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

General Description

Holtek’s human body infrared detector micro modules, the HT7M2xxx 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.

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

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

Note: When the HT7M21xx selects Network mode and the internal enable bit ACTEN is high, if an effective trigger signal is detected the MODE/ACT pin will output a high pulse signal. The signal pulse width is determined by the Titv15~Titv0 bits at address of 03H with a default value of 10 seconds.

Stand-alone Application Circuit – Stand-alone Mode
# Interface & Pin Assignment – Stand-alone Mode

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

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** \( C_T = 0.22\mu F \), \( V_{DD} = 3.3V \)

<table>
<thead>
<tr>
<th>RA+RT Resistance (Ω)</th>
<th>1.8K</th>
<th>2.2K</th>
<th>2.7K</th>
<th>3K</th>
<th>3.3K</th>
<th>3.6K</th>
<th>3.9K</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRO output duration time *</td>
<td>3sec.</td>
<td>10sec.</td>
<td>38sec.</td>
<td>1mins.</td>
<td>3mins.</td>
<td>5mins.</td>
<td>10mins.</td>
</tr>
</tbody>
</table>

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) Standby 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 HT7M2xxx device using the I²C interface.

Detection Range

The main differences for the HT7M2126/2127/2136/2156/2176, hereinafter referred to as the HT7M21xx series, are the field of view, FOV, and detection range. The HT7M21xx series visual field is shown in the following figure. For example, the HT7M2126 has a horizontal view of 121°, and a pitch angle of 77°, here the farthest detection range will be approximately 3.5 to 6 meters. This device series also contains the HT7M2127, HT7M2136, HT7M2156, HT7M2176 modules except for the HT7M2126 device, which have different detection ranges and horizontal angles as well as pitch angles.

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Viewing Angle H/V</th>
<th>Farthest Center Distance</th>
<th>Lens Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>HT7M2126</td>
<td>121° / 77°</td>
<td>3.5 ~ 6.0 meters</td>
<td>Nature</td>
</tr>
<tr>
<td>HT7M2127</td>
<td>121° / 77°</td>
<td>2.8 ~ 5.0 meters</td>
<td>Black</td>
</tr>
<tr>
<td>HT7M2136</td>
<td>91° / 10°</td>
<td>5.5 ~ 8.0 meters</td>
<td>Nature</td>
</tr>
<tr>
<td>HT7M2156</td>
<td>20° / 10°</td>
<td>8 ~ 12 meters</td>
<td>Nature</td>
</tr>
<tr>
<td>HT7M2176</td>
<td>86° / 75°</td>
<td>5.0 ~ 7.5 meters</td>
<td>Nature</td>
</tr>
</tbody>
</table>

Note: 1. The measured background temperature is 25°C.
2. Taking the human body as a detection object, moving transverse horizontally with a speed of 1m/s~1.5m/s right in front of the module.
3. Setup the Threshold[2:0] and PGAC[4:0] registers to their default values.
Absolute Maximum Ratings

