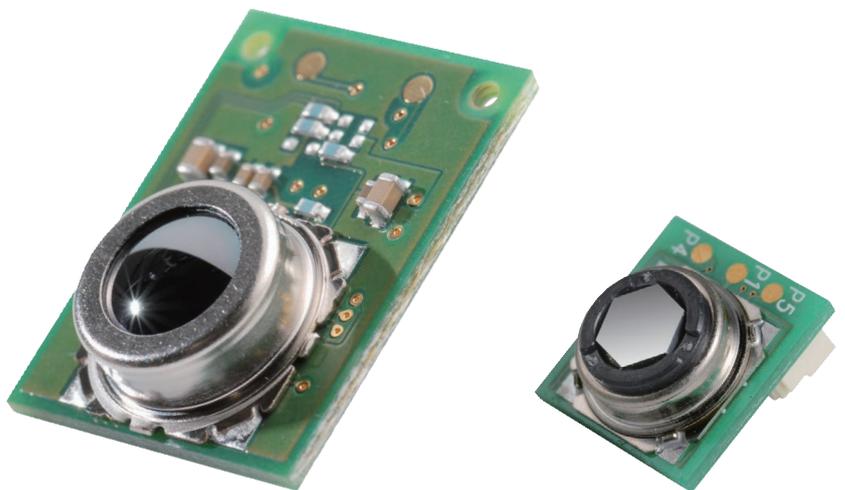


MEMS非接触温度传感器 D6T

用户手册

MEMS非接触温度传感器



目录

1. 概要	2
2. 工作原理	2
3. 产品特点	3
4. 使用方法	4
4.1 接插件	4
4.2 电气连接示例	4
4.3 表面外壳部件	7
4.4 固定方法	8
5. 开发、评估工具	9
6. 常见疑问	16
7. 用语说明	17
8. 承诺事项	18

D6T 通信规格	20
D6T-1A-01, D6T-1A-02	21
D6T-8L-09	25
D6T-8L-09H	30
D6T-44L-06, D6T-44L-06H	35
D6T-32L-01A	41

1. 概要

本用户手册介绍了 MEMS 非接触温度传感器 D6T 系列的使用方法和特别记载事项等。此外，本手册是产品样本的补充资料，在实际使用时，请参照产品样本一起使用。

2. 工作原理

MEMS 非接触温度传感器的测量动作概要如下所示。

- 通过硅透镜，将物体发出的辐射热（远红外线）聚集到模组内的热电堆型传感器上。（*1）
- 通过聚集起来的辐射热（远红外线），热电堆传感器会产生电动势。
- 基于测量的电动势值和内部温度传感器值，用内置 lookup table 进行插补运算，从而算出测量值（对象物温度）。（*2）
- 测量值通过 I²C 总线，由上位系统读取使用。

*1. D6T-1A-01/02 使用硅过滤器。

*2. D6T-1A-01/02、D6T-8L-09/09H 通过 ASIC 内的温度换算电路算出测量值（对象物温度）。

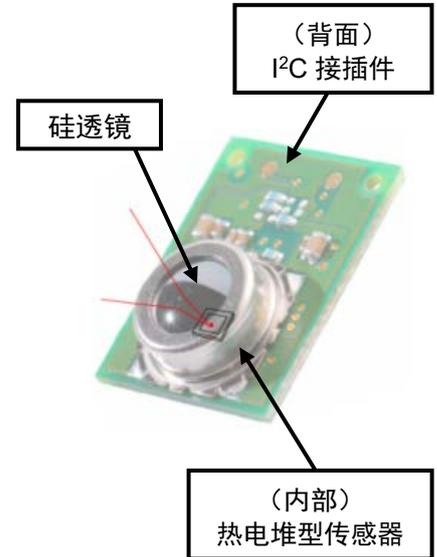
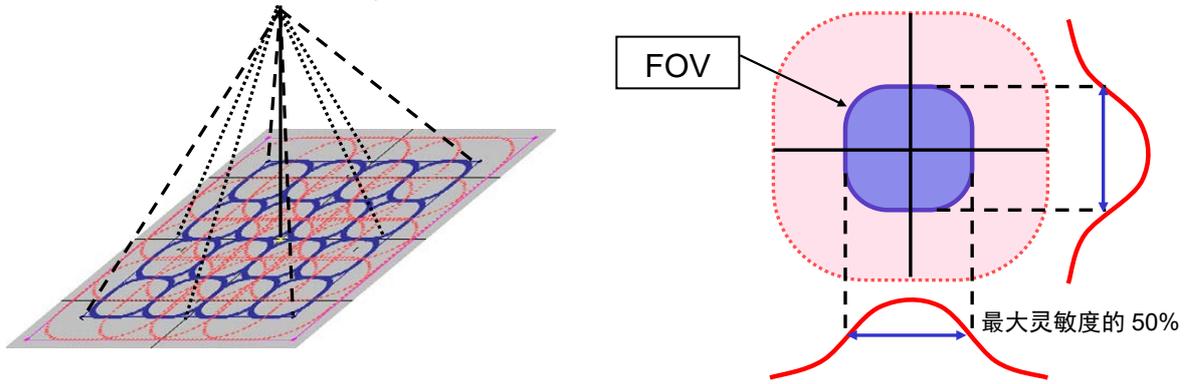


图 1 模组结构

3. 产品特点

MEMS 非接触温度传感器采用光学设计的硅透镜，具有设定好的光灵敏度特性。我公司非接触温度传感器和一般传感器相同，将最大灵敏度 50% 的区域角度记载为 FOV（Field of View）。



a) D6T-44L-06/06H 的 FOV (16ch) 示意图

b) 1 元件的 FOV 和 XY 轴的灵敏度特性示意图

图 2 视角 (FOV) 的灵敏度特性形状示意图

灵敏度区域是比 FOV 规格宽度更广的区域。另外，测量对象物的尺寸比灵敏度区域还小的情况下，有可能会受到对象物之外背景温度的影响。

本公司的非接触温度传感器采用基准热源（黑体炉）补偿温度测量值，不过，还需要多留心测量对象物的不同材质所产生的不同放射率、表面形状和灵敏度感测区域内的占有率等对测量值产生影响的因素。

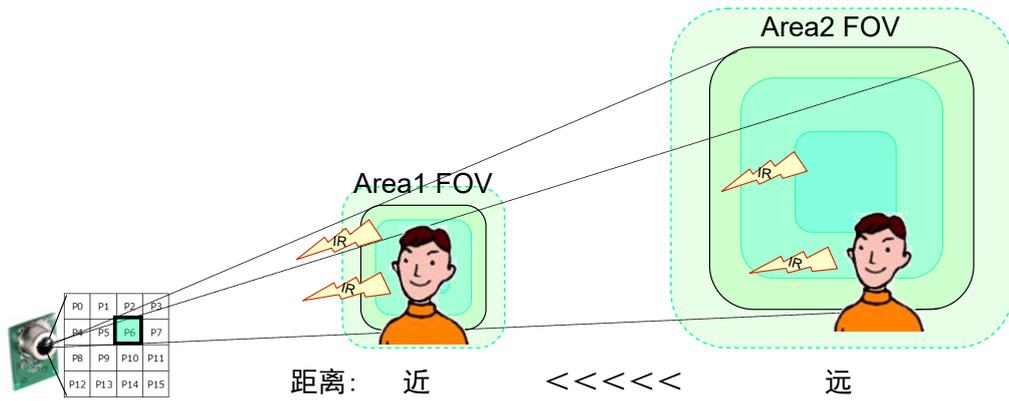


图 3 距离引起测量值变化的主要因素

距离越远，观测的面积（FOV）越广。FOV 里对象物（人物）的面积占有率随距离而变，距离越远面积占有率越小。因此，距离越远，对测量值占比（影响度）而言，背景温度对测量值所产生的影响比对象物（人物）温度所产生的影响还大。换言之，只有对象物体面积比 FOV 面积足够大，对温度才能进行正确测量。

将 MEMS 非接触温度传感器作为人体感应传感器使用时，只单纯测量温度值的情况下，仅用于近距离测量。要检测距离较远的对象物，则需要基于时间变化、热源位置、人物行为信息等因素，通过软件处理提升判断准确度。

4. 使用方法

4.1 接插件

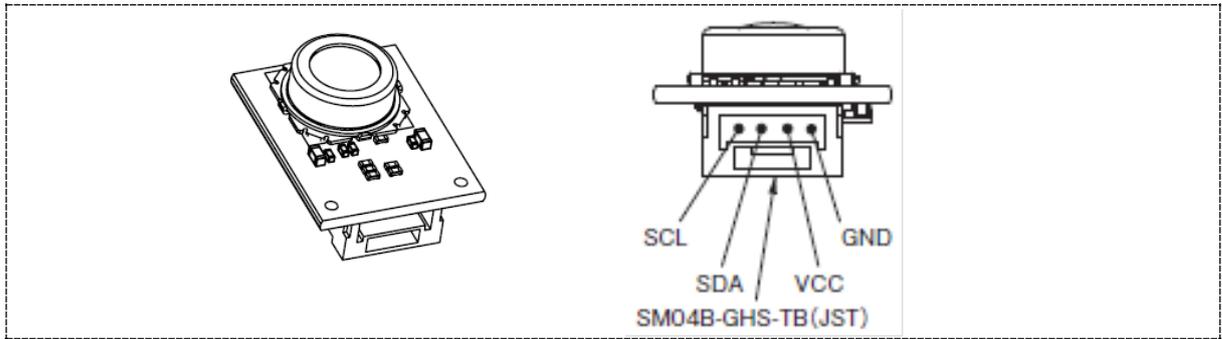


图4 产品外观（参考）

连接端子

表 1. 连接端子表

1	GND	GND 电源端子
2	VCC	VCC 电源端子 (5V ±10%)
3	SDA	I ² C (5V) 数据
4	SCL	I ² C (5V) 时钟

接插件部件

使用接插件部件型号：SM04B-GHS-TB (JST)

触头：SSHL-002T-P0.2 (JST)

外壳：GHR-04V-S (JST)

型号不同，其镜头高度和基板尺寸也不同。关于详细尺寸，请通过产品样本资料进行确认。与系统连接时，使用以上 4 个端子接插件。

4.2 电气连接示例

情况 1： 5V MCU 直接连接（微控制器电源电压相同的情况下）

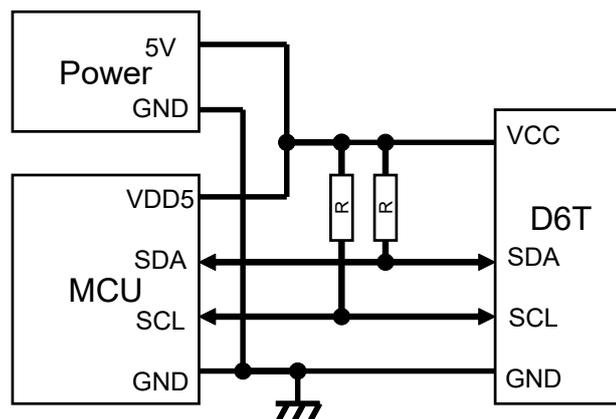


图5 和 5V 微控制器直接连接

情况 2: 3V MCU (I²C port 为 5V 容许规格的情况下)

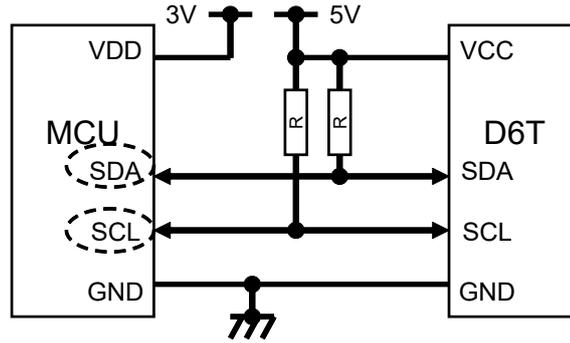


图 6 5V 容许规格的情况

情况 3: 使用 I²C 电平转换器
(在非 5V 容许规格、总线上存在 I²C (3V) 规格的其他设备的情况下)

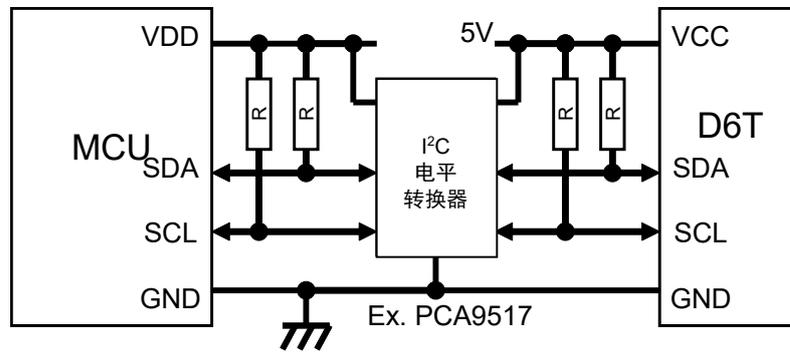


图 7 使用电平转换器时

情况 4: 使用双向开漏的 GPIO 端子, 进行软式 I²C 通信处理
(MCU 内部没有 I²C 功能时)

*1. D6T-44L-06/D6T-44L-06H/D6T-32L-01A 需要时钟拉伸应对。请参考后述的 D6T 通信规格的时钟拉伸章节。

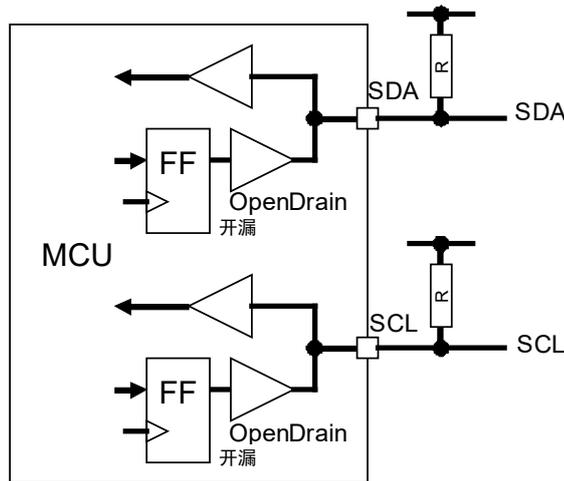


图 8 使用 GPIO 端子

情况 5: 使用 I²C 总线切换 IC (连接多个 D6T 传感器)
 (因该传感器无法更改从站地址)

