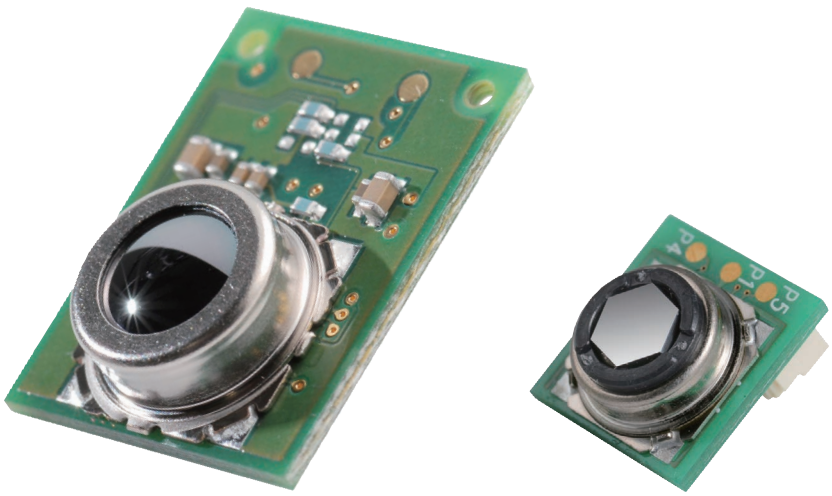
OMRON

MEMS 非接触温度センサ
形D6T

ユーザーズマニュアル

MEMS 非接触温度センサ



CDSC-019F

目次

1. 概要

本ユーザーズマニュアルは、ＭＥＭＳ非接触温度センサＤ６Ｔシリーズの使用方法、特記事項などを示すものです。なお、本資料は製品カタログを補足するものであり、実際のご使用にあたっては製品カタログとあわせてご使用ください。

2. 動作原理

ＭＥＭＳ非接触温度センサの測定動作の概要は、下記のとおりです。

- シリコンレンズにより、物体が発生している放射熱（遠赤外線）をモジュール内のサーモパイルセンサ上に集光します。(＊１)
- 集光した放射熱（遠赤外線）により、サーモパイルセンサで起電力が発生します。
- 起電力の値と、内部の温度センサの値を測定し、それらの値を元に、内蔵のルックアップテーブルを用いた補間演算により測定値（対象物温度）を算出します。(＊２)
- 測定値は、I^2C バスを介して、上位システムから読み出して使用します。

＊１. D6T-1A-01/02 はシリコンフィルタを使用。

＊２. D6T-1A-01/02、D6T-8L-09/09H は ASIC 内の温度換算回路を用いて測定値（対象物温度）を算出します。

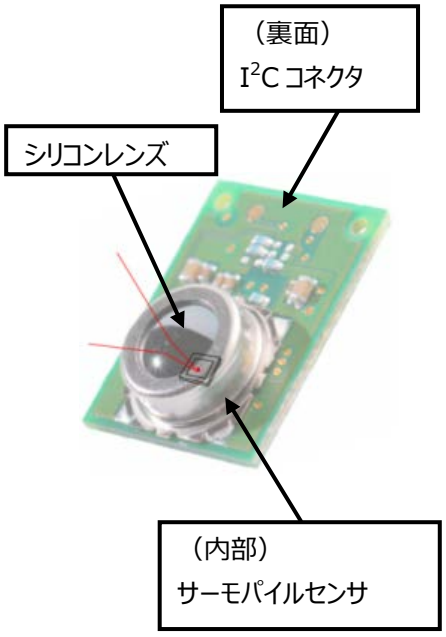
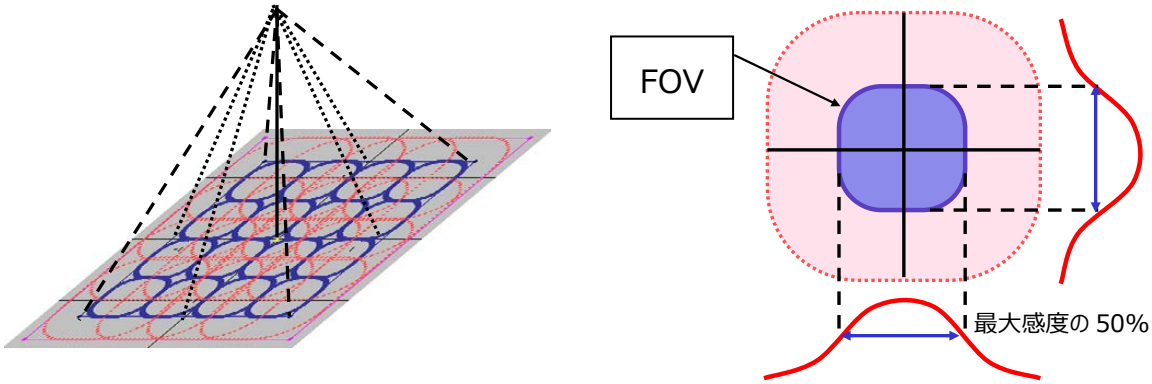


図１　モジュール構成

3. 製品の特徴

ＭＥＭＳ非接触温度センサは、光学設計されたシリコンレンズにより、決められた感度特性を持っています。弊社の非接触温度センサでは、一般的なセンサと同じく、最大感度の 50%となる領域角度を FOV（Field of View）として記載しています。



a）D6T-44L-06/06H の FOV（16ch）イメージ図　　b）1 素子の FOV と XY 軸の感度特性イメージ図

図 2　視野角（FOV）の感度特性の形状イメージ

感度を持つ領域は、FOV 仕様幅よりも広い領域となります。また、測定対象物の大きさが、感度を持つ領域よりも小さい場合には、対象物以外の背景温度の影響が含まれることになります。

弊社の非接触温度センサは、基準熱源（黒体炉）を用いて温度測定値を補正していますが、測定対象物の材質による放射率の違い、表面形状、感度を持つ領域内の占有率などが、測定値に影響することに留意ください。

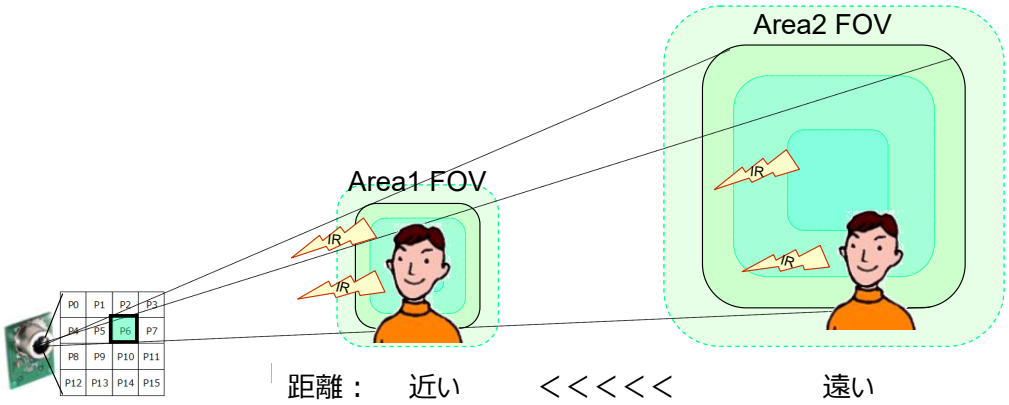


図 3　距離による測定値の変化要因

距離が遠くなると、観測する面積（FOV）は広がっていきます。FOV 内での対象物（人物）の占有面積率は、距離に応じて小さくなっていきます。よって距離が遠くなると、測定値に占める割合（影響度）は、対象物（人物）温度よりも背景温度の方が大きくなっていきます。言い換えると、温度を正しく測定するためには、対象物体が、ＦＯＶの面積よりも十分大きい必要があります。

ＭＥＭＳ非接触温度センサを、人感センサとして利用する場合には、温度値のみによる単純な判定では近距離用途に限定されてしまいます。検出距離を長くするために、時間的な変化や熱源の位置、人の振る舞い情報などを元に、ソフトウェア処理によって判定確度を向上させる必要があります。

4. 使用方法

4.1 接続コネクタ

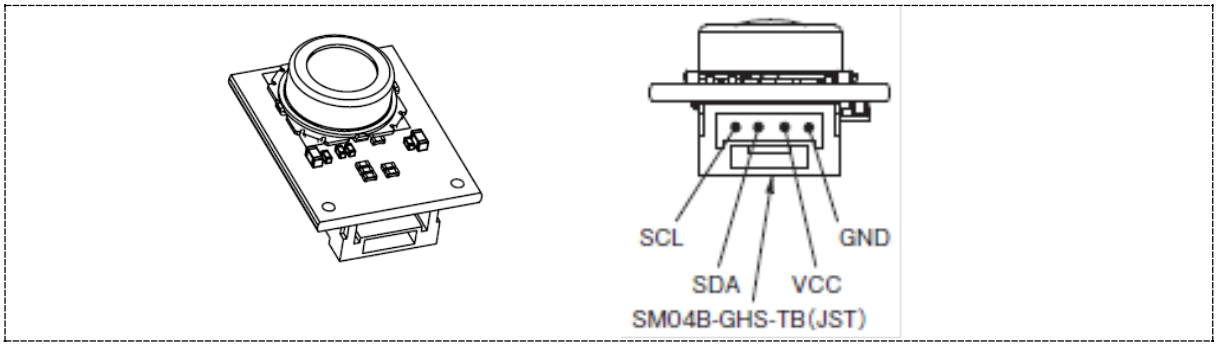


図 4　製品外観（参考）

接続端子

表 1.　接続端子表

1	GND	GND 電源端子
2	VCC	VCC 電源端子（5V ±10%）
3	SDA	I^2C(5V) データ
4	SCL	I^2C(5V) クロック

接続コネクタ部材

使用コネクタ部品型式 ：SM04B-GHS-TB (JST 社製)
コンタクト　：　SSHL-002T-P0.2 (JST 社製)
ハウジング　：　GHR-04V-S (JST 社製)

形式によって、レンズ部の高さや基板サイズが異なります。詳細寸法については、製品のカタログ資料をご確認ください。システムとの接続には、上記の 4 端子コネクタを使用します。

4.2 電気的接続例

ケース 1:　5V MCU 直結（マイコン電源電圧が同じ場合）

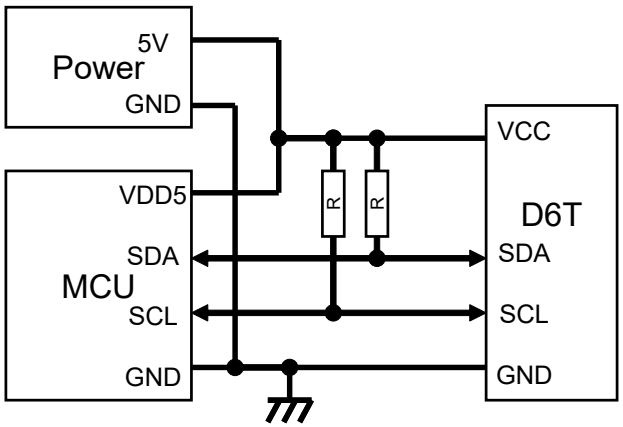


図 5　5V マイコンと直結

ケース 2:　3V MCU (I^2C port が 5V トレラント仕様の場合)

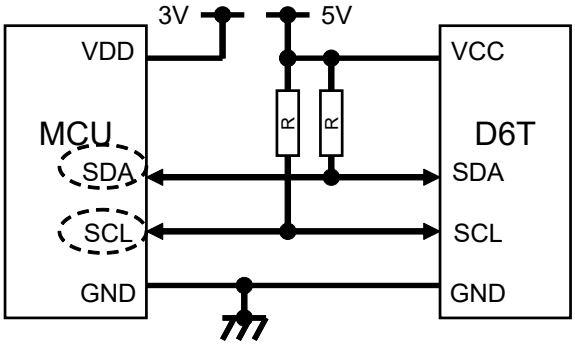


図 6　5V トレラント仕様の場合

ケース 3:　I^2C レベルコンバータを使用

(5V トレラント仕様でない場合や、バス上に I^2C（3V）仕様の他デバイスが存在する場合)

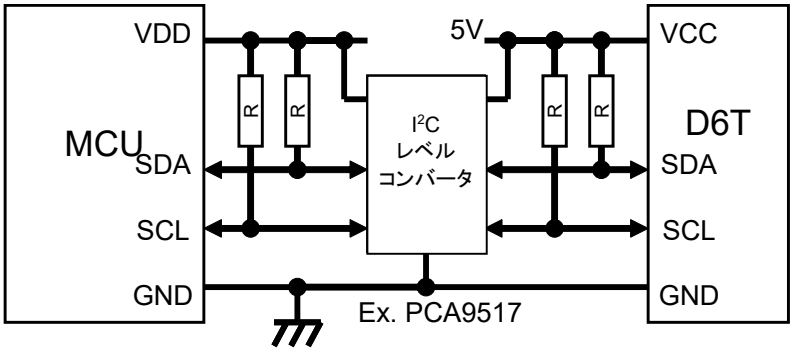


図 7　レベルコンバータ利用時

ケース 4:　双方向オープンドレインの GPIO 端子を使い、 ソフト的に I^2C 通信処理を行う

(MCU が内部に I^2C 機能を持たない場合)

＊1. D6T-44L-06 / D6T-44L-06H / D6T-32L-01A については、クロックストレッチ対応が必須となります。後述の D6T 通信仕様のクロックストレッチの章を参考にしてください。

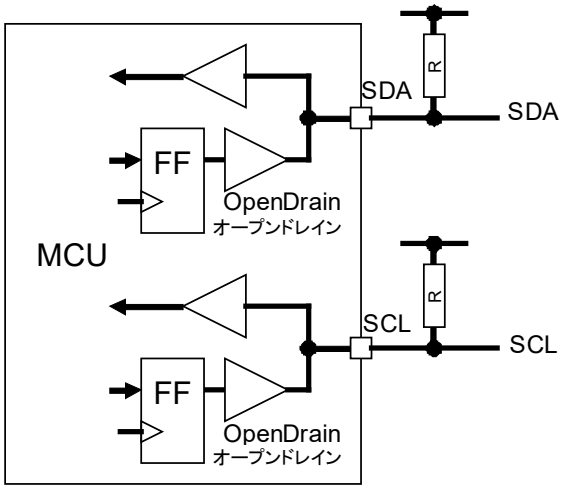


図 8　GPIO 端子を使用

ケース 5:　I²C バス切り替え IC 使用 (複数の D6T センサ接続)

(本センサがスレーブアドレスを変更できないため)