Supply Voltage: $V_{SS} - 0.3V$ to $V_{SS} + 6.0V$
Input Voltage: $V_{DD}$
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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Test Conditions</th>
<th>$V_{DD}$ Conditions</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{DD}$</td>
<td>Operating Voltage</td>
<td>—</td>
<td>—</td>
<td>2.7</td>
<td>3.3</td>
<td>5.5</td>
<td>V</td>
</tr>
<tr>
<td>$I_{DD1}$</td>
<td>Operating Current</td>
<td>3.3V</td>
<td>Moving objects to be detected</td>
<td>—</td>
<td>1.2</td>
<td>2.0</td>
<td>mA</td>
</tr>
<tr>
<td>$I_{DD2}$</td>
<td>Operating Current</td>
<td>3.3V</td>
<td>Standby with detection mode, the ACC wake-up time is 4ms.</td>
<td>—</td>
<td>30</td>
<td>50</td>
<td>µA</td>
</tr>
<tr>
<td>$T_{PIR}$</td>
<td>PIR Stabilization Time</td>
<td>3.3V</td>
<td>—</td>
<td>—</td>
<td>12</td>
<td>—</td>
<td>s</td>
</tr>
<tr>
<td>$V_{IL}$</td>
<td>Input Low Voltage (I/O)</td>
<td>—</td>
<td>—</td>
<td>0</td>
<td>—</td>
<td>0.2$V_{DD}$</td>
<td>V</td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>Input High Voltage (I/O)</td>
<td>—</td>
<td>—</td>
<td>0.8$V_{DD}$</td>
<td>—</td>
<td>$V_{DD}$</td>
<td>V</td>
</tr>
<tr>
<td>$I_{OL}$</td>
<td>I/O Port Sink Current</td>
<td>3.3V</td>
<td>$V_{OL}=0.1V_{DD}$</td>
<td>6</td>
<td>12</td>
<td>—</td>
<td>mA</td>
</tr>
<tr>
<td>$I_{OL}$</td>
<td>I/O Port Sink Current</td>
<td>5V</td>
<td>$V_{OL}=0.0V_{DD}$</td>
<td>10</td>
<td>25</td>
<td>—</td>
<td>mA</td>
</tr>
<tr>
<td>$I_{OH}$</td>
<td>I/O Port Source Current</td>
<td>3.3V</td>
<td>$V_{OH}=0.9V_{DD}$</td>
<td>-2</td>
<td>-4</td>
<td>—</td>
<td>mA</td>
</tr>
<tr>
<td>$I_{OH}$</td>
<td>I/O Port Source Current</td>
<td>5V</td>
<td>$V_{OH}=0.0V_{DD}$</td>
<td>-5</td>
<td>-8</td>
<td>—</td>
<td>mA</td>
</tr>
<tr>
<td>$R_{PH}$</td>
<td>Pull-high Resistance (FTS)</td>
<td>—</td>
<td>—</td>
<td>-5%</td>
<td>680</td>
<td>+5%</td>
<td>kΩ</td>
</tr>
</tbody>
</table>

$LVDEN = 1, V_{UD} = 2.0V$
$LVDEN = 1, V_{UD} = 2.2V$
$LVDEN = 1, V_{UD} = 2.4V$
$LVDEN = 1, V_{UD} = 2.7V$
$LVDEN = 1, V_{UD} = 3.0V$
$LVDEN = 1, V_{UD} = 3.3V$
$LVDEN = 1, V_{UD} = 3.6V$
$LVDEN = 1, V_{UD} = 4.0V$

$V_{DD}$

$V_{SS}$
### A.C. Characteristics – I²C Interface

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Test Conditions</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>VDD</strong> Conditions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Conditions</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Ta=25°C</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f_SCL</td>
<td>Clock Frequency</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>400 kHz</td>
</tr>
<tr>
<td>t_BUF</td>
<td>Bus Free Time</td>
<td>—</td>
<td>—</td>
<td>1.3</td>
<td>—</td>
<td>μs</td>
</tr>
<tr>
<td>t_HLD</td>
<td>Start Condition Hold Time</td>
<td>—</td>
<td>0.6</td>
<td>—</td>
<td>—</td>
<td>μs</td>
</tr>
<tr>
<td>t_Low</td>
<td>SCL Low Time</td>
<td>—</td>
<td>1.3</td>
<td>—</td>
<td>—</td>
<td>μs</td>
</tr>
<tr>
<td>t_HIGH</td>
<td>SCL High Time</td>
<td>—</td>
<td>0.6</td>
<td>—</td>
<td>—</td>
<td>μs</td>
</tr>
<tr>
<td>t_SU.DAT</td>
<td>Start Condition Setup Time</td>
<td>—</td>
<td>0.6</td>
<td>—</td>
<td>—</td>
<td>μs</td>
</tr>
<tr>
<td>t_HLD.DAT</td>
<td>Data Hold Time</td>
<td>—</td>
<td>0</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>t_SDA &amp; SCL Rise Time</td>
<td>— Note</td>
<td>—</td>
<td>100</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>t_SDA &amp; SCL Fall Time</td>
<td>— Note</td>
<td>—</td>
<td>0.3</td>
<td>—</td>
<td>—</td>
<td>μs</td>
</tr>
<tr>
<td>t_SU.STO</td>
<td>Stop Condition Setup time</td>
<td>—</td>
<td>0.6</td>
<td>—</td>
<td>—</td>
<td>μs</td>
</tr>
<tr>
<td>t_A</td>
<td>Output Valid from Clock</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μs</td>
</tr>
<tr>
<td>t_SP</td>
<td>Input Filter Time Constant (SCL)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>50 ns</td>
</tr>
</tbody>
</table>