*1. 许多总线切换 IC 中包含电源电压转换功能。

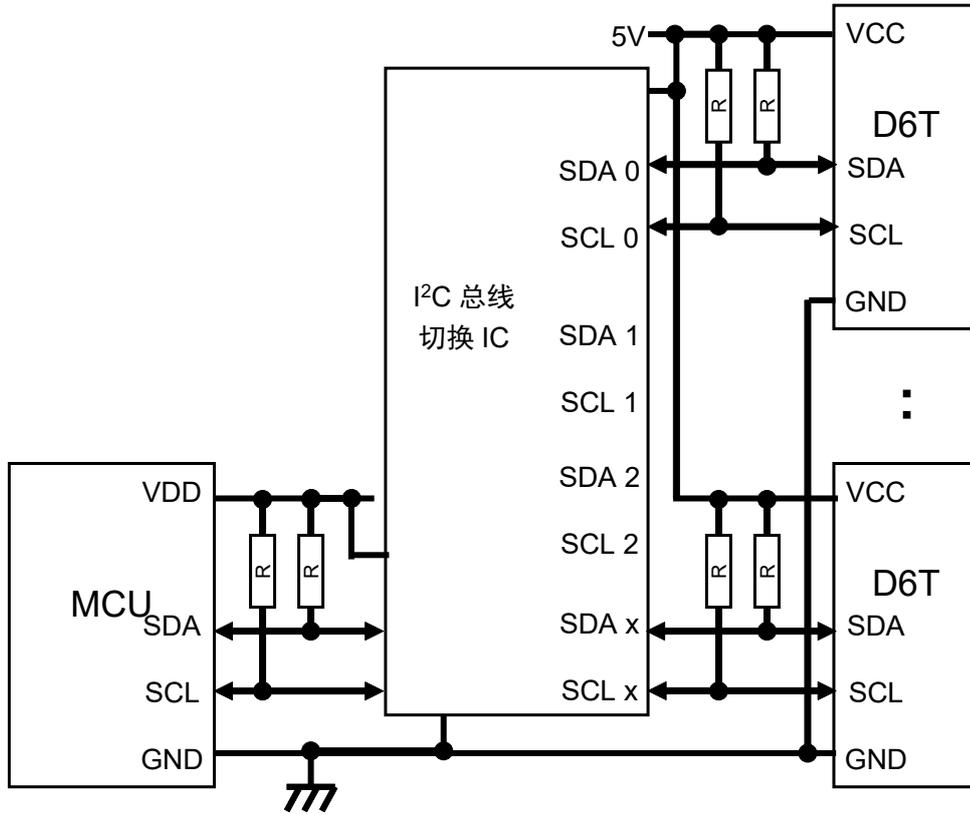


图 9 使用 I²C 总线切换 IC 时

关于上拉电阻值

上拉电阻的适当值根据配线的容量成分等用户使用条件而异，请评估后确定。
 (请确认 I²C 规格。多数情况下，大约在 3k~10kΩ 之间)

4.3 表面外壳部件

嵌入式设置 MEMS 非接触温度传感器时, 请充分留意外壳部件具辐射热(远红外线)穿透性能。远红外线可穿透级别的高密度聚乙烯(HDPE)具备相对廉价、易加工等特点, 经常被人们用作外壳材料。不同厚度的外壳, 其衰减率也各不相同, 越薄对检测性能造成的影响越小。不过, 如以下照片所示, 使用薄外壳时, 会透过其看到内部的传感器。

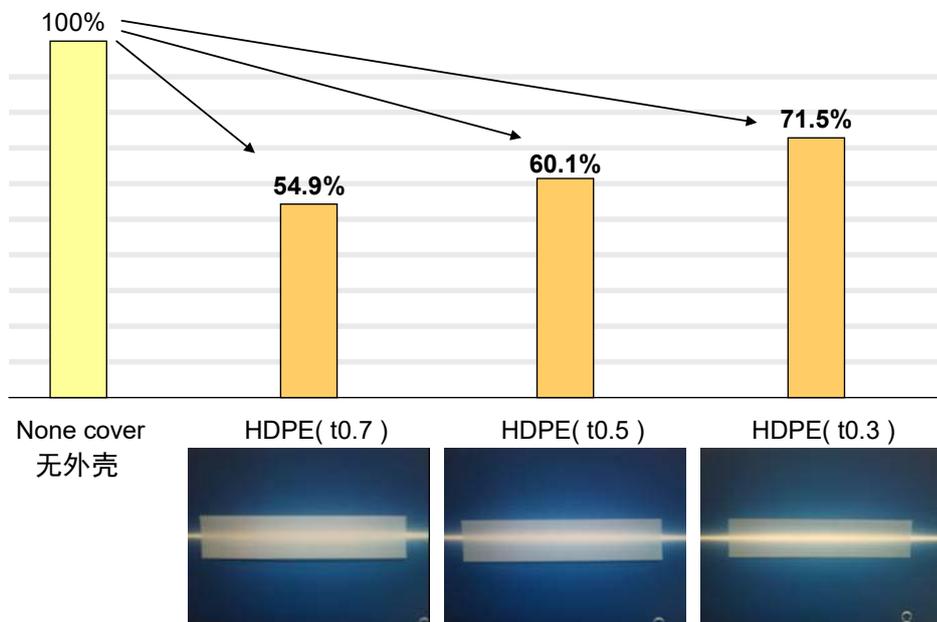


图 10 HDPE 板厚度和通透度的关系 (参考)

4.4 固定方法

通过使用外盒夹住等方式，将 MEMS 非接触温度传感器固定在可保持和固定的区域（斜线部）。

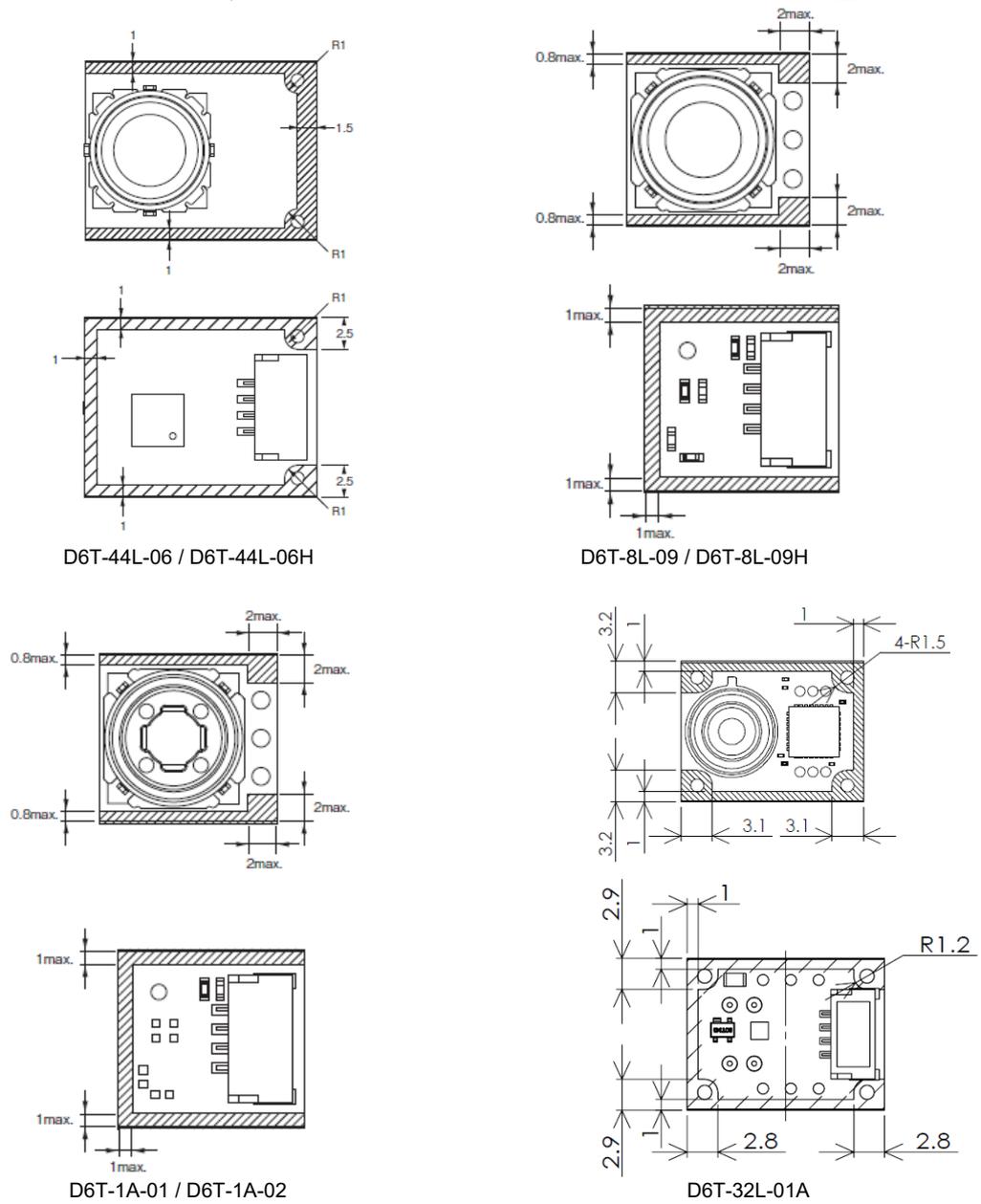


图 11 保持、固定区域（斜线部）

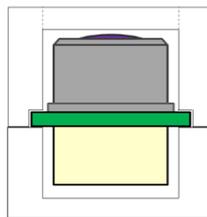


图 12 保持、固定方法（参考）

5. 开发、评估工具

备有 2 种示例代码，作为软件开发支援工具。

注 1. 本目录中的产品 2JCIE-EV01-RP1、2JCIE-EV01-AR1、2JCIE-EV01-FT1、2JCIE-HARNESS-01 将在 2025 年 1 月底停止接收订单。

1. Raspberry Pi 用 示例代码
2. Arduino 用 示例代码

Raspberry Pi 用示例代码和 Arduino 用示例代码可以与欧姆龙评估板一起使用，从而实现简单的硬件连接。

欧姆龙评估板支持以下 3 种平台，将非接触温度传感器 D6T、评估板、线束连接至平台，即可轻松评估。

评估板 URL: (<http://www.omron.co.jp/ecb/sensor/evaluation-board/2jcie>)

表 2 评估板一览表

平台	评估板	连接用线束 (评估板与 D6T 之间)	示例代码
Raspberry Pi *1 连接用	2JCIE-EV01-RP1 	2JCIE-HARNESS-01	https://github.com/omron-devhub/d6t-2jcieev01-raspberrypi
Arduino *2 连接用	2JCIE-EV01-AR1 	2JCIE-HARNESS-01	https://github.com/omron-devhub/d6t-2jcieev01-arduino
ESP32 Feather *3 连接用	2JCIE-EV01-FT1 	2JCIE-HARNESS-01	https://github.com/omron-devhub/d6t-2jcieev01-arduino

*1. Raspberry Pi 是 Raspberry Pi 财团的注册商标。

*2. Arduino 是 Arduino LLC 和 Arduino SRL 的注册商标。

*3. Feather 是 Adafruit Industries LLC 的注册商标。

即使不使用评估板，也可利用示例代码。但需由用户对传感器进行接线。

示例代码使用 C 语言记载。根据客户需使用的 MCU，可以通过变更 I²C 控制的函数，利用示例代码进行软件开发。

此外，示例代码用于评估，欧姆龙不保证正常运行。

1. Raspberry Pi 用示例代码运行顺序

(1) 将 D6T、线缆、欧姆龙评估板(2JCIE-EV01-RP1)与 Raspberry Pi 连接

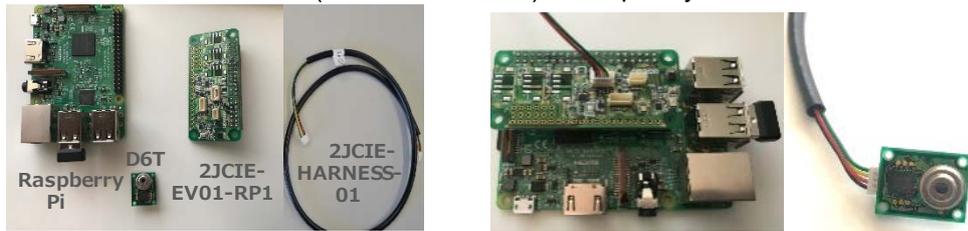


图 13 Set-up

(2) 启用 I²C

启动 Raspberry Pi, 从 Start 菜单中打开“Preferences”>“Raspberry Pi Configuration”, 将 I²C 设定为“Enable”后重启。

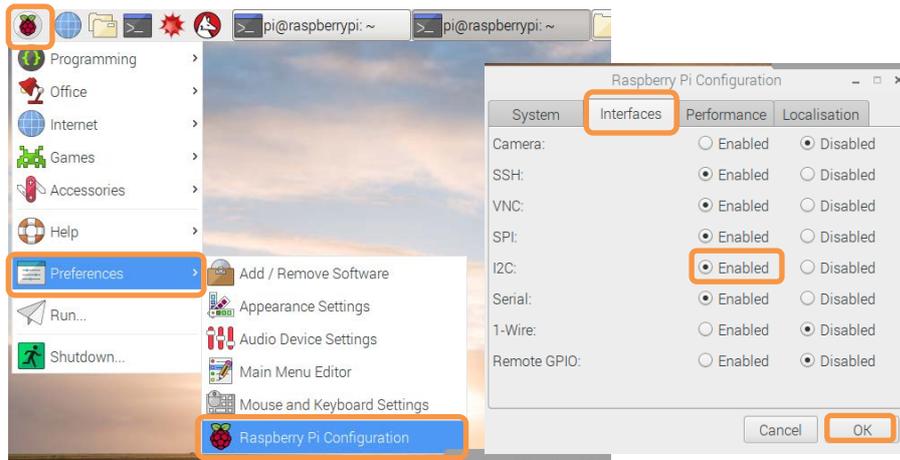


图 14 Enable I²C

(3) 下载示例代码

从以下 URL 访问 GitHub, 下载 Zip file。

GitHub URL: <https://github.com/omron-devhub/d6t-2jcieev01-raspberrypi>

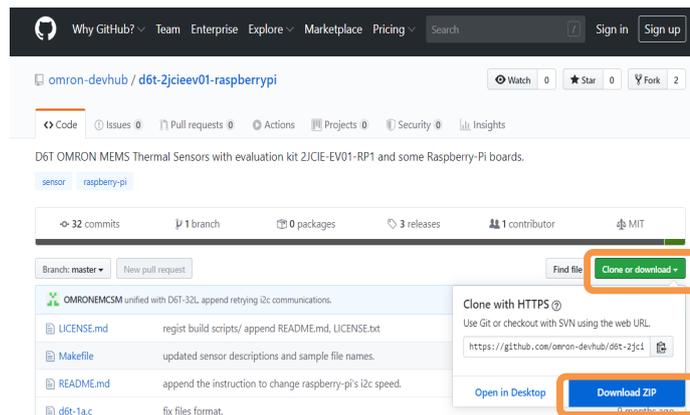


图 15 Download sample code

或者, 打开 Terminal, 执行以下指令。

```
~$ git clone https://github.com/omron-devhub/d6t-2jcieev01-raspberrypi
```