＊1．多くのバス切り替え IC では、電源電圧変換の機能が含まれています。

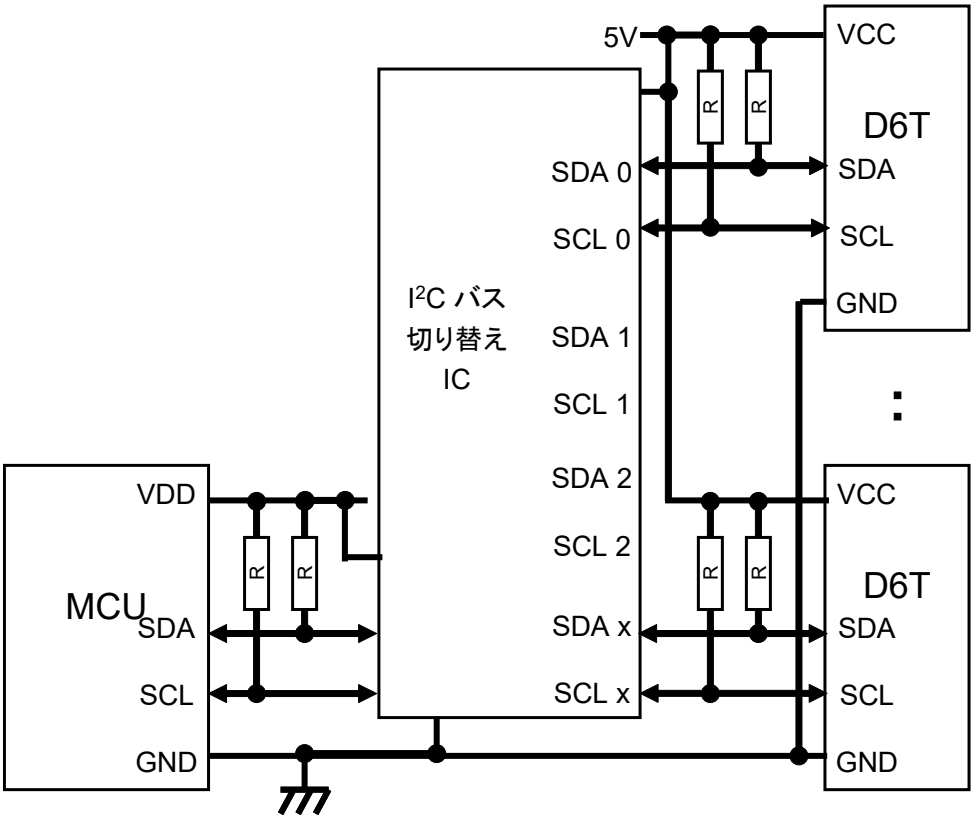


図 9　I²C バス切り替え IC を使用する場合

プルアップ抵抗の値について

プルアップ抵抗の適切な値は、配線の容量成分などのユーザーの使用条件に応じて異なりますので、評価頂いた上で決定してください。

(I²C 仕様をご確認ください。多くの場合、約 3k～10kΩの範囲となります)

4.3 **表面カバー部材**

ＭＥＭＳ非接触温度センサを組み込み設置する際は、カバー部材の放射熱（遠赤外）通過性能に十分にご注意ください。遠赤外線透過グレードの高密度ポリエチレン（HDPE）が、比較的安価で加工しやすいカバー材として良く利用されます。カバーの厚みにより減衰率が異なりますので、出来るだけ薄い方が検出性能への影響を低く抑えられます。ただし、下記写真の様に、薄いカバーでは内部のセンサが透けて見えることになります。

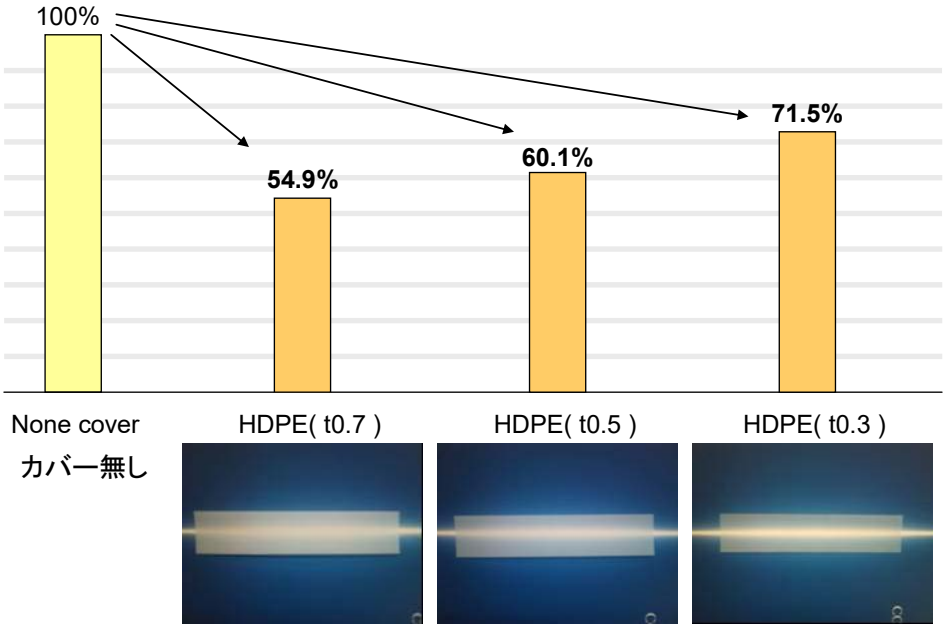
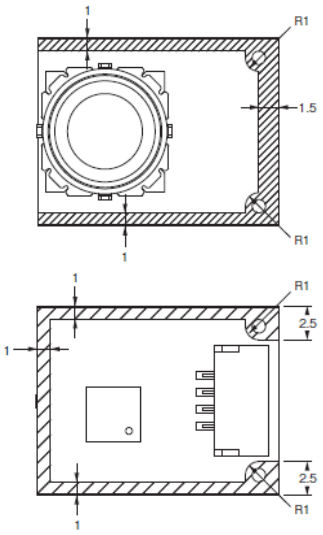


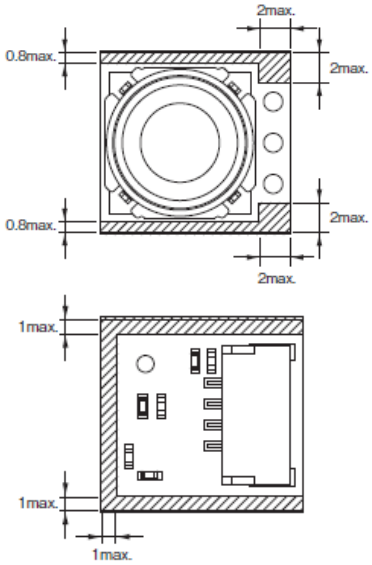
図 10　HDPE 板厚と透過量の関係（参考）

4.4 固定方法

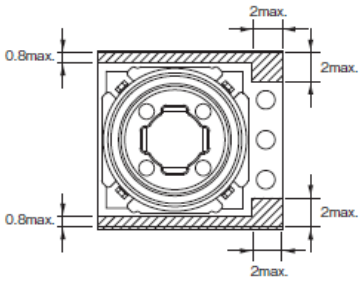
ＭＥＭＳ非接触温度センサの保持・固定可能エリア（斜線部）をケースで挟むなどして固定してください。

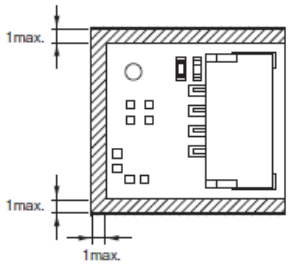


D6T-44L-06 / D6T-44L-06H

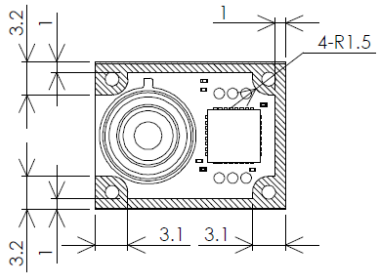


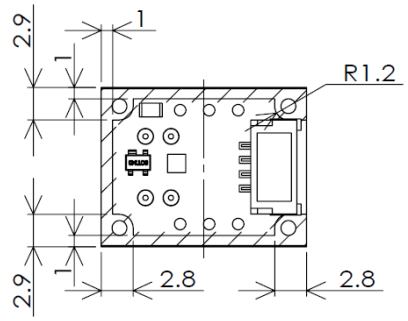
D6T-8L-09 / D6T-8L-09H





D6T-1A-01 / D6T-1A-02





D6T-32L-01A

図 11　保持・固定エリア（斜線部）

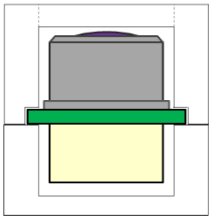


図 12　保持・固定方法（参考）

5. 開発・評価ツール

注 1. 本カタログ掲載の形 2JCIE-EV01-RP1、形 2JCIE-EV01-AR1、形 2JCIE-EV01-FT1、形 2JCIE-HARNESS-01 は、2025 年 1 月末に受注終了

ソフト開発支援ツールとして、2 種類のサンプルコードを用意しています。

1. Raspberry Pi 用　サンプルコード
2. Arduino 用　サンプルコード

Raspberry Pi 用サンプルコードと Arduino 用サンプルコードは、オムロン評価ボードと一緒に使用頂くことで、簡単にハード接続可能です。
オムロン評価ボードは下記の 3 種類のプラットフォームに対応しており、非接触温度センサ D6T、評価ボード、ハーネスをプラットフォームに接続することで、簡単に評価することが可能です。
評価ボード URL: (http://www.omron.co.jp/ecb/sensor/evaluation-board/2jcie)

表 2　評価ボード一覧

プラットフォーム	評価ボード	接続用ハーネス （評価ボード･D6T 間）	サンプルコード
Raspberry Pi *1 接続用	形 2JCIE-EV01-RP1 	形 2JCIE-HARNESS-01	https://github.com/omron-devhub/d6t-2jcieev01-raspberrypi
Arduino *2 接続用	形 2JCIE-EV01-AR1 	形 2JCIE-HARNESS-01	https://github.com/omron-devhub/d6t-2jcieev01-arduino
ESP32 Feather *3 接続用	形 2JCIE-EV01-FT1 	形 2JCIE-HARNESS-01	https://github.com/omron-devhub/d6t-2jcieev01-arduino

＊1. Raspberry Pi は、Raspberry Pi 財団の登録商標です。
＊2. Arduino は、Arduino LLC および Arduino SRL の登録商標です。
＊3. Feather は、Adafruit Industries LLC の登録商標です。

評価ボードを使用頂かなくても、サンプルコードは利用可能です。ただし、お客様でセンサを配線頂く必要があります。

サンプルコードは C 言語で記載されています。お客様が使用したい MCU に合わせて、I^2C 制御の関数を変更することにより、サンプルコードを活用してソフト開発をすることが可能です。

なお、サンプルコードは評価用目的のためのコードであり、オムロンは動作保証しておりません。

1. Raspberry pi 用サンプルコード動作手順

(1) D6T、ハーネス、オムロン評価ボード（2JCIE-EV01-RP1）を Raspberry Pi を接続

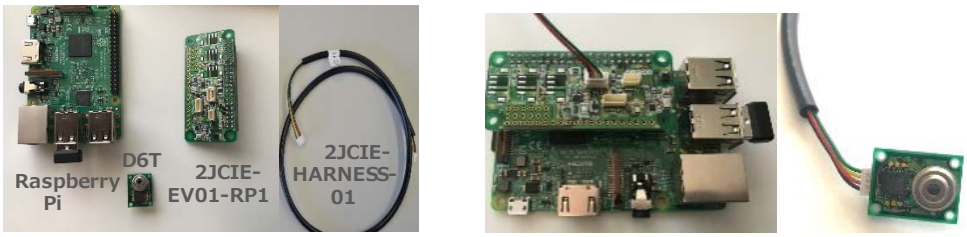


図 13　Set-up

(2) I^2C を有効化

Raspberry Pi を立ち上げ、Start メニューから、"Preferences" > "Raspberry Pi Configuration" を開き、I^2C を"Enable"し、再起動。

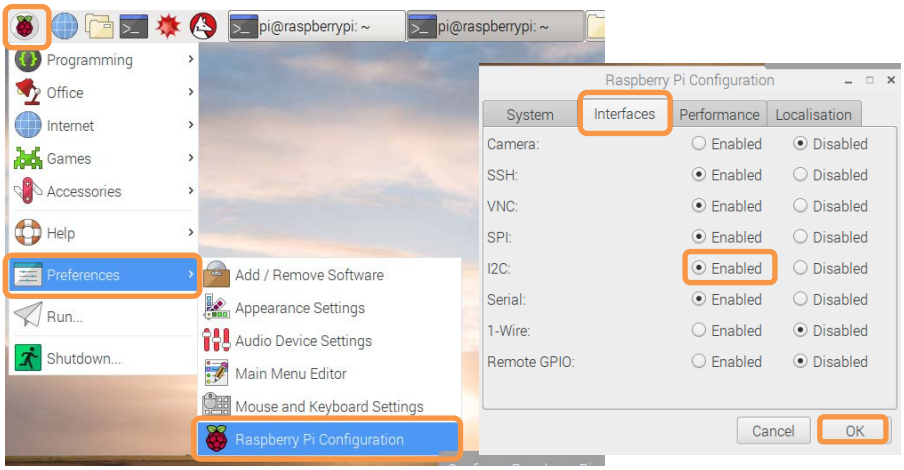


図 14　Enable I^2C

(3) サンプルコードをダウンロード

下記の URL から GitHub にアクセスして、Zip file をダウンロード。

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

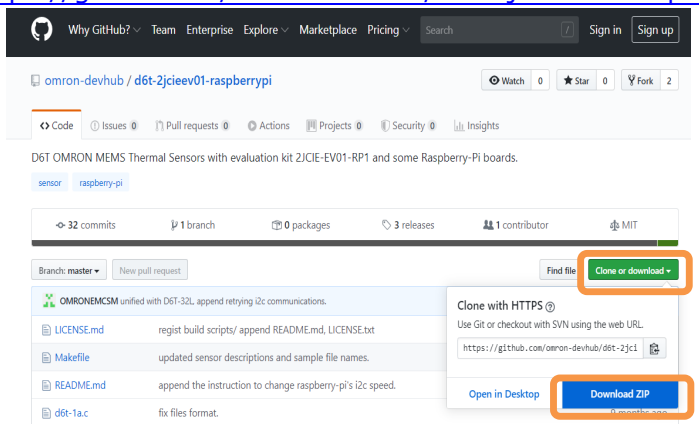


図 15　Download sample code

または、Terminal を開き、下記のコマンドを実行。

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

または、インターネットに接続された他の PC で Zip file をダウンロードした後, USB メモリ等で Raspberry Pi に Zip file を移動させる。

(4) Make file

Terminal を開き、下記のコマンドを実行。

```
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
```

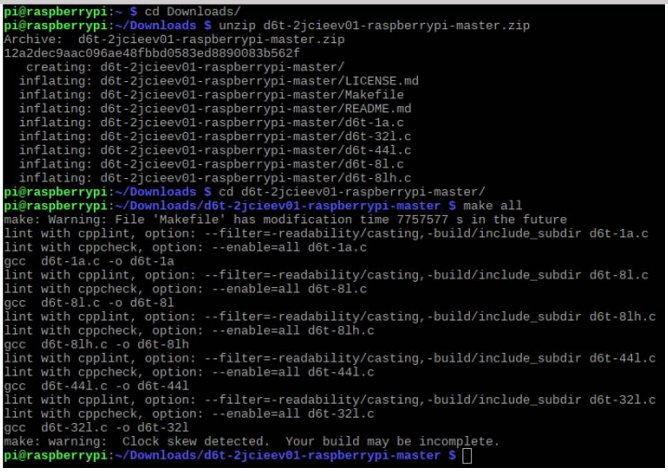


図 16 Make file

(5) Run the file

データ取得するために下記のコマンドを実行。

-- D6T-1A-01 / D6T-1A-02 の場合

```
pi@raspberrypi:~/Downloads/d6t-2jcieev01-raspberrypi-master $ ./d6t-1a
```