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

### Temperature Sensor Characteristics

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Test Conditions</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>VDD</strong> Conditions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Conditions</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Ta=25°C</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V_DD</td>
<td>Analog Voltage</td>
<td>—</td>
<td>2.7</td>
<td>—</td>
<td>5.5</td>
<td>V</td>
</tr>
<tr>
<td>V_REFO</td>
<td>Bandgap Output Voltage</td>
<td>3V No Load</td>
<td>-3%</td>
<td>1.04</td>
<td>+3%</td>
<td>V</td>
</tr>
<tr>
<td>V_TPS</td>
<td>Temperature Sensor Voltage</td>
<td>Bypass pre-buffer</td>
<td>-10%</td>
<td>0.91</td>
<td>+10%</td>
<td>V</td>
</tr>
<tr>
<td>T_slope</td>
<td>Temperature Sensor Slope</td>
<td>Bypass pre-buffer</td>
<td>—</td>
<td>3.12</td>
<td>—</td>
<td>mV/°C</td>
</tr>
</tbody>
</table>

### Power-on Reset Characteristics

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Test Conditions</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>VDD</strong> Conditions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Conditions</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Ta=25°C</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V_POR</td>
<td>V_DD Start Voltage to Ensure Power-on Reset</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>100 mV</td>
</tr>
<tr>
<td>R_RDV</td>
<td>V_DD Rising Rate to Ensure Power-on Reset</td>
<td>—</td>
<td>0.035</td>
<td>—</td>
<td>—</td>
<td>V/ms</td>
</tr>
<tr>
<td>t_POR</td>
<td>Minimum Time for V_DD Stays at V_POR to Ensure Power-on Reset</td>
<td>—</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>ms</td>
</tr>
</tbody>
</table>
Timing Diagrams

I²C Timing

PIR Module Outline Dimensions

HT7M2126 – Dimensions in mm (Typical Value, tolerance: ±0.2mm)
HT7M2127 – Dimensions in mm (Typical Value, tolerance: ±0.2mm)

HT7M2136 – Dimensions in mm (Typical Value, tolerance: ±0.2mm)
HT7M2156 – Dimensions in mm (Typical Value, tolerance: ±0.2mm)

HT7M2176 – Dimensions in mm (Typical Value, tolerance: ±0.2mm)
Appendix A – Communication Protocol

Protocol Definitions

Module I2C Address Defined

<table>
<thead>
<tr>
<th>I2C Address</th>
<th>Bit7</th>
<th>Bit6</th>
<th>Bit5</th>
<th>Bit4</th>
<th>Bit3</th>
<th>Bit2</th>
<th>Bit1</th>
<th>Bit0</th>
<th>R/W</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IICA6</td>
<td>IICA5</td>
<td>IICA4</td>
<td>IICA3</td>
<td>IICA2</td>
<td>IICA1</td>
<td>IICA0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>