或者,通过连接英特网的其他 PC 下载 Zip file 后,用 USB 存储器等将 Zip file 转移到 Raspberry Pi 中。

(4) Make file

打开 Terminal, 执行以下指令。

```
pi@raspberrypi:~ $ cd Downloads/
pi@raspberrypi:~/Downloads $ unzip d6t-2jcieev01-raspberrypi-master.zip
pi@raspberrypi:~/Downloads $ cd d6t-2jcieev01-raspberrypi-master/
pi@raspberrypi:~/Downloads/d6t-2jcieev01-raspberrypi-master $ make all
pi@raspberrypi:~ $ cd Downloads/
pi@raspberrypi:~/Downloads $ unzip d6t-2jcieev01-raspberrypi-master.zip
Archive: d6t-2jcieev01-raspberrypi-master.zip
12e24ee9aae096ae48fbb0533ed889083b562f
  creating: d6t-2jcieev01-raspberrypi-master/
  inflating: d6t-2jcieev01-raspberrypi-master/LICENSE.md
  inflating: d6t-2jcieev01-raspberrypi-master/Makefile
  inflating: d6t-2jcieev01-raspberrypi-master/README.md
  inflating: d6t-2jcieev01-raspberrypi-master/d6t-1a.c
  inflating: d6t-2jcieev01-raspberrypi-master/d6t-32l.c
  inflating: d6t-2jcieev01-raspberrypi-master/d6t-44l.c
  inflating: d6t-2jcieev01-raspberrypi-master/d6t-8l.c
  inflating: d6t-2jcieev01-raspberrypi-master/d6t-8lh.c
pi@raspberrypi:~/Downloads $ cd d6t-2jcieev01-raspberrypi-master/
pi@raspberrypi:~/Downloads/d6t-2jcieev01-raspberrypi-master $ make all
make: Warning: File 'Makefile' has modification time 7757577 s in the future
lint with cpplint, option: --filter=readability/casting,-build/include_subdir d6t-1a.c
lint with cppcheck, option: --enable=all d6t-1a.c
gcc d6t-1a.c -o d6t-1a
lint with cpplint, option: --filter=readability/casting,-build/include_subdir d6t-8l.c
lint with cppcheck, option: --enable=all d6t-8l.c
gcc d6t-8l.c -o d6t-8l
lint with cpplint, option: --filter=readability/casting,-build/include_subdir d6t-8lh.c
lint with cppcheck, option: --enable=all d6t-8lh.c
gcc d6t-8lh.c -o d6t-8lh
lint with cpplint, option: --filter=readability/casting,-build/include_subdir d6t-44l.c
lint with cppcheck, option: --enable=all d6t-44l.c
gcc d6t-44l.c -o d6t-44l
lint with cpplint, option: --filter=readability/casting,-build/include_subdir d6t-32l.c
lint with cppcheck, option: --enable=all d6t-32l.c
gcc d6t-32l.c -o d6t-32l
make: warning: Clock skew detected. Your build may be incomplete.
pi@raspberrypi:~/Downloads/d6t-2jcieev01-raspberrypi-master $
```

图 16 Make file

(5) Run the file

为了获取数据, 执行以下指令。

```
-- D6T-1A-01 / D6T-1A-02 时
pi@raspberrypi:~/Downloads/d6t-2jcieev01-raspberrypi-master $ ./d6t-1a
-- D6T-8L-09 时
pi@raspberrypi:~/Downloads/d6t-2jcieev01-raspberrypi-master $ ./d6t-8l
-- D6T-8L-09H 时
pi@raspberrypi:~/Downloads/d6t-2jcieev01-raspberrypi-master $ ./d6t-8lh
-- D6T-44L-06 / D6T-44L-06H 时
pi@raspberrypi:~/Downloads/d6t-2jcieev01-raspberrypi-master $ ./d6t-44l
-- D6T-32L-01A 时
pi@raspberrypi:~/Downloads/d6t-2jcieev01-raspberrypi-master $ ./d6t-32l
(需停止获取数据时, 请同时按下“Ctrl”键和“C”键。)
```

```
pi@raspberrypi:~/Downloads/d6t-2jcieev01-raspberrypi-master $ ./d6t-8l
PTAT: 24.4 [degC], Temperature: 23.5, 24.0, 24.3, 23.2, 23.0, 23.1, 23.5, 24.6, [degC]
PTAT: 24.4 [degC], Temperature: 23.5, 24.2, 24.3, 23.3, 23.0, 23.1, 23.4, 24.5, [degC]
PTAT: 24.4 [degC], Temperature: 23.5, 24.2, 24.4, 23.4, 23.0, 23.1, 23.4, 24.4, [degC]
PTAT: 24.4 [degC], Temperature: 23.5, 24.2, 24.4, 23.3, 23.0, 23.1, 23.4, 24.6, [degC]
PTAT: 24.4 [degC], Temperature: 23.6, 24.2, 24.4, 23.3, 23.0, 23.2, 23.5, 24.8, [degC]
PTAT: 24.4 [degC], Temperature: 23.6, 24.2, 24.4, 23.3, 23.0, 23.1, 23.5, 25.0, [degC]
PTAT: 24.4 [degC], Temperature: 23.7, 24.2, 24.3, 23.3, 23.0, 23.2, 23.5, 25.0, [degC]
PTAT: 24.4 [degC], Temperature: 23.6, 24.1, 24.3, 23.2, 23.0, 23.1, 23.5, 25.0, [degC]
PTAT: 24.4 [degC], Temperature: 23.5, 24.1, 24.3, 23.2, 23.0, 23.0, 23.4, 24.9, [degC]
PTAT: 24.5 [degC], Temperature: 23.4, 24.0, 24.2, 23.2, 22.9, 23.0, 23.4, 24.9, [degC]
PTAT: 24.4 [degC], Temperature: 23.4, 24.0, 24.2, 23.2, 23.0, 23.0, 23.4, 24.9, [degC]
PTAT: 24.4 [degC], Temperature: 23.4, 24.0, 24.2, 23.2, 23.0, 23.1, 23.5, 24.9, [degC]
PTAT: 24.4 [degC], Temperature: 23.4, 24.0, 24.3, 23.2, 23.0, 23.1, 23.4, 24.9, [degC]
PTAT: 24.5 [degC], Temperature: 23.4, 24.0, 24.3, 23.2, 23.0, 23.0, 23.4, 24.9, [degC]
PTAT: 24.5 [degC], Temperature: 23.5, 24.1, 24.4, 23.3, 23.0, 23.2, 23.5, 24.9, [degC]
PTAT: 24.5 [degC], Temperature: 23.5, 24.1, 24.4, 23.4, 23.1, 23.2, 23.5, 25.0, [degC]
PTAT: 24.5 [degC], Temperature: 23.5, 24.1, 24.4, 23.4, 23.0, 23.1, 23.5, 25.1, [degC]
PTAT: 24.5 [degC], Temperature: 23.5, 24.1, 24.4, 23.3, 23.0, 23.1, 23.6, 25.4, [degC]
PTAT: 24.5 [degC], Temperature: 23.5, 24.1, 24.3, 23.2, 23.0, 23.1, 23.6, 25.8, [degC]
```

图 17 Run the file

2. Arduino 用示例代码运行顺序

(1) 将 D6T、线束、欧姆龙评估板(2JCIE-EV01-AR1)与 Arduino 连接

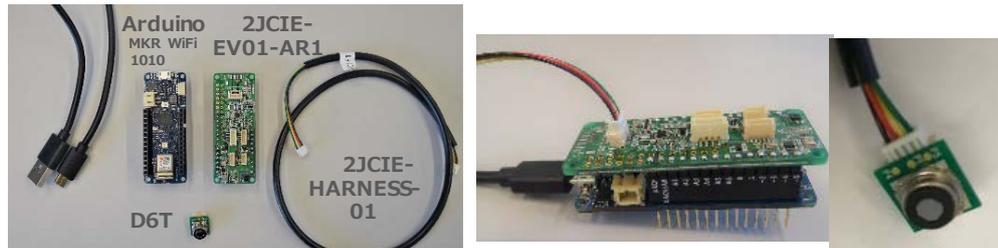


图 18 Set up

(2) 下载 Arduino IDE

从以下 URL 下载 Arduino IDE。

<https://www.arduino.cc/en/Main/Software>

(3) 在 Arduino IDE 上识别 Arduino

打开 Arduino IDE，经由 USB 将 Arduino 连接至 PC。

显示以下内容时，安装 Arduino MKR 用的 Package。



图 19 Arduino IDE

安装 Driver。

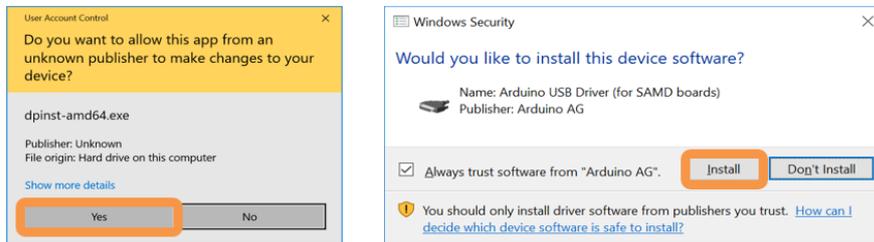


图 20 Driver

在 Windows 的开始菜单中检索“Device Manager”。

确认 PC 识别的 Arduino 的 COM 端口编号。

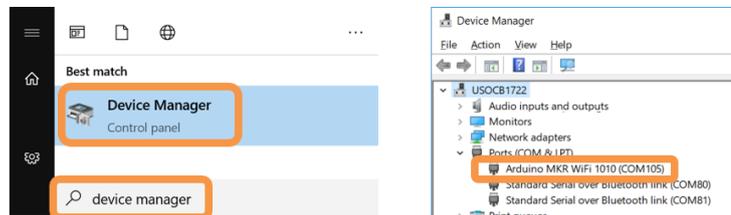


图 21 Device manager

“Tools” -> “Board Arduino” -> “Arduino MKR WiFi101”.

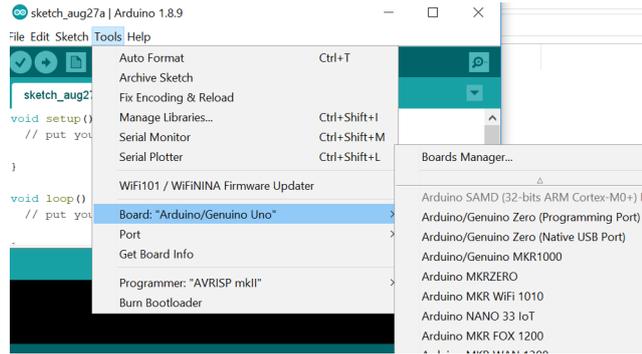


图 22 Select Arduino

选择“Tools” -> “PORT” -> Arduino 的 COM 端口。

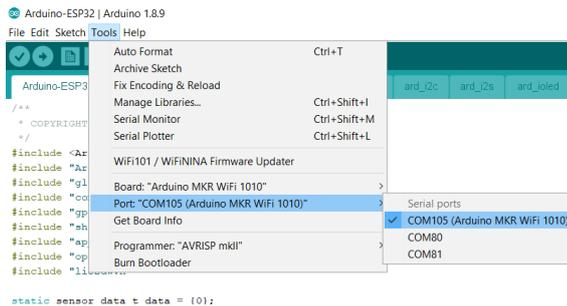


图 23 Select COM port

(4) 下载示例代码

连接至以下 GitHub 的 URL，下载 zip file。

GitHub URL: <https://github.com/omron-devhub/d6t-2jcieev01-arduino>

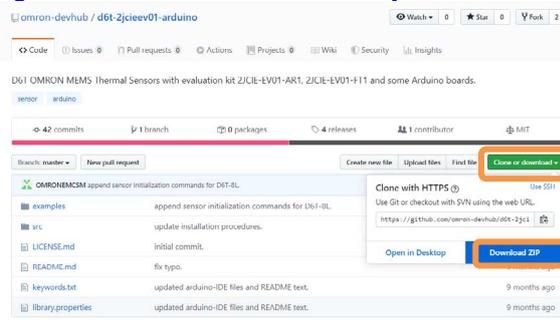


图 24 Download sample code

(5) 将示例代码上传到 Arduino。

“Sketch” -> “Include Library” -> “Add .ZIP Library”

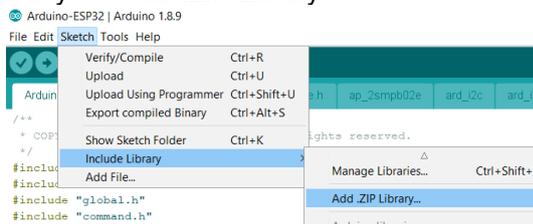


图 25 Add zip

选择“Downloads”文件夹。选择“d6t-2jcieev01-arduino-master.zip”。

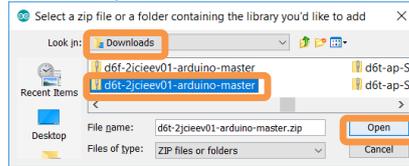


图 26 Select zip file

根据传感器的型号选择文件。

“File” -> “Examples” -> “D6T-2JCIE-EV01”

-> “d6t-1a” or “d6t-8l” or “d6t-8lh” or “d6t-44l” or “d6t-32l”

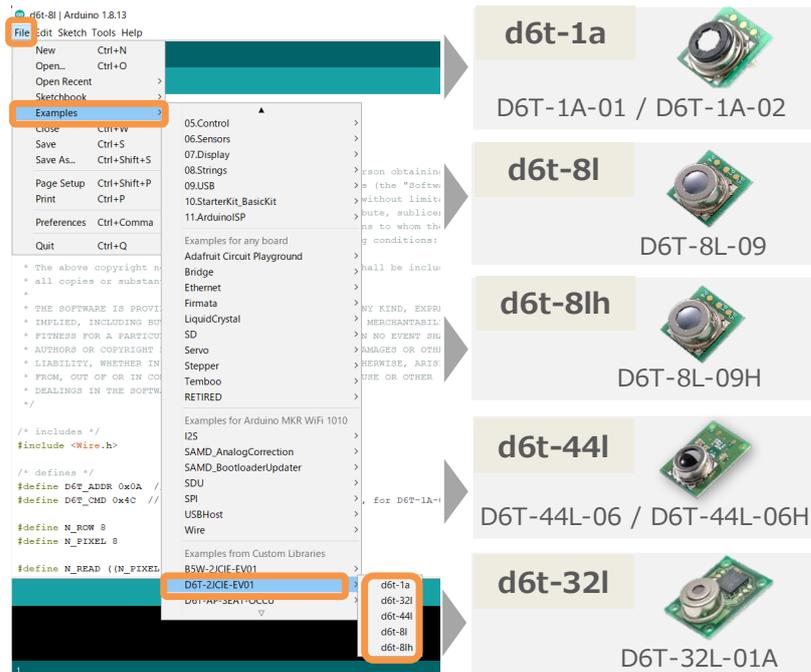


图 27 Select file

点击“Verify”。确认是否出现错误。



图 28 Verify

点击“Upload”。确认是否显示“CPU reset”。



图 29 Upload

(6) 获取数据

“Tools” -> “Serial Monitor”

