

簡易湿度計(LM74)

概要

温度センサーにはLM35LM60LM61などの、温度に比例した電圧が出力されるタイプの物が、よく使用されています。

そして、このようなセンサーを使用した温度計の仕組み(流れ)は、概ね次のようになります。【センサー】 【オペアンプ】A/D変換+基準電圧 【マイコン】LCD また、精度を高めるためには、特にオペアンプA/D変換、基準電圧の回路設計に留意する必要があります。

そこで今回は、回路設計の手間を省いて、尚且つ、高い精度の得られる温度計を製作してみます。センサーには、-55 ~+150 の動作温度範囲に対して、12ビット+サインの温度分解能 (LSB 当たり、0.0625) を備えたナショナル セミコンダクター社の「LM74」を使用します。

<簡易温度計(LM74)の仕様>

- 測定範囲:-55 ~150
- 温度分解能:0.0625
- 表示内容:現在温度、最高温度、最低温度、測定経過時間

動作原理

PICに温度センサーをシリアル接続し、周期的に温度を測定し、結果をLCDに表示します。

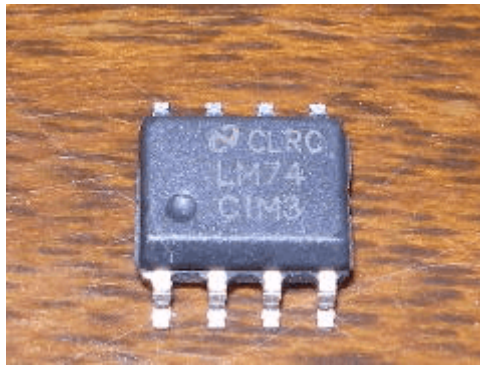
動作原理(ハードウェア)

温度センサー

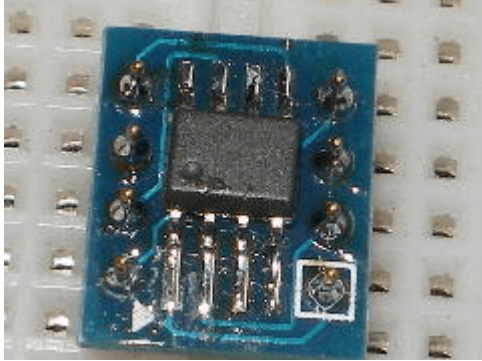
- ナショナル セミコンダクター社の「LM74」を使用します。

<LM74の仕様>

- 0.0625 の温度分解能
- シャットダウン・モードによる温度読み出し間の節電
- SPI およびMICROWIRE バス・インタフェースを装備
- 電源電圧3.0V あるいは2.65V ~ 5.5V
- 電源電流通常動作時265μA (代表値)、520μA (最大)、シャットダウン時3μA (代表値)

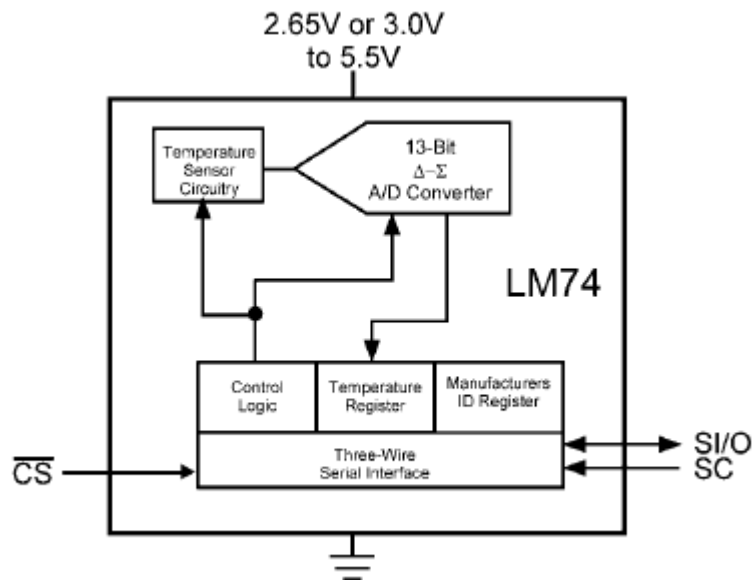


<LM74の概観と変換基板への実装例>



◎LM74の制御線

- スレーブ出力(SI/O):シリアルバス双方向データライン(PIC□□LM74)
- スレーブクロック(SC):シリアルバス用クロック(PIC□LM74)
- チップセレクト(CS):チップセレクト用の信号(PIC□LM74)



