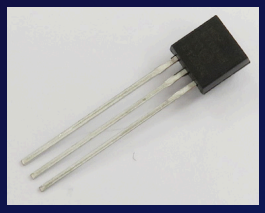
組込システム　I
–9–

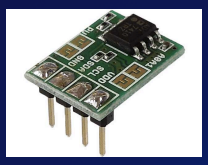
説明・演習
1. 温度センサ
2. 加速度センサ

センサ

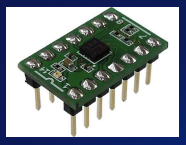
1. アナログ温度センサ（MCP9700A）
   - 測定温度範囲：-40～125℃（精度：±2℃）
   - 出力電圧：500mV（0℃）
   - 温度係数：10.0mV/℃



2. 13/16bit温度センサモジュール（ADT7410）
   - I2Cシリアルインタフェース



3. 3軸加速度センサモジュール（LIS3DH）
   - SPIシリアルインタフェース



シリアル通信準備

- Raspberry Piのシリアルインタフェース設定
  - 1線通信（1wire）
  - 2線通信（I²C / UART）
  - 3線通信（SPI：Serial Peripheral Interface）

➡ 設定 > Raspberry Pi の設定

センサ ①

- アナログ温度センサ　MCP9700A
  - 電源電圧 2.3V～5.5V（3.3V使用）
  - 測定温度範囲：-40～125℃
  - 精度：±2℃
  - 出力電圧：500mV（0℃時）
  - 温度係数：10.0mV /℃

➡ 例：

  - 0℃ → 500mV
  - 20℃ → 500mV + （10.0 x 20）mV

- MCP3002でAD変換して値を読み込む

センサ ②

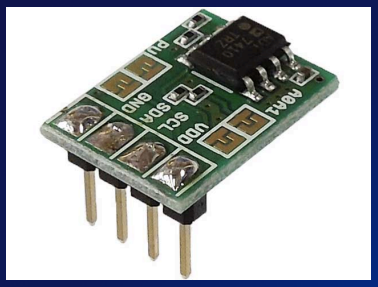
- 13/16bit温度センサモジュール　ADT7410
  - センサ出力：
    - ➡ 13ビット 2の補数表現（0x0000～0x1FFF）
    - ➡ 16ビット 2の補数表現（0x0000～0xFFFF）

初期状態

  - 温度分解能：
    - ➡0.0625℃（13ビット設定）
    - ➡0.0078℃（16ビット設定）

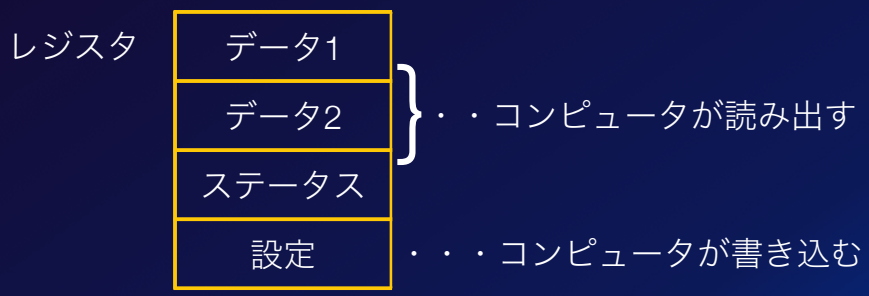
初期状態

  - 温度精度：
    - ➡ ±0.5℃（-40℃～105℃）
  - I2Cインタフェース
    - ➡アドレス 0x48



センサのアクセス方法

- アナログセンサ
  - 物理情報 → 電圧 → A/D変換 → コンピュータ
- センサモジュール（デジタルセンサ）
  - 物理情報 → 電圧 → A/D変換+符号化 → コンピュータ
  - ＋用途に応じた条件設定
  - ➡センサモジュールのアクセス方法