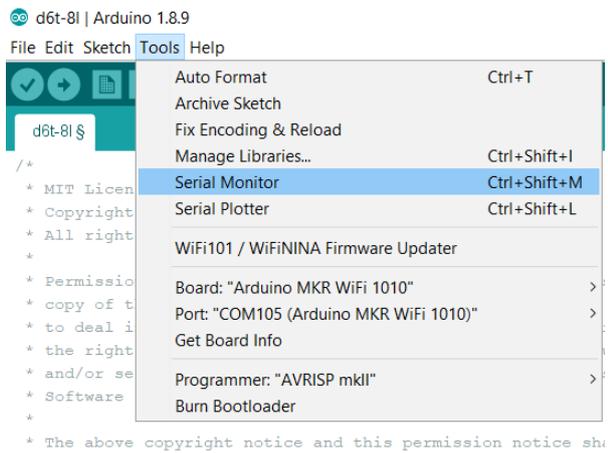


图 30 Device manager

显示数据。

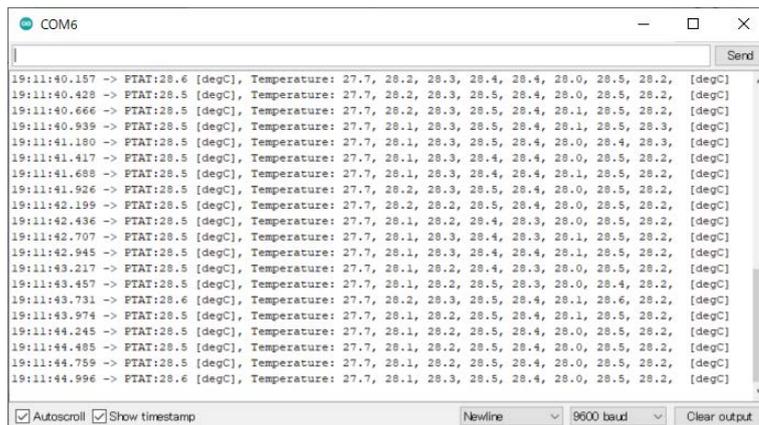


图 31 data

6. 常见疑问

疑问	从接通电源以后到可以测量要花多长时间。
回答	接通电源几秒后，输出温度便会进入温度精度范围内，但到完全稳定下来需要 15 分钟左右。（参考值）
疑问	与传感器之间发生通信异常。或者输出异常值。是否有可能的原因。
答案	请确认以下观点。 <ul style="list-style-type: none">●D6T 通用<ul style="list-style-type: none">· 电压是否正确供给 5V。· 上拉电阻是否为适当值。· 转换传感器数据时，是否适当转换。 (例) P0(Low):0x20, P0 (High):0x01 时, $\text{Data} = (\text{int}16_t) (((\text{uint}16_t)\text{P0}(\text{Low})) + (((\text{uint}16_t)\text{P0}(\text{High})) << 8)) = 0x0120 = 288$●D6T-1A-01 / D6T-1A-02<ul style="list-style-type: none">· 测量数据时是否保持 100ms 以上的间隔。●D6T-8L-09<ul style="list-style-type: none">· 测量数据时是否保持 250ms 以上的间隔。· 电源启动后是否发送初始化指令。●D6T-8L-09H<ul style="list-style-type: none">· 测量数据时是否保持 250ms 以上的间隔。· 电源启动后是否发送初始化指令。· 是否正确转换传感器数据。（D6T-8L-09H 以温度的 5 倍值输出。） 例) 温度输出值: 560 -> 560/5 = 112.0 (°C)●D6T-44L-06 / D6T-44L-06H<ul style="list-style-type: none">· 是否启用所用 MCU 的 I²C 时钟拉伸。· 测量数据时是否保持 300ms 以上的间隔。●D6T-32L-01A<ul style="list-style-type: none">· 是否启用所用 MCU 的 I²C 时钟拉伸。· I²C 的 Read 函数是否有足够的数据缓存（2051 byte）。· 测量数据时是否保持 200ms 以上的间隔。
疑问	可以与传感器通信，但温度高（或低）。或者温度不稳定。是否有可能的原因？
答案	可能有以下原因。 <ul style="list-style-type: none">· 物体比 FOV 小，因此温度也包含背景温度。· 对象物体以外的物体进入视野。· 测量温度不稳定的物体。· 气温发生了变化。· 受到用户框体的影响。（框体进入视野。框体内部发热。）
疑问	能否进一步降低功耗呢？
回答	D6T 系列未设置“休眠省电模式”，因此，必须切断电源以降低功耗。
疑问	无法更改电源电压 3[V]驱动和从站地址吗？
回答	D6T 系列不具备此类功能。
疑问	是否可以进一步扩大视角呢？
回答	受到硅透镜的厚度和折射率因素的制约，本传感器已设定好了 FOV。 1 个元件的 FOV 扩大，可能会造成测量检测能力降低的后果，这也是视角无法扩大的理由之一。要对大范围进行测量，需要设置、使用多个传感器。
疑问	能区分出人与动物和电器吗。
回答	非接触温度传感器只用于获取测量温度数据。需要基于该测量数据的行为动向，使用用户方的软件对对象物进行区分。根据使用情况，开发判断软件，因此，有可能提升判断精度。
疑问	作为人体感应传感器，可以检测的距离是多少。
回答	将会很大程度上受到设置状况和判断算法的影响。 这和热电堆传感器 1 个元件的 FOV 面积和测量对象物体的尺寸都有关系，大约距离为 5~6[m]。

疑问	红外线遥控会引起误动作吗。
回答	波长 1.2[μm]以下的可视光~近红外线间的光几乎不会透过正在使用的硅透镜，所以，不会出现因遥控红外线引起的误动作。作为辐射热发出的远红外线约为 4~14[μm]之间。

7. 用语说明

- 热电堆
串联热电偶增加电动势。热电堆。
将各热电偶的测温端串联起来
- NETD（产品样本记载）
Noise Equivalent Temperature Difference 的略称。将噪声换算成测量温度。
是指可判断测量温度变化的最小参考值，也被称为温度分辨率。
- FOV
Field of View 的略称。视野角度指标。灵敏度峰值多为 50%。

$^{\circ}\text{C}$ 是 Royal Philips、NXP 在荷兰及其他国家的注册商标。
其他记载的公司名称和产品名称均为各公司的注册商标或商标。

8. 承诺事项

承蒙对欧姆龙株式会社(以下简称“本公司”)产品的一贯厚爱和支持,藉此机会再次深表谢意。如果未特别约定,无论贵司从何处购买的产品,都将适用本承诺事项中记载的事项。请在充分了解这些注意事项基础上订购。

1. 定义

本承诺事项中的术语定义如下。

- (1) “本公司产品”:是指“本公司”的 FA 系统机器、通用控制器、传感器、电子 / 结构部件。
- (2) “产品目录等”:是指与“本公司产品”有关的欧姆龙综合产品目录、FA 系统设备综合产品目录、安全组件综合产品目录、电子 / 机构部件综合产品目录以及其他产品目录、规格书、使用说明书、操作指南等,包括以电子数据方式提供的资料。
- (3) “使用条件等”:是指在“产品目录等”资料中记载的“本公司产品”的使用条件、额定值、性能、运行环境、操作使用方法、使用时的注意事项、禁止事项以及其他事项。
- (4) “客户用途”:是指客户使用“本公司产品”的方法,包括将“本公司产品”组装或运用到客户生产的部件、电子电路板、机器、设备或系统等产品中。
- (5) “适用性等”:是指在“客户用途”中“本公司产品”的 (a) 适用性、(b) 动作、(c) 不侵害第三方知识产权、(d) 法规法令的遵守以及 (e) 满足各种规格标准。

2. 关于记载事项的注意事項

对“产品目录等”中的记载内容,请理解如下要点。

- (1) 额定值及性能值是在单项试验中分别在各条件下获得的值,并不构成对各额定值及性能值的综合条件下获得值的承诺。
- (2) 提供的参考数据仅作为参考,并非可在该范围内一直正常运行的保证。
- (3) 应用示例仅作参考,不构成对“适用性等”的保证。
- (4) 如果因技术改进等原因,“本公司”可能会停止“本公司产品”的生产或变更“本公司产品”的规格。

3. 使用时的注意事項

选用及使用本公司产品时请理解如下要点。

- (1) 除了额定值、性能指标外,使用时还必须遵守“使用条件等”。
- (2) 客户应事先确认“适用性等”,进而再判断是否选用“本公司产品”。“本公司”对“适用性等”不做任何保证。
- (3) 对于“本公司产品”在客户的整个系统中的设计用途,客户应负责事先确认是否已进行了适当配电、安装等事项。
- (4) 使用“本公司产品”时,客户必须采取如下措施:(i) 相对额定值及性能指标,必须在留有余量的前提下使用“本公司产品”,并采用冗余设计等安全设计(ii) 所采用的安全设计必须确保即使“本公司产品”发生故障时也可将“客户用途”中的危险降到最小程度、(iii) 构建随时提示使用者危险的完整安全体系、(iv) 针对“本公司产品”及“客户用途”定期实施各项维护保养。
- (5) 因 DDoS 攻击(分布式 DoS 攻击)、计算机病毒以及其他技术性有害程序、非法侵入,即使导致“本公司产品”、所安装软件、或者所有的计算机器材、计算机程序、网络、数据库受到感染,对于由此而引起的直接或间接损失、损害以及其他费用,“本公司”将不承担任何责任。对于(i) 杀毒保护、(ii) 数据输入输出、(iii) 丢失数据的恢复、(iv) 防止“本公司产品”或者所安装软件感染计算机病毒、(v) 防止对“本公司产品”的非法侵入,请客户自行负责采取充分措施。
- (6) “本公司产品”是作为应用于一般工业产品的通用产品而设计生产的。如果客户将“本公司产品”用于以下所列用途,则本公司对产品不作任何保证。但“本公司”已表明可用于特殊用途,或已与客户有特殊约定时,另行处理。
 - (a) 必须具备很高安全性的用途(例:核能控制设备、燃烧设备、航空/宇宙设备、铁路设备、升降设备、娱乐设备、医疗设备、安全装置、其他可能危及生命及人身安全的用途)
 - (b) 必须具备很高可靠性的用途(例:燃气、自来水、电力等供应系统、24 小时连续运行系统、结算系统、以及其他处理权利、财产的用途等)
 - (c) 具有苛刻条件或严酷环境的用途(例:安装在室外的设备、会受到化学污染的设备、会受到电磁波影响的设备、会受到振动或冲击的设备等)
 - (d) “产品目录等”资料中未记载的条件或环境下的用途
- (7) 除了不适用于上述 3.(6)(a) 至(d) 中记载的用途外,“本产品目录等资料中记载的产品”也不适

用于汽车(含二轮车,以下同)。请勿配置到汽车上使用。关于汽车配置用产品,请咨询本公司销售人员。

4.保修条件

“本公司产品”的保修条件如下。

- (1) 保修期限 自购买之日起 1 年。(但是,“产品目录等”资料中有明确说明时除外。)
- (2) 保修内容 对于发生故障的“本公司产品”,由“本公司”判断并可选择以下其中之一方式进行保修。
 - (a) 在本公司的维修保养服务点对发生故障的“本公司产品”进行免费修理(但是对于电子、结构部件不提供修理服务。)
 - (b) 对发生故障的“本公司产品”免费提供同等数量的替代品
- (3) 当故障因以下任何一种情形引起时,不属于保修的范围。
 - (a) 将“本公司产品”用于原本设计用途以外的用途
 - (b) 超过“使用条件等”范围的使用
 - (c) 违反本注意事项“3. 使用时的注意事项”的使用
 - (d) 非因“本公司”进行的改装、修理导致故障时
 - (e) 非因“本公司”出品的软件导致故障时
 - (f) “本公司”生产时的科学、技术水平无法预见的原因
 - (g) 除上述情形外的其它原因,如“本公司”或“本公司产品”以外的原因(包括天灾等不可抗力)

5.责任限制

本承诺事项中记载的保修是关于“本公司产品”的全部保证。对于因“本公司产品”而发生的其他损害,“本公司”及“本公司产品”的经销商不负任何责任。

6.出口管理

客户若将“本公司产品”或技术资料出口或向境外提供时,请遵守中国及各国关于安全保障进出口管理方面的法律、法规。否则,“本公司”有权不予提供“本公司产品”或技术资料。

D6T 通信规格

D6T-1A-01

D6T-1A-02

D6T-8L-09

D6T-8L-09H

D6T-44L-06

D6T-44L-06H

D6T-32L-01A

通信规格

D6T-1A-01

D6T-1A-02

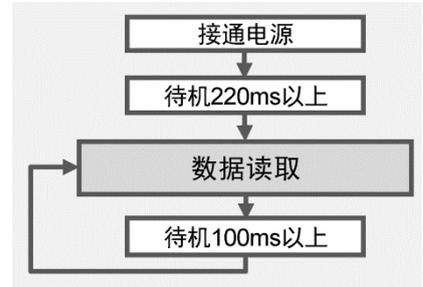


(1) I²C 接口概述

从站地址	7bit (0001_010b) 8bit (with R/W bit)表现: Read : 15h , Write : 14h
数据位宽度	8bit (MSB-first)
时钟速度	max 100kHz
时钟拉伸应对	不支持

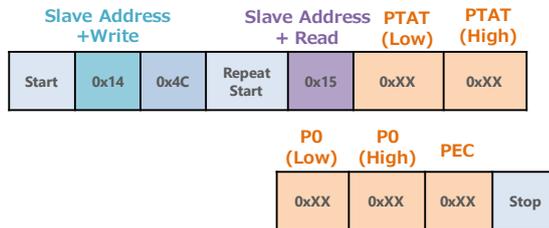
(2) 通信步骤

与传感器的通信步骤如右所示。本传感器为标准规格品时，每 100ms 以内完成对测量数据的更新。该操作与通信无关，重复进行，外部无法控制测量时间。



数据读取

请发送以下指令，执行数据获取。
接收数据有 5byte。



[接收数据项目]

· PTAT:

传感器内部的参照温度数据。带 16bit 符号的整数（2 的补数），10 倍温度值[°C]。

（计算示例）

PTAT(Low):0x20、PTAT(High):0x01 时，

按 Int16 计算: 0x0120 -> 288 -> 288/10 = 28.8 (°C)

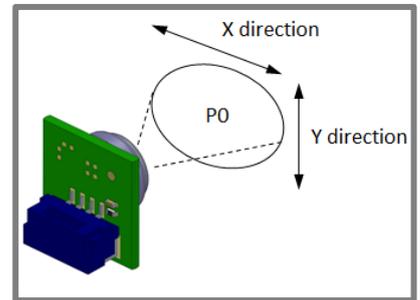
· P0:

