## MPL115A2（I2C を用いた絶対圧力計）のデータ取り込み

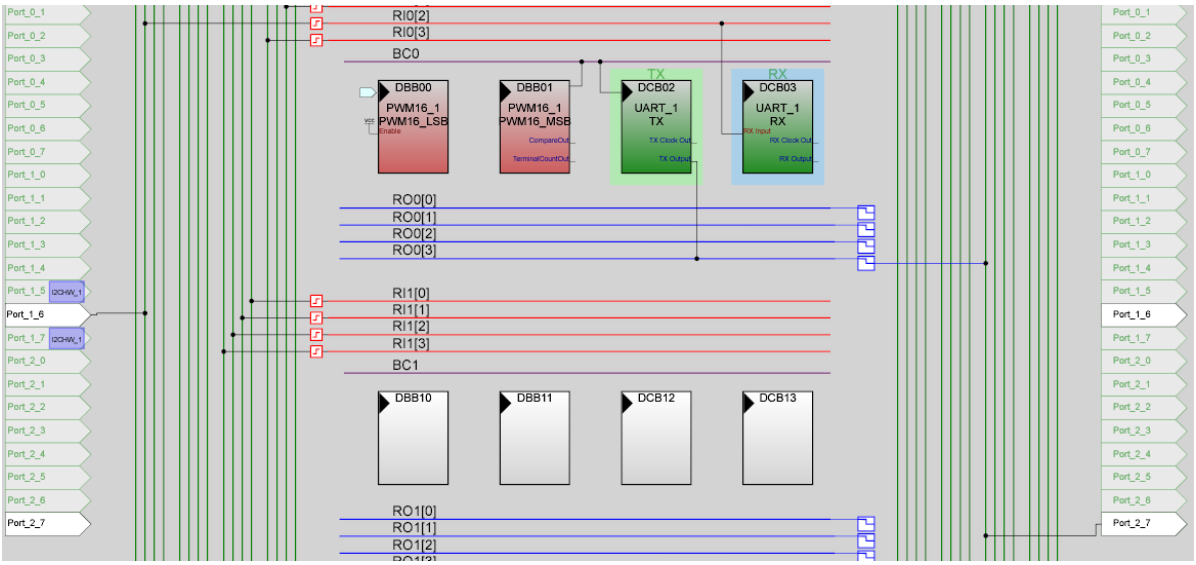
ストロベリーリナックスなどを通して，購入可能な，I2C 通信で行う絶対圧力センサ MPL115A2 と PSoC を用い，RS232 で出力するようにプログラムをする。

PSoC には，P16 に Rx，P27 に Tx を接続し，シリアル通信できるようにした。
MPL115A2 との配線は次のようにする。

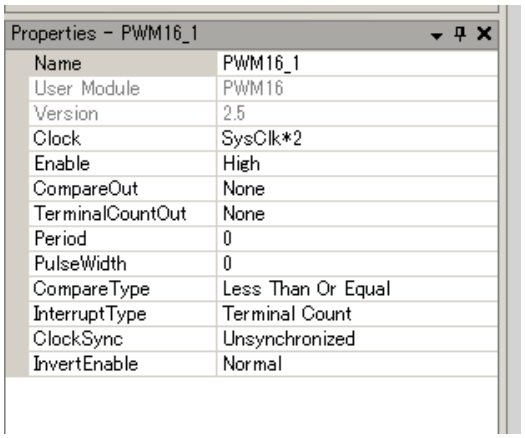
| MPL115A2　PSoC |
|---|
| GND(1)<------GND(14) |
| SHDN(2)<----VDD(28) |
| RST(3)<------- VDD(28) |
| GND(4) <------GND(14) |
| VDD(5) <----VDD(28) |
| NC(7)　　SCL,SDA 10k pullup |
| SCL(6)<--|----P17(10) |
| SDA(5)---|-->P15(11) |

センサへの電源は，5V を許容しているので，配線，書き込みは，それほど注意する必要はないが，逆差しすると壊れると書いてあるので注意すること。

## 全体のブロック構成



## PWM16 の設定



| Properties - PWM16_1 | |
|---|---|
| Name | PWM16_1 |
| User Module | PWM16 |
| Version | 2.5 |
| Clock | SysClk*2 |
| Enable | High |
| CompareOut | None |
| TerminalCountOut | None |
| Period | 0 |
| PulseWidth | 0 |
| CompareType | Less Than Or Equal |
| InterruptType | Terminal Count |
| ClockSync | Unsynchronized |
| InvertEnable | Normal |