温度センサ（ADT7410）の使用方法

◉ ADT7410のレジスタ

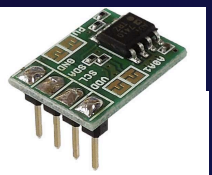


Table 6. ADT7410 Registers

	Register Address	Description	Power-On Default
データ上位	0x00	Temperature value most significant byte	0x00
データ下位	0x01	Temperature value least significant byte	0x00
状態	0x02	Status	0x00
設定	0x03	Configuration	0x00
	0x04	T_{HIGH} setpoint most significant byte	0x20 (64°C)
	0x05	T_{HIGH} setpoint least significant byte	0x00 (64°C)
	0x06	T_{LOW} setpoint most significant byte	0x05 (10°C)
	0x07	T_{LOW} setpoint least significant byte	0x00 (10°C)
	0x08	T_{CRIT} setpoint most significant byte	0x49 (147°C)
	0x09	T_{CRIT} setpoint least significant byte	0x80 (147°C)
	0x0A	T_{HYST} setpoint	0x05 (5°C)
	0x0B	ID	0xCX
	0x2F	Software reset	0xXX

温度センサ（ADT7410）の使用方法

B_{12} ～ B_0

◉ データレジスタフォーマット（0x00, 0x01：13bitデータ）

Table 8. Temperature Value MSB Register (Register Address 0x00) （上位レジスタ・0X00）

Bit	Default Value	Type	Name	Description
[14:8]	0000000	R	Temp	Temperature value in twos complement format
15	0	R	Sign	Sign bit, indicates if the temperature value is negative or positive

B_{12},B_{11},B_{10},B_9,B_8,B_7,B_6,B_5

Table 9. Temperature Value LSB Register (Register Address 0x01) （下位レジスタ・0X01）

Bit	Default Value	Type	Name	Description
0	0	R	T_{LOW} flag/LSB0	Flags a T_{LOW} event if the configuration register, Register Address 0x03[7] = 0 (13-resolution). When the temperature value is below T_{LOW}, this bit it set to 1. Contains the Least Significant Bit 0 of the 15-bit temperature value if the config register, Register Address 0x03[7] = 1 (16-bit resolution).
1	0	R	T_{HIGH} flag/LSB1	Flags a T_{HIGH} event if the configuration register, Register Address 0x03[7] = 0 (13-resolution). When the temperature value is above T_{HIGH}, this bit it set to 1. Contains the Least Significant Bit 1 of the 15-bit temperature value if the config register, Register Address 0x03[7] = 1 (16-bit resolution).
2	0	R	T_{CRIT} flag/LSB2	Flags a T_{CRIT} event if the configuration register, Register Address 0x03[7] = 0 (13-resolution). When the temperature value exceeds T_{CRIT}, this bit it set to 1. Contains the Least Significant Bit 2 of the 15-bit temperature value if the config register, Register Address 0x03[7] = 1 (16-bit resolution).
[7:3]	00000	R	Temp	Temperature value in twos complement format.

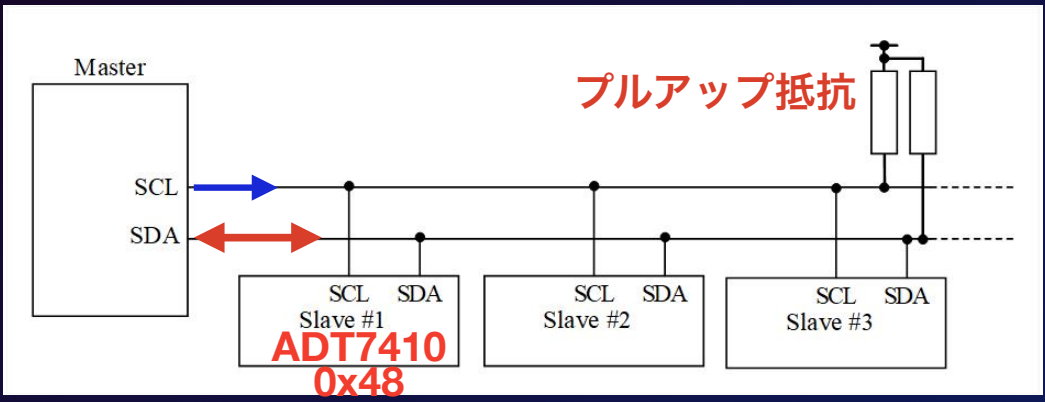
B_4,B_3,B_2,B_1,B_0,＊,＊,＊

I²C (Inter-Integrated Circuit)

- 2線同期式シリアル通信インタフェース
  - ➡ 特徴
    - マイコンと周辺ICの通信
    - 標準モード100kbps/ファーストモード400kbps
    - マスタ-スレーブ方式
    - デバイスにはアドレスが振られている
  - ➡ 信号線
    - Serial DAta （SDA:データ送受信）
    - Serial CLock（SCL:クロック信号）

I²C (Inter-Integrated Circuit)

- 接続例



- Raspberry Pi
  - ➡ SDA (GPIO2)
  - ➡ SCL (GPIO3)
    - I²C用のプルアップ抵抗が接続されているために他のGPIOピンとは特性が異なる.