Register Pointer

<table>
<thead>
<tr>
<th>HOST_CMD</th>
<th>Bit7</th>
<th>Bit6</th>
<th>Bit5</th>
<th>Bit4</th>
<th>Bit3</th>
<th>Bit2</th>
<th>Bit1</th>
<th>Bit0</th>
<th>Pointer Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7~D4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Pointer Bit</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Register Pointer (hex)</th>
<th>Msb/ Lsb</th>
<th>Bit Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>00H R/W</td>
<td>Temp7</td>
<td>Temp6</td>
</tr>
<tr>
<td></td>
<td>Lab</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>01H R/W</td>
<td>VLVD2</td>
<td>VLVD1</td>
</tr>
<tr>
<td></td>
<td>Lab</td>
<td>Threshold2</td>
</tr>
<tr>
<td>02H R/W</td>
<td>Lumil8</td>
<td>Luml5</td>
</tr>
<tr>
<td></td>
<td>Lab</td>
<td>Madd6</td>
</tr>
<tr>
<td>03H R/W</td>
<td>Titv15</td>
<td>Titv14</td>
</tr>
<tr>
<td></td>
<td>Lab</td>
<td>Titv7</td>
</tr>
<tr>
<td>04H R/W</td>
<td>Dbit7</td>
<td>Dbit6</td>
</tr>
<tr>
<td></td>
<td>Lab</td>
<td>EEOK</td>
</tr>
<tr>
<td>05H R</td>
<td>PirRAW7</td>
<td>PirRAW6</td>
</tr>
<tr>
<td></td>
<td>Lab</td>
<td>PirRAW7</td>
</tr>
<tr>
<td>06H R</td>
<td>LumiRAW7</td>
<td>LumiRAW6</td>
</tr>
<tr>
<td></td>
<td>Lab</td>
<td>LumiRAW7</td>
</tr>
<tr>
<td>07H R</td>
<td>TsRAW7</td>
<td>TsRAW6</td>
</tr>
<tr>
<td></td>
<td>Lab</td>
<td>LumiRAW7</td>
</tr>
<tr>
<td>08H R</td>
<td>Blumi</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>Lab</td>
<td>Ini</td>
</tr>
<tr>
<td>09H R</td>
<td>Ver15</td>
<td>Ver14</td>
</tr>
<tr>
<td></td>
<td>Lab</td>
<td>Ver7</td>
</tr>
</tbody>
</table>

*—*: Unimplemented, read as “0”.

### 0. Config standby with detecting mode

- **Config Register → ADDRESS: 00H**

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Temp7</td>
<td>Temp6</td>
<td>Temp5</td>
<td>Temp4</td>
<td>Temp3</td>
<td>Temp2</td>
<td>Temp1</td>
<td>Temp0</td>
</tr>
<tr>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
</tbody>
</table>

- **Bit 15~8** Temp7~Temp0: Current Temperature (°C)
- **Bit 7~2** Unimplemented, read as “0”
- **Bit 1~0** ACC1~ACC0: Define the period of standby with detecting mode
  - 00: 4ms (Default)
  - 01: 8ms
  - 10: 16ms
  - 11: 32ms

The module will detect the human body movement with a specific frequency corresponding to the selected period in the standby with detecting mode.
1. Sensor Config Register – CONFIG

- Sensor Config Register → ADDRESS: 01H

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>VLVD2</td>
<td>VLVD1</td>
<td>VLVD0</td>
<td>LVDEN</td>
<td>PirEN</td>
<td>D10</td>
<td>Trig mode</td>
<td>ACTEN</td>
</tr>
<tr>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Threshold2</td>
<td>Threshold1</td>
<td>Threshold0</td>
<td>PGAC4</td>
<td>PGAC3</td>
<td>PGAC2</td>
<td>PGAC1</td>
<td>PGAC0</td>
</tr>
<tr>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
</tbody>
</table>

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

When the ACTEN bit is set high and an available trigger signal is detected, a high pulse signal will be output on the MODE/ACT pin. If the ACTEN bit is set low as the MODE/ACT pin output high, the MODE/ACT pin status will remain high. After the ACTEN bit is set high again with another available trigger signal being detected and the high pulse signal is output on the MODE/ACT pin, then the MODE/ACT pin status will go low.
### Bit 7~5  
**Threshold2~Threshold0**

