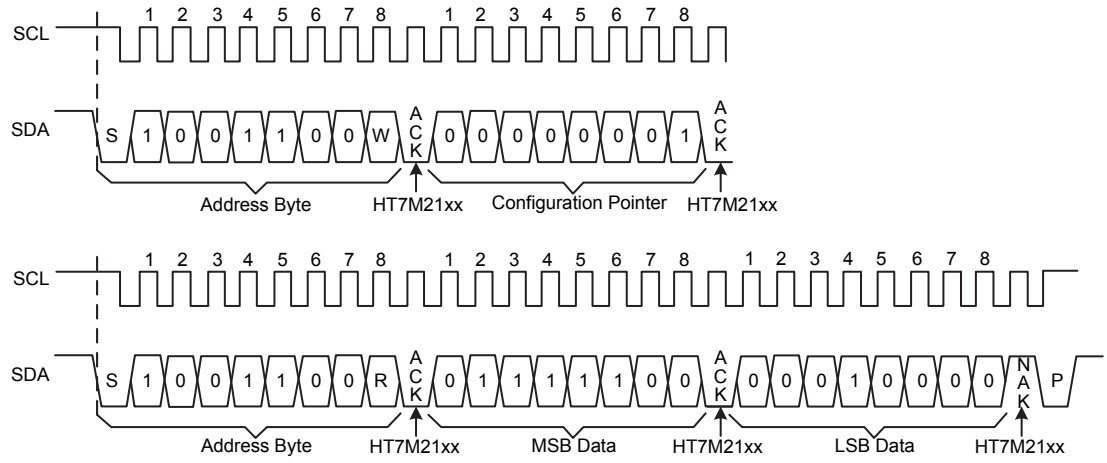
Bit 4~0 **PGAC4~PGAC0**
OPA2 Gain Control: gain = 32 + (PGAC × 2), Default gain = 64
Note: Higher sensitivity when at high magnification



Note: This is an example routine: (See Appendix B: "Source Code")

```
i2c_start();                    // send START command
i2c_write(AddressByte & 0xFE);  // WRITE Command
                                // also, make sure bit 0 is cleared '0'
i2c_write(0x01);                // Write CONFIG Register
i2c_write(0x7c);                // Write config data
i2c_write(0x10);                // Write config data
i2c_stop();                     // send STOP command
```

- **Reading the CONFIG Register**



Note: 1. It is not necessary to select the register pointer if it was set from the previous read/write.

2. This is an example routine: (See Appendix B: "Source Code")

```
i2c_start();                    // send START command
i2c_write(AddressByte & 0xFE);  // WRITE Command
                                // also, make sure bit 0 is cleared '0'
i2c_write(0x01);                // Write CONFIG Register
i2c_start();                    // send Repeat START command
i2c_write(AddressByte | 0x01);  // READ Command
```

```
                                    // also, make sure bit 0 is set '1'
UpperByte = i2c_read(ACK);          // READ 8 bits
                                    // and Send ACK bit
LowerByte = i2c_read(NAK);          // READ 8 bits
                                    // and Send NAK bit
```

2. Config Module Address – Madd

- **CONGIF MODULE ADDRESS → ADDRESS: 02H**

Bit	15	14	13	12	11	10	9	8
Name	Lumi6	Lumi5	Lumi4	Lumi3	Lumi2	Lumi1	Lumi0	LUMIEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Name	Madd6	Madd5	Madd4	Madd3	Madd2	Madd1	Madd0	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 15~9 **Lumi6~Lumi0**

These bits define the photo transistor A/D conversion thresholds for threshold triggering of brightness setting.

When the value is larger, environment needs more dark (Default 1Fh).

Bit 8 **LUMIEN**

1: Enable Brightness detection and make it associate with PIR detect
PIR detection will be started when the brightness less than Lumi setting which is the brightness value of the corresponding.