Clock は，SysClk*2 とし，VC1～VC3 までの設定に影響が出ないようにしている。また，Period と，PulseWidth は，C の main.c の方で設定する。

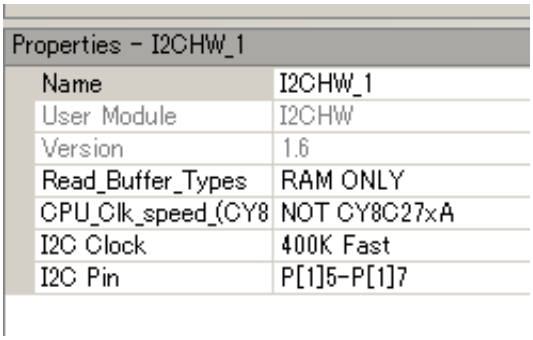
ここでは，出力は出ていないが，出力は，上の BC0 の線から DBB01 を選択し結んでいる。

## UART の設定

| Properties - UART_1 | |
|---|---|
| Name | UART_1 |
| User Module | UART |
| Version | 5.2 |
| Clock | Row_0_Broadcast |
| RX Input | Row_0_Input_2 |
| TX Output | Row_0_Output_3 |
| TX Interrupt Mode | TXRegEmpty |
| ClockSync | Sync to SysClk*2 |
| RxCmdBuffer | Enable |
| RxBufferSize | 16 |
| CommandTerminator | 13 |
| Param_Delimiter | 32 |
| IgnoreCharsBelow | 32 |
| Enable_BackSpace | Disable |
| RX Output | None |
| RX Clock Out | None |
| TX Clock Out | None |
| InvertRX Input | Normal |

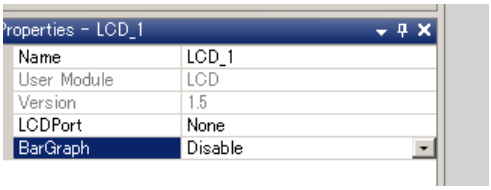
これらの設定は，ほぼデフォルトであり，入出力が Port2_7（Tx）,Port1_6（Rx）につながるように配線を設定している。