-- D6T-8L-09 の場合

```
pi@raspberrypi:~/Downloads/d6t-2jcieev01-raspberrypi-master $ ./d6t-8l
```

-- D6T-8L-09H の場合

```
pi@raspberrypi:~/Downloads/d6t-2jcieev01-raspberrypi-master $ ./d6t-8lh
```

-- D6T-44L-06 / D6T-44L-06H の場合

```
pi@raspberrypi:~/Downloads/d6t-2jcieev01-raspberrypi-master $ ./d6t-44l
```

-- D6T-32L-01A の場合

```
pi@raspberrypi:~/Downloads/d6t-2jcieev01-raspberrypi-master $ ./d6t-32l
```

(データ取得を停止したい場合は、"Ctrl"キーと"C"キーを同時に押してください。)

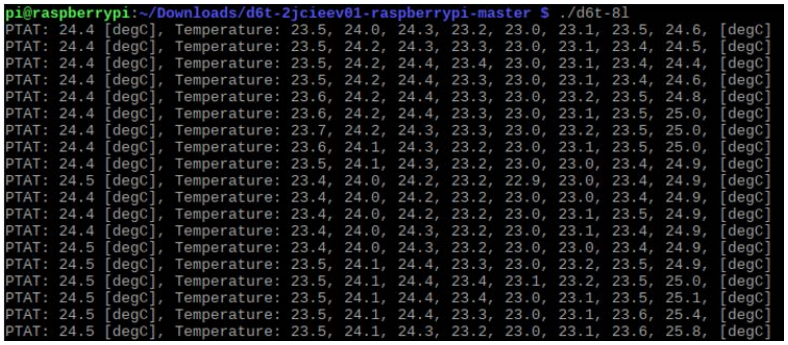


図 17 Run the file

2. Arduino 用サンプルコード動作手順

(1) D6T、ハーネス、オムロン評価ボード（2JCIE-EV01-AR1）を Arduino に接続

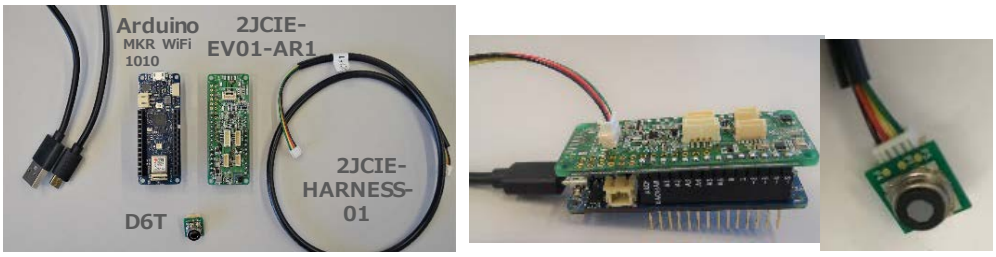


図 18 Set up

(2) Arduino IDE をダウンロード

下記 URL から Arduino IDE をダウンロード。

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

(3) Arduino IDE 上で Arduino を認識

Arduino IDE を開き、Arduino を PC に USB 経由で接続。

下記のような表示が出た場合、Arduino MKR 用の Package をインストール。

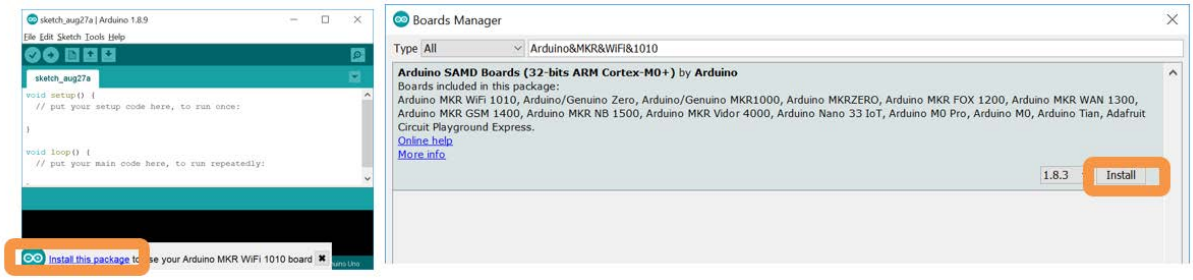


図 19 Arduino IDE

Driver をインストール。

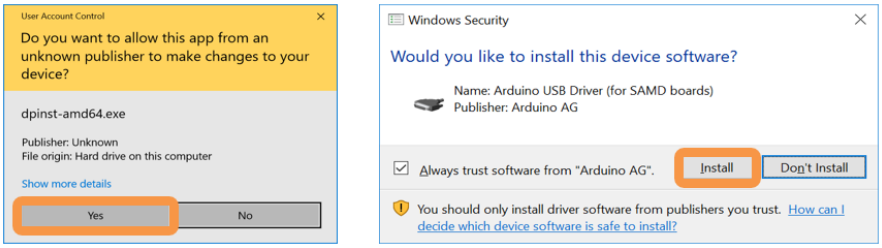


図 20 Driver

Windows のスタートメニューで “Device Manager” と検索。

PC に認識された Arduino の COM ポート番号を確認。

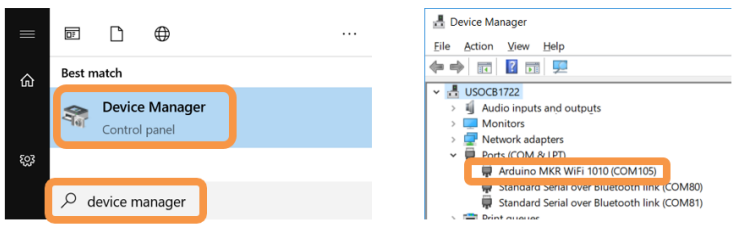


図 21 Device manager

“Tools” -> “Board Arduino” -> “Arduino MKR WiFi1010”.

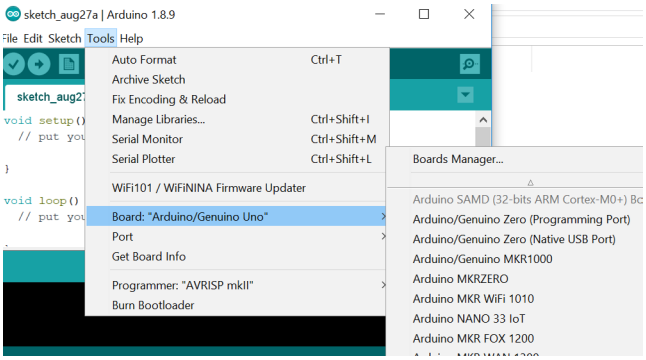


図 22 Select Arduino

“Tools” -> “PORT” -> Arduino の COM ポートを選択.

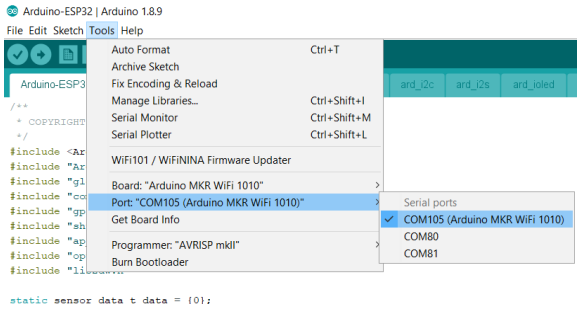


図 23 Select COM port

(4) サンプルコードをダウンロード

下記の GitHub の URL に接続して zip file をダウンロード。

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

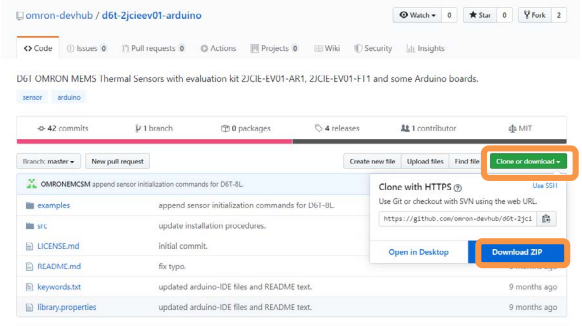


図 24 Download sample code

(5) サンプルコードを Arduino にアップロード。

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

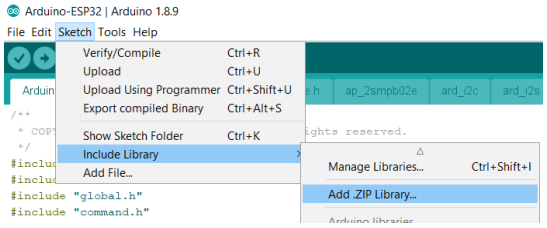


図 25 Add zip

“Downloads” フォルダを選択。“d6t-2jcieev01-arduino-master.zip”を選択。

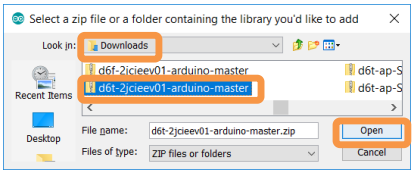


図 26 Select zip file

センサの型式に合わせて、ファイルを選択。

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

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

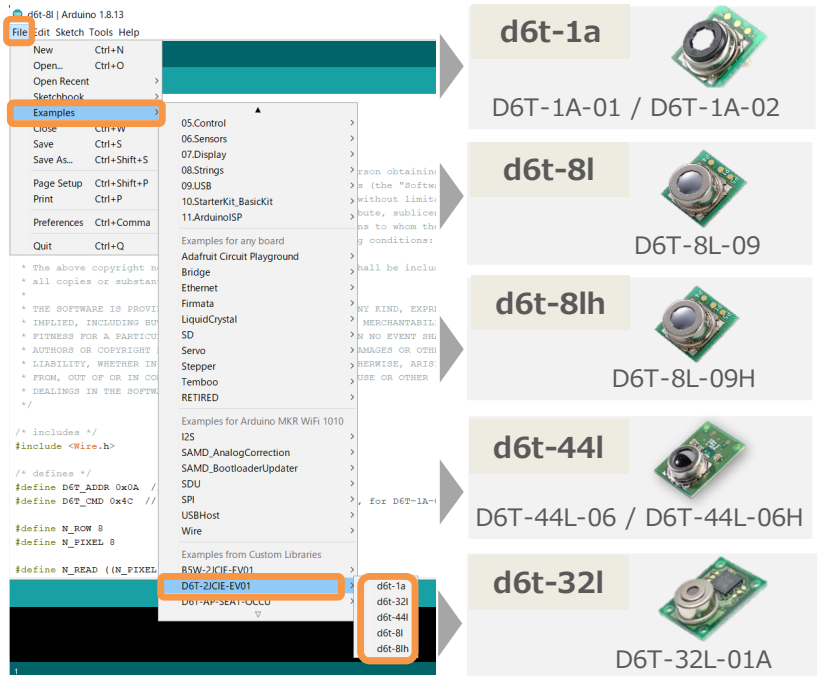


図 27 Select file

“Verify”をクリック。エラーが出ていないか確認。

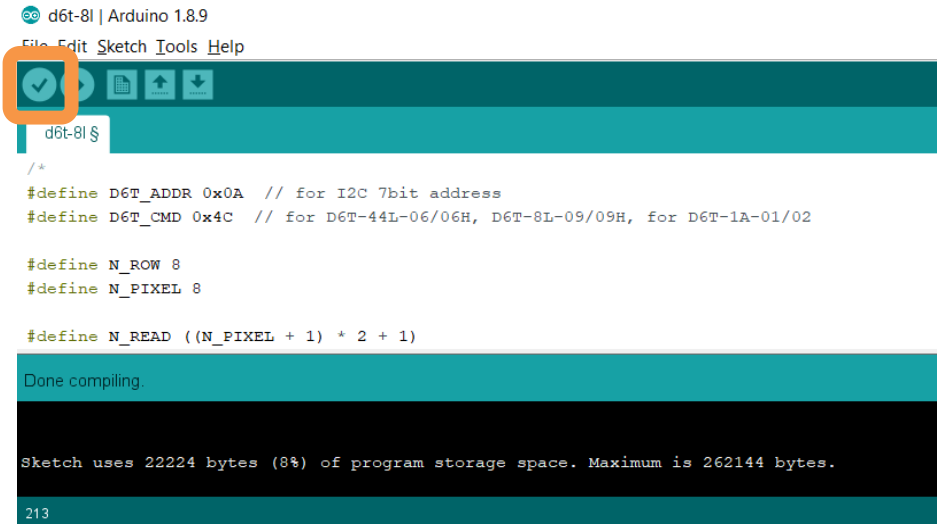


図 28 Verify

“Upload”をクリック。“CPU reset” が表示されるか確認。

図 29 Upload

(6) データ取得

“Tools” -> “Serial Monitor”

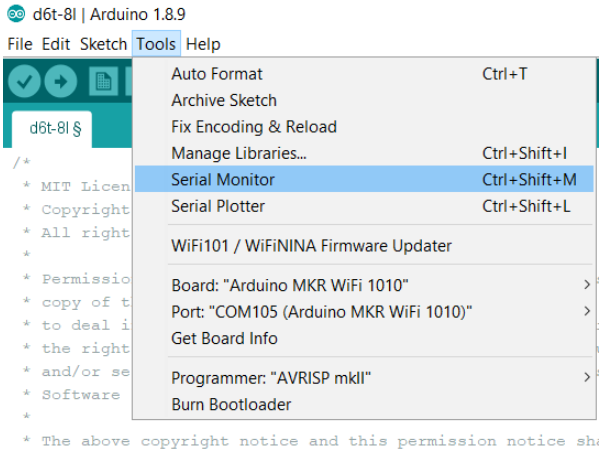


図 30 Serial monitor

データが表示される。

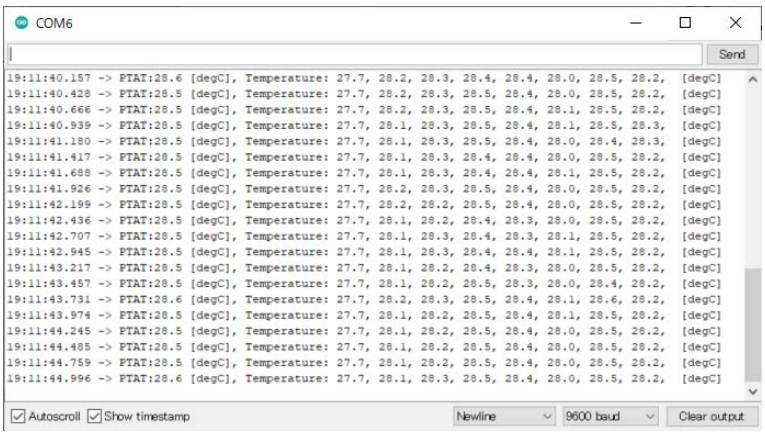


図 31 data

6. よくある質問

質 問　電源投入後、測定可能までの時間はどれくらいか。

回 答　出力温度は電源投入後に数秒で温度精度範囲内に入りますが、完全に安定するまでには 15 分程度かかります。（参考値）

質 問　センサとの通信異常が生じる。または異常値が出力される。考えられる要因はあるか。

回答　下記の観点をご確認ください。

●D6T 共通

・電圧が 5 V 正しく供給できているか。

・プルアップ抵抗が適切な値か。

・センサデータ変換時に、適切に変換できているか。

(例) P0(Low):0x20, P0 (High):0x01 の時、

Data = (int16_t) (((uint16_t)P0(Low)) + (((uint16_t)P0(High))<< 8))