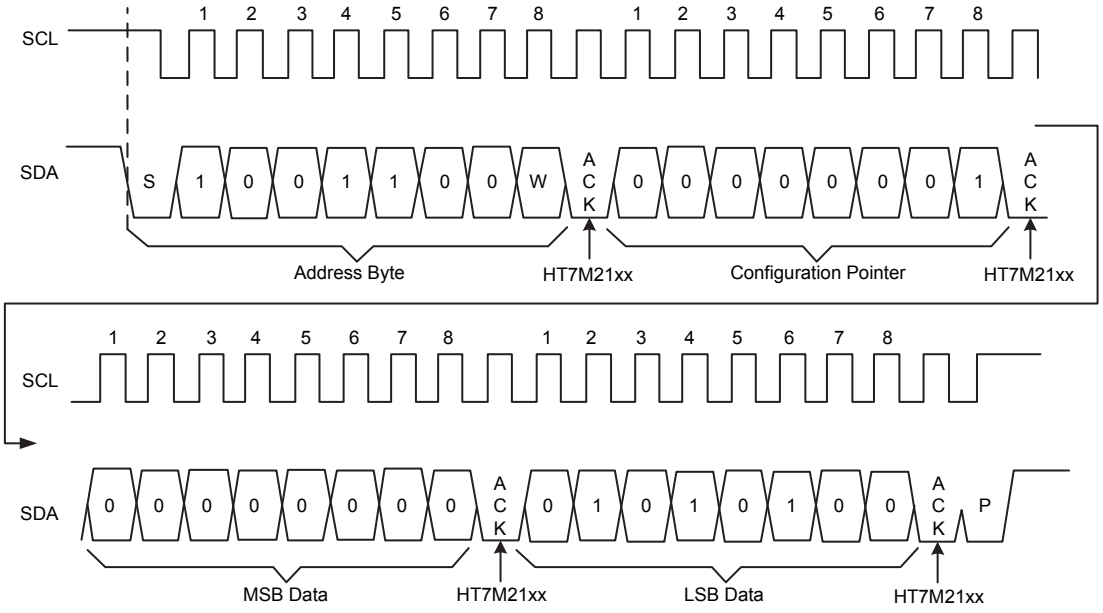
0: Disable Brightness detection and make it associate with PIR detect

Bit 7~1 **Madd6~Madd0**: Config Module I²C Address

The Address can not be changed, must be fixed as 4Ch.

Bit 0 **D0**: Reserved bit.

Note: Address is MSB 7 bits, bit 0 reserved.



Note: This is an example routine: (See Appendix B: "Source Code")

```
i2c_start();                        // send START command
i2c_write(AddressByte & 0xFE);      // WRITE Command
                                    // also, make sure bit 0 is cleared '0'
i2c_write(0x02);                    // Write MADD Register
i2c_write(0x00);                    // Write data
i2c_write(0x54);                    // Write data
i2c_stop();                         // send STOP command

i2c_start();                        // send START command
i2c_write(0x54& 0xFE);              // WRITE Command  MUST use the new address
……
```

3. Trig Time Interval

- **MODULE TRIG TIME INTERVAL → ADDRESS: 03H**

Bit	15	14	13	12	11	10	9	8
Name	Titv15	Titv14	Titv13	Titv12	Titv11	Titv10	Titv9	Titv8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Name	Titv7	Titv6	Titv5	Titv4	Titv3	Titv2	Titv1	Titv0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 15~0 **Titv15~Titv0**

Trigger flags keep time: [Titv15:Titv0]×100ms (default 10 seconds)

Note: Ftrg flag keep time that can be used for delay switching of the lighting control products. Because of the PIR signal characteristics, we recommend to set retention time above 500ms, otherwise there will be a single detection trigger repeat status when using single trigger function. While the continuity trigger that will not have this condition.

4. EEPROM ACCESS

- **MODULE EEPROM ACCESS → ADDRESS: 04H**

Bit	15	14	13	12	11	10	9	8
Name	Dbit7	Dbit6	Dbit5	Dbit4	Dbit3	Dbit2	Dbit1	Dbit0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Name	—	EEOK	R/W	—	EEADD3	EEADD2	EEADD1	EEADD0
R/W	—	R/W	R/W	—	R/W	R/W	R/W	R/W

Bit 15~8 **Dbit7~Dbit0**: EEPROM data

Bit 7 Unimplemented, read as 0

Bit 6 **EEOK**

1: Read /Write EEPROM Finish, need to clr by user

0: Read /Write command not execute yet or failed

Bit 5 **R/W**: EEPROM Read/Write Control

1: Read EEPROM

0: Write EEPROM

Bit 4 Unimplemented, read as 0

Bit 3~0 **EEADD3~EEADD0**: EEPROM Address

Note: 0000~1111 Addresses for custom defined.