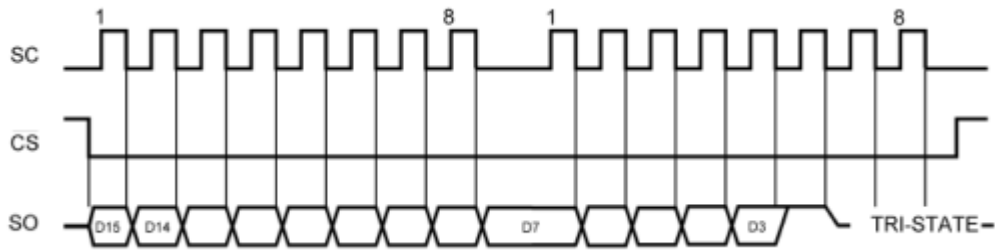
<LM74のブロックダイアグラム>

動作原理(ソフトウェア)

温度の測定

- LM74のタイミングチャートに従って、温度レジスタ(16ビット)の内容を取得します。
- 温度レジスタの、D15~D3の13ビットを使用します。
- LM74の温度とデータの関係より、温度値を求めます。

<LM74のタイミングチャート>

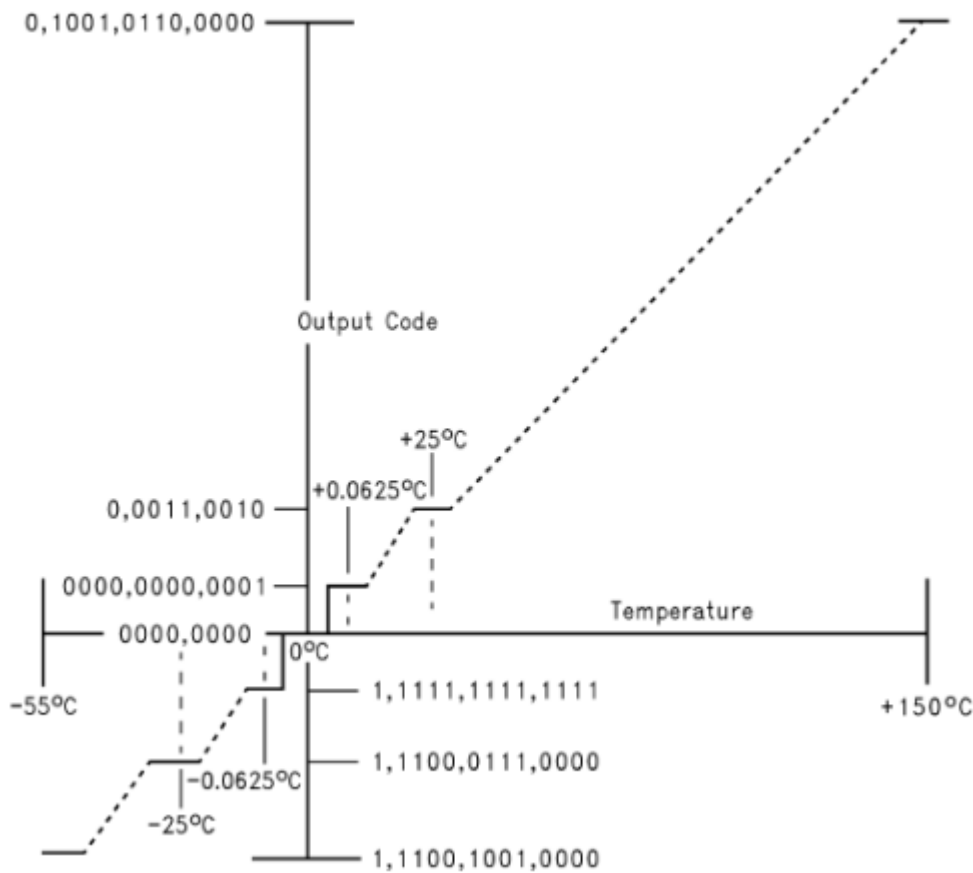


D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
MSB	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	LSB	1	X	X