I2C (Inter-Integrated Circuit)

- デバイスのアドレスを調べる。（LXTerminal）

  $ i2cdetect -y 1

I2C (Inter-Integrated Circuit)

- 通信方法（データ線１本での送受信）

Master→Slave

スレーブアドレス+W

Table 13. START r when master is wri レジスタ /te to slav データ STOP

Master	ST	SAD + W		SUB		DATA		SP
Slave			SAK		SAK		SAK	

ACK（該当スレーブ） ACK ACK

Slave→Master

スレーブアドレス+W　リスタート・スレーブアドレス+R　STOP

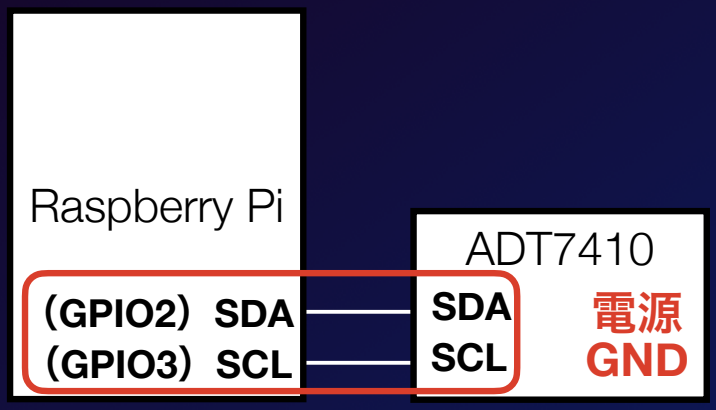
Table 1 START nsfer when m レジスタ eiving (reading) one byte o No Master ACK

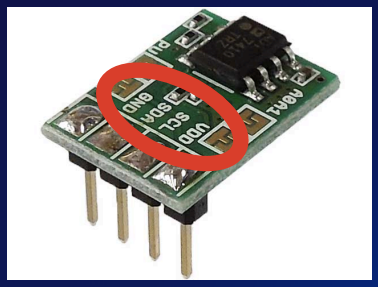
Master	ST	SAD + W		SUB		SR	SAD + R			NMAK	SP
Slave			SAK		SAK			SAK	DATA		

ACK ACK ACK データ

温度センサモジュール測定方法

- センサ出力
  - 13bit 2の補数表現（0x0000～0x1FFF）
  - 分解能 0.0625℃（13ビット設定）





温度センサモジュール測定ブロック図

温度センサモジュールサンプルプログラム

```
import smbus
import time
i2c = smbus.SMBus(1)
address = 0x48

def read_adt7410( ):
    byte_data = i2c.read_byte_data(address,0x00)
    data = byte_data[redacted]
    byte_data = i2c.read_byte_data(address,0x01)
    data = [redacted]
    data = [redacted]
    return data

try:
    while True:
        print(read_adt7410( ) )
        time.sleep(0.5)

except KeyboardInterrupt:
    pass
```

インスタンスの作成（バス番号）

スレーブアドレス

上位8bit

下位5bit

13bitの整数値が得られるので温度に換算する（ 0x000 ～ 0x1FFF ）

温度センサモジュールサンプルプログラム

```
import smbus
import time
i2c = smbus.SMBus(1)
address = 0x48

def read_adt7410( ):
    byte_data = i2c.read_byte_data(address,0x00)
    data = byte_data<<8
    byte_data = i2c.read_byte_data(address,0x01)
    data = data | byte_data
    data = data>>3
    return data

try:
    while True:
        print(read_adt7410( ) )
        time.sleep(0.5)

except KeyboardInterrupt:
    pass
```