Note: This is an example routine: (See Appendix B: "Source Code")

```
i2c_start();                    // send START command
i2c_write(AddressByte & 0xFE);  // WRITE Command
                                // also, make sure bit 0 is cleared '0'
i2c_write(0x04);                // Write EEPROM Register
i2c_write(0x05);                // Write data(Write mode,Write address 0x05)
i2c_write(0xAA);                // Write data(Write data 0xAA)
i2c_stop();                     // send STOP command

i2c_start();                    // send START command
i2c_write(AddressByte & 0xFE);  // WRITE Command
                                // also, make sure bit 0 is cleared '0'
i2c_write(0x04);                // Write EEPROM Register
i2c_write(0x25);                // Write data(Read mode,read address 0x05)
i2c_write(0x00);                // Write data(Write any data will not affect result)
i2c_stop();

i2c_start();                    // send START command
i2c_write(AddressByte & 0xFE);  // WRITE Command
                                // also, make sure bit 0 is cleared '0'
i2c_write(0x04);                // Write EEPROM Register
i2c_start();                    // send Repeat START command
i2c_write(AddressByte | 0x01);  // READ Command
                                // also, make sure bit 0 is set '1'
UpperByte = i2c_read(ACK);      // READ 8 bits  (UpperByte = 0x25)
                                // and Send ACK bit
LowerByte = i2c_read(NAK);      // READ 8 bits (LowerByte = 0xAA)
                                // and Send NAK bit
```

5. PIR RAW DATA

• PIR A/D CONVERSION RAW DATA → ADDRESS: 05H

Bit	15	14	13	12	11	10	9	8
Name	—	—	—	—	PirRAW11	PirRAW10	PirRAW9	PirRAW8
R/W	—	—	—	—	R	R	R	R

Bit	7	6	5	4	3	2	1	0
Name	PirRAW7	PirRAW6	PirRAW5	PirRAW4	PirRAW3	PirRAW2	PirRAW1	PirRAW0
R/W	R	R	R	R	R	R	R	R

Bit 15~12 Unimplemented read as 0

Bit 11~0 **PirRAW11~PirRAW0**: Pir Signal A/D Conversion Raw data

Note: Can be used for developed recognition algorithms or for custom define recognition function. Recommended polling time is greater than 4ms.

Note: This is an example routine: (See Appendix B: "Source Code")

```
i2c_start();                    // send START command
i2c_write(AddressByte & 0xFE);  // WRITE Command
                                // also, make sure bit 0 is cleared '0'
i2c_write(0x05);                // Write PIRRAW Register
i2c_start();                    // send Repeat START command
i2c_write(AddressByte | 0x01);  // READ Command
                                // also, make sure bit 0 is set '1'
```

```
UpperByte = i2c_read(ACK);          // READ 8 bits  (UpperByte = High 4bit)
                                    // and Send ACK bit
LowerByte = i2c_read(NAK);          // READ 8 bits (LowerByte = Low 8bit)
                                    // and Send NAK bit
```

6. Optical Sensor RAW DATA

- **Optical Sensor A/D CONVERSION RAW DATA → ADDRESS: 06H**

Bit	15	14	13	12	11	10	9	8
Name	—	—	—	—	LumiRAW11	LumiRAW10	LumiRAW9	LumiRAW8
R/W	—	—	—	—	R	R	R	R

Bit	7	6	5	4	3	2	1	0
Name	LumiRAW7	LumiRAW6	LumiRAW5	LumiRAW4	LumiRAW3	LumiRAW2	LumiRAW1	LumiRAW0
R/W	R	R	R	R	R	R	R	R

Bit 15~12 Unimplemented read as 0

Bit 11~0 **LumiRAW11~LumiRAW0**: Optical Sensor Signal A/D Conversion Raw Data

Note: To Read the environment brightness value that can be used to setting brightness threshold value.