<LM74の温度レジスタ>

※D15~D3の13ビットを使用します。

<LM74の温度とデータの関係(1)>



<LM74の温度とデータの関係(2)>

Temperature	Digital Output	
	Binary	Hex
+150°C	0100 1011 0000 0111	4B 07h
+125°C	0011 1110 1000 0111	3E 87h
+25°C	0000 1100 1000 0111	0C 87h
+0.0625°C	0000 0000 0000 1111	00 0Fh
0°C	0000 0000 0000 0111	00 07h
-0.0625°C	1111 1111 1111 1111	FF FFh
-25°C	1111 0011 1000 0111	F3 87h
-55°C	1110 0100 1000 0111	E4 87h

最高温度、最低温度の算出

- 現在温度と前回までの最高温度を比較し、最高温度および最低温度を求めます。
- スイッチ(SW)が押下されると、最高温度および最低温度をリセットします。

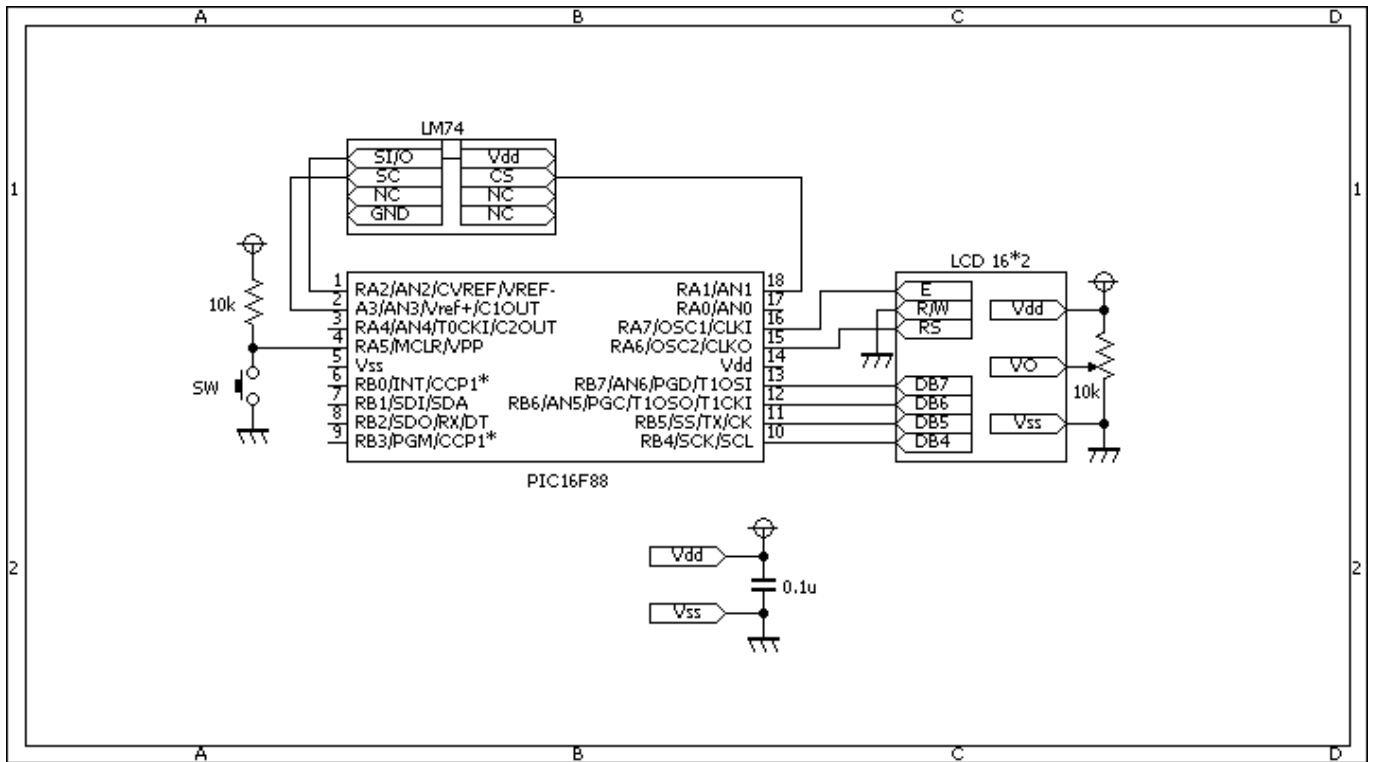
測定経過時間の算出

- 100msec周期の割り込みを発生させ、それを測定経過時間(時分秒)を計る基準クロックとします。
- スイッチ(SW)が押下されると、測定経過時間をリセットします。