上位8bit $B_{12},B_{11},B_{10},B_9,B_8,B_7,B_6,B_5$

下位5bit $B_4,B_3,B_2,B_1,B_0,*,*,*$

データを上位に移動する

上位と下位を合わせる

データを下位にそろえる

data

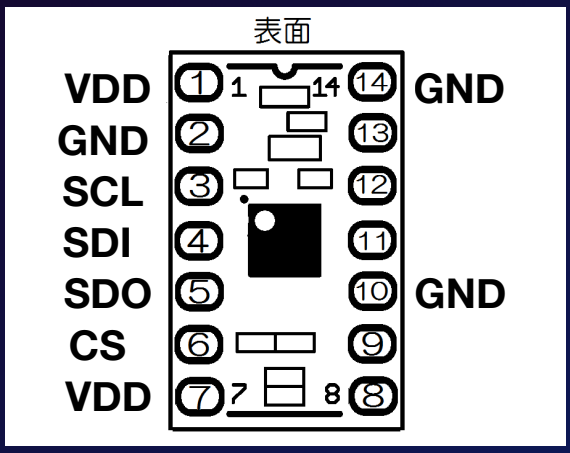
$B_{12},B_{11},B_{10},B_9,B_8,B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$

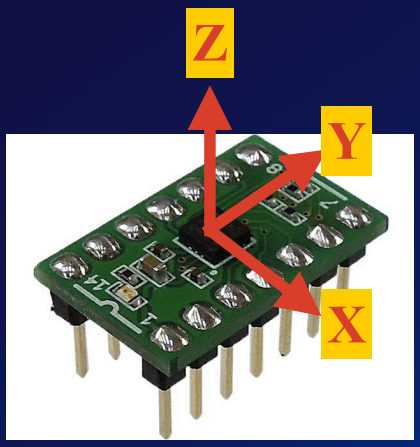
演習（1）

- 2つの温度センサを同時に接続して、計測値（単位℃）を比較しなさい。
  - アナログ温度センサ（MCP9700A）は、出力端子の電圧をAD変換して Raspberry Pi に取り込み、デジタル値をセンサ出力の電圧に変換し、さらに温度を求める。
  - 温度センサモジュール（ADT7410）では、I2Cで取り込んだデジタル値から温度を求める。
  - できるだけ温度範囲を広く計測する。
  - ひとつのプログラムで実行する。
    - ➡ 温度の異なる時間帯に何度か計測する。
    - ➡ エアコンで室温を変えながら計測する。

センサ ③

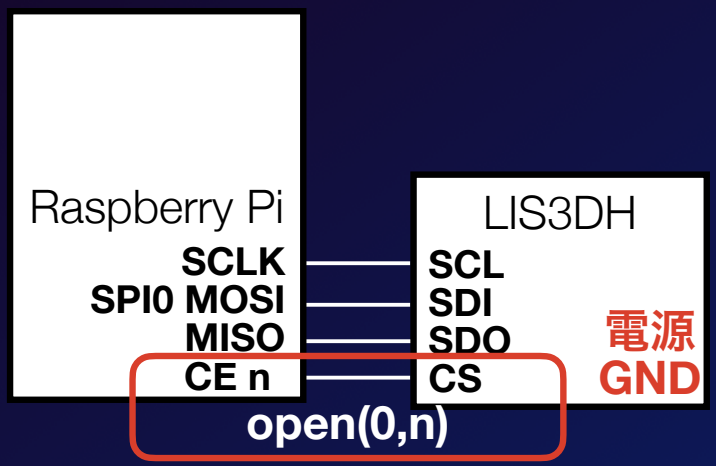
- 3軸加速度センサモジュール（LIS3DH）
  - ３軸の加速度ｇを12bit計測（**2の補数表現**）
  - フルスケールを±2g～16g で設定（**初期状態±2g**）
  - SPI インタフェース





加速度センサモジュール測定方法

- センサ出力（初期設定）
  - フルスケール　±2g
  - 12bit 2の補数表現（0x000～0xFFF）
  - 分解能は?