## I2CHW の設定




I2C Pin は，P15,P17 とした。これにより MiniProg の書き込みとバッティングしなくて済む。

## LCD の設定

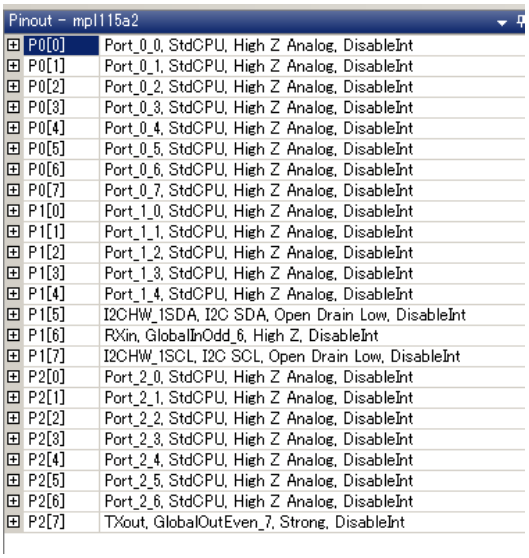



ここでは，LCD は使用していないが，ブロックとして組み込んでいる。理由は，LCD ブロックの関数に，50μSec のタイマーがあるため，C のプログラム中で

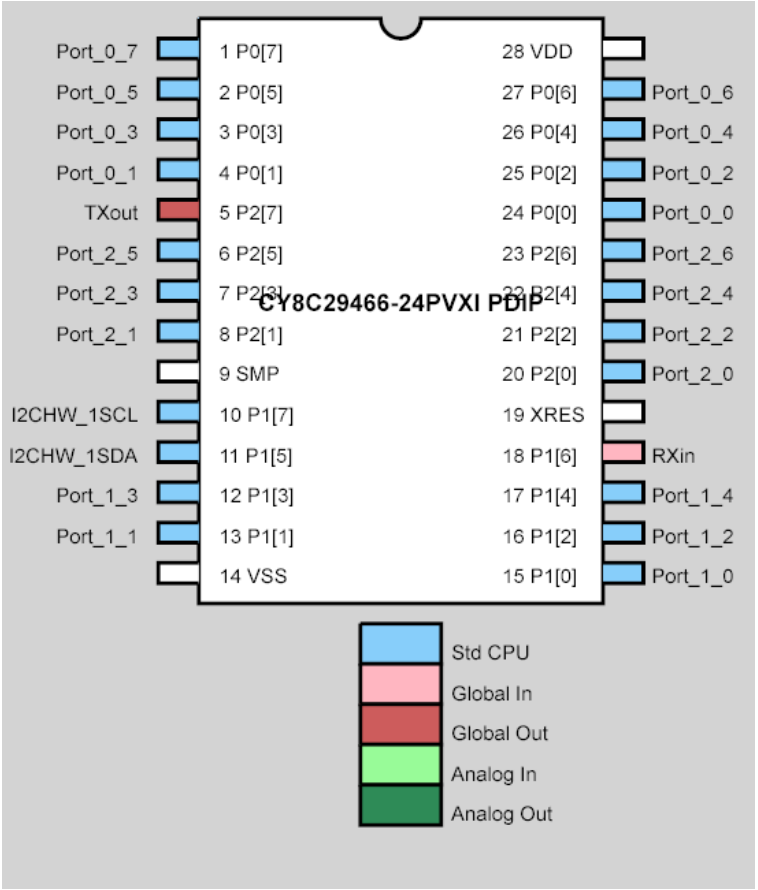
```
void wait_ms(unsigned int ms){
        int i,j;
        if(ms > 12) {
                i = (int)(ms / 12);
                for(j = 0;j < i;j++) {
                        LCD_1_Delay50uTimes(240);
                        ms -= 12;
                }
        }
        LCD_1_Delay50uTimes(ms * 20);
}
```

を使いたいためである。

## Pinout



割り込みは，すべて DisableInt とし，P2_7 は，Strong,P16 は，High Z としている。
また，I2C の仕様により，P15，P17 は，10kΩの抵抗でプルアップしている。




## プログラム

MPL115A は，I2C 通信を用いることになる．PSoC では，I2C 通信用のブロック，モジュールが用意されているので，それらを用いて行う．
基本的に読み書きは，ハイレベル関数である以下の関数を用いる．

| |
|---|
| I2CHW_fReadBytes() |
| I2CHW_bWriteBytes() |
| I2CHW_bWriteCBytes() |

## マニュアルの解読

マニュアルによると，デバイスメモリマップを 16 バイト読み込むには，
[Start],0x60+[W],0x00[Stop]
[Restart],0x60+[R], [0x00], [0x01], [0x02], [0x03], [0x04], [0x05], [0x06], [0x07], [0x08], [0x09], [0x0A], [0x0B], [0x0C],[0x0D],[0x0E],[0x0F],[Stop]
とする。これをハイレベル関数で記述すると以下のようになる。

```
txBuffer[0]=0x00;
I2CHW_1_bWriteBytes(SLAVE_ADDRESS, txBuffer, 1,I2CHW_1_CompleteXfer);
```

```
while(!(I2CHW_1_bReadI2CStatus() & I2CHW_WR_COMPLETE));
I2CHW_1_ClrWrStatus();
I2CHW_1_fReadBytes(SLAVE_ADDRESS, rxBuffer, 16,I2CHW_1_RepStart);
while(!(I2CHW_1_bReadI2CStatus() & I2CHW_RD_COMPLETE));
I2CHW_1_ClrRdStatus();
```

これらを元に I2C 通信により１６バイトのデータを読み込み表示したものを示す.