(To read special brightness and then to write in lumi (02H Msb) registers)

Note: This is an example routine: (See Appendix B: "Source Code")

```
i2c_start();                    // send START command
i2c_write(AddressByte & 0xFE);  // WRITE Command
                                // also, make sure bit 0 is cleared '0'
i2c_write(0x06);                // Write PHORAW Register
i2c_start();                    // send Repeat START command
i2c_write(AddressByte | 0x01);  // READ Command
                                // also, make sure bit 0 is set '1'
UpperByte = i2c_read(ACK);      // READ 8 bits  (UpperByte = High 4bit)
                                // and Send ACK bit
LowerByte = i2c_read(NAK);      // READ 8 bits (LowerByte = Low 8bit)
                                // and Send NAK bit
```

7. Temperature RAW DATA

- **Temperature Sensor A/D CONVERSTION RAW DATA → ADDRESS: 07H**

Bit	15	14	13	12	11	10	9	8
Name	—	—	—	—	TsRAW11	TsRAW10	TsRAW9	TsRAW8
R/W	—	—	—	—	R	R	R	R

Bit	7	6	5	4	3	2	1	0
Name	TsRAW7	TsRAW6	TsRAW5	TsRAW4	TsRAW3	TsRAW2	TsRAW1	TsRAW0
R/W	R	R	R	R	R	R	R	R

Bit 15~12 Unimplemented read as 0

Bit 11~0 **TsRAW11~TsRAW0**: Temperature Signal A/D Conversion Raw data

- **Temperature sensor Specifications**

Symbol	Parameter	Condition	Min.	Typ.	Max.	Unit
V_{DD}	Analog Voltage	—	2.7	—	5.5	V
V_{REFO}	Bandgap output voltage	No load @ 3V	-3%	1.04	+3%	V
V_{TPS}	Temperature Sensor Voltage	—	-10%	0.91	+10%	V
T_{slope}	Temp. sensor Slope	—	—	3.12	—	mV/°C

Note: This is an example routine: (See Appendix B: "Source Code")

```
i2c_start();                    // send START command
i2c_write(AddressByte & 0xFE);  // WRITE Command
                                // also, make sure bit 0 is cleared '0'
i2c_write(0x07);                // Write TSRAW Register
i2c_start();                    // send Repeat START command
i2c_write(AddressByte | 0x01);  // READ Command
                                // also, make sure bit 0 is set '1'
UpperByte = i2c_read(ACK);      // READ 8 bits  (UpperByte = High 4bit)
                                // and Send ACK bit
LowerByte = i2c_read(NAK);      // READ 8 bits (LowerByte = Low 8bit)
                                // and Send NAK bit
```

8. Module Status

- **MODULE STATUS → ADDRESS: 08H**

Bit	15	14	13	12	11	10	9	8
Name	Ini	—	—	—	—	—	—	LVD
R/W	R	—	—	—	—	—	—	R

Bit	7	6	5	4	3	2	1	0
Name	BLumi	—	—	—	—	Fnoise	Fagtrg	Ftrg
R/W	R	—	—	—	—	R	R	R

Bit 15 **Ini**
1: module initialing/module initial failed
0: module initial ok

Bit 14~9 Unimplemented read as 0

Bit 8 **LVD**
1: Low Voltage Detect
0: No Low Voltage Detect

Bit 7 **BLumi**
1: Night, darkness detected
0: Day, brightness detected

Bit 6~3 Unimplemented read as 0

Bit 2 **Fnoise**
1: PIR noise detected
0: No PIR noise detected

Bit 1 **Fagtrg**
1: PIR trig again (notice: This bit can be trig when Trig mode = 1)
0: No PIR trig again

Bit 0 **Ftrg**

1: PIR Trigged

0: No PIR Trigged

Note: This is an example routine: (See Appendix B: "Source Code")

```
i2c_start();                    // send START command
i2c_write(AddressByte & 0xFE);  // WRITE Command
                                // also, make sure bit 0 is cleared '0'
i2c_write(0x08);                // Write PIRRAW Register
i2c_start();                    // send Repeat START command
i2c_write(AddressByte | 0x01);  // READ Command
                                // also, make sure bit 0 is set '1'
UpperByte = i2c_read(ACK);      // READ 8 bits  (UpperByte = MSB)
                                // and Send ACK bit
LowerByte = i2c_read(NAK);      // READ 8 bits (LowerByte =LSB)
                                // and Send NAK bit
```