- 000: Threshold Trigger 1 (offset ± 0.2V), (Default)
- 001: Threshold Trigger 2 (offset ± 0.3V)
- 010: Threshold Trigger 3 (offset ± 0.4V)
- 011: Threshold Trigger 4 (offset ± 0.5V)
- 100: Threshold Trigger 5 (offset ± 0.6V)
- 101: Threshold Trigger 6 (offset ± 0.7V)
- 110: Threshold Trigger 7 (offset ± 0.8V)
- 111: Threshold Trigger 8 (offset ± 0.9V)

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”)

```c
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(0x01);  // Write config data
i2c_stop();       // send STOP command
```
• Reading the CONFIG Register

```
SCL   S   1  0  0  1  1  0  0  A  C  0  0  0  0  0  1  K 000000011
SDA   S   1  0  0  1  1  0  0  A  C  0  1  1  0  0  R  A 011111001
```

Address Byte HT7M2xxx  Configuration Pointer HT7M2xxx

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

• CONFIG MODULE ADDRESS → ADDRESS: 02H

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>R/W</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lumi6</td>
<td>R/W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Lumi5</td>
</tr>
<tr>
<td></td>
<td>Lumi5</td>
<td>R/W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Lumi4</td>
</tr>
<tr>
<td></td>
<td>Lumi4</td>
<td>R/W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Lumi3</td>
</tr>
<tr>
<td></td>
<td>Lumi3</td>
<td>R/W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Lumi2</td>
</tr>
<tr>
<td></td>
<td>Lumi2</td>
<td>R/W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Lumi1</td>
</tr>
<tr>
<td></td>
<td>Lumi1</td>
<td>R/W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Lumi0</td>
</tr>
<tr>
<td></td>
<td>Lumi0</td>
<td>R/W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LUMIEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R/W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>R/W</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Madd6</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>D0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Madd5</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Madd4</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Madd3</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Madd2</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Madd1</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Madd0</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>R/W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 (Default)

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 FC 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.

---

**3. Trig Time Interval**

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Titv15</td>
<td>Titv14</td>
<td>Titv13</td>
<td>Titv12</td>
<td>Titv11</td>
<td>Titv10</td>
<td>Titv9</td>
<td>Titv8</td>
</tr>
<tr>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
</tbody>
</table>

---

**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**

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Dbit7</td>
<td>Dbit6</td>
<td>Dbit5</td>
<td>Dbit4</td>
<td>Dbit3</td>
<td>Dbit2</td>
<td>Dbit1</td>
<td>Dbit0</td>
</tr>
<tr>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>—</td>
<td>EEOK</td>
<td>R/W</td>
<td>—</td>
<td>EEADD3</td>
<td>EEADD2</td>
<td>EEADD1</td>
<td>EEADD0</td>
</tr>
<tr>
<td>R/W</td>
<td>—</td>
<td>R/W</td>
<td>R/W</td>
<td>—</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
</tbody>
</table>

**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

For some reasons, a write operation to the EEPROM address 00H cannot be executed after the module is powered on. Users can first execute a read operation to the EEPROM address 00H then a write operation can be executed to the address 00H. Users can also write data to one of the EEPROM address from 01H to 0FH then write data to the EEPROM address 0000H.

Note: 00H~0FH Addresses for custom defined.

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

```c
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(0xAA);     // Write data(Write data 0xAA)
i2c_write(0x05);     // Write data(Write mode,Write address 0x05)
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(0x00);     // Write data(Write any data will not affect result)
i2c_write(0x25);     // Write data(Read mode,read address 0x05)
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_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

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>PirRAW11</td>
<td>PirRAW10</td>
<td>PirRAW9</td>
<td>PirRAW8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R/W</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>PirRAW11</td>
<td>PirRAW10</td>
<td>PirRAW9</td>
<td>PirRAW8</td>
</tr>
</tbody>
</table>

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”)

```c
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

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>LumiRAW11</td>
<td>LumiRAW10</td>
<td>LumiRAW9</td>
<td>LumiRAW8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R/W</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>LumiRAW11</td>
<td>LumiRAW10</td>
<td>LumiRAW9</td>
</tr>
</tbody>
</table>