= 0x0120　= 288

●D6T-1A-01 / D6T-1A-02

・100ms 以上の間隔をあけてデータを測定しているか。

●D6T-8L-09

・250ms 以上の間隔をあけてデータを測定しているか。

・電源起動後、初期化のコマンドを送信しているか。

●D6T-8L-09H

・250ms 以上の間隔をあけてデータを測定しているか。

・電源起動後、初期化のコマンドを送信しているか。

・センサのデータを正しく変換できているか。(D6T-8L-09H は温度の 5 倍値で出力。)

例）温度出力値：560　->　560/5 = 112.0 (℃)

●D6T-44L-06 / D6T-44L-06H

・ご使用の MCU の I^2C クロックストレッチを有効にしているか。

・300ms 以上の間隔をあけてデータを測定しているか。

●D6T-32L-01A

・ご使用の MCU の I^2C クロックストレッチを有効にしているか。

・I^2C の Read 関数が十分なデータバッファ（2051 byte 分）を持っているか。

・200ms 以上の間隔をあけてデータを測定しているか。

質 問　センサと通信はできているが、温度が高い（または低い）。または温度が安定していない。考えられる要因はあるか？

回答　下記の要因が考えられます。

・物体が FOV と比べて小さいため、背景の温度も含めた温度になっている。

・視野に対象物以外のものが入っている。

・温度が安定しない物体を測定している。

・気温が変化している。

・お客様の筐体が影響している。（視野に筐体が入っている。筐体内部が発熱している。）

質 問　消費電力をさらに小さくできないか。

回 答　D6T シリーズには、「省電力スリープのための動作モード」の設定はありませんので、消費電力を抑えるためには電源を遮断する必要があります。

質 問	電源電圧 3[V]駆動やスレーブアドレスの変更はできないか。
回 答	D6T シリーズにはそれらの機能はありません。
質 問	もっと視野角を拡げることが出来ないか。
回 答	シリコンレンズの厚みと屈折率による制約を考慮して、設定した FOV になっています。 １素子あたりの FOV が大きくなると測定検出能力が低下することも、単純に視野角を広げられない理由のひとつです。広い範囲を測定するためには、センサを可動させるか、複数個設置する必要があります。
質 問	人と動物、電化製品などとの区別がつけられるか。
回 答	非接触温度センサでは、測定温度のデータ取得のみになります。その測定データの振る舞いを元に、ユーザー側のソフトウェアにより対象物の区別を行う必要があります。ご使用状況を考慮して判定ソフトウェアを開発することで、判定精度を向上出来る可能性があります。
質 問	人感センサとして検出可能な距離はどれぐらいか。
回 答	設置状況および判定アルゴリズムの性能に大きく影響を受けます。 サーモパイルセンサ 1 素子あたりの FOV 面積と測定対象物体のサイズによりますが、おおまかな距離目安としては、5～6[m]程度となります。
質 問	赤外線リモコンによる誤動作は起きないか。
回 答	使用しているシリコンレンズは、波長 1.2[μm]以下の可視光～近赤外線をほとんど透過しないため、リモコンの赤外線による誤動作は起きません。放射熱として発せられている遠赤外線はおよそ 4～14[μm]になります。

7. 用語の説明

- サーモパイル
  熱電対（サーモカップル）を縦続接続して起電力を増やしたもの。熱電対列。
  各熱電対の温接点を隣接位置に並ぶ様に配置している

- NETD (カタログに記載)
  Noise Equivalent Temperature Difference の略。ノイズ量を測定温度に換算したもの。
  測定温度の変化を判定できる最小値の目安で、温度分解能と呼ばれる場合もある。

- FOV
  Field of View の略。視野角の指標。感度ピーク 50%で規定することが多い。

8. ご承諾事項

平素はオムロン株式会社（以下「当社」）の商品をご愛用いただき誠にありがとうございます。
「当社商品」のご購入について特別の合意がない場合には、お客様のご購入先にかかわらず、本ご承諾事項記載の条件を適用いたします。ご承諾のうえご注文ください。

1. 定義

本ご承諾事項中の用語の定義は次のとおりです。

(1)「当社商品」：「当社」のＦＡ システム機器、汎用制御機器、センシング機器、電子・機構部品
(2)「カタログ等」：「当社商品」に関する、ベスト制御機器オムロン、電子・機構部品総合カタログ、その他のカタログ、仕様書、取扱説明書、マニュアル等であって電磁的方法で提供されるものも含みます。
(3)「利用条件等」：「カタログ等」に記載の、「当社商品」の利用条件、定格、性能、動作環境、取り扱い方法、利用上の注意、禁止事項その他
(4)「お客様用途」：「当社商品」のお客様におけるご利用方法であって、お客様が製造する部品、電子基板、機器、設備またはシステム等への「当社商品」の組み込み又は利用を含みます
(5)「適合性等」：「お客様用途」での「当社商品」の(a) 適合性、(b) 動作、(c) 第三者の知的財産の非侵害、(d) 法令の遵守および(e) 各種規格の遵守

2. 記載事項のご注意

「カタログ等」の記載内容については次の点をご理解ください。

(1) 定格値および性能値は、単独試験における各条件のもとで得られた値であり、各定格値および性能値の複合条件のもとで得られる値を保証するものではありません。
(2) 参考データはご参考として提供するもので、その範囲で常に正常に動作することを保証するものではありません。
(3) 利用事例はご参考ですので、「当社」は「適合性等」について保証いたしかねます。
(4)「当社」は、改善や当社都合等により、「当社商品」の生産を中止し、または「当社商品」の仕様を変更することがあります。

3. ご利用にあたってのご注意

ご採用およびご利用に際しては次の点をご理解ください。

(1) 定格・性能ほか「利用条件等」を遵守しご利用ください。
(2) お客様ご自身にて「適合性等」をご確認いただき、「当社商品」のご利用の可否をご判断ください。「当社」は「適合性等」を一切保証いたしかねます。
(3)「当社商品」がお客様のシステム全体の中で意図した用途に対して、適切に配電・設置されていることをお客様ご自身で、必ず事前に確認してください。
(4)「当社商品」をご使用の際には、(i) 定格および性能に対し余裕のある「当社商品」のご利用、冗長設計などの安全設計、(ii)「当社商品」が故障しても、「お客様用途」の危険を最小にする安全設計、(iii) 利用者に危険を知らせるための、安全対策のシステム全体としての構築、(iv)「当社商品」および「お客様用途」の定期的な保守、の各事項を実施してください。
(5)「当社」は DDoS 攻撃（分散型 DoS 攻撃）、コンピュータウイルスその他の技術的な有害プログラム、不正アクセスにより、「当社商品」、インストールされたソフトウェア、またはすべてのコンピュータ機器、コンピュータプログラム、ネットワーク、データベースが感染したとしても、そのことにより直接または間接的に生じた損失、損害その他の費用について一切責任を負わないものとします。お客様ご自身にて、①アンチウイルス保護、②データ入

出力、③紛失データの復元、④「当社商品」またはインストールされたソフトウェアに対するコンピュータウイルス感染防止、⑤「当社商品」に対する不正アクセス防止についての十分な措置を講じてください。

(6) 「当社商品」は、一般工業製品向けの汎用品として設計製造されています。従いまして、次に掲げる用途での使用は意図しておらず、お客様が「当社商品」をこれらの用途に使用される際には、「当社」は「当社商品」に対して一切保証をいたしません。ただし、次に掲げる用途であっても、「当社」の意図した特別な商品用途の場合や特別の合意がある場合は除きます。

(a) 高い安全性が必要とされる用途（例：原子力制御設備、燃焼設備、航空・宇宙設備、鉄道設備、昇降設備、娯楽設備、医用機器、安全装置、その他生命・身体に危険が及びうる用途）

(b) 高い信頼性が必要な用途（例：ガス・水道・電気等の供給システム、24 時間連続運転システム、決済システムほか権利・財産を取扱う用途など）

(c) 厳しい条件または環境での用途（例：屋外に設置する設備、化学的汚染を被る設備、電磁的妨害を被る設備、振動・衝撃を受ける設備など）

(d) 「カタログ等」に記載のない条件や環境での用途

(7) 上記 3.(6) (a)から(d)に記載されている他、「本カタログ等記載の商品」は自動車（二輪車含む。以下同じ）向けではありません。自動車に搭載する用途には利用しないでください。自動車搭載用商品については当社営業担当者にご相談ください。

4. 保証条件

「当社商品」の保証条件は次のとおりです。

(1) 保証期間　ご購入後 1 年間といたします。
（ただし「カタログ等」に別途記載がある場合を除きます。）

(2) 保証内容　故障した「当社商品」について、以下のいずれかを「当社」の任意の判断で実施します。

(a) 当社保守サービス拠点における故障した「当社商品」の無償修理
（ただし、電子・機構部品については、修理対応は行いません。）

(b) 故障した「当社商品」と同数の代替品の無償提供

(3) 保証対象外　故障の原因が次のいずれかに該当する場合は、保証いたしません。

(a) 「当社商品」本来の使い方以外のご利用

(b) 「利用条件等」から外れたご利用

(c) 本ご承諾事項「3. ご利用にあたってのご注意」に反するご利用

(d) 「当社」以外による改造、修理による場合

(e) 「当社」以外の者によるソフトウェアプログラムによる場合

(f) 「当社」からの出荷時の科学・技術の水準では予見できなかった原因

(g) 上記のほか「当社」または「当社商品」以外の原因（天災等の不可抗力を含む）

5. 責任の制限

本ご承諾事項に記載の保証が、「当社商品」に関する保証のすべてです。

「当社商品」に関連して生じた損害について、「当社」および「当社商品」の販売店は責任を負いません。

6. 輸出管理

「当社商品」または技術資料を、輸出または非居住者に提供する場合は、安全保障貿易管理に関する日本および関係各国の法令・規制を遵守ください。お客様が法令・規則に違反する場合には、「当社商品」または技術資料をご提供できない場合があります。

D6T 通信仕様

D6T-1A-01

D6T-1A-02

D6T-8L-09

D6T-8L-09H

D6T-44L-06

D6T-44L-06H

D6T-32L-01A

通信仕様

D6T-1A-01

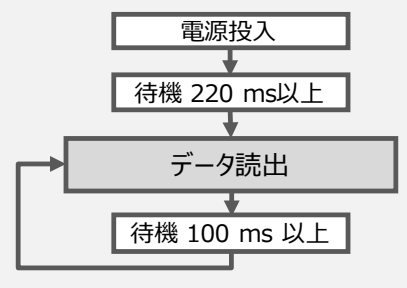
D6T-1A-02

(1) I^2C インターフェイス概要

スレーブアドレス	7bit (0001_010b) 8bit (with R/W bit)表現： Read：15h , Write：14h
データビット幅	8bit (MSB-first)
クロックスピード	max 100kHz
クロックストレッチ対応	非対応

(2) 通信手順

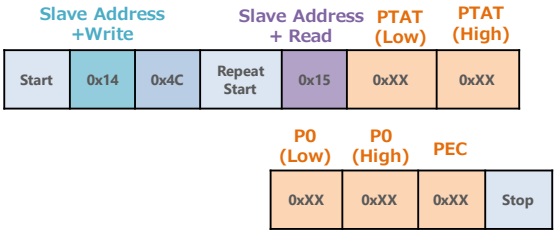
センサとの通信手順は右記の通りです。本センサは、標準仕様品の場合、毎 100ms 以内に測定データを更新しています。これは通信と無関係に繰り返しており、測定タイミングを外部から制御できません。



<u>データ読 出</u>

下記のコマンドを送信し、データ取得を実行してください。

受信データは 5byte あります。



［受信データ項目］

・PTAT:

センサ内部の参照温度データ。16bit 符号つき整数(2 の補数)で、10 倍温度値[℃]。

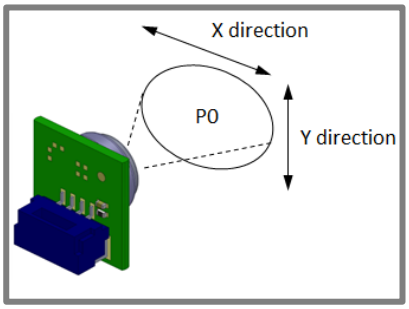
（計算例）

PTAT(Low):0x20, PTAT(High):0x01 の時、

Int16 で 0x0120 -> 288 -> 288/10 = 28.8 (℃)

・P0:

各画素の温度データ。配置は図参照。16bit 符号つき整数(2 の補数)で、10 倍温度値[℃]



（計算例）

P0(Low):0x87, P0(High):0xFE の時、

Int16 で 0xFE87 -> -121 -> -121/10 = -12.1 (℃)

・PEC:

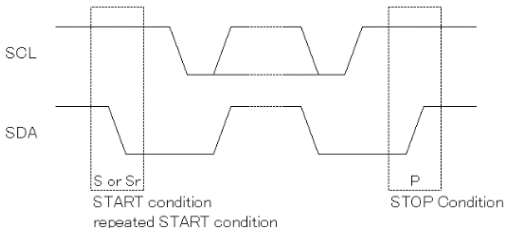
Packet error check code. PEC は CRC-8 方式を用いたエラーチェック用のデータで、通信出力の最後に付加されています。PEC の対象範囲は、Slave Address+ Write から P0(High)のデータまでで、各 1byte の値を用いて PEC 値を作成します。この PEC 値を利用して、ユーザーが通信障害を検出し、データの信頼性を向上させることができます。

(詳細については SMBus 仕様を参照ください)

(3) I²C アクセスプロトコル

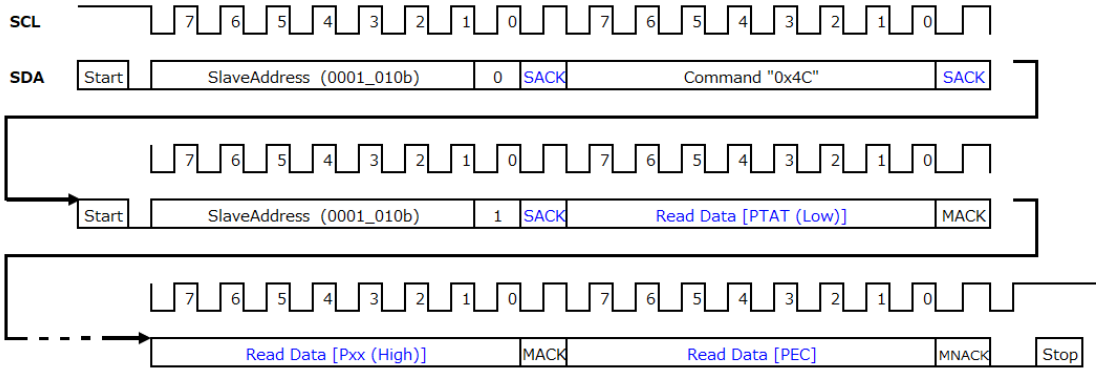
I²C アクセスプロトコルの各用語は下記の通りです。

“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 リードアクセスプロトコル

まず write モードにて、Command を送信してください。次に、Repeat Start Condition にしてから read モードに切替えて、データを読み出してください。