各像素的温度数据。配置参照图。带 16bit 符号的整数(2 的补数)，10 倍温度值[°C]

（计算示例）

P0(Low):0x87、P0(High):0xFE 时，

按 Int16 计算: 0xFE87 -> -121 -> -121/10 = -12.1 (°C)



· PEC:

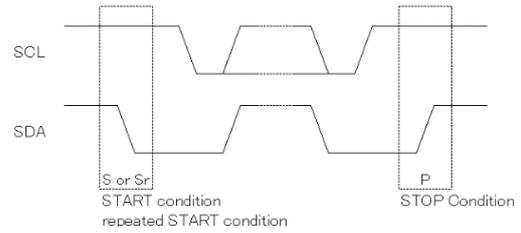
Packet error check code. PEC 将通过以 CRC-8 方式进行错误检查用的数据，附加在通信输出最后。PEC 的对象范围从 Slave Address+ Write 开始到 P0(High)的数据结束，使用各 1byte 的值创建 PEC 值。用户可以使用该 PEC 值检查出通信问题，提高数据的可靠性。

（详情请参阅 SMBus 规格）

(3) I²C 访问协议

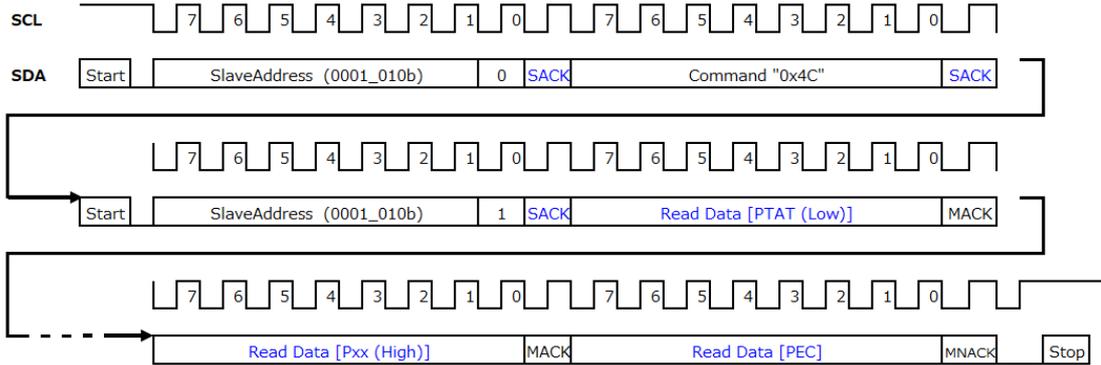
I²C 访问协议的各用语如下所示。

“S”	: Start Condition
“Sr”	: Repeat Start Condition
“P”	: Stop Condition
“W/R”	: Write (Lo) / Read (Hi)
“SACK”	: Acknowledge by Slave
“MACK”	: Acknowledge by Master
“MNACK”	: No-acknowledge by Master



I²C 读访问协议

首先，请在 write 模式下发送 Command。然后，请设为 Repeat Start Condition，再切换至 read 模式，读取数据。



*1. Black characters : Master → Slave 、 Blue characters : Slave → Master

本传感器中 SDA 或 SCL 以下时间 Low 输入持续的情况下，将作为通信超时时间进入停止状态。

· D6T-1A-01 / D6T-1A-02 : 70msec

另外，传感器判断出通信超时时间时，Write 连接时将反馈 NACK，Read 连接时，读取值为 FFFFh。使用 PEC 进行数据检查时，可以判断出读取值异常，所以推荐用 PEC 进行数据检查。

(4) 示例代码

Raspberry Pi 用示例代码在以下 URL 中。

<https://github.com/omron-devhub/d6t-2jcieev01-raspberrypi/blob/master/d6t-1a.c>

Arduino 用示例代码在以下 URL 中。

<https://github.com/omron-devhub/d6t-2jcieev01-arduino/blob/master/examples/d6t-1a/d6t-1a.ino>

示例代码的结构如下所示。

```

/** <!-- main - Thermal sensor {{{1 -->
 * Read data
 */
int main() {
    int i;
    int16_t itemp;

    delay(220);

    while(1){
        // Read data via I2C
        memset(rbuf, 0, N_READ);
        uint32_t ret = i2c_read_reg8(D6T_ADDR, D6T_CMD, rbuf, N_READ);
        D6T_checkPEC(rbuf, N_READ - 1);

        //Convert to temperature data (degC)
        ptat = (double)conv8us_s16_le(rbuf, 0) / 10.0;
        for (i = 0; i < N_PIXEL; i++) {
            itemp = conv8us_s16_le(rbuf, 2 + 2*i);
            pix_data[i] = (double)itemp / 10.0;
        }

        //Output results
        printf("PTAT: %4.1f [degC], Temperature: ", ptat);
        for (i = 0; i < N_PIXEL; i++) {
            printf("%4.1f, ", pix_data[i]);
        }
        printf("[degC]\n");

        delay(100);
    }
}

```

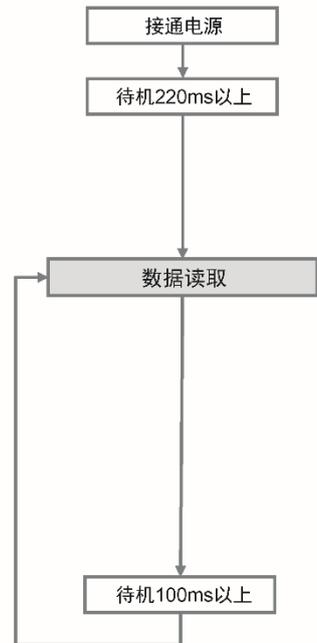
I²C Read

PEC确认

转换为PTAT

转换为温度值

在画面上显示



通信规格

D6T-8L-09



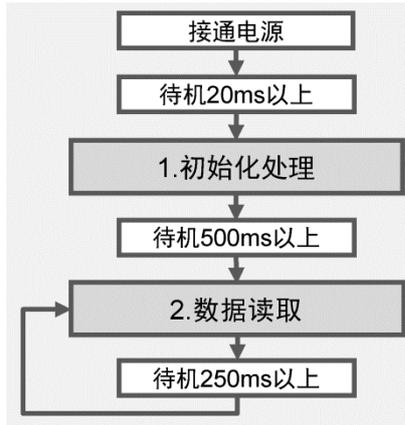
(1) I²C 接口概述

从站地址	7bit (0001_010b) 8bit (with R/W bit)表现: Read : 15h , Write : 14h
数据位宽度	8bit (MSB-first)
时钟速度	max 100kHz
时钟拉伸应对	不支持

(2) 通信步骤

与传感器的通信步骤如下所示。

本传感器为标准规格品时，每 250ms 以内完成对测量数据的更新。该操作与通信无关，重复进行，外部无法控制测量时间。



1. 初始化处理

请如下发送、接收指令。

Slave Address + Write

Start	0x14	0x02	0x00	0x01	0xEE	Stop
Start	0x14	0x05	0x90	0x3A	0xB8	Stop
Start	0x14	0x03	0x00	0x03	0x8B	Stop
Start	0x14	0x03	0x00	0x07	0x97	Stop
Start	0x14	0x02	0x00	0x00	0xE9	Stop

Start	0x14	0x02	Repeat Start	0x15	0XX	0XX	Stop
Start	0x14	0x05	Repeat Start	0x15	0XX	0XX	Stop
Start	0x14	0x03	Repeat Start	0x15	0XX	0XX	Stop

Slave Address + Read Should be 0x00 Should be 0x00

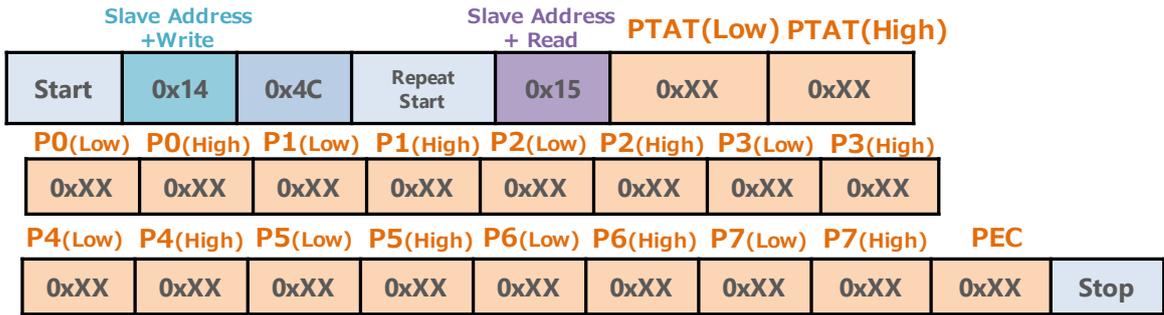
Should be 0x90 Should be 0x3A

Should be 0x00 Should be 0x07

确认是否正确
写入(可省略)

2. 数据读取

请发送以下指令，执行数据获取。接收数据有 19byte。



[接收数据项目]

· PTAT:

传感器内部的参照温度数据。带 16bit 符号的整数（2 的补数），10 倍温度值[°C]。

（计算示例）

PTAT(Low):0x20、PTAT(High):0x01 时，

按 Int16 计算：0x0120 -> 288 -> 288/10 = 28.8 (°C)

· P0 ~ P7:

各像素的温度数据。配置参照图。带 16bit 符号的整数（2 的补数），10 倍温度值[°C]

（计算示例）

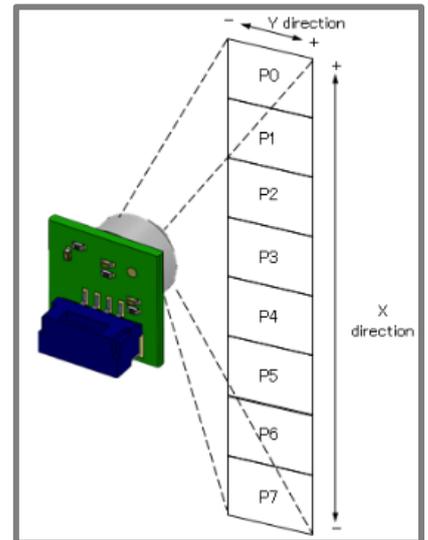
P0(Low):0x87、P0(High):0xFE 时，

按 Int16 计算：0xFE87 -> -121 -> -121/10 = -12.1 (°C)

· PEC:

Packet error check code. PEC 将通过以 CRC-8 方式进行错误检查用的数据，附加在通信输出最后。PEC 的对象范围从 Slave Address+ Write 开始到 P7(High)的数据结束，使用各 1byte 的值创建 PEC 值。用户可以使用该 PEC 值检查出通信问题，提高数据的可靠性。

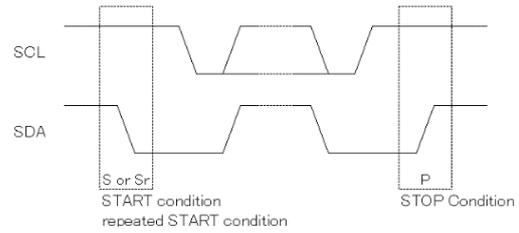
（详情请参阅 SMBus 规格）



(3) I²C 访问协议

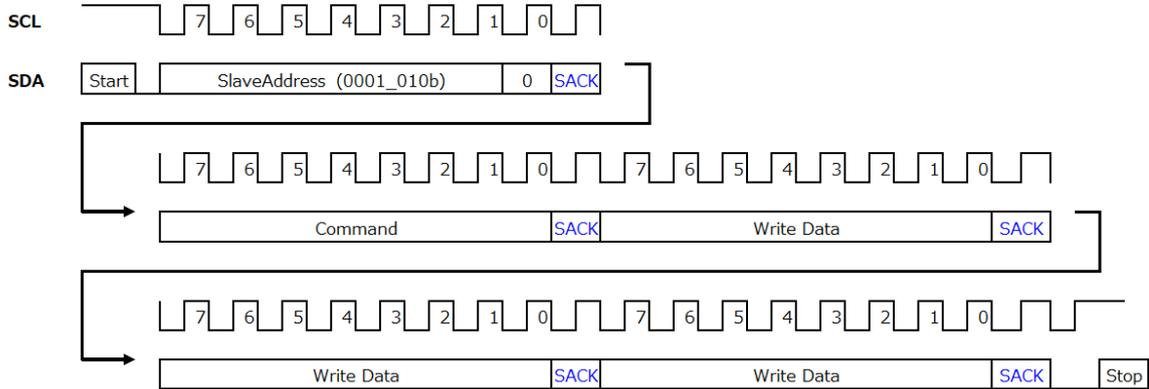
I²C 访问协议的各用语如下所示。

“S”	: Start Condition
“Sr”	: Repeat Start Condition
“P”	: Stop Condition
“W/R”	: Write (Lo) / Read (Hi)
“SACK”	: Acknowledge by Slave
“MACK”	: Acknowledge by Master
“MNACK”	: No-acknowledge by Master



I²C 写访问协议

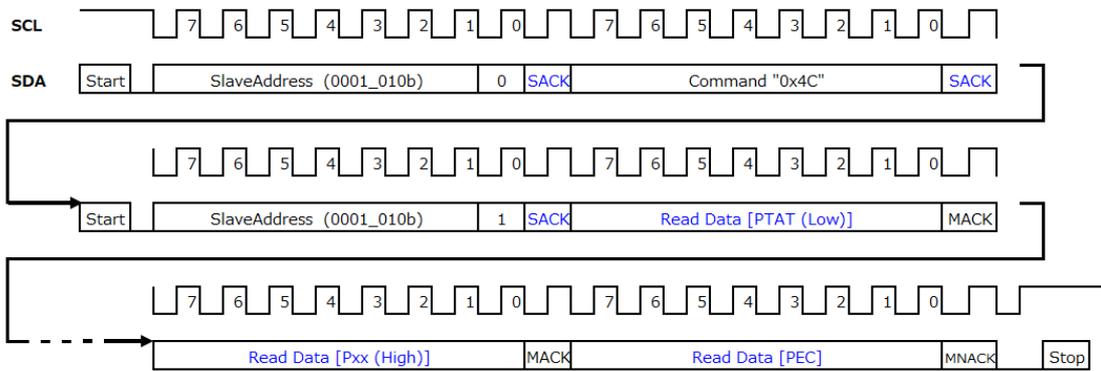
Start Condition 后，请发送 Slave Address (bit7~1)加上写入信号 (=“0”at bit0)的数据，设为 write 模式。然后，请持续发送 Write 数据，设为 Stop Condition。