;MPL115A2 Data= 64 C0 75 80 3D 9B B9 4A C7 46 32 BC 02 20 FD 60

;MPL115A2 Data= 65 00 76 80 3D 9B B9 4A C7 46 32 BC 02 20 FD 60

;MPL115A2 Data= 65 40 76 80 3D 9B B9 4A C7 46 32 BC 02 20 FD 60

;MPL115A2 Data= 65 00 76 C0 3D 9B B9 4A C7 46 32 BC 02 20 FD 60

;MPL115A2 Data= 65 80 75 C0 3D 9B B9 4A C7 46 32 BC 02 20 FD 60

;MPL115A2 Data= 65 00 75 40 3D 9B B9 4A C7 46 32 BC 02 20 FD 60

このデータの解析をする。マニュアルによれば,

**Table 5. Device Memory Map**

| Address | Name | Description | Size (bits) | NVM |
|---|---|---|---|---|
| $00 | POUTH | 10-bit pressure output value MSB | 8 | |
| $01 | POUTL | 10-bit pressure output value LSB | 2 | |
| $02 | TOUTH | 10-bit temperature output value MSB | 8 | |
| $03 | TOUTL | 10-bit temperature output value LSB | 2 | |
| $04 | COEF1 | 96-bit coefficient data 1st byte | 8 | 8 |
| $05 | COEF2 | 96-bit coefficient data 2nd byte | 8 | 8 |
| $06 | COEF3 | 96-bit coefficient data 3rd byte | 8 | 8 |
| $07 | COEF4 | 96-bit coefficient data 4th byte | 8 | 8 |
| $08 | COEF5 | 96-bit coefficient data 5th byte | 8 | 8 |
| $09 | COEF6 | 96-bit coefficient data 6th byte | 8 | 8 |
| $0A | COEF7 | 96-bit coefficient data 7th byte | 8 | 8 |
| $0B | COEF8 | 96-bit coefficient data 8th byte | 8 | 8 |
| $0C | COEF9 | 96-bit coefficient data 9th byte | 8 | 8 |
| $0D | COEF10 | 96-bit coefficient data 10th byte | 8 | 8 |
| $0E | COEF11 | 96-bit coefficient data 11th byte | 8 | 8 |
| $0F | COEF12 | 96-bit coefficient data 12th byte | 8 | 8 |
| $10 | PRESS | Start Pressure Conversion | | |
| $11 | TEMP | Start Temperature Conversion | | |
| $12 | BOTH | Start both Conversions | | |

となっている.

係数がマイナスの場合には，２の補数で表示されるため，それらの変換も行う必要がある.

まず，符号を判断し，

```
bitand(hex2dec(AA(3,:)),2.^15)==2.^15
```

マイナス符号があるならば，数を引いてマイナスの数にする.

```
hex2dec(AA(3,:))-2^16*(bitand(hex2dec(AA(3,:)),2.^15)==2.^15)
```

以下がマニュアルに書いてあったものを MATLAB 用のスクリプトに変換したものである。

```
AA=['6580';'75C0';'3D9B';'B94A';'C746';'32BC';'0220';'FD60']
uiPadc=hex2dec(AA(1,:))/2^6;
uiTadc = hex2dec(AA(2,:))/2^6;
```

```
(hex2dec(AA(3,:)))
sia0=hex2dec(AA(3,:))-2^16*(bitand(hex2dec(AA(3,:)),2.^15)==2.^15);
sib1=hex2dec(AA(4,:))-2^16*(bitand(hex2dec(AA(4,:)),2.^15)==2.^15);
sib2=hex2dec(AA(5,:))-2^16*(bitand(hex2dec(AA(5,:)),2.^15)==2.^15);
sic12=hex2dec(AA(6,:))-2^16*(bitand(hex2dec(AA(6,:)),2.^15)==2.^15);
sic11=hex2dec(AA(7,:))-2^16*(bitand(hex2dec(AA(7,:)),2.^15)==2.^15);
sic22=hex2dec(AA(8,:))-2^16*(bitand(hex2dec(AA(8,:)),2.^15)==2.^15);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
lt1 = sic11;          lt2 = uiPadc;          lt3 = lt1 * lt2;si_c11x1 = lt3;
lt1 = sib1*2^14;   lt2 = si_c11x1;      lt3 = lt1 + lt2;si_a11     = lt3/2^14;
lt1 = sic12;          lt2 = uiTadc;          lt3 = lt1 * lt2;si_c12x2 = lt3;
lt1 = si_a11*2^11;lt2 = si_c12x2;      lt3 = lt1 + lt2;si_a1       = lt3/2^11;
lt1 = sic22;          lt2 = uiTadc;          lt3 = lt1 * lt2;si_c22x2 = lt3;
lt1 = sib2*2^15;   lt2 = si_c22x2/2^1;lt3 = lt1 + lt2;si_a2       = lt3/2^16;
lt1 = si_a1;          lt2 = uiPadc;          lt3 = lt1 * lt2;si_a1x1    = lt3;
lt1 = sia0*2^10;   lt2 = si_a1x1;        lt3 = lt1 + lt2;si_y1       = lt3/2^10;
lt1 = si_a2;          lt2 = uiTadc;          lt3 = lt1 * lt2;si_a2x2    = lt3;
lt1 = si_y1*2^10; lt2 = si_a2x2;        lt3 = lt1 + lt2;siPcomp    = lt3/2^13;
decPcomp = ((65.0/1023.0)*siPcomp)+50
```