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

本センサは SDA もしくは SCL が下記の時間 Low 入力が継続すると、通信タイムアウトとしてストップ状態にします。

・D6T-1A-01 / D6T-1A-02 : 70msec

なお、センサが通信タイムアウトと判断した場合は、Write アクセス時には NACK を返しますが、Read アクセス時は読み出し値が FFFFh となります。PEC を用いたデータチェックを実施すると読み出し値が異常と判断できますので、PEC でのデータチェックを推奨します。

(4) サンプルコード

Raspberry Pi 用のサンプルコードは下記の URL にあります。

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

Arduino 用のサンプルコードは下記の URL にあります。

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

サンプルコードの構成は下記の通りです。

```
/** <!-- 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);
        }
}
```

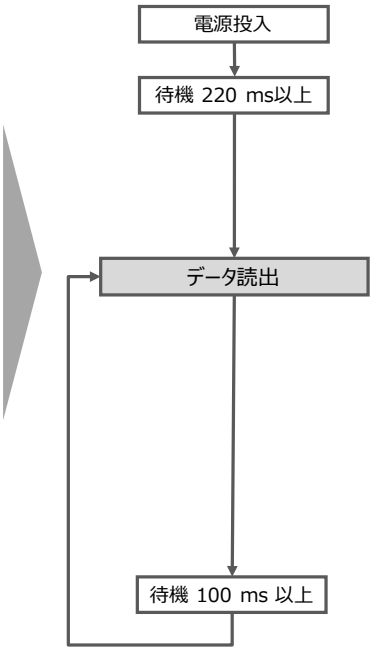
I²C Read

PEC確認

PTAT に変換

温度値に変換

画面に表示



通信仕様

D6T-8L-09

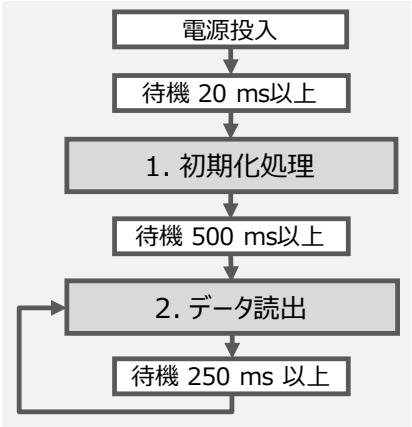
(1) I^2C インターフェイス概要

スレーブアドレス	7bit (0001_010b) 8bit (with R/W bit)表現： Read：15h , Write：14h
データビット幅	8bit (MSB-first)
クロックスピード	max 100kHz
クロックストレッチ対応	非対応

(2) 通信手順

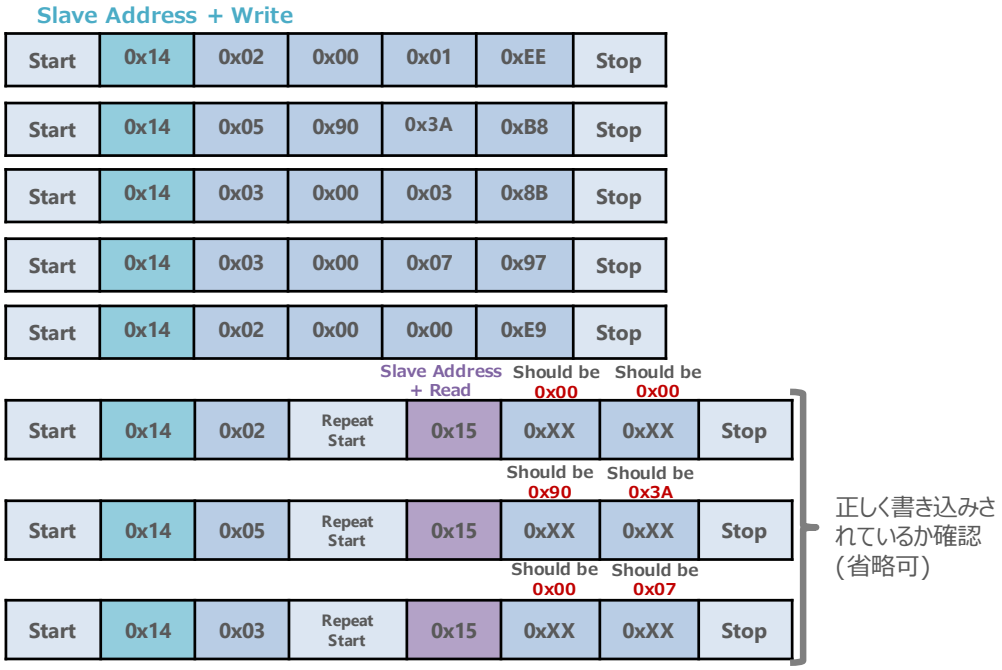
センサとの通信手順は下記の通りです。

本センサは、標準仕様品の場合、毎 250ms 以内に測定データを更新しています。これは通信と無関係に繰り返しており、測定タイミングを外部から制御できません。



1.初期化処理

下記のように、コマンドを送信・受信してください。



2.データ読 出

下記のコマンドを送信し、データ取得を実行してください。受信データは 19byte あります。

	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	0xXX	Stop

［受信データ項目］

・PTAT:

センサ内部の参照温度データ。16bit 符号つき整数(2 の補数)で、10 倍温度値[℃]。

（計算例）

PTAT(Low):0x20, PTAT(High):0x01 の時、

Int16 で 0x0120 -> 288 -> 288/10 = 28.8 (℃)

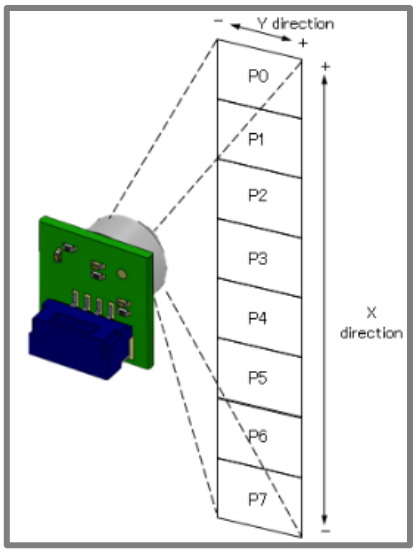
・P0 ～ P7:

各画素の温度データ。配置は図参照。16bit 符号つき整数(2 の補数)で、10 倍温度値[℃]

（計算例）

P0(Low):0x87, P0(High):0xFE の時、

Int16 で 0xFE87 -> -121 -> -121/10 = -12.1 (℃)



・PEC:

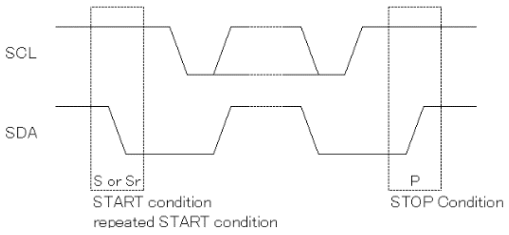
Packet error check code. PEC は CRC-8 方式を用いたエラーチェック用のデータで、通信出力の最後に付加されています。PEC の対象範囲は、Slave Address+ Write から P7(High)のデータまでで、各 1byte の値を用いて PEC 値を作成します。この PEC 値を利用して、ユーザーが通信障害を検出し、データの信頼性を向上させることができます。

(詳細については SMBus 仕様を参照ください)

(3) I²C アクセスプロトコル

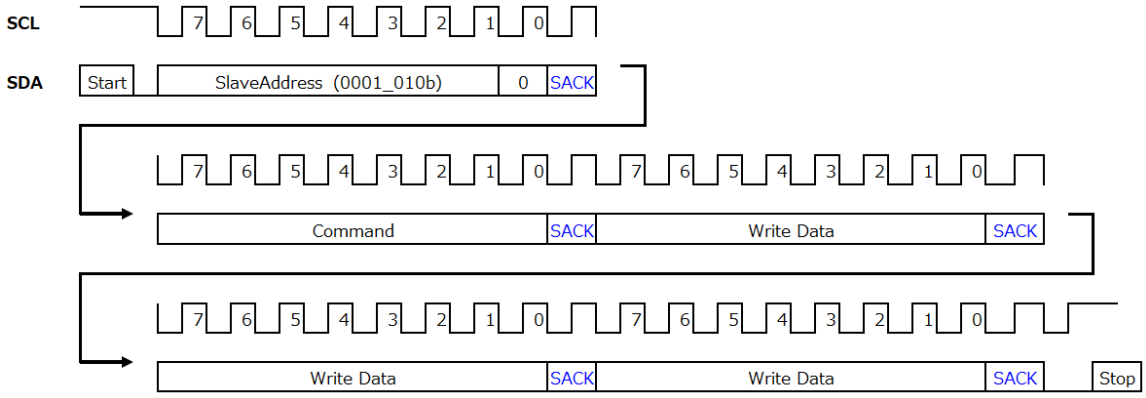
I²C アクセスプロトコルの各用語は下記の通りです。

“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 ライトアクセスプロトコル

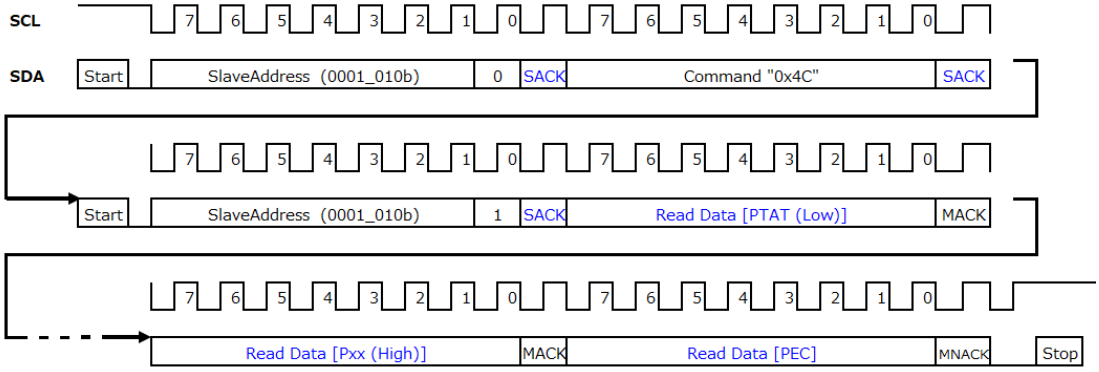
Start Condition 後、Slave Address (bit7～1)に書込み信号(=”0”at bit0)を加えたデータを送信し、write モードに設定してください。その後、Write データを送信しつづけ、Stop Condition にしてください。



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

I²C リードアクセスプロトコル

まず write モードにて、Command を送信してください。次に、Repeat Start Condition にしてから read モードに切替えて、データを読み出してください。



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

本センサは SDA もしくは SCL が下記の時間 Low 入力が継続すると、通信タイムアウトとしてストップ状態にします。

・D6T-8L-09： 70msec

なお、センサが通信タイムアウトと判断した場合は、Write アクセス時には NACK を返しますが、Read アクセス時は読み出し値が FFFFh となります。PEC を用いたデータチェックを実施すると読み出し値が異常と判断できますので、PEC でのデータチェックを推奨します。

(4) サンプルコード

Raspberry Pi 用のサンプルコードは下記の URL にあります。

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

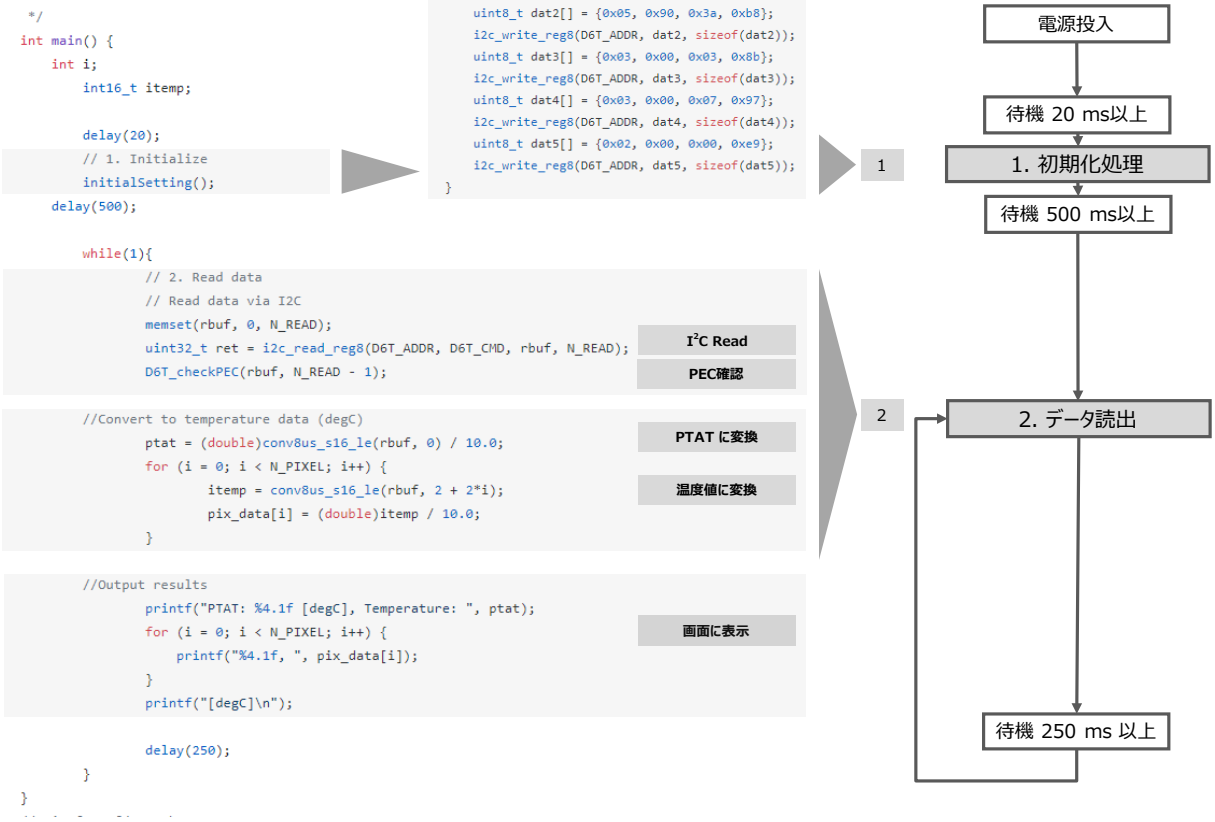
Arduino 用のサンプルコードは下記の URL にあります。

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

サンプルコードの構成は下記の通りです。

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

        delay(20);
        // 1. Initialize
        initialSetting();
    delay(500);

        while(1){
                // 2. Read data
                // 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(250);
        }
}
```