9. Manufacture ID

- **MANUFACTURE ID (MID) → ADDRESS: 09H**

Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	1	0	0
R/W	R	R	R	R	R	R	R	R

Bit	7	6	5	4	3	2	1	0
Name	1	1	0	1	1	0	0	1
R/W	R	R	R	R	R	R	R	R

Bit 15~0 Manufacture ID = 0x04D9

Note: This is an example routine: (See Appendix B: "Source Code")

```
i2c_start();                    // send START command
i2c_write(AddressByte & 0xFE);  // WRITE Command
                                // also, make sure bit 0 is cleared '0'
i2c_write(0x09);                // Write MID Register
i2c_start();                    // send Repeat START command
i2c_write(AddressByte | 0x01);  // READ Command
                                // also, make sure bit 0 is set '1'
UpperByte = i2c_read(ACK);      // READ 8 bits  (UpperByte = 0x04)
                                // and Send ACK bit
LowerByte = i2c_read(NAK);      // READ 8 bits (LowerByte = 0xd9)
                                // and Send NAK bit
```

10. Firmware Version

- **Firmware Version → ADDRESS: 0AH**

Bit	15	14	13	12	11	10	9	8
Name	Ver15	Ver14	Ver13	Ver12	Ver11	Ver10	Ver9	Ver8
R/W	R	R	R	R	R	R	R	R

Bit	7	6	5	4	3	2	1	0
Name	Ver7	Ver6	Ver5	Ver4	Ver3	Ver2	Ver1	Ver0
R/W	R	R	R	R	R	R	R	R

Bit 15~0 **Ver15~Ver0**: Firmware Version

Note: This is an example routine: (See Appendix B: "Source Code")

```
i2c_start();                    // send START command
i2c_write(AddressByte & 0xFE);  // WRITE Command
                                // also, make sure bit 0 is cleared '0'
i2c_write(0x0a);                // Write PID Register
i2c_start();                    // send Repeat START command
i2c_write(AddressByte | 0x01);  // READ Command
                                // also, make sure bit 0 is set '1'
UpperByte = i2c_read(ACK);      // READ 8 bits  (UpperByte = 0x02)
                                // and Send ACK bit
LowerByte = i2c_read(NAK);      // READ 8 bits (LowerByte = 0x00)
                                // and Send NAK bit
```

- **EEPROM Planning**

The storage area is used to record the user-defined data

00h: user define	01h: user define	02h: user define	03h: user define
04h: user define	05h: user define	06h: user define	07h: user define
08h: user define	09h: user define	0Ah: user define	0Bh: user define
0Ch: user define	0Dh: user define	0Eh: user define	0Fh: user define