SPI インタフェース

open (0,n)
0：ポート番号
n：デバイス番号

加速度センサモジュール測定ブロック図

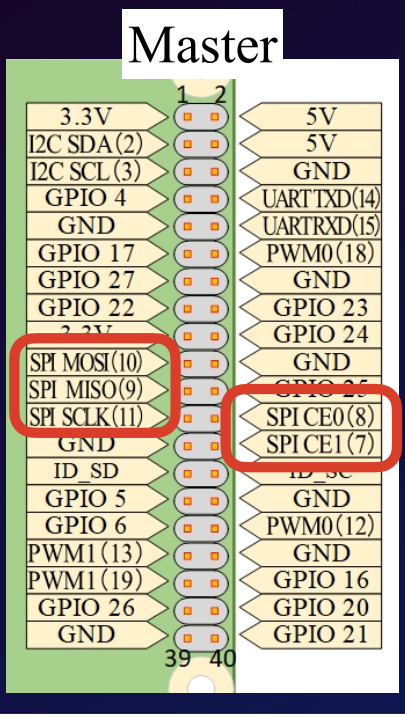
Serial Peripheral Interface

- Raspberry PI の SPI 端子

open(ch,n)

ch : バス番号
n ：デバイス番号

注意：
SPIはCE信号でデバイスを選択する.



bus SPI0

device
CE0／CS0
CE1／CS1

bus SPI1

SPI1を使用する場合は設定が必要

加速度センサ（LIS3DH）の使用方法

- LIS3DHのレジスタ

7 Register mapping

The table given below provides a listing of the 8 bit registers embedded in the device and the related addresses:

Table 17. Register address map

Name	Type	Register address		Default	Comment
		Hex	Binary		
Reserved (do not modify)		00 - 06			Reserved
STATUS_REG_AUX	r	07	000 0111		
OUT_ADC1_L	r	08	000 1000	output	
OUT_ADC1_H	r	09	000 1001	output	
OUT_ADC2_L	r	0A	000 1010	output	
OUT_ADC2_H	r	0B	000 1011	output	
OUT_ADC3_L	r	0C	000 1100	output	
OUT_ADC3_H	r	0D	000 1101	output	
INT_COUNTER_REG	r	0E	000 1110		

OUT_ADC3_L	r	0C	000 1100	output	
OUT_ADC3_H	r	0D	000 1101	output	
INT_COUNTER_REG	r	0E	000 1110		
WHO_AM_I	r	0F	000 1111	00110011	Dummy register
Reserved (do not modify)		10 - 1E			Reserved
TEMP_CFG_REG	rw	1F	001 1111		
CTRL_REG1	rw	20	010 0000	00000111	
CTRL_REG2	rw	21	010 0001	00000000	
CTRL_REG3	rw	22	010 0010	00000000	
CTRL_REG4	rw	23	010 0011	00000000	
CTRL_REG5	rw	24	010 0100	00000000	
CTRL_REG6	rw	25	010 0101	00000000	
REFERENCE	rw	26	010 0110	00000000	
STATUS_REG2	r	27	010 0111	00000000	
OUT_X_L	r	28	010 1000	output	
OUT_X_H	r	29	010 1001	output	
OUT_Y_L	r	2A	010 1010	output	
OUT_Y_H	r	2B	010 1011	output	
OUT_Z_L	r	2C	010 1100	output	
OUT_Z_H	r	2D	010 1101	output	
FIFO_CTRL_REG	rw	2E	010 1110	00000000	
FIFO_SRC_REG	r	2F	010 1111		
INT1_CFG	rw	30	011 0000	00000000	

加速度センサ（LIS3DH）の使用方法

- LIS3DHのレジスタ（デバイス確認用 0x0F）

r/w　レジスタ番号　初期値

（16進）（2進）（0x33）

デバイスの確認用レジスタ

OUT_ADC3_H					
WHO_AM_I	r	0F	000 1111	00110011	Dummy register
Reserved (do not modify)		10 - 1E			Reserved
TEMP_CFG_REG	rw	1F	001 1111		
CTRL_REG1	rw	20	010 0000	00000111	
CTRL_REG2	rw	21	010 0001	00000000	
CTRL_REG3	rw	22	010 0010	00000000	
CTRL_REG4	rw	23	010 0011	00000000	