```
void initialSetting(void) {
    uint8_t dat1[] = {0x02, 0x00, 0x01, 0xee};
    i2c_write_reg8(D6T_ADDR, dat1, sizeof(dat1));
    uint8_t dat2[] = {0x05, 0x90, 0x3a, 0xb8};
    i2c_write_reg8(D6T_ADDR, dat2, sizeof(dat2));
    uint8_t dat3[] = {0x03, 0x00, 0x03, 0x8b};
    i2c_write_reg8(D6T_ADDR, dat3, sizeof(dat3));
    uint8_t dat4[] = {0x03, 0x00, 0x07, 0x97};
    i2c_write_reg8(D6T_ADDR, dat4, sizeof(dat4));
    uint8_t dat5[] = {0x02, 0x00, 0x00, 0xe9};
    i2c_write_reg8(D6T_ADDR, dat5, sizeof(dat5));
}
```



通信仕様

D6T-8L-09H

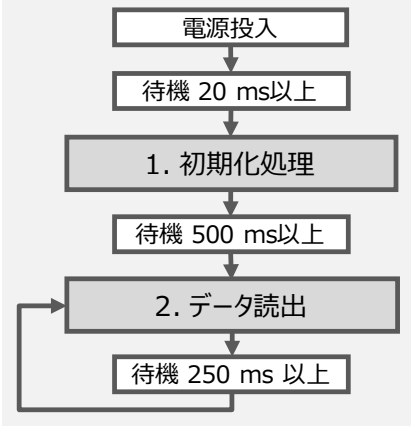
(1) I^2C インターフェイス概要

スレーブアドレス	7bit (0001_010b) 8bit (with R/W bit)表現： Read：15h , Write：14h
データビット幅	8bit (MSB-first)
クロックスピード	max 100kHz
クロックストレッチ対応	非対応

(2) 通信手順

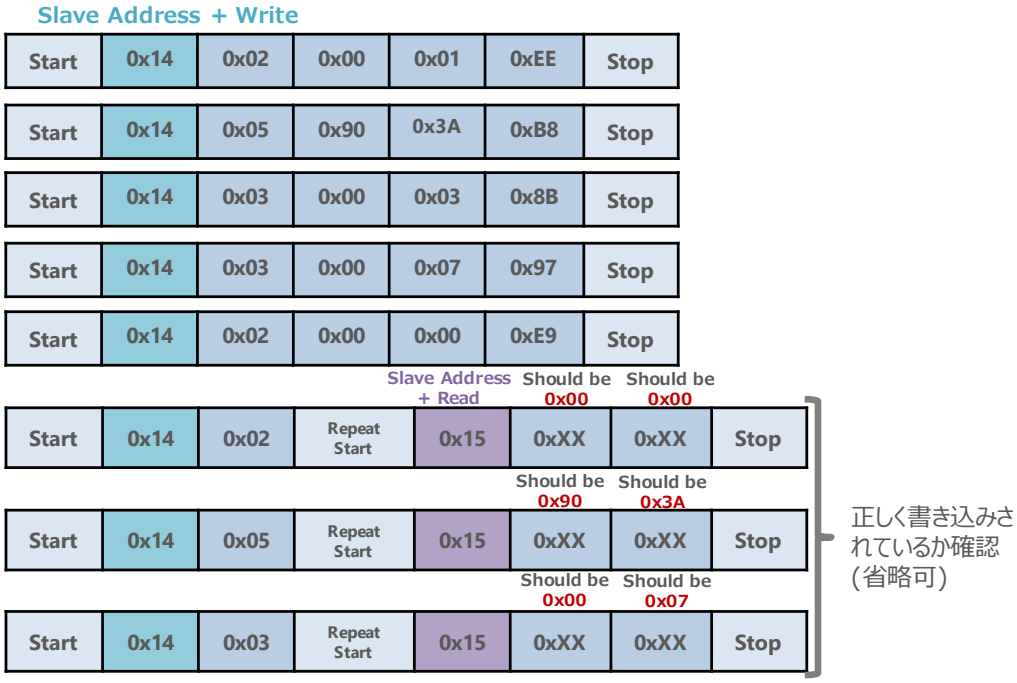
センサとの通信手順は下記の通りです。

本センサは、標準仕様品の場合、毎 250ms 以内に測定データを更新しています。これは通信と無関係に繰り返しており、測定タイミングを外部から制御できません。



1.初期化処理

下記のように、コマンドを送信・受信してください。



2.データ読 出

下記のコマンドを送信し、データ取得を実行してください。受信データは 19byte あります。

	Slave Address +Write			Slave Address + Read		
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	0xXX	Stop

［受信データ項目］

・PTAT:

センサ内部の参照温度データ。16bit 符号つき整数(2 の補数)で、10 倍の温度値[℃]。

（計算例）

PTAT(Low):0x20, PTAT(High):0x01 の時、

Int16 で 0x0120 -> 288 -> 288/10 = 28.8 (℃)

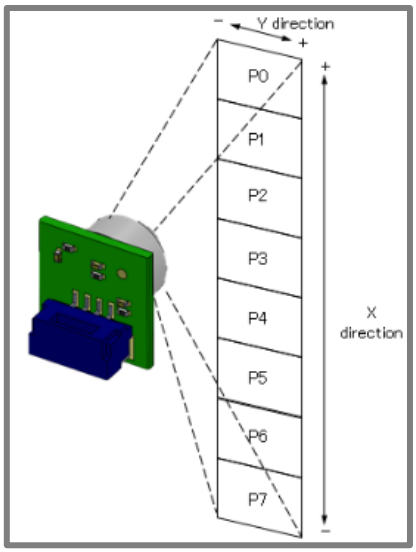
・P0 ～ P7:

各画素の温度データ。配置は図参照。16bit 符号つき整数(2 の補数)で、5 倍の温度値[℃]

（計算例）

P0(Low):0x30, P0(High):0x02 の時、

Int16 で 0x0230 -> 560 -> 560/5 = 112.0 (℃)



・PEC:

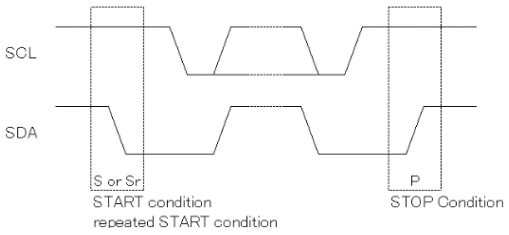
Packet error check code. PEC は CRC-8 方式を用いたエラーチェック用のデータで、通信出力の最後に付加されています。PEC の対象範囲は、Slave Address+ Write から P7(High)のデータまでで、各 1byte の値を用いて PEC 値を作成します。この PEC 値を利用して、ユーザーが通信障害を検出し、データの信頼性を向上させることができます。

(詳細については SMBus 仕様を参照ください)

(3) I²C アクセスプロトコル

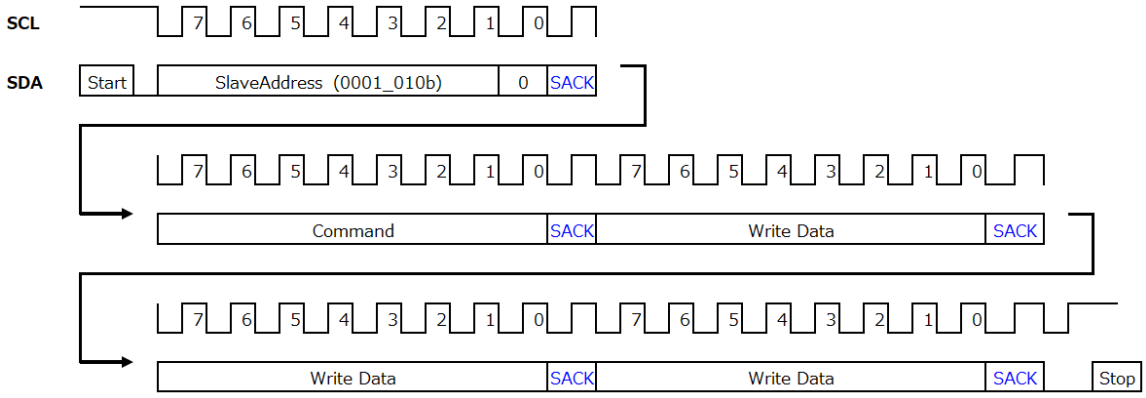
I²C アクセスプロトコルの各用語は下記の通りです。

“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 ライトアクセスプロトコル

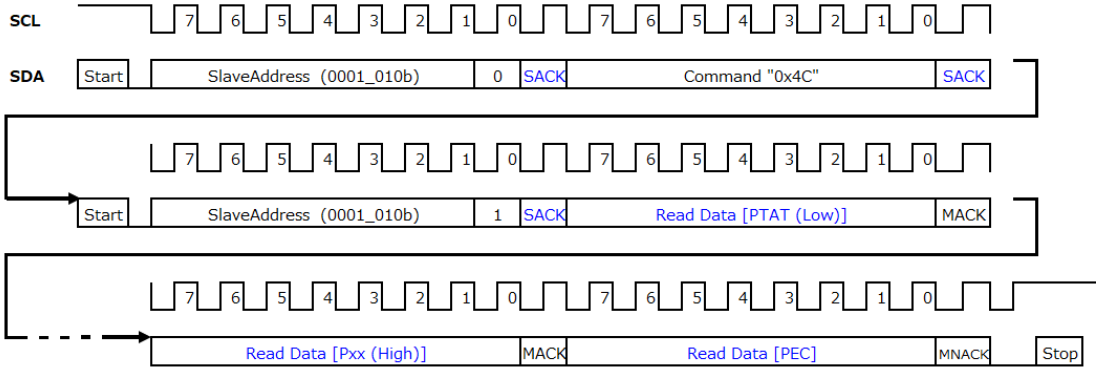
Start Condition 後、Slave Address (bit7～1)に書込み信号(=”0”at bit0)を加えたデータを送信し、write モードに設定してください。その後、Write データを送信しつづけ、Stop Condition にしてください。



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

I²C リードアクセスプロトコル

まず write モードにて、Command を送信してください。次に、Repeat Start Condition にしてから read モードに切替えて、データを読み出してください。



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

本センサは SDA もしくは SCL が下記の時間 Low 入力が継続すると、通信タイムアウトとしてストップ状態にします。

・D6T-8L-09H： 70msec

なお、センサが通信タイムアウトと判断した場合は、Write アクセス時には NACK を返しますが、Read アクセス時は読み出し値が FFFFh となります。PEC を用いたデータチェックを実施すると読み出し値が異常と判断できますので、PEC でのデータチェックを推奨します。

(4) サンプルコード

Raspberry Pi 用のサンプルコードは下記の URL にあります。

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

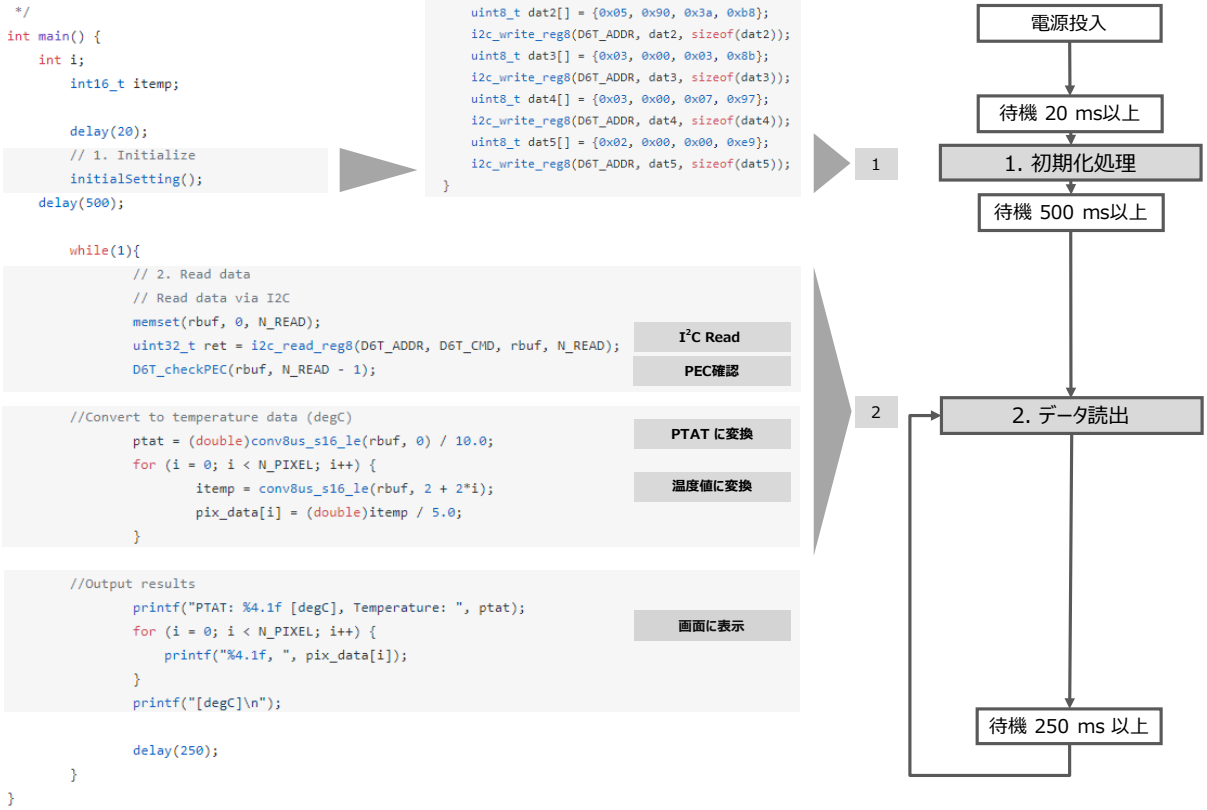
Arduino 用のサンプルコードは下記の URL にあります。

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

サンプルコードの構成は下記の通りです。

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

        delay(20);
        // 1. Initialize
        initialSetting();
    delay(500);

        while(1){
                // 2. Read data
                // 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 / 5.0;
                }

        //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(250);
        }
}
```

```
void initialSetting(void) {
    uint8_t dat1[] = {0x02, 0x00, 0x01, 0xee};
    i2c_write_reg8(D6T_ADDR, dat1, sizeof(dat1));
    uint8_t dat2[] = {0x05, 0x90, 0x3a, 0xb8};
    i2c_write_reg8(D6T_ADDR, dat2, sizeof(dat2));
    uint8_t dat3[] = {0x03, 0x00, 0x03, 0x8b};
    i2c_write_reg8(D6T_ADDR, dat3, sizeof(dat3));
    uint8_t dat4[] = {0x03, 0x00, 0x07, 0x97};
    i2c_write_reg8(D6T_ADDR, dat4, sizeof(dat4));
    uint8_t dat5[] = {0x02, 0x00, 0x00, 0xe9};
    i2c_write_reg8(D6T_ADDR, dat5, sizeof(dat5));
}
```