として計算できる．ちょっと変換式が長いので整理すると，次のようにかける．

```
format long g
AA=['6580';'75C0';'3D9B';'B94A';'C746';'32BC';'0220';'FD60']
uiPadc=hex2dec(AA(1,:))/2^6;
uiTadc = hex2dec(AA(2,:))/2^6;
(hex2dec(AA(3,:)))
A0=hex2dec(AA(3,:))-2^16*(bitand(hex2dec(AA(3,:)),2.^15)==2.^15);
B1=hex2dec(AA(4,:))-2^16*(bitand(hex2dec(AA(4,:)),2.^15)==2.^15);
B2=hex2dec(AA(5,:))-2^16*(bitand(hex2dec(AA(5,:)),2.^15)==2.^15);
C12=hex2dec(AA(6,:))-2^16*(bitand(hex2dec(AA(6,:)),2.^15)==2.^15);
C11=hex2dec(AA(7,:))-2^16*(bitand(hex2dec(AA(7,:)),2.^15)==2.^15);
C22=hex2dec(AA(8,:))-2^16*(bitand(hex2dec(AA(8,:)),2.^15)==2.^15);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
siPcomp    = ((A0*2^10 + ((((B1*2^14 + C11 * uiPadc)/2^14)*2^11 + C12 * uiTadc)/2^11)
* uiPadc) + ((B2*2^15 + (C22 * uiTadc)/2^1)/2^16)* uiTadc)/2^13;


decPcomp = (((65.0/1023.0)*siPcomp)+50)*10
```