測定結果の表示

- 現在温度、最高温度、最低温度、測定経過時間をLCDに表示します。

回路図



ソースコード

[thermo_meter_lm74.c](#)

```

//*****
*
/*
  < 簡易湿度計 LM74 >
*/
//*****
*
//LCD
sbit LCD_RS at RA6_bit;
sbit LCD_EN at RA7_bit;
sbit LCD_D7 at RB7_bit;
sbit LCD_D6 at RB6_bit;
sbit LCD_D5 at RB5_bit;
sbit LCD_D4 at RB4_bit;
sbit LCD_RS_Direction at TRISA6_bit;
sbit LCD_EN_Direction at TRISA7_bit;
sbit LCD_D7_Direction at TRISB7_bit;
sbit LCD_D6_Direction at TRISB6_bit;
sbit LCD_D5_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB4_bit;
//LM74
sbit LM74_CS at RA1_bit;
sbit LM74_S0 at RA2_bit;
sbit LM74_SC at RA3_bit;
sbit LM74_CS_Direction at TRISA1_bit;

```

```
sbit LM74_S0_Direction at TRISA2_bit;
sbit LM74_SC_Direction at TRISA3_bit;
//
sbit SW at RA5_bit;
//
#define BYTE    unsigned short
#define WORD    unsigned int
#define DWORD   unsigned long
//*****
*
// 関数宣言
extern void    main();
extern int     measurement();
extern int     convert(int thermo);
extern void    display(char row, char column, int thermo);
extern void    init_timer();
extern void    interrupt();
extern void    trigger();
extern void    get_clock(char *msg);
extern DWORD  clock;
extern short   start_flg;
//*****
*
static char    buf[16];
//*****
*
// メイン関数
void    main()
{
    int     thermo, max, min;
    //
    OSCCON = 0b01110000;
    ANSEL  = 0b00000000;
    TRISA  = 0b00111111;
    TRISB  = 0b00000110;
    //
    Lcd_Init();
    Lcd_Cmd(_LCD_CURSOR_OFF);
    Lcd_Cmd(_LCD_CLEAR);
    Lcd_Out(1, 1, "Thermometer LM74");
    Delay_ms(1000);
    Lcd_Cmd(_LCD_CLEAR);
    Lcd_Out(1, 6, "□C");
    Lcd_Out(2, 6, "□C");
    Lcd_Out(2, 14, "□C");
    //
    LM74_CS_Direction = 0;
    LM74_S0_Direction = 1;
    LM74_SC_Direction = 0;
    LM74_SC = 0;
    LM74_CS = 1;
}
```