通信仕様

D6T-44L-06

D6T-44L-06H

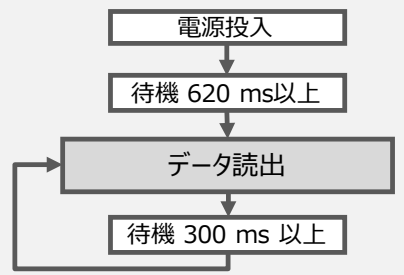
(1) I^2C インターフェイス概要

スレーブアドレス	7bit (0001_010b) 8bit (with R/W bit)表現： Read：15h , Write：14h
データビット幅	8bit (MSB-first)
クロックスピード	max 100kHz
クロックストレッチ対応	対応

(2) 通信手順

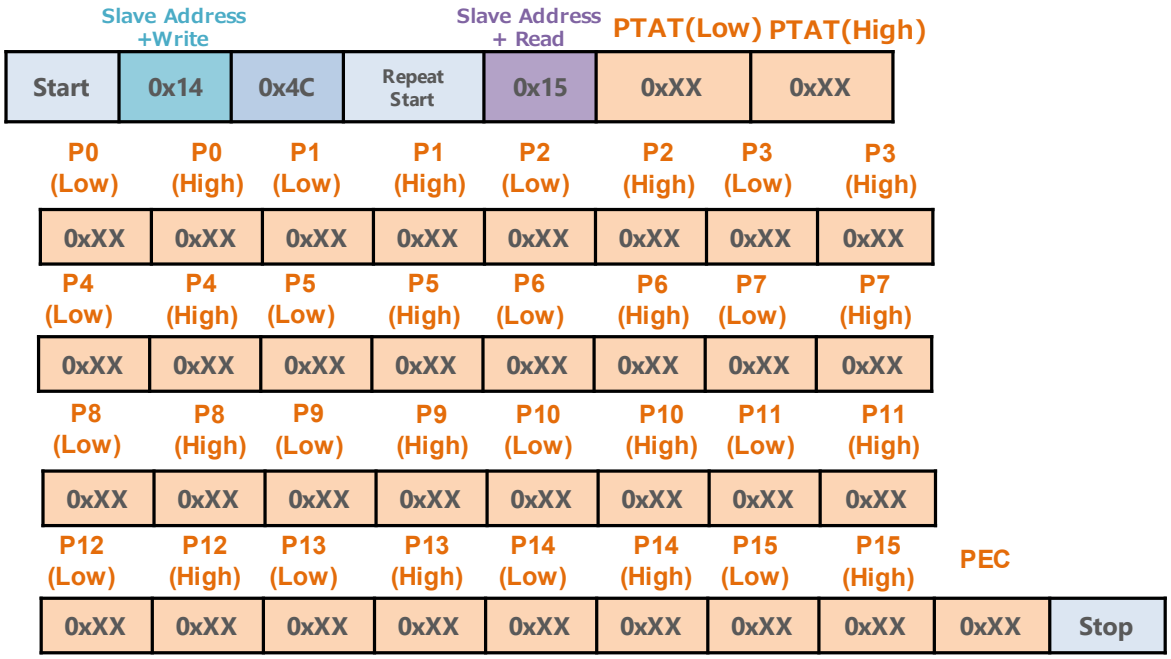
センサとの通信手順は下記の通りです。

本センサは、標準仕様品の場合、毎 300 ms 以内に測定データを更新しています。これは通信と無関係に繰り返しており、測定タイミングを外部から制御できません。



データ読 出

下記のコマンドを送信し、データ取得を実行してください。受信データは 35byte あります。



［受信データ項目］

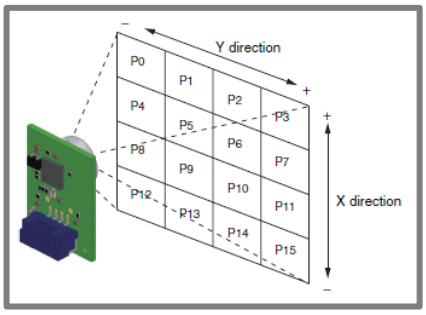
・PTAT:

センサ内部の参照温度データ。16bit 符号つき整数(2 の補数)で、10 倍の温度値[℃]。

（計算例）

PTAT(Low):0x20, PTAT(High):0x01 の時、

Int16 で 0x0120 -> 288 -> 288/10 = 28.8 (℃)



・P0 ～ P15:

各画素の温度データ。配置は図参照。16bit 符号つき整数(2 の補数)で、10 倍の温度値[℃]

（計算例）

P0(Low):0x87, P0(High):0xFE の時、

Int16 で 0xFE87 -> -121 -> -121/10 = -12.1 (℃)

・PEC:

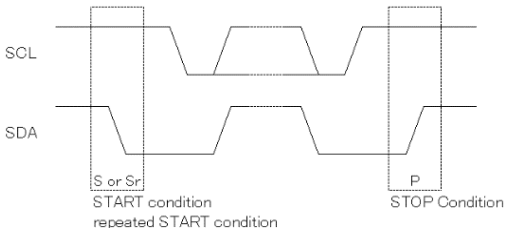
Packet error check code. PEC は CRC-8 方式を用いたエラーチェック用のデータで、通信出力の最後に付加されています。PEC の対象範囲は、Slave Address+ Write から P15(High)のデータまでで、各 1byte の値を用いて PEC 値を作成します。この PEC 値を利用して、ユーザーが通信障害を検出し、データの信頼性を向上させることができます。

(詳細については SMBus 仕様を参照ください)

(3) I²C アクセスプロトコル

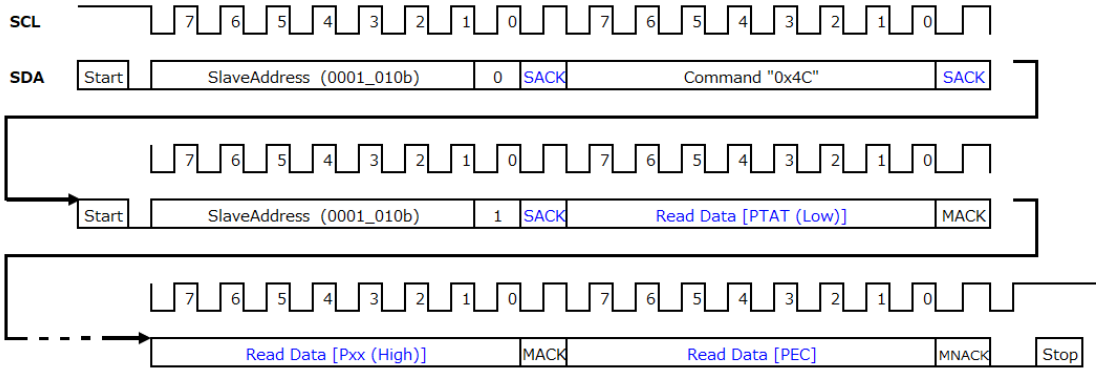
I²C アクセスプロトコルの各用語は下記の通りです。

“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 リードアクセスプロトコル

まず write モードにて、Command を送信してください。次に、Repeat Start Condition にしてから read モードに切替えて、データを読み出してください。



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

本センサは SDA もしくは SCL が下記の時間 Low 入力が継続すると、通信タイムアウトとしてストップ状態にします。

・D6T-44L-06 / D6T-44L-06H : 1sec

なお、センサが通信タイムアウトと判断した場合は、Write アクセス時には NACK を返しますが、Read アクセス時は読み出し値が FFFFh となります。PEC を用いたデータチェックを実施すると読み出し値が異常と判断できますので、PEC でのデータチェックを推奨します。

(4) サンプルコード

Raspberry Pi 用のサンプルコードは下記の URL にあります。

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

Arduino 用のサンプルコードは下記の URL にあります。

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

サンプルコードの構成は下記の通りです。

```
/** <!-- 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);

        //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(300);
        }
}
```

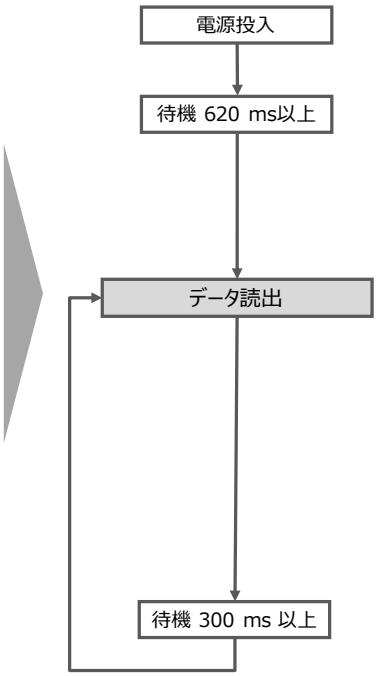
I²C Read

PEC確認

PTAT に変換

温度値に変換

画面に表示

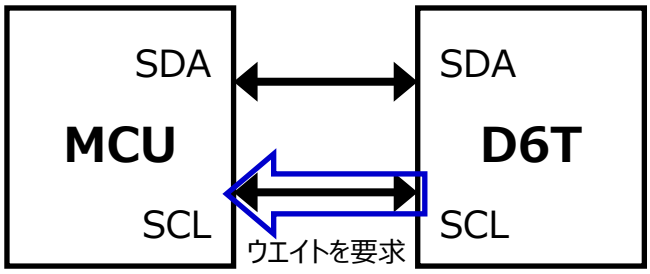


（5）クロックストレッチ

クロックストレッチとは、I²C スレーブの処理が間に合わない時にスレーブ側が SCL の LOW 期間を延長してマスタ側を待機させる事ができる機能です。スレーブ側である D6T がこの機能を持つため、マスタ側の MCU もこの機能に対応する必要があります。

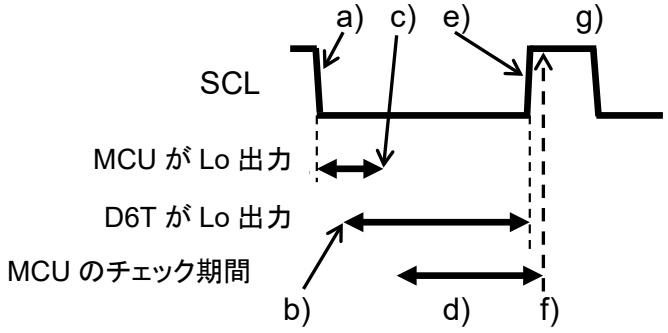
多くの MCU に内蔵されている I²C モジュールには、自動対応する機能があります。お使いの MCU の仕様に合わせて、クロックストレッチを有効にしてお使いください。

内臓 I²C モジュールのない MCU など、ソフトウェアによる I²C ライブラリを使用される場合は、ウエイト対応機能の有無を確認の上で御使用ください。機能が無い場合は、下記に示す様なウエイト検出ルーチンを、SCL 出力部分に追加する必要があります。



ウエイト検出ルーチン

I²C マスタ	I²C スレーブ（センサ）
a)　SCL に Lo 出力（毎 Ack タイミング）	SCL 端子の Lo 検知チェック
（固定ウエイト） **c)　SCL 出力を Hi-Z に変更** **SCL 端子を入力モードに変更** **d)　SCL 端子を Hi になるかチェック** **チェック待機（LOOP）**	b)　SCL に Lo 出力（ウエイト要求） ウエイト中 … : :
	ウエイト完了 e)　SCL 出力を Hi-Z に変更.
f)　チェック完（Hi 検知） **SCL 端子を出力モードに変更** g)　次の処理へ移る	



ウエイト検出ルーチンの設定が難しい場合は、毎 Ack タイミングに 160μsec のウエイトタイムをプログラムに追加してください。

通信仕様

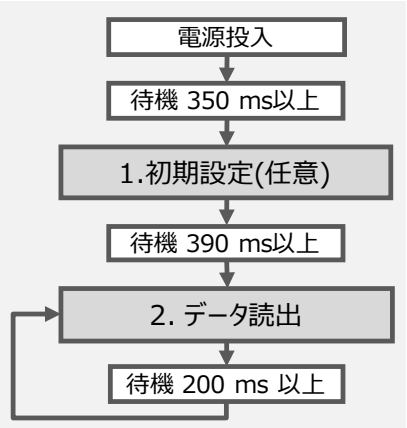
D6T-32L-01A

(1) I²C インターフェイス概要

スレーブアドレス	7bit (0001_010b) 8bit (with R/W bit)表現： Read：15h , Write：14h
データビット幅	8bit (MSB-first)
クロックスピード	max 1000kHz (Fast-Mode Plus)
クロックストレッチ対応	対応

(2) 通信手順

センサとの通信手順は下記の通りです。本センサは、標準仕様品の場合、毎 200ms 以内に測定データを更新しています。これは通信と無関係に繰り返しており、測定タイミングを外部から制御できません。



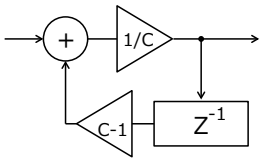
1.初期設定(任意)

下記のように、コマンドを送信してください。設定変更しない場合は、初期設定の送信は不要です。設定変更しない場合は、 Default の設定で動作します。初期設定を実施する場合は、電源投入後、350ms 経過後に実施してください。一度変更した設定も電源を OFF すると、設定は消去され、Default の設定となります。

下記がレジスタの詳細です。

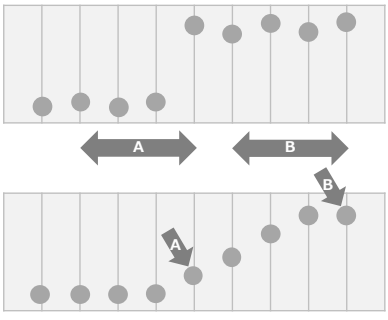
Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0x01	IIR[3]	IIR[2]	IIR[1]	IIR[0]	Avg[3]	Avg[2]	Avg[1]	Avg[0]
Default	0	0	0	0	0	1	0	0

IIR[3：0]： IIR フィルタ設定。 IIR フィルタの係数設定は、
0（スルー）、1～15 が設定可能です。
設定値の 4 倍を C とすると、
Y＝（（C-1）× Yold ＋ X ）／C となります。
IIR フィルタの係数が大きいほどノイズは小さくなりますが、
Response time は遅くなります。