以上の解読結果をもとに作成したプログラムを以下に示す。このプログラムでは，まず，シリアル通信の確認をするため，文字を入力しエンターを押すとエコーバックするプログラムとなっている。“Z”+Enter により取り込みプログラムがスタートする。

```
#include <m8c.h>
#include "PSoCAPI.h"
#include "I2CHW_1Mstr.h"
#define SLAVE_ADDRESS 0x60
//P16 Rx
//P27 Tx
/*
MPL115A2  PSoC
GND(1)<-----GND(14)
SHDN(2)<----VDD(28)
RST(3)<-----VDD(28)
GND(4) <----GND(14)
VDD(5) <----VDD(28)
NC(7)
SCL(6)<-----P17(10)
SDA(5)----->P15(11)
AA=['5F00';'7F00';'43EF';'AE8B';'BA33';'3878';'0000';'0000']
AA=['5F40';'7F00';'43EF';'AE8B';'BA33';'3878';'0000';'0000']
AA=['5F80';'7F00';'43EF';'AE8B';'BA33';'3878';'0000';'0000']
AA=['5F00';'7FC0';'43EF';'AE8B';'BA33';'3878';'0000';'0000']
AA=['5F40';'7FC0';'43EF';'AE8B';'BA33';'3878';'0000';'0000']
AA=['5F80';'7FC0';'43EF';'AE8B';'BA33';'3878';'0000';'0000']

%AA=['6580';'75C0';'3D9B';'B94A';'C746';'32BC';'0220';'FD60']
uiPadc=hex2dec(AA(1,:))/2^6;
uiTadc = hex2dec(AA(2,:))/2^6;
(hex2dec(AA(3,:)))
A0=hex2dec(AA(3,:))-2^16*(bitand(hex2dec(AA(3,:)),2.^15)==2.^15);
B1=hex2dec(AA(4,:))-2^16*(bitand(hex2dec(AA(4,:)),2.^15)==2.^15);
B2=hex2dec(AA(5,:))-2^16*(bitand(hex2dec(AA(5,:)),2.^15)==2.^15);
C12=hex2dec(AA(6,:))-2^16*(bitand(hex2dec(AA(6,:)),2.^15)==2.^15);
C11=hex2dec(AA(7,:))-2^16*(bitand(hex2dec(AA(7,:)),2.^15)==2.^15);
C22=hex2dec(AA(8,:))-2^16*(bitand(hex2dec(AA(8,:)),2.^15)==2.^15);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
siPcomp  = (((A0*2^10 + ((((B1*2^14 + C11 * uiPadc)/2^14)*2^11 + C12 * uiTadc)/2^11) *
uiPadc)/2^10)*2^10 + ((B2*2^15 + (C22 * uiTadc)/2^1)/2^16)* uiTadc)/2^13;
decPcomp = (((65.0/1023.0)*siPcomp)+50)*10

*/
/* setup a 64 byte buffer */
char   txBuffer[32];
char    rxBuffer[32];
void wait_ms(unsigned int ms){
        int i,j;
        if(ms > 12) {
                i = (int)(ms / 12);
                for(j = 0;j < i;j++) {
                        LCD_1_Delay50uTimes(240);
                        ms -= 12;
                }
        }
        LCD_1_Delay50uTimes(ms * 20);
}
void main(void) {
        char * strPtr;
        int flag;
        double A0,B1,B2,C12,C11,C22;
        unsigned int tmp;
        double siPcomp;
        long decPcomp;
        int iPadc,iTadc;
        double uiPadc,uiTadc;
        PWM16_1_WritePeriod(625-1);PWM16_1_WritePulseWidth(312);// 9600 bps
        // PWM16_1_WritePeriod(313-1);PWM16_1_WritePulseWidth(156);//19200 bps
```

```
        // PWM16_1_WritePeriod(156-1);PWM16_1_WritePulseWidth(78);//38400 bps
        // PWM16_1_WritePeriod(104-1);PWM16_1_WritePulseWidth(52);//57600 bps
        // PWM16_1_WritePeriod(52-1);PWM16_1_WritePulseWidth(26);//115200 bps
        // PWM16_1_WritePeriod(26-1);PWM16_1_WritePulseWidth(13);//230400 bps x
        PWM16_1_Start();
        UART_1_CmdReset(); // Initialize receiver/cmd
        UART_1_IntCntl(UART_1_ENABLE_RX_INT); // Enable RX interrupts
        UART_1_Start(UART_1_PARITY_NONE); // Enable UART
    I2CHW_1_Start();
    I2CHW_1_EnableMstr();
    M8C_EnableGInt;
    I2CHW_1_EnableInt();
        UART_1_CPutString("Welcome to PSoC UART test program. V1.1 ");
        while(1) {
                if(UART_1_bCmdCheck()) { // Wait for command
                        if(strPtr = UART_1_szGetParam()) { // More than delimiter?
                                UART_1_CPutString("Found valid command=>");
                                UART_1_PutString(strPtr);
                                if(strPtr[0] == 'Z') break;
                                UART_1_CPutString("Paramaters:");
                                while(strPtr = UART_1_szGetParam()) { // loop on each
parameter
                                        UART_1_CPutString(" <");
                                        UART_1_PutString(strPtr); // Print each
parameter
                                        UART_1_CPutString(">");
                                }
                        }
                        UART_1_CmdReset(); // Reset command buffer
                }
        }
        txBuffer[0]=0x04;
        I2CHW_1_bWriteBytes(SLAVE_ADDRESS, txBuffer, 1,I2CHW_1_CompleteXfer);
    while(!(I2CHW_1_bReadI2CStatus() & I2CHW_WR_COMPLETE));
        I2CHW_1_ClrWrStatus();
        I2CHW_1_fReadBytes(SLAVE_ADDRESS, rxBuffer, 12,I2CHW_1_RepStart);
    while(!(I2CHW_1_bReadI2CStatus() & I2CHW_RD_COMPLETE));
        I2CHW_1_ClrRdStatus();
        tmp = ((unsigned int)(rxBuffer[0]) << 8) + ((unsigned int)(rxBuffer[1]) & 0x00FF);
//      UART_1_PutSHexInt((int)tmp);
        A0 = (double)(tmp - 0x10000*((0x8000 & tmp)== 0x8000));
        tmp = ((unsigned int)(rxBuffer[2]) << 8) + ((unsigned int)(rxBuffer[3]) & 0x00FF);
//      UART_1_PutSHexInt((int)tmp);
        B1 = (double)(tmp - 0x10000*((0x8000 & tmp)== 0x8000));
        tmp = ((unsigned int)(rxBuffer[4]) << 8) + ((unsigned int)(rxBuffer[5]) & 0x00FF);
//      UART_1_PutSHexInt((int)tmp);
        B2 = (double)(tmp - 0x10000*((0x8000 & tmp)== 0x8000));
        tmp = ((unsigned int)(rxBuffer[6]) << 8) + ((unsigned int)(rxBuffer[7]) & 0x00FF);
//      UART_1_PutSHexInt((int)tmp);
        C12= (double)(tmp - 0x10000*((0x8000 & tmp)== 0x8000));
        tmp = ((unsigned int)(rxBuffer[8]) << 8) + ((unsigned int)(rxBuffer[9]) & 0x00FF);
//      UART_1_PutSHexInt((int)tmp);
        C11= (double)(tmp - 0x10000*((0x8000 & tmp)== 0x8000));
        tmp = ((unsigned int)(rxBuffer[10]) << 8) + ((unsigned int)(rxBuffer[11]) &
0x00FF);
//      UART_1_PutSHexInt((int)tmp);
        C22= (double)(tmp - 0x10000*((0x8000 & tmp)== 0x8000));
        for(;;) {
                txBuffer[0]=0x12;txBuffer[1]=0x01;
                I2CHW_1_bWriteBytes(SLAVE_ADDRESS, txBuffer, 2, I2CHW_1_CompleteXfer);
      while(!(I2CHW_1_bReadI2CStatus() & I2CHW_WR_COMPLETE));
                I2CHW_1_ClrWrStatus();
                wait_ms(3);
                txBuffer[0]=0x00;
                I2CHW_1_bWriteBytes(SLAVE_ADDRESS, txBuffer, 1, I2CHW_1_RepStart);
      while(!(I2CHW_1_bReadI2CStatus() & I2CHW_WR_COMPLETE));
                I2CHW_1_ClrWrStatus();
                I2CHW_1_fReadBytes(SLAVE_ADDRESS, rxBuffer, 4, I2CHW_1_CompleteXfer);
      while(!(I2CHW_1_bReadI2CStatus() & I2CHW_RD_COMPLETE));
                I2CHW_1_ClrRdStatus();
                iPadc = ((int)(rxBuffer[0]) << 8) + ((int)(rxBuffer[1]) & 0x00FF);
                iTadc = ((int)(rxBuffer[2]) << 8) + ((int)(rxBuffer[3]) & 0x00FF);
```

```
                    iTadc = (int)(iTadc - 0x10000*((0x8000 & iTadc)== 0x8000));
                    uiPadc = (double)iPadc / 64.0;
                    uiTadc = (double)iTadc / 64.0;
                    siPcomp  = ((A0 * 1024.0 + ((((B1 * 16384.0 + C11 * uiPadc) / 16384.0) *
2048.0
                    + C12 * uiTadc) / 2048.0) * uiPadc)
                    + ((B2 * 32768.0 + (C22 * uiTadc) / 2.0) / 65536.0) * uiTadc) / 8192.0;
                    decPcomp = (long)((((65.0/1023.0) * siPcomp) + 50.) * 100000.);
                    UART_1_PutSHexInt((int)iPadc);
                    UART_1_PutSHexInt((int)iTadc);
                    UART_1_PutSHexInt((int)(decPcomp >> 16));
                    UART_1_PutSHexInt((int)(decPcomp & 0xffff));
                    UART_1_PutChar(10);
          }
}
```