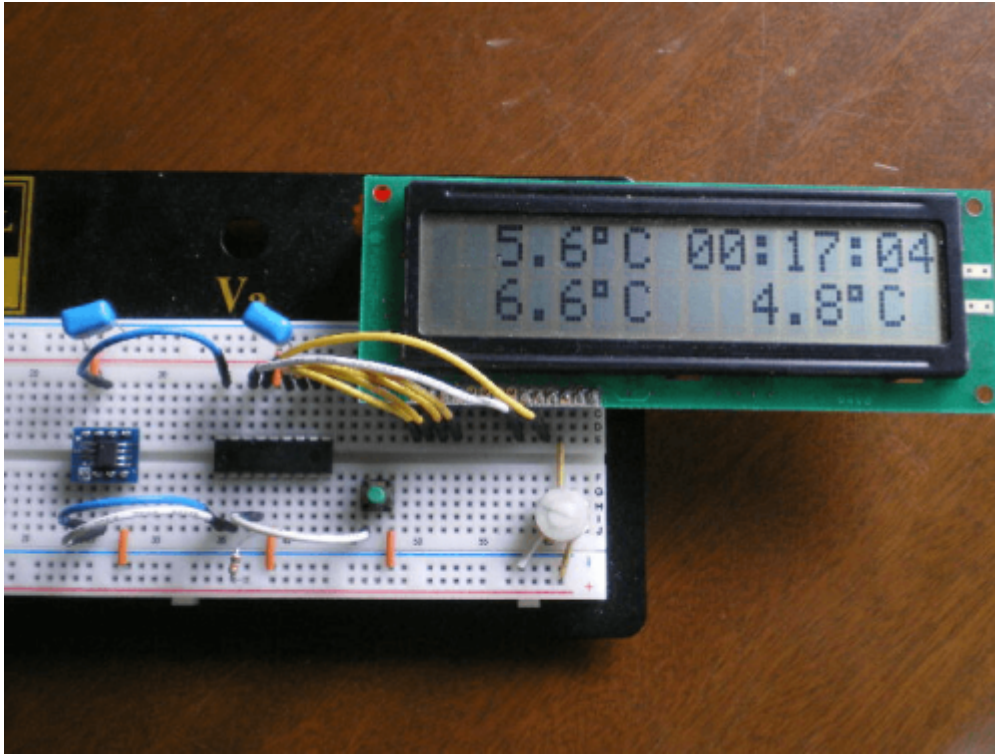
```
//
max = 0;
min = 1000;
clock = 0;
start_flg = 1;
//
init_timer();
//
while (1) {
    //温度の測定と表示(現在値、最高値、最低値)
    thermo = measurement();
    thermo = convert(thermo);
    max = (thermo < max) ? max : thermo;
    min = (thermo > min) ? min : thermo;
    display(1, 1, thermo);
    display(2, 1, max);
    display(2, 9, min);
    //経過時間の取得と表示
    get_clock(buf);
    Lcd_Out(1, 9, buf);
    //
    if (SW == 0) {
        start_flg = 0;
        clock = 0;
        max = 0;
        min = 1000;
        start_flg = 1;
    }
}
}
//*****
*
// 温度換算関数
void display(char row, char column, int thermo)
{
    if (thermo >= 0) {
        IntToStr(thermo, buf);
        buf[1] = ' ';
        buf[2] = buf[3];
        buf[3] = buf[4];
        buf[4] = '.';
    } else {
        IntToStr(thermo, buf);
        buf[1] = '-';
        buf[2] = (buf[3] == '-') ? ' ' : buf[3];
        buf[3] = (buf[4] == '-') ? ' ' : buf[4];
        buf[4] = '.';
    }
    Lcd_Out(row, column, &buf[1]);
}
//*****
```

```
*
int    convert(int thermo)
{
    double d;
    int    tmp;
    //
    if ((thermo & 0b1000000000000000) == 0) {
        d = thermo >> 3;
        d = d * 0.0625 * 10;
    } else {
        d = (thermo >> 3) & 0x0FFF;
        d = (d - 4096) * 0.0625 * 10;
    }
    tmp = d;
    return (tmp);
}
//*****
*
// 温度測定関数
int    measurement()
{
    DWORD    thermo;
    short    cnt;
    //
    thermo = 0;
    LM74_CS = 0;
    for (cnt = 0; cnt < 16; cnt++) {
        LM74_SC = 1;
        if (LM74_SO == 1) {
            thermo |= 1;
            thermo <<= 1;
        } else {
            thermo |= 0;
            thermo <<= 1;
        }
        LM74_SC = 0;
    }
    LM74_CS = 1;
    thermo >>= 1;
    return (thermo);
}
//*****
*
//■100msecの周期割り込みを発生させる関数です。
void    init_timer()
{
    // CCPの設定
    PIE1.CCP1IE = 1;
    PIR1.CCP1IF = 0;
    CCP1CON.CCP1M3 = 1;
    CCP1CON.CCP1M2 = 0;
}
```