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”)

```c
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
```
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 CONVERSION RAW DATA → ADDRESS: 07H

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TsRAW11</td>
<td>TsRAW10</td>
<td>TsRAW9</td>
<td>TsRAW8</td>
</tr>
<tr>
<td>R/W</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

Bit 15~12 Unimplemented read as 0

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

- Temperature sensor Specifications

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Condition</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD</td>
<td>Analog Voltage</td>
<td>—</td>
<td>2.7</td>
<td>—</td>
<td>5.5</td>
<td>V</td>
</tr>
<tr>
<td>VREFO</td>
<td>Bandgap output voltage</td>
<td>No load @ 3V</td>
<td>-3%</td>
<td>1.04</td>
<td>+3%</td>
<td>V</td>
</tr>
<tr>
<td>VTPS</td>
<td>Temperature Sensor Voltage</td>
<td>—</td>
<td>-10%</td>
<td>0.91</td>
<td>+10%</td>
<td>V</td>
</tr>
<tr>
<td>Tslope</td>
<td>Temp. sensor Slope</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>mV/°C</td>
</tr>
</tbody>
</table>

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

```c
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**

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Ini</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>LVD</td>
</tr>
<tr>
<td>R/W</td>
<td>R</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>R</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>BLumi</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Fnoise</td>
<td>Fagtrg</td>
</tr>
<tr>
<td>R/W</td>
<td>R</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

Bit 15 **Ini**
1: module initializing/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”)

```c
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**

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R/W</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>R/W</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

Bit 15~0    Manufacture ID = 0x04D9

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

```c
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 = 0x0D)
    // and Send NAK bit
```

10. Firmware Version

- **Firmware Version → ADDRESS: 0AH**

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Ver15</td>
<td>Ver14</td>
<td>Ver13</td>
<td>Ver12</td>
<td>Ver11</td>
<td>Ver10</td>
<td>Ver9</td>
<td>Ver8</td>
</tr>
<tr>
<td>R/W</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Ver7</td>
<td>Ver6</td>
<td>Ver5</td>
<td>Ver4</td>
<td>Ver3</td>
<td>Ver2</td>
<td>Ver1</td>
<td>Ver0</td>
</tr>
<tr>
<td>R/W</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

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

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

```c
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

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00h:</td>
<td>user define</td>
</tr>
<tr>
<td>01h:</td>
<td>user define</td>
</tr>
<tr>
<td>02h:</td>
<td>user define</td>
</tr>
<tr>
<td>03h:</td>
<td>user define</td>
</tr>
<tr>
<td>04h:</td>
<td>user define</td>
</tr>
<tr>
<td>05h:</td>
<td>user define</td>
</tr>
<tr>
<td>06h:</td>
<td>user define</td>
</tr>
<tr>
<td>07h:</td>
<td>user define</td>
</tr>
<tr>
<td>08h:</td>
<td>user define</td>
</tr>
<tr>
<td>09h:</td>
<td>user define</td>
</tr>
<tr>
<td>0Ah:</td>
<td>user define</td>
</tr>
<tr>
<td>0Bh:</td>
<td>user define</td>
</tr>
<tr>
<td>0Ch:</td>
<td>user define</td>
</tr>
<tr>
<td>0Dh:</td>
<td>user define</td>
</tr>
<tr>
<td>0Eh:</td>
<td>user define</td>
</tr>
<tr>
<td>0Fh:</td>
<td>user define</td>
</tr>
</tbody>
</table>

Note: EEPROM 16×8

Appendix B

/***************************************************************************/
/* File Name: I2C.c */
/* Processor: HT66F60 Microcontrollers */
/* Compiler: 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 SCL_IN() _pc4 = 1
#define SCL_OUT() _pc4 = 0
#define SCL_PUH() _pcpu4 = 1
#define IIC_SCL _pc4 //SCL
#define READ_SCL _pc4
#define IIC_SDA _pc3 //SDA
#define READ_SDA _pc3 //SDA
/**********************************
* 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();
}
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 from 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 from 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
********************************************************************/

```c
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;
}
```