## 出力例

最初の 8 桁分の数値は，4 桁ごとに，圧力，温度表記である。

最後の 8 桁分は，hPa を単位とした気圧を表している。

```
5FC07F00009B123E
5F407F00009B7AE0
5FC07F00009B123E
5F407F40009B67F4
5F807F40009B33A9
5F407F00009B7AE0
5FC07F00009B123E
5F407F40009B67F4
5F807F00009B468F
5FC07F00009B123E
5F407F00009B7AE0
5F807F00009B468F
5FC07F40009AFF5D
5F807F40009B33A9
5F807F80009B20C2
5F407F40009B67F4
5F407F40009B67F4
5F407F00009B7AE0
5F807F00009B468F
5F807F00009B468F
5F407F40009B67F4
5F807F40009B33A9
5F407F40009B67F4
```

```
5F007F40009B9C3F
5F407F40009B67F4
5F807F40009B33A9
5F807F40009B33A9
5FC07F40009AFF5D
5F407F40009B67F4
5F407F00009B7AE0
5F407F00009B7AE0
5F807F40009B33A9
```

ちなみに，hex2dec 関数により変換できるが，この場合の気圧は，

```
>> hex2dec('009A1D20')/10000
ans =
        1010
>> hex2dec('009BA3C0')/10000
ans =
        1020
```

となる。