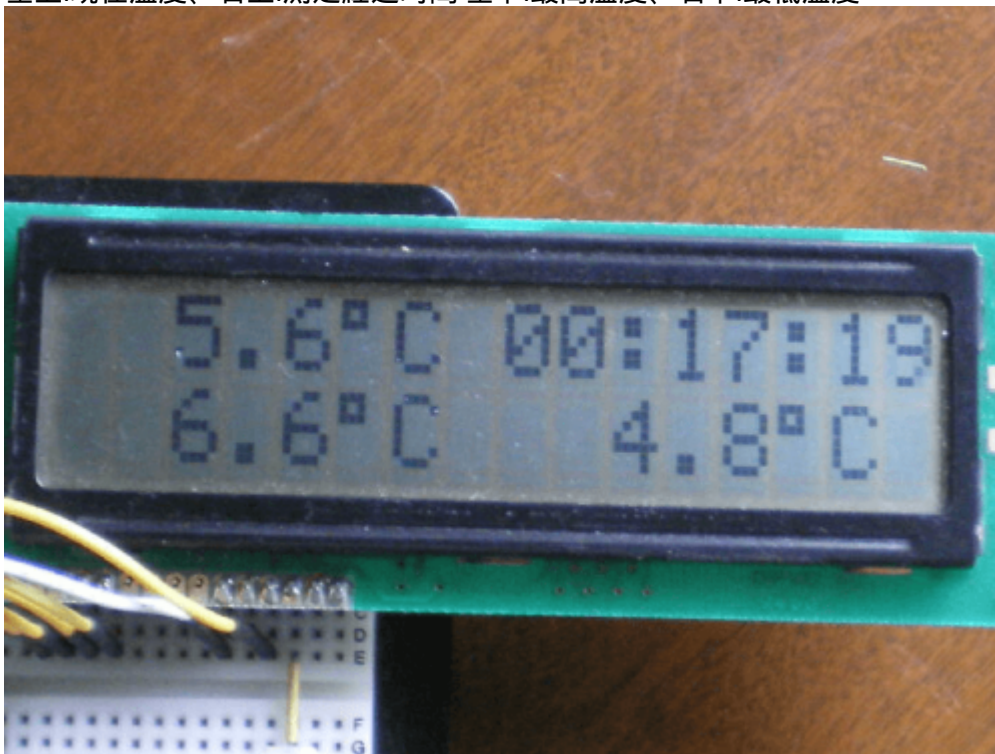
```
    CCP1CON.CCP1M1 = 1;
    CCP1CON.CCP1M0 = 1;
    CCPR1L = 0xA8;          // 100msec...クロックが8Mhzの時
    CCPR1H = 0x61;          //
100msec... (1÷16000000)*4*8*25000(0x61A8)
    // TIMER1の設定
    PIE1.TMR1IE = 0;
    PIR1.TMR1IF = 0;
    TMR1L = 0;
    TMR1H = 0;
    T1CON.TMR1CS = 0;
    T1CON.T1CKPS0 = 1;
    T1CON.T1CKPS1 = 1;
    T1CON.TMR1ON = 1;
    //
    INTCON.PEIE = 1;
    INTCON.GIE = 1;
}
//*****
*
// 割り込み関数です。
DWORD   clock = 0;
short   start_flg = 0;
//
void    interrupt()
{
    if (PIR1.CCP1IF == 1) {
        PIR1.CCP1IF = 0;
        //
        if (start_flg == 1) {
            clock++;
        }
    }
}
//*****
*
// 測定タイミングをチェックする関数です。
void    trigger()
{
    DWORD   tmp;
    //
    tmp = clock;
    while ((tmp % 10) != 0) {
        Delay_ms(1);
    }
}
//*****
*
void    ByteToStr2(unsigned short number, char *output)
{
    ByteToStr(number, output);
}
```

```
output[0] = (output[1] == ' ') ? '0' : output[1];
output[1] = output[2];
output[2] = 0x00;
}
//*****
*
//經過時間取得関数
void get_clock(char *msg)
{
    DWORD tmp;
    BYTE hh, mm, ss;
    //
    tmp = clock;
    tmp /= 10;
    hh = tmp / 3600;
    mm = (tmp % 3600) / 60;
    ss = (tmp % 3600) % 60;
    ByteToStr2(hh, buf);
    buf[2] = ':';
    ByteToStr2(mm, &buf[3]);
    buf[5] = ':';
    ByteToStr2(ss, &buf[6]);
    buf[8] = ' ';
}
//*****
*
```

動作確認



左上:現在温度、右上:測定経過時間 左下:最高温度、右下:最低温度



著作権表示 **copyright notice**

このページは稲崎様の閉鎖したHPのコピーで、著作権は稲崎様にあります。[詳細](#) This page is a copy of Mr. Inasaki's closed website, and the copyright is held by him.[Details](#)

From:
<http://www.deepsky.jp/wiki/> - うごくといいな

Permanent link:
<http://www.deepsky.jp/wiki/doku.php?id=elechobby:picdic:pic16f88:153&rev=1588326594>

Last update: **2025/10/17 14:28**