Note: EEPROM 16×8

Appendix B

```
/*************************************************************
FileName: I2C.c
Processor:HT66F60 Microcontrollers
Complier:  IDE-3000 V7.71  V2 compiler
Company: holtek semiconductor .Inc
#include <HT66F60.h> // This code is developed for HT66F
//It can be modified to be used with any HTmicro  With GPIO
/** PRIVATE PROTOTYPES*********************************/
void i2c_init(void);
void i2c_start(void);
void i2c_stop(void);
unsigned char  i2c_wait_ack ()
void i2c_ack()
void i2c_nack()
unsigned char i2c_write( unsigned char txd );
unsigned char i2c_read( unsigned char ack );
#define SDA_IN()   _pcc3=1
#define SDA_OUT()  _pcc3=0
#define SDA_PUH()  _pcpu3 = 1
#define IIC_SDA    _pc3 //SDA
#define READ_SDA   _pc3 //SDA
#define SCL_IN()   _pcc4 = 1
#define SCL_OUT()  _pcc4 = 0
#define SCL_PUH()  _pcpu4 = 1
#define IIC_SCL    _pc4 //SCL
#define READ_SCL   _pc4
/*************************************************************
* Function Name: i2c_init
* Return Value: void
* Parameters: Set IO status
* Description: This function sets up
* HT66F device for use with a IO simulate I2C
*************************************************************/
void i2c_init(void) {
          SCL_OUT();           //scl set output
          SDA_OUT();           //sda set output
          SCL_PUH();
          SDA_PUH();}
/*************************************************************************
* Function Name: i2c_start
* Return Value: void
* Parameters: void
* Description: Send I2C Start Command
*************************************************************************/
void i2c_start(void) {
          SDA_OUT();           //sda output setting
          IIC_SDA=1;
          IIC_SCL_H();         //IIC_SCL=1;
          _delay(4);
          IIC_SDA=0;           //START:when CLK is high,DATA change form high to low
          _delay(4);
          IIC_SCL_L();}
```

```
/*********************************************************************
* Function Name: i2c_stop
* Return Value: void
* Parameters: void
* Description: Send I2C Stop command
*********************************************************************/
void i2c_stop(void) {
          SDA_OUT();     //sda output
          IIC_SCL_L();   //IIC_SCL=0;
          IIC_SDA=0;     //STOP:when CLK is high DATA change form low to high
          _delay(4);
          IIC_SCL_H();
          IIC_SDA=1;
          _delay(4);
}

/*********************************************************************
* Function Name: i2c_write
* Return Value:
* Parameters: Single data byte for I2C2 bus.
* Description: This routine writes a single byte to the
* I2C2 bus.
*********************************************************************/
void i2c_write( unsigned char txd ) {
          uint8 t,buf;
          SDA_OUT();
          IIC_SCL_L();//IIC_SCL=0;
          for(t=0;t<8;t++)
          {
              buf=txd&0x80;
              IIC_SDA = buf>>7;     //IIC_SDA=(txd&0x80)>>7;
              txd<<=1;
              _delay(2);
              IIC_SCL_H();          //IIC_SCL=1;
              _delay(2);
              IIC_SCL_L();          //IIC_SCL=0;
              _delay(2);
          }
}
/*********************************************************************
* Function Name: i2c_read
* Return Value: read iic data
* Parameters: ack = 1 and nak = 0
* Description: Read a byte from I2C bus and ACK/NAK device
*********************************************************************/
unsigned char i2c_read( unsigned char ack ) {
          unsigned char i,receive=0;
          SDA_IN();
          for(i=0;i<8;i++ )
          {
              IIC_SCL_L();      //IIC_SCL=0;
              _delay(2);
              IIC_SCL_H();      //IIC_SCL=1;
              receive<<=1;
              if(READ_SDA)receive++;
              _delay(2);
          }
```

```
        if (ack)
        i2c_ack();        //send ACK
        else
        i2c_nack();       //send ACK
        return receive;
}
/*********************************************************************
* Function Name: i2c_wait_ack
* Return Value: 1:get ack success ,0:get ack failed
* Parameters:
* Description: wait ack from i2c bus
*********************************************************************/
unsigned char  i2c_wait_ack ()
{
  uint8 ucErrTime=0;
  SDA_IN();
  IIC_SDA=1;_delay(1);
  IIC_SCL_H();//IIC_SCL=1;
  _delay(1);
  while(READ_SDA)
  {
    ucErrTime++;
    if(ucErrTime>50)
    {
      IIC_Stop();
      return 1;
    }
  _delay(1);
  }
  IIC_SCL_L();    //IIC_SCL=0;
  return 0;
}
/*********************************************************************
* Function Name: i2c_ack
* Return Value:
* Parameters:
* Description: generate ack to i2c bus
*********************************************************************/
void i2c_ack()
{
  IIC_SCL_L();    //IIC_SCL=0;
  SDA_OUT();
  IIC_SDA=0;
  _delay(2);
  IIC_SCL_H();    //IIC_SCL=1;
  _delay(2);
  IIC_SCL_L();    //IIC_SCL=0;
}
 /*********************************************************************
* Function Name: i2c_ nack
* Return Value:
* Parameters:
* Description: generate nack to i2c bus
*********************************************************************/
```

```
void i2c_nack()
{
  IIC_SCL_L();    //IIC_SCL=0;
  SDA_OUT();
  IIC_SDA=1;
  _delay(2);
  IIC_SCL_H();    //IIC_SCL=1;
  _delay(2);
  IIC_SCL_L();    //IIC_SCL=0;
}
```