*1. Black characters : Master → Slave 、 Blue characters : Slave → Master

I²C 读访问协议

首先，请在 write 模式下发送 Command。然后，请设为 Repeat Start Condition，再切换至 read 模式，读取数据。



*1. Black characters : Master → Slave 、 Blue characters : Slave → Master

本传感器中 SDA 或 SCL 以下时间 Low 输入持续的情况下，将作为通信超时时间进入停止状态。

· D6T-8L-09: 70msec

另外，传感器判断出通信超时时间时，Write 连接时将反馈 NACK，Read 连接时，读取值为 FFFFh。使用 PEC 进行数据检查时，可以判断出读取值异常，所以推荐用 PEC 进行数据检查。

(4) 示例代码

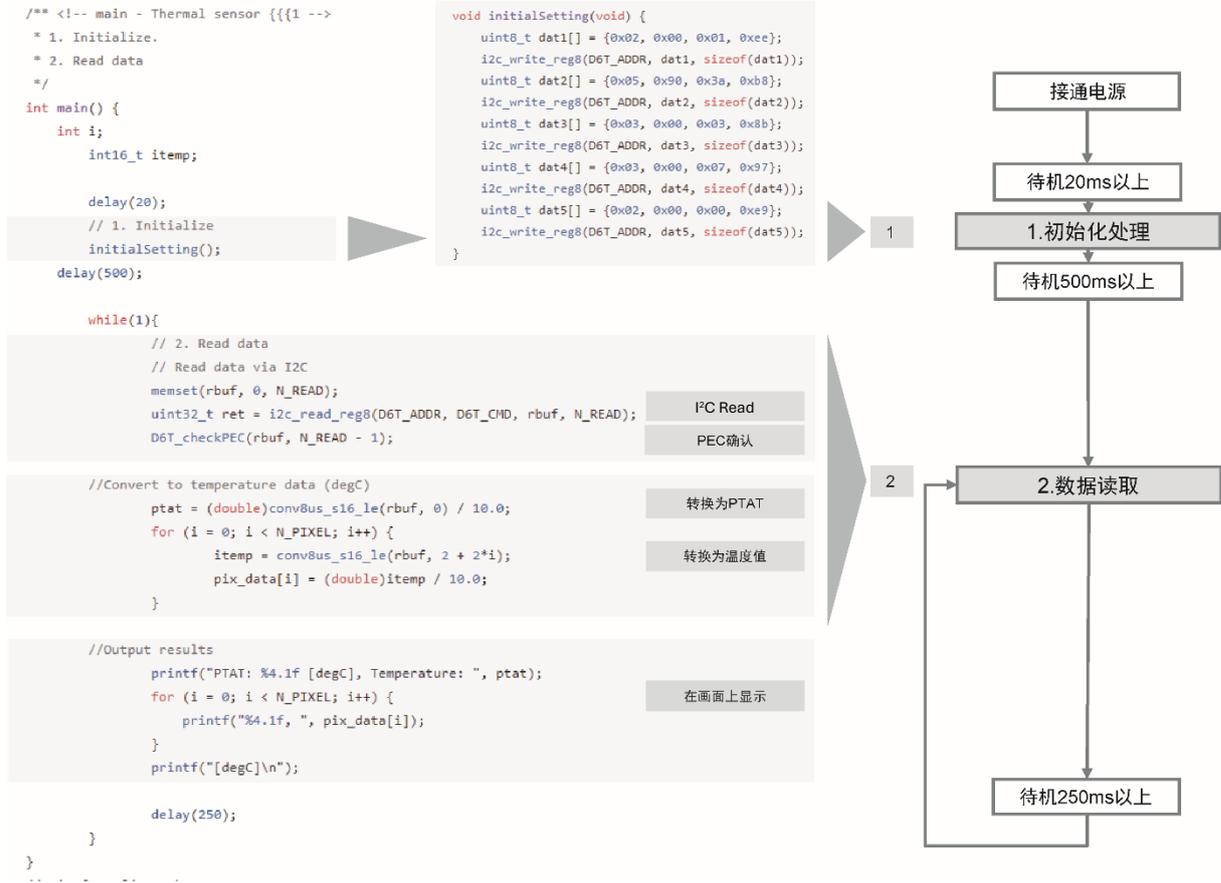
Raspberry Pi 用示例代码在以下 URL 中。

<https://github.com/omron-devhub/d6t-2jcieev01-raspberrypi/blob/master/d6t-8l.c>

Arduino 用示例代码在以下 URL 中。

<https://github.com/omron-devhub/d6t-2jcieev01-arduino/blob/master/examples/d6t-8l/d6t-8l.ino>

示例代码的结构如下所示。



通信规格

D6T-8L-09H



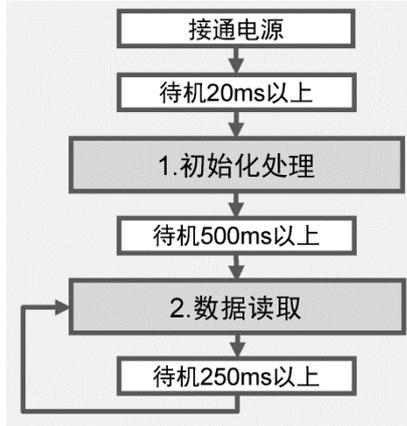
(1) I²C 接口概述

从站地址	7bit (0001_010b) 8bit (with R/W bit)表现: Read : 15h , Write : 14h
数据位宽度	8bit (MSB-first)
时钟速度	max 100kHz
时钟拉伸应对	不支持

(2) 通信步骤

与传感器的通信步骤如下所示。

本传感器为标准规格品时，每 250ms 以内完成对测量数据的更新。该操作与通信无关，重复进行，外部无法控制测量时间。



1.初始化处理

请如下发送、接收指令。

Slave Address + Write

Start	0x14	0x02	0x00	0x01	0xEE	Stop
Start	0x14	0x05	0x90	0x3A	0xB8	Stop
Start	0x14	0x03	0x00	0x03	0x8B	Stop
Start	0x14	0x03	0x00	0x07	0x97	Stop
Start	0x14	0x02	0x00	0x00	0xE9	Stop

Start	0x14	0x02	Repeat Start	0x15	0xFF	0xFF	Stop
Start	0x14	0x05	Repeat Start	0x15	0xFF	0xFF	Stop
Start	0x14	0x03	Repeat Start	0x15	0xFF	0xFF	Stop

Slave Address + Read Should be 0x00 Should be 0x00
 Should be 0x90 Should be 0x3A
 Should be 0x00 Should be 0x07

确认是否正确
写入(可省略)

2.数据读取

请发送以下指令，执行数据获取。接收数据有 19byte。

	Slave Address +Write			Slave Address + Read		PTAT(Low) PTAT(High)			
Start	0x14	0x4C	Repeat Start	0x15	0xXX	0xXX			
	P0(Low) P0(High)		P1(Low) P1(High)		P2(Low) P2(High)		P3(Low) P3(High)		
	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	
	P4(Low) P4(High)		P5(Low) P5(High)		P6(Low) P6(High)		P7(Low) P7(High)		PEC
	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	Stop

[接收数据项目]

· PTAT:

传感器内部的参照温度数据。带 16bit 符号的整数（2 的补数），10 倍的温度值[°C]。

（计算示例）

PTAT(Low):0x20、PTAT(High):0x01 时，

按 Int16 计算：0x0120 -> 288 -> 288/10 = 28.8 (°C)

· P0 ~ P7:

各像素的温度数据。配置参照图。带 16bit 符号的整数（2 的补数），5 倍的温度值[°C]

（计算示例）

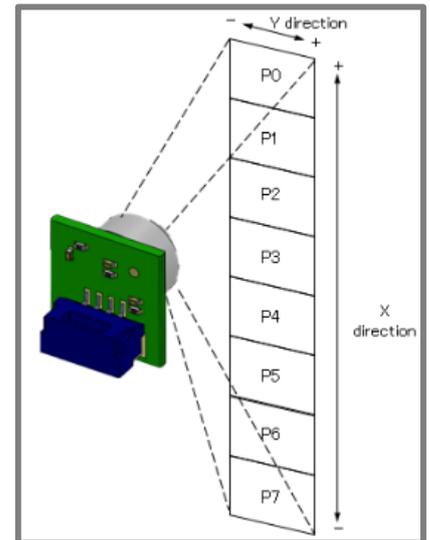
P0(Low):0x30、P0(High):0x02 时，

按 Int16 计算：0x0230 -> 560 -> 560/5 = 112.0 (°C)

· PEC:

Packet error check code. PEC 将通过以 CRC-8 方式进行错误检查用的数据，附加在通信输出最后。PEC 的对象范围从 Slave Address+ Write 开始到 P7(High)的数据结束，使用各 1byte 的值创建 PEC 值。用户可以使用该 PEC 值检查出通信问题，提高数据的可靠性。

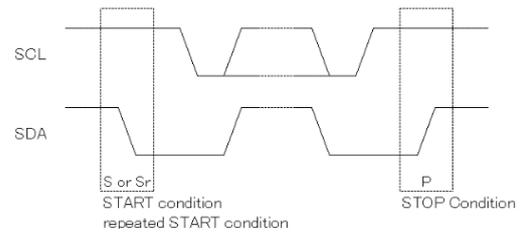
（详情请参阅 SMBus 规格）



(3) I²C 访问协议

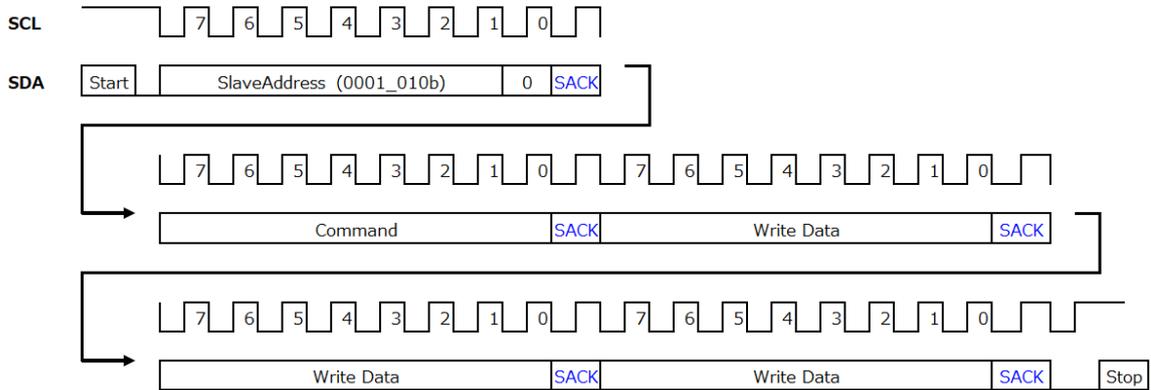
I²C 访问协议的各用语如下所示。

"S"	: Start Condition
"Sr"	: Repeat Start Condition
"P"	: Stop Condition
"W/R"	: Write (Lo) / Read (Hi)
"SACK"	: Acknowledge by Slave
"MACK"	: Acknowledge by Master
"MNACK"	: No-acknowledge by Master



I²C 写访问协议

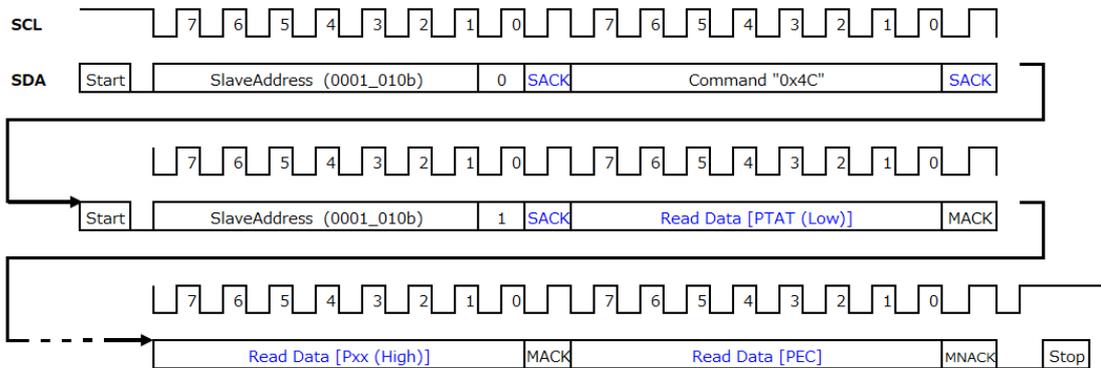
Start Condition 后，请发送 Slave Address (bit7~1)加上写入信号 (=“0”at bit0)的数据，设为 write 模式。然后，请持续发送 Write 数据，设为 Stop Condition。



*1. Black characters : Master → Slave 、 Blue characters : Slave → Master

I²C 读访问协议

首先，请在 write 模式下发送 Command。然后，请设为 Repeat Start Condition，再切换至 read 模式，读取数据。



*1. Black characters : Master → Slave 、 Blue characters : Slave → Master

本传感器中 SDA 或 SCL 以下时间 Low 输入持续的情况下，将作为通信超时时间进入停止状态。

· D6T-8L-09H: 70msec

另外，传感器判断出通信超时时间时，Write 连接时将反馈 NACK，Read 连接时，读取值为 FFFFh。使用 PEC 进行数据检查时，可以判断出读取值异常，所以推荐用 PEC 进行数据检查。