加速度センサ（LIS3DH）の使用方法

- LIS3DHのレジスタ（データ用 0x28 ～ 0X2D）

		r/w	レジスタ番号（16進）	レジスタ番号（2進）	初期値	
REFERENCE						
データレジスタ						
OUT_X_L	X軸	r	28	010 1000	output	$B_3,B_2,B_1,B_0,*,*,*,*$
OUT_X_H		r	29	010 1001	output	$B_{11},B_{10},B_9,B_8,B_7,B_6,B_5,B_4$
OUT_Y_L	Y軸	r	2A	010 1010	output	
OUT_Y_H		r	2B	010 1011	output	
OUT_Z_L	Z軸	r	2C	010 1100	output	
OUT_Z_H		r	2D	010 1101	output	
FIFO_CTRL_REG		rw	2E	010 1110	00000000	
FIFO_SRC_REG		r	2F	010 1111		
INT1_CFG		rw	30	011 0000	00000000	

加速度センサ（LIS3DH）の使用方法

- LIS3DHのレジスタ（設定用 0x20）

	r/w	レジスタ番号（16進）	レジスタ番号（2進）	初期値	
WHO_AM_I	r	0F	000 1111	00110011	Dummy register
Reserved (do not modify)					Reserved
TEMP_CFG_REG					
CTRL_REG1	rw	20	010 0000	00000111	
CTRL_REG2	rw	21	010 0001	00000000	
CTRL_REG3	rw	22	010 0010	00000000	
CTRL_REG4	rw	23	010 0011	00000000	
CTRL_REG5	rw	24	010 0100	00000000	
CTRL_REG6	rw	25	010 0101	00000000	
REFERENCE	rw	26	010 0110	00000000	
STATUS_REG2	r	27	010 0111	00000000	

加速度センサ（LIS3DH）の使用方法

◉ LIS3DHのレジスタ（設定用 0x20）

8.8 CTRL_REG1 (20h)

(0x20)

Table 29. CTRL_REG1 register

ODR3	ODR2	ODR1	ODR0	LPen	Zen	Yen	Xen

Table 30. CTRL_REG1 description

ODR[3:0]	Data rate selection. Default value: 0000 (0000: power-down mode; others: Refer to *Table 31: Data rate configuration*)
LPen	Low-power mode enable. Default value: 0 (0: high-resolution mode / normal mode, 1: low-power mode) (Refer to section *Section 3.2.1: High-resolution, normal mode, low-power mode*)
Zen	Z-axis enable. Default value: 1 (0: Z-axis disabled; 1: Z-axis enabled)
Yen	Y-axis enable. Default value: 1 (0: Y-axis disabled; 1: Y-axis enabled)
Xen	X-axis enable. Default value: 1 (0: X-axis disabled; 1: X-axis enabled)

0：無効 / 1：有効

EBSI2025

加速度センサ（LIS3DH）の使用方法

◉ LIS3DHのレジスタ（設定用 0x20）

ODR[3:0] is used to set the power mode and ODR selection. The following table indicates the frequency of each combination of ODR[3:0].

Table 31. Data rate configuration

	ODR3	ODR2	ODR1	ODR0	Power mode selection
(0x0)	0	0	0	0	Power-down mode
	0	0	0	1	HR / Normal / Low-power mode (1 Hz)
(0x2)	0	0	1	0	HR / Normal / Low-power mode (10 Hz)
	0	0	1	1	HR / Normal / Low-power mode (25 Hz)
	0	1	0	0	HR / Normal / Low-power mode (50 Hz)
	0	1	0	1	HR / Normal / Low-power mode (100 Hz)
	0	1	1	0	HR / Normal / Low-power mode (200 Hz)
	0	1	1	1	HR / Normal / Low-power mode (400 Hz)
	1	0	0	0	Low power mode (1.60 kHz)
	1	0	0	1	HR / normal (1.344 kHz); Low-power mode (5.376 kHz)