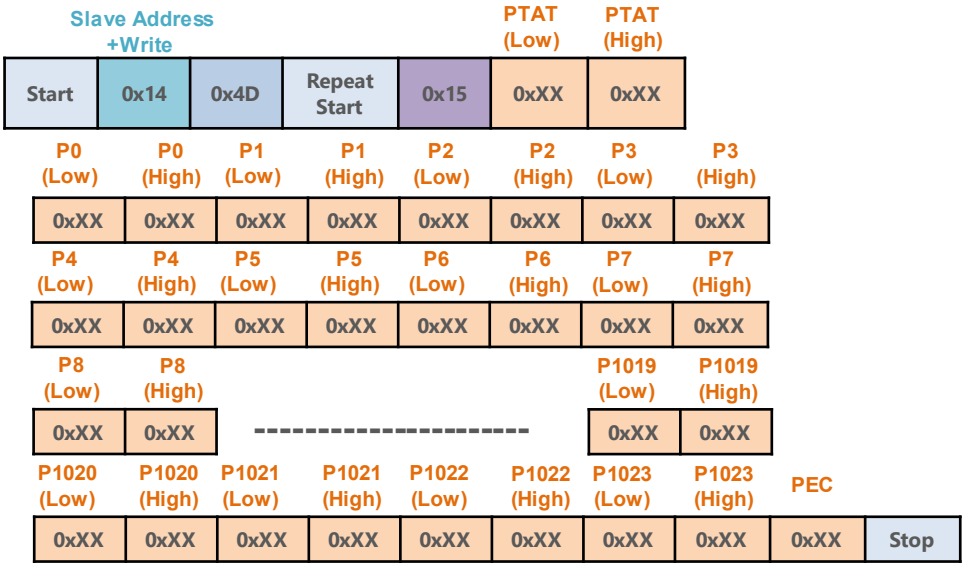
Avg[3：0]： 移動平均設定。移動平均は、
0～10 で設定可能です。 0 は 1 と同じです。
Default 値は 4 です。
移動平均回数が大きいほどノイズは小さく
なりますが、Response time は遅くなります。

右図は移動平均の例です。移動平均設定
が 4 の時、過去 4 回分のデータを平均した値と
なります。



2.データ読出

下記のコマンドを送信し、データ取得を実行してください。受信データは 2051byte あります。



［受信データ項目］

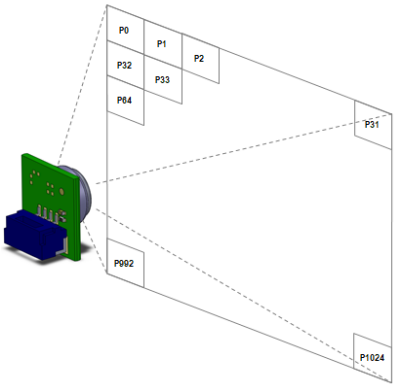
・PTAT:

センサ内部の参照温度データ。16bit 符号つき整数(2 の補数)で、10 倍温度値[℃]。

（計算例）

PTAT(Low):0x20, PTAT(High):0x01 の時、

Int16 で 0x0120 -> 288 -> 288/10 = 28.8 (℃)



・P0 ～ P1023:

各画素の温度データ。配置は図参照。16bit 符号つき整数(2 の補数)で、10 倍温度値[℃]

（計算例）

P0(Low):0x87, P0(High):0xFE の時、

Int16 で 0xFE87 -> -121 -> -121/10 = -12.1 (℃)

・PEC:

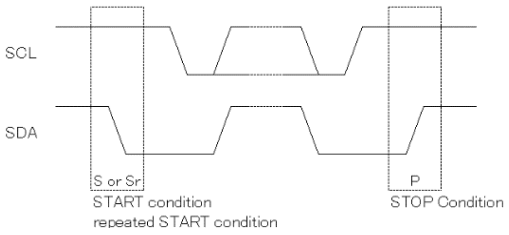
Packet error check code. PEC は CRC-8 方式を用いたエラーチェック用のデータで、通信出力の最後に付加されています。PEC の対象範囲は、PTAT(Low)から P1023(High)のデータまでで、各 1byte の値を用いて PEC 値を作成します。この PEC 値を利用して、ユーザーが通信障害を検出し、データの信頼性を向上させることができます。

(詳細については SMBus 仕様を参照ください)

(3) I²C アクセスプロトコル

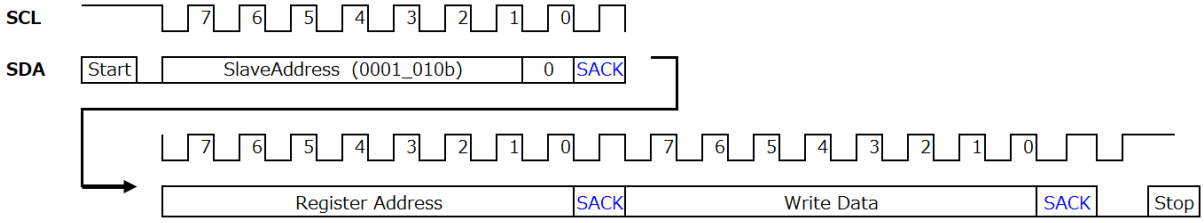
I²C アクセスプロトコルの各用語は下記の通りです。

“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 ライトアクセスプロトコル

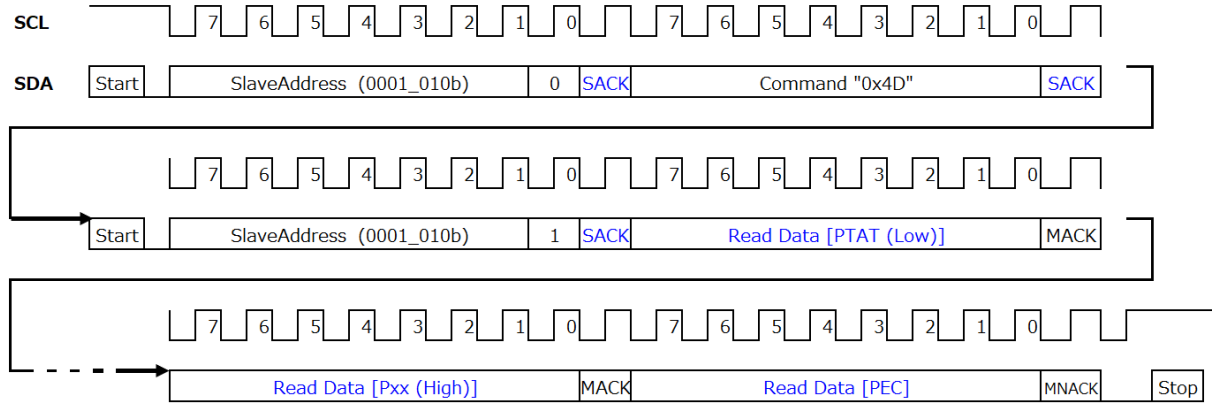
Start Condition 後、Slave Address (bit7～1)に書込み信号(=”0”at bit0)を加えたデータを送信し、write モードに設定してください。その後、Write データを送信し、Stop Condition にしてください。



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

I²C リードアクセスプロトコル

まず write モードにて、Command を送信してください。次に、Repeat Start Condition にしてから read モードに切替えて、データを読み出してください。



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

本センサは SDA もしくは SCL が下記の時間 Low 入力が継続すると、通信タイムアウトとしてストップ状態にします。

・D6T-32L-01A： 1sec

なお、センサが通信タイムアウトと判断した場合は、Write アクセス時には NACK を返しますが、Read アクセス時は読み出し値が FFFFh となります。PEC を用いたデータチェックを実施すると読み出し値が異常と判断できますので、PEC でのデータチェックを推奨します。

(4) サンプルコード

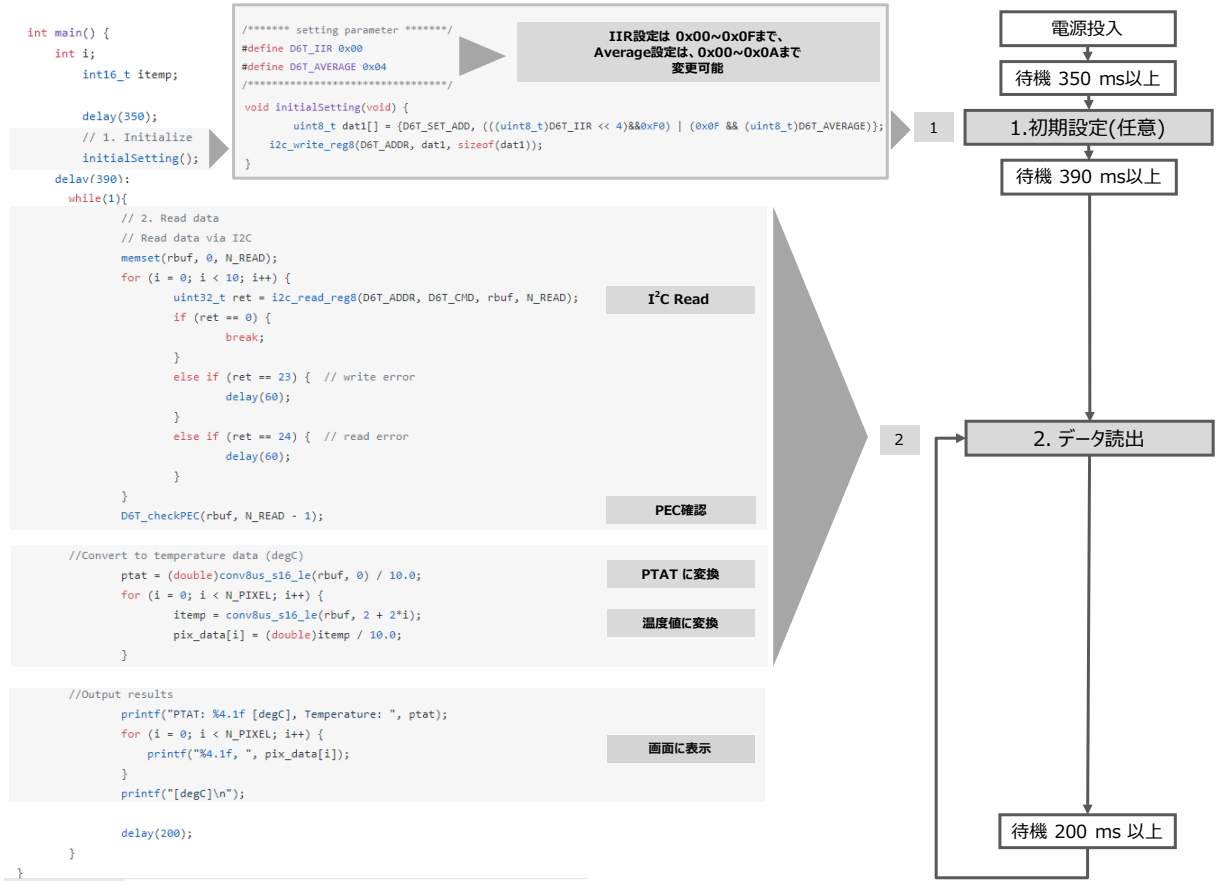
Raspberry Pi 用のサンプルコードは下記の URL にあります。

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

Arduino 用のサンプルコードは下記の URL にあります。

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

サンプルコードの構成は下記の通りです。

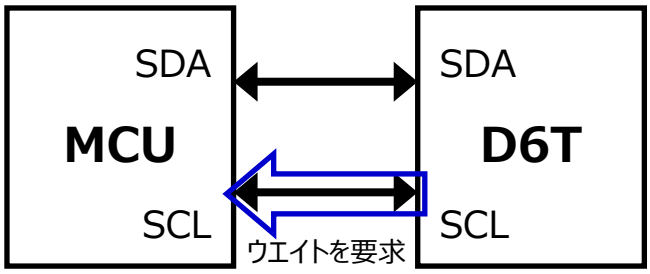


(5) クロックストレッチ

クロックストレッチとは、I²C スレーブの処理が間に合わない時にスレーブ側が SCL の LOW 期間を延長してマスタ側を待機させる事ができる機能です。スレーブ側である D6T がこの機能を持つため、マスタ側の MCU もこの機能に対応する必要があります。

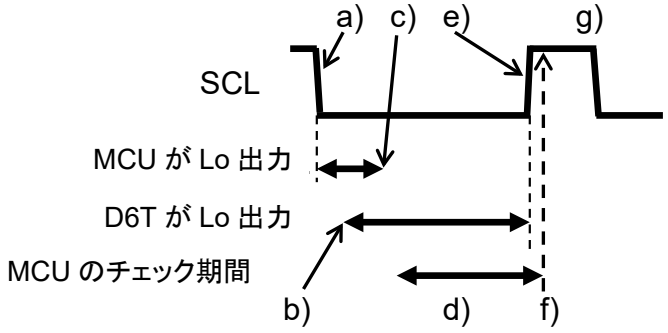
多くの MCU に内蔵されている I²C モジュールには、自動対応する機能があります。お使いの MCU の仕様に合わせて、クロックストレッチを有効にしてお使いください。

内臓 I²C モジュールのない MCU など、ソフトウェアによる I²C ライブラリを使用される場合は、ウエイト対応機能の有無を確認の上で御使用ください。機能が無い場合は、下記に示す様なウエイト検出ルーチンを、SCL 出力部分に追加する必要があります。



ウエイト検出ルーチン

I²C マスタ	I²C スレーブ（センサ）
a) SCL に Lo 出力（毎 Ack タイミング）	SCL 端子の Lo 検知チェック
（固定ウエイト） **c) SCL 出力を Hi-Z に変更** **SCL 端子を入力モードに変更** **d) SCL 端子を Hi になるかチェック** **チェック待機（LOOP）**	b) SCL に Lo 出力（ウエイト要求） ウエイト中 … : :
	ウエイト完了 e) SCL 出力を Hi-Z に変更.
f) チェック完（Hi 検知） **SCL 端子を出力モードに変更** g) 次の処理へ移る	



ウエイト検出ルーチンの設定が難しい場合は、毎 Ack タイミングに 160μsec のウエイトタイムをプログラムに追加してください。

オムロン株式会社 インダストリアルオートメーションビジネスカンパニー

製品に関するお問い合わせ先

お客様相談室

フリー通話 **0120-919-066**（クイック　オムロン）

携帯電話の場合、

055-982-5015（有料）をご利用ください。

受付時間：9:00～17:00（土・日・12/31～1/3を除く）

オムロンFAクイックチャット

www.fa.omron.co.jp/contact/tech/chat/

技術相談員にチャットでお問い合わせいただけます。（I-Webメンバーズ限定）

受付時間：平日9:00～12:00 / 13:00～17:00（土日祝日・年末年始・当社休業日を除く）

※受付時間、営業日は変更の可能性がございます。最新情報はリンク先をご確認ください。

その他のお問い合わせ：納期・価格・サンプル・仕様書は貴社のお取引先、または貴社担当オムロン販売員にご相談ください。オムロン制御機器販売店やオムロン販売拠点は、Webページでご案内しています。

オムロン制御機器の最新情報をご覧いただけます。緊急時のご購入にもご利用ください。 **www.fa.omron.co.jp**

本誌には主に機種のご選定に必要な内容を掲載しており、ご使用上の注意事項等を掲載していない製品も含まれています。
本誌に注意事項等の掲載のない製品につきましては、ユーザーズマニュアル掲載のご使用上の注意事項等、ご使用の際に必要な内容を必ずお読みください。

●本誌に記載の商品の価格は、お取引き商社にお問い合わせください。
●ご注文の際には下記URLに掲載の「ご承諾事項」を必ずお読みください。
適合用途の条件、保証内容などご注文に際してのご承諾事項をご説明しております。
https://components.omron.com/jp-ja/sales_terms-and-conditions

オムロン商品のご用命は

Man. No. CDSC-019F　2024年4月現在