(4) 示例代码

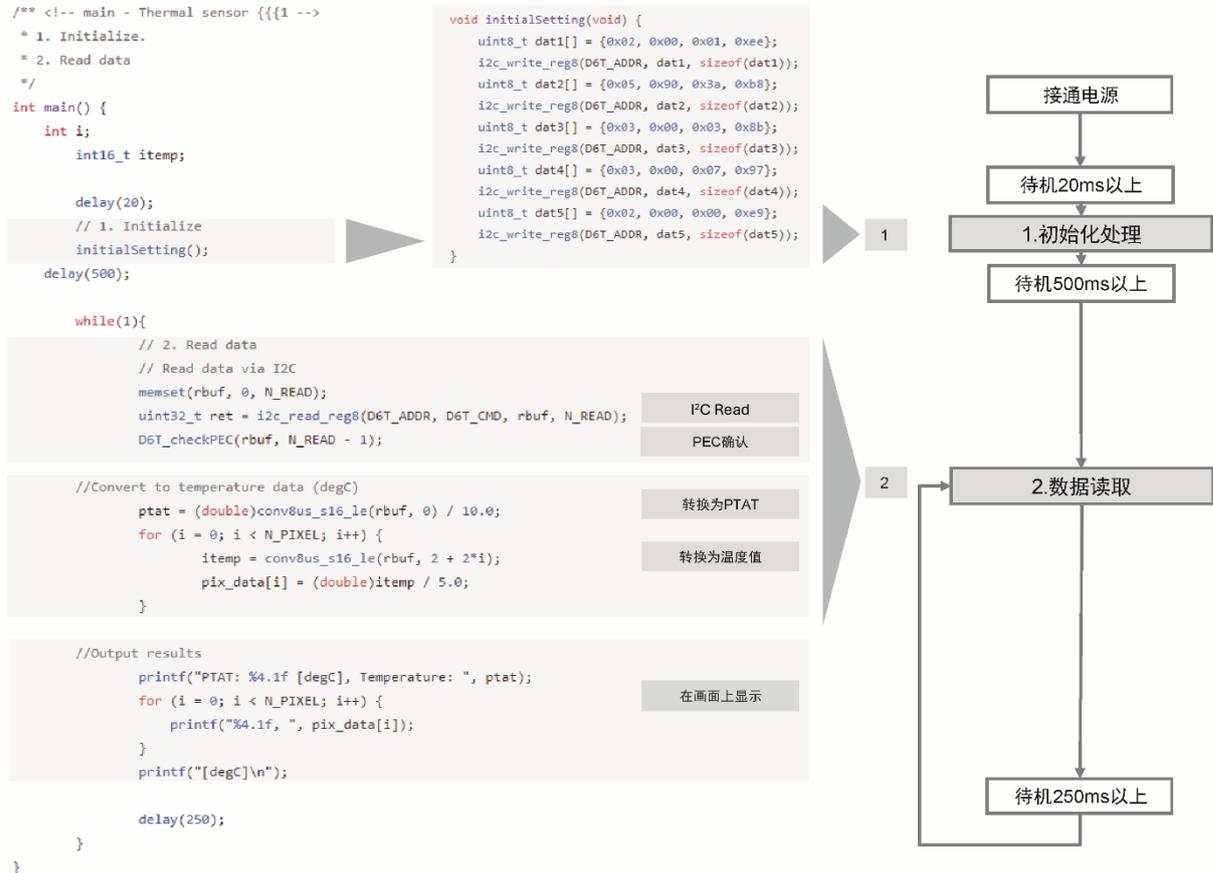
Raspberry Pi 用示例代码在以下 URL 中。

<https://github.com/omron-devhub/d6t-2jcieev01-raspberrypi/blob/master/d6t-8lh.c>

Arduino 用示例代码在以下 URL 中。

<https://github.com/omron-devhub/d6t-2jcieev01-arduino/blob/master/examples/d6t-8lh/d6t-8lh.ino>

示例代码的结构如下所示。



通信规格
D6T-44L-06
D6T-44L-06H



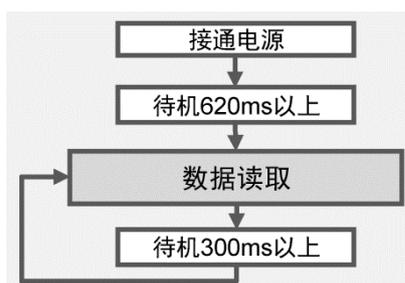
(1) I²C 接口概述

从站地址	7bit (0001_010b) 8bit (with R/W bit)表现: Read : 15h , Write : 14h
数据位宽度	8bit (MSB-first)
时钟速度	max 100kHz
时钟拉伸应对	支持

(2) 通信步骤

与传感器的通信步骤如下所示。

本传感器为标准规格品时，每 300ms 以内完成对测量数据的更新。该操作与通信无关，重复进行，外部无法控制测量时间。



数据读取

请发送以下指令，执行数据获取。接收数据有 35byte。

Slave Address + Write		Slave Address + Read		PTAT(Low)		PTAT(High)			
Start	0x14	0x4C	Repeat Start	0x15	0xXX	0xXX			
P0 (Low)	P0 (High)	P1 (Low)	P1 (High)	P2 (Low)	P2 (High)	P3 (Low)	P3 (High)		
0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX		
P4 (Low)	P4 (High)	P5 (Low)	P5 (High)	P6 (Low)	P6 (High)	P7 (Low)	P7 (High)		
0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX		
P8 (Low)	P8 (High)	P9 (Low)	P9 (High)	P10 (Low)	P10 (High)	P11 (Low)	P11 (High)		
0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX		
P12 (Low)	P12 (High)	P13 (Low)	P13 (High)	P14 (Low)	P14 (High)	P15 (Low)	P15 (High)	PEC	
0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	Stop

[接收数据项目]

· PTAT:

传感器内部的参照温度数据。带 16bit 符号的整数（2 的补数），10 倍的温度值[°C]。

（计算示例）

PTAT(Low):0x20、PTAT(High):0x01 时，

按 Int16 计算：0x0120 -> 288 -> 288/10 = 28.8 (°C)

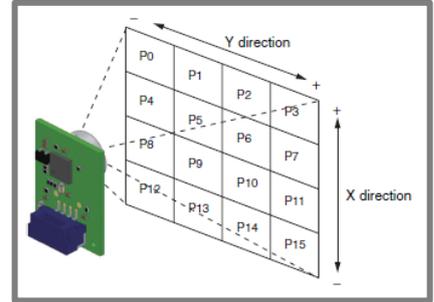
· P0 ~ P15:

各像素的温度数据。配置参照图。带 16bit 符号的整数（2 的补数），10 倍的温度值[°C]

（计算示例）

P0(Low):0x87、P0(High):0xFE 时，

按 Int16 计算：0xFE87 -> -121 -> -121/10 = -12.1 (°C)



· PEC:

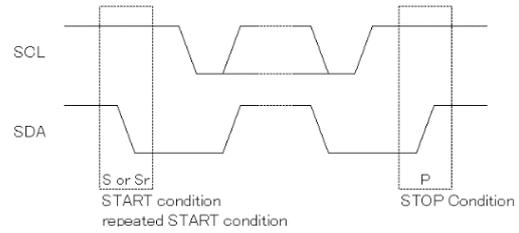
Packet error check code. PEC 将通过以 CRC-8 方式进行错误检查用的数据, 附加在通信输出最后。PEC 的对象范围从 Slave Address+ Write 开始到 P15(High)的数据结束, 使用各 1byte 的值创建 PEC 值。用户可以使用该 PEC 值检查出通信问题, 提高数据的可靠性。

（详情请参阅 SMBus 规格）

(3) I²C 访问协议

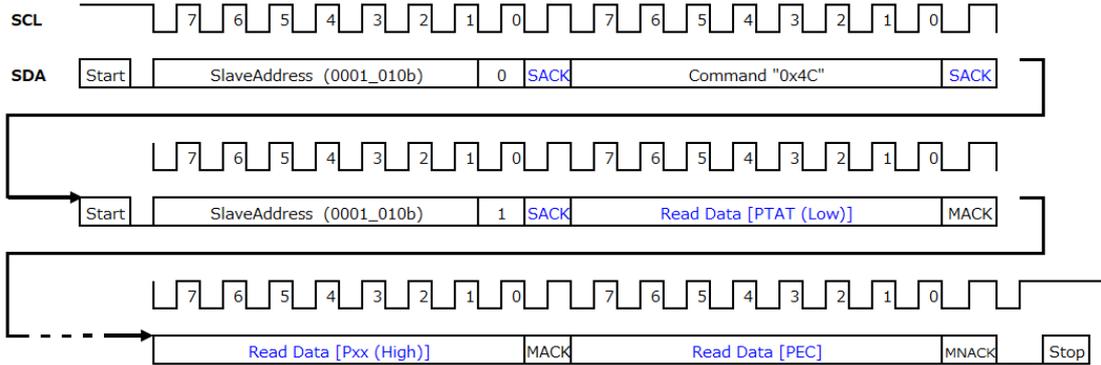
I²C 访问协议的各用语如下所示。

“S”	: Start Condition
“Sr”	: Repeat Start Condition
“P”	: Stop Condition
“W/R”	: Write (Lo) / Read (Hi)
“SACK”	: Acknowledge by Slave
“MACK”	: Acknowledge by Master
“MNACK”	: No-acknowledge by Master



I²C 读访问协议

首先，请在 write 模式下发送 Command。然后，请设为 Repeat Start Condition，再切换至 read 模式，读取数据。



*1. Black characters : Master → Slave 、 Blue characters : Slave → Master

本传感器中 SDA 或 SCL 以下时间 Low 输入持续的情况下，将作为通信超时时间进入停止状态。

· D6T-44L-06 / D6T-44L-06H : 1sec

另外，传感器判断出通信超时时间时，Write 连接时将反馈 NACK，Read 连接时，读取值为 FFFFh。使用 PEC 进行数据检查时，可以判断出读取值异常，所以推荐用 PEC 进行数据检查。

(4) 示例代码

Raspberry Pi 用示例代码在以下 URL 中。

<https://github.com/omron-devhub/d6t-2jcieev01-raspberrypi/blob/master/d6t-44l.c>

Arduino 用示例代码在以下 URL 中。

<https://github.com/omron-devhub/d6t-2jcieev01-arduino/blob/master/examples/d6t-44l/d6t-44l.ino>

示例代码的结构如下所示。

```
/** <!-- main - Thermal sensor {{{1 -->
```

```
* Read data
*/
```

```
int main() {
```

```
    int i;
```

```
    int16_t itemp;
```

```
    delay(620);
```

```
    while(1){
```

```
        // Read data via I2C
```

```
        memset(rbuf, 0, N_READ);
```

```
        uint32_t ret = i2c_read_reg8(D6T_ADDR, D6T_CMD, rbuf, N_READ);
```

```
        D6T_checkPEC(rbuf, N_READ - 1);
```

I²C Read

PEC确认

```
        //Convert to temperature data (degC)
```

```
        ptat = (double)conv8us_s16_le(rbuf, 0) / 10.0;
```

```
        for (i = 0; i < N_PIXEL; i++) {
```

```
            itemp = conv8us_s16_le(rbuf, 2 + 2*i);
```

```
            pix_data[i] = (double)itemp / 10.0;
```

```
        }
```

转换为PTAT

转换为温度值

```
        //Output results
```

```
        printf("PTAT: %4.1f [degC], Temperature: ", ptat);
```

```
        for (i = 0; i < N_PIXEL; i++) {
```

```
            printf("%4.1f, ", pix_data[i]);
```

```
        }
```

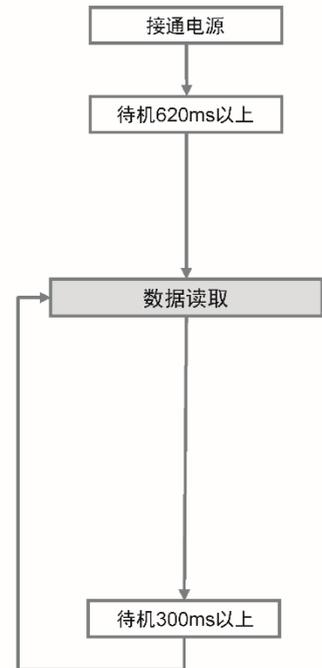
```
        printf("[degC]\n");
```

在画面上显示

```
        delay(300);
```

```
    }
```

```
}
```

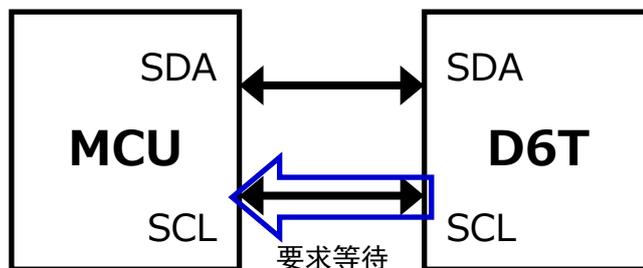


(5) 时钟拉伸

时钟拉伸是指，I²C 从站的处理来不及，从站侧可以延长 SCL 的 LOW 时间，使主站侧待机的功能。从站侧的 D6T 具有该功能，因此主站侧的 MCU 也需要支持该功能。

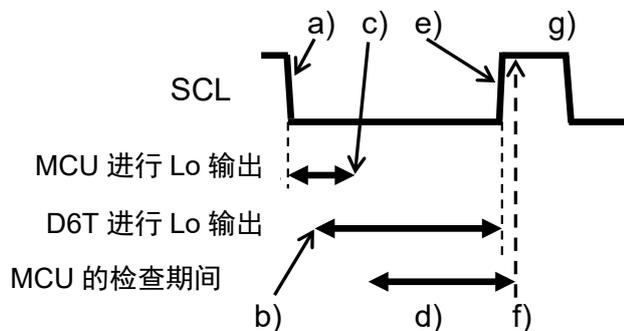
内置于许多 MCU 里的 I²C 模组都具备自动应对功能。请根据所用 MCU 的规格，启用时钟拉伸后再使用。

在未内置 I²C 模组的 MCU 等中，通过软件使用 I²C 库时，请确认好其是否具备等待对应功能后再继续使用。若不具备该功能，需将以下所示等待检测程序添加到 SCL 输出部分。



等待检测程序

I ² C 主站	I ² C 从站 (传感器)
a) 向 SCL 输出 Lo (每 Ack 时间)	SCL 端子的 Lo 检测确认
(固定等待)	b) 向 SCL 输出 Lo (要求等待)
c) 将 SCL 输出变更为 Hi-Z	等待中...
SCL 端子变更为输入模式	:
d) 检查 SCL 端子是否成为 Hi	:
检查待机 (LOOP)	等待完成
f) 检查完成 (Hi 检测)	e) 将 SCL 输出变更为 Hi-Z
将 SCL 端子变更为输出模式	
g) 进入下一个处理	



等待检测程序设定困难时，请将每 Ack 时间 160μse 的等待时间添加至程序中。

通信规格

D6T-32L-01A



(1) I²C 接口概述

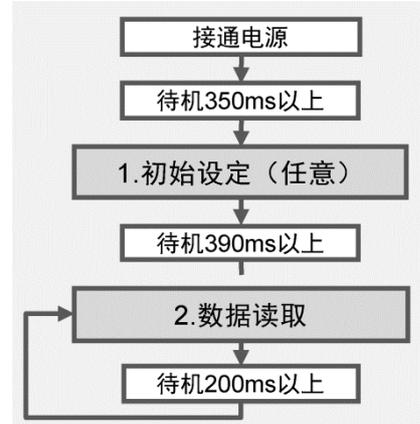
从站地址	7bit (0001_010b) 8bit (with R/W bit)表现: Read : 15h , Write : 14h
数据位宽度	8bit (MSB-first)
时钟速度	max 1000kHz (Fast-Mode Plus)
时钟拉伸应对	支持

2) 通信步骤

与传感器的通信步骤如下所示。本传感器为标准规格品时，每 200ms 以内完成对测量数据的更新。该操作与通信无关，重复进行，外部无法控制测量时间。

1.初始设定（任意）

请如下发送指令。不变更设定时，无需发送初始设定。不变更设定时，根据 Default 设定进行动作。实施初始设定时，请在电源接通 350ms 后实施。设定一经变更，在关闭电源后也会被删除，变为 Default 设定。