加速度センサ（LIS3DH）の使用方法

- LIS3DHのレジスタ（設定用 0x20）

8.8 CTRL_REG1 (20h) (0x20) ←0x27

Table 29. CTRL_REG1 register

ODR3	ODR2	ODR1	ODR0	LPen	Zen	Yen	Xen
0	0	1	0	0	1	1	1

Table 30. CTRL_REG1 description

ODR[3:0]	Data rate selection. Default value: 0000 (0000: power-down mode; others: Refer to *Table 31: Data rate configuration*)
LPen	Low-power mode enable. Default value: 0 (0: high-resolution mode / normal mode, 1: low-power mode) (Refer to section *Section 3.2.1: High-resolution, normal mode, low-power mode*)
Zen	Z-axis enable. Default value: 1 (0: Z-axis disabled; 1: Z-axis enabled)
Yen	Y-axis enable. Default value: 1 (0: Y-axis disabled; 1: Y-axis enabled)
Xen	X-axis enable. Default value: 1 (0: X-axis disabled; 1: X-axis enabled)

加速度センサモジュールサンプルプログラム（2/2）

```
import spidev
import time

lis = spidev.SpiDev()
lis.open(0,1)
lis.max_speed_hz = 10000
lis.bits_per_word=8

lisread = 0x80
liswrite = 0x00
lissingle = 0x00
#lismulti = 0x40

def readlis3dh(reg2):
    rcv = lis.xfer2([lisread|lissingle|reg2,0x00])
    g = rcv[1]
    rcv = lis.xfer2([lisread|lissingle|reg2+1,0x00])
    g = (g | rcv[1]<<8)>>4
    return g
```

デバイス番号注意

0x80 | 0x00 | 0x28 = 0xA8

加速度センサモジュールサンプルプログラム（2/2）

```
try:
    rcv = lis.xfer2([lisread|lissingle|0x0f,0x00])
    if rcv[1] == 0x33:
        print("LIS3DH ok,")
        lis.xfer2([liswrite|lissingle|0x20,0x27])
        while True:
            x = readlis3dh(0x28)
            y = readlis3dh(0x2a)
            z = readlis3dh(0x2c)
            print("x=",x,"  y=",y,"  z=",z)
            time.sleep(1)
except KeyboardInterrupt:
    pass
lis.close()
```

Dummy Data（接続確認）

センサ設定（3軸有効・測定周期10Hz）

各軸12bitの整数値が得られるので加速度に換算する.

注意：
I（大文字アイ）
l（小文字エル）
|（演算記号or）

演習（2）

- 加速度センサモジュールによる測定と、アナログ温度センサによる測定をひとつのプログラムの中で行いなさい。
  - 加速度センサを傾けたときに各軸が相応しく変化することを確認しなさい.
  - SPI通信では、AD変換器をデバイス0（CE0）、加速度センサをデバイス1（CE1）とすること.

演習（3）

- アナログ温度センサ又は加速度センサが異常値（設定値より大きな値）となった場合にLEDを点滅させるシステムを作成しなさい.
  - 条件
    - ➡演習（2）の回路にLEDを追加する.
    - ➡異常値となったセンサによって点滅方法を変える.
    - ➡加速度、温度の異常値は任意とする.
    - ➡加速度単位はg、温度単位は℃で指定すること.
    - ➡異常値はソースコード上に明記（定義）すること.

問い

- 演習（1）において、氷点下10度のときのAD変換器の出力値（10bit）と、ADT7410の出力値（13bit）を計算で求めなさい.

課題

- 演習(1)（2）（3)について全体が把握できる様に詳細にまとめること．最低限、以下の項目は含むこと．（課題～考察で1500字以上）
  - 表紙（授業名、学籍番号・氏名、提出日）
  - 本文
    - ➡課題
    - ➡使用部品
    - ➡回路の説明（RaspberryPiの端子を明記）
    - ➡アルゴリズムの説明
    - ➡結果（写真，説明を工夫すること）
    - ➡考察
    - ➡問いの解答
    - ➡参考文献（任意）
  - 図表
    - ➡回路図と回路の写真は必須

レポー作成上の注意

- 締め切り 6月19日（厳守）
- 提出物
  - レポート(学籍番号.pdf)
  - 演習（1）（2）（3）ソースファイル（＊＊＊.py）
- 注意
  - 実体配線図は回路図として認めない.
  - 授業資料のコピペ、流用は認めない.
  - レポートはA4版としてPDFファイルで提出する.
  - 印刷して適切なレポートであること.
  - キャプチャ画像等の文字は読めるものであること.
  - PDFファイルに変換後、必ず、確認すること.