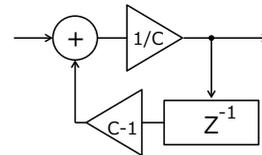


	Slave Address + Write	Register Address	Write data	
Start	0x14	0x01	0xXX	Stop

寄存器详情如下所示。

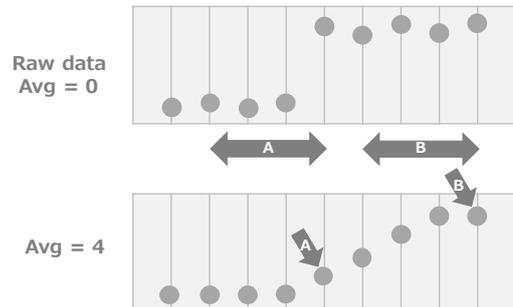
Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0x01	IIR[3]	IIR[2]	IIR[1]	IIR[0]	Avg[3]	Avg[2]	Avg[1]	Avg[0]
Default	0	0	0	0	0	1	0	0

IIR[3:0]: IIR 滤波器设定。
IIR 滤波器系数可以设定 0（直通）、1~15。
以设定值的 4 倍为 C 时，
 $Y = ((C-1) \times Y_{old} + X) / C$ 。
IIR 滤波器的系数越大，噪声越小，但 Response time 越慢。



Avg[3:0]: 移动平均设定。
移动平均可在 0~10 范围内设定。
0 与 1 相同。Default 值为 4。
移动平均次数越大，噪声越小，
但 Response time 越慢。

右图为移动平均的一例。
移动平均设定为 4 时，
值为以往 4 次数据的平均值。



2. 数据读取

请发送以下指令，执行数据获取。接收数据有 2051byte。

Slave Address +Write				PTAT (Low)	PTAT (High)			
Start	0x14	0x4D	Repeat Start	0x15	0xXX	0xXX		
P0 (Low)	P0 (High)	P1 (Low)	P1 (High)	P2 (Low)	P2 (High)	P3 (Low)	P3 (High)	
0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	
P4 (Low)	P4 (High)	P5 (Low)	P5 (High)	P6 (Low)	P6 (High)	P7 (Low)	P7 (High)	
0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	
P8 (Low)	P8 (High)	-----				P1019 (Low)	P1019 (High)	
0xXX	0xXX					0xXX	0xXX	
P1020 (Low)	P1020 (High)	P1021 (Low)	P1021 (High)	P1022 (Low)	P1022 (High)	P1023 (Low)	P1023 (High)	PEC
0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX
Stop								

[接收数据项目]

· PTAT:

传感器内部的参照温度数据。带 16bit 符号的整数（2 的补数），10 倍温度值[°C]。

（计算示例）

PTAT(Low):0x20、PTAT(High):0x01 时，

按 Int16 计算：0x0120 -> 288 -> 288/10 = 28.8 (°C)

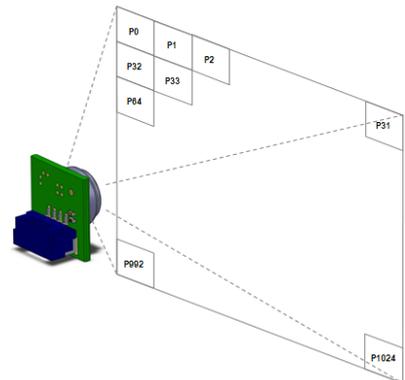
· P0 ~ P1023:

各像素的温度数据。配置参照图。带 16bit 符号的整数（2 的补数），10 倍温度值[°C]

（计算示例）

P0(Low):0x87、P0(High):0xFE 时，

按 Int16 计算：0xFE87 -> -121 -> -121/10 = -12.1 (°C)



· PEC:

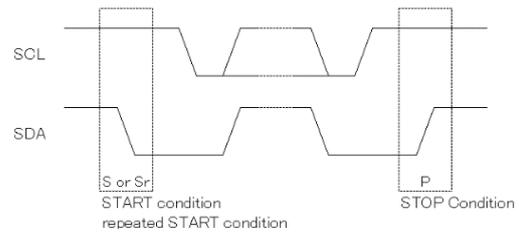
Packet error check code. PEC 将通过以 CRC-8 方式进行错误检查用的数据，附加在通信输出最后。PEC 的对象范围从 PTAT(Low)开始到 P1023(High)的数据结束，使用各 1byte 的值创建 PEC 值。用户可以使用该 PEC 值检查出通信问题，提高数据的可靠性。

（详情请参阅 SMBus 规格）

(3) I²C 访问协议

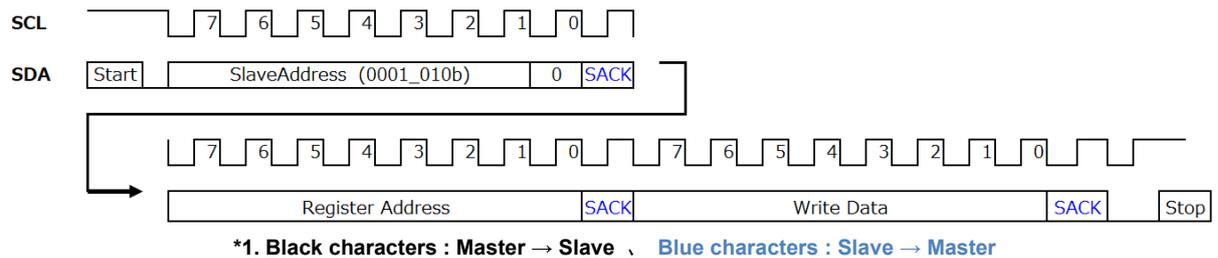
I²C 访问协议的各用语如下所示。

"S"	: Start Condition
"Sr"	: Repeat Start Condition
"P"	: Stop Condition
"W/R"	: Write (Lo) / Read (Hi)
"SACK"	: Acknowledge by Slave
"MACK"	: Acknowledge by Master
"MNACK"	: No-acknowledge by Master



I²C 写访问协议

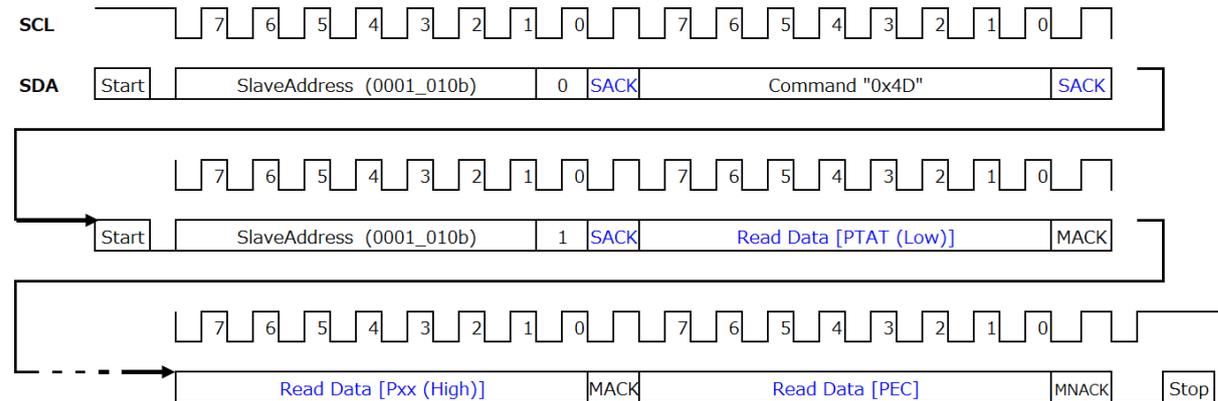
Start Condition 后，请发送 Slave Address (bit7~1)加上写入信号 (=“0”at bit0) 的数据，设为 write 模式。然后，请发送 Write 数据，设为 Stop Condition。



*1. Black characters : Master → Slave 、 Blue characters : Slave → Master

I²C 读访问协议

首先，请在 write 模式下发送 Command。然后，请设为 Repeat Start Condition，再切换至 read 模式，读取数据。



*1. Black characters : Master → Slave 、 Blue characters : Slave → Master

本传感器中 SDA 或 SCL 以下时间 Low 输入持续的情况下，将作为通信超时时间进入停止状态。

· D6T-32L-01A: 1sec

另外，传感器判断出通信超时时间时，Write 连接时将反馈 NACK，Read 连接时，读取值为 FFFFh。使用 PEC 进行数据检查时，可以判断出读取值异常，所以推荐用 PEC 进行数据检查。

(4) 示例代码

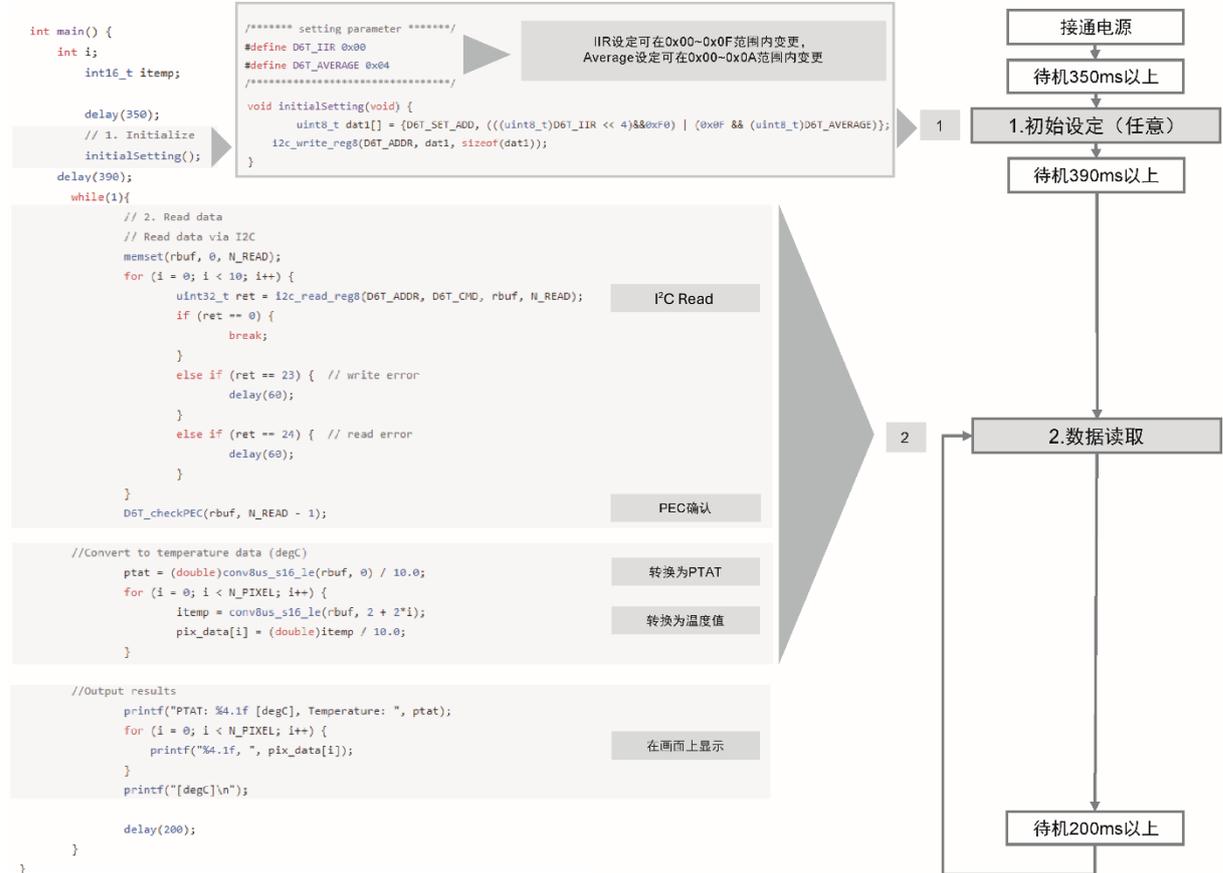
Raspberry Pi 用示例代码在以下 URL 中。

<https://github.com/omron-devhub/d6t-2jcieev01-raspberrypi/blob/master/d6t-32l.c>

Arduino 用示例代码在以下 URL 中。

<https://github.com/omron-devhub/d6t-2jcieev01-arduino/blob/master/examples/d6t-32l/d6t-32l.ino>

示例代码的结构如下所示。

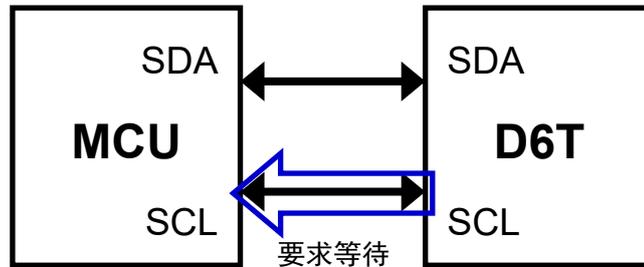


(5) 时钟拉伸

时钟拉伸是指，I²C 从站的处理来不及，从站侧可以延长 SCL 的 LOW 时间，使主站侧待机的功能。从站侧的 D6T 具有该功能，因此主站侧的 MCU 也需要支持该功能。

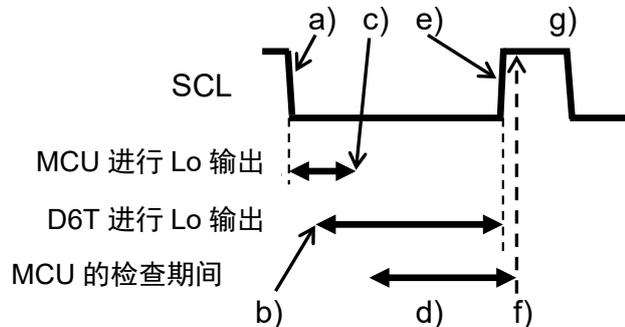
内置于许多 MCU 里的 I²C 模组都具备自动应对功能。请根据所用 MCU 的规格，启用时钟拉伸后再使用。

在未内置 I²C 模组的 MCU 等中，通过软件使用 I²C 库时，请确认好其是否具备等待对应功能后再继续使用。若不具备该功能，需将以下所示等待检测程序添加到 SCL 输出部分。



等待检测程序

I ² C 主站	I ² C 从站 (传感器)
a) 向 SCL 输出 Lo (每 Ack 时间)	SCL 端子的 Lo 检测确认
(固定等待)	b) 向 SCL 输出 Lo (要求等待)
c) 将 SCL 输出变更为 Hi-Z	等待中...
SCL 端子变更为输入模式	:
d) 检查 SCL 端子是否成为 Hi	:
检查待机 (LOOP)	等待完成
f) 检查完成 (Hi 检测)	e) 将 SCL 输出变更为 Hi-Z
将 SCL 端子变更为输出模式	
g) 进入下一个处理	



等待检测程序设定困难时，请将每 Ack 时间 160μse 的等待时间添加至程序中。

订购前请务必阅读我司网站上的“注意事项”。

欧姆龙电子部品(中国)统辖集团

网站

欧姆龙电子部品贸易(上海)有限公司

<https://components.omron.com.cn>

Cat. No. **CDSC-CN1-019E**

2024年4月

© OMRON Corporation 2024 All Rights Reserved.
规格等随时可能更改,恕不另行通知。