簡易オシロV2(LCD:128×64)

概要

以前に製作した『簡易オシロ(グラフィック液晶表示)』は、122×32ドットでしたが、今回は、128×64ドットのLCDを使用しました。 従ってオシロのY軸が2倍になりましたので、以前よりも細かいデータを観測することが出来ます。

<LCDグラフィック・ディスプレイモジュール(SG12864A)の仕様>

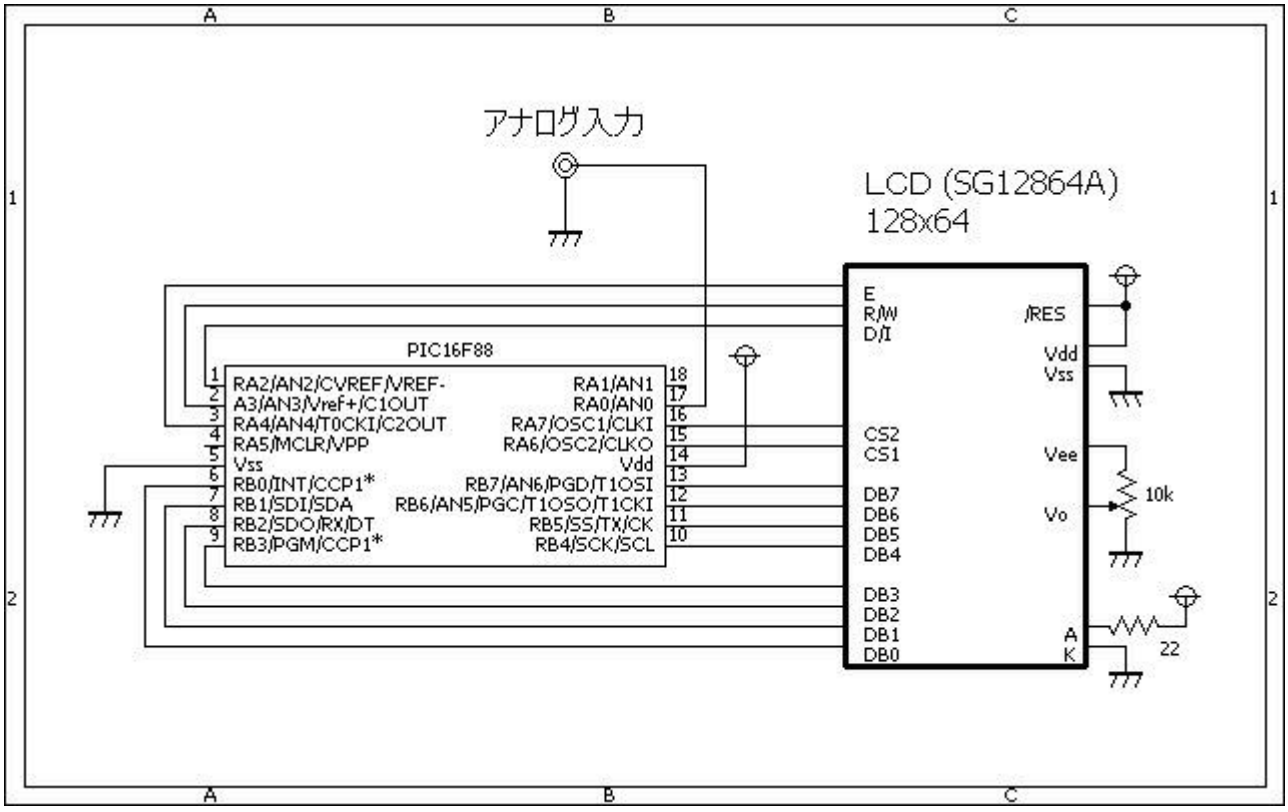
- SUNLIKE社グラフィックLEDモジュールです。
- 128×64ドット
- 外形サイズ:93x72mm
- 液晶表示サイズ:72x40mm
- 5V単一電源(3.3V動作可能)
- H8マイコン、PICマイコンなどに直結できます。
- 接続端子:20ピン20ピン端子付き

動作原理

PICのA/D変換機能でアナログデータを取り込みそれを単純に液晶に表示させるだけのものです。 但し、PICのA/Dの分解能は10ビットなので4ビット分右にシフトして6ビット(64)にします。

尚、以前に使用したモジュールは、液晶駆動クロック(2kHz~3kHz)を外部から供給する必要がありましたが、今回使用したモジュールはその必要がありません。 また、コントラスト調整端子(Vo)への電源の供給方法は、以前に使用したモジュールは、VddとVssでしたが、今回使用したモジュールは、VeeとVssです。 Veeはモジュールの内部で負電圧(約-9V)を発生させているようです。

回路図



ソースコード

glcd.c

```
//*********************************************************************
*
/*
  <簡易オシロ>

  ★使用□□□グラフィック・ディスプレイモジュール
□□□□□□□□□□□□□□□□□□ドット）
*/
//*********************************************************************
*

#define       cntl_E        PORTA.F4
#define       cntl_RW        PORTA.F3
#define       cntl_DI        PORTA.F2
#define       cntl_CS2    PORTA.F7
#define       cntl_CS1    PORTA.F6

void   gLcdData(unsigned char cs, unsigned char data)
{
    cntl_CS1 = (((cs == 1) || (cs == 0)) ? 1 : 0);
    cntl_CS2 = (((cs == 2) || (cs == 0)) ? 1 : 0);
    cntl_RW = 0;
    cntl_DI = 1;
    cntl_E = 1;
```

```
    PORTB = data;
    Delay_us(30);
    cntl_E = 0;
    cntl_CS1 = 0;
    cntl_CS2 = 0;
}

void   gLcdCommand(unsigned char cs, unsigned char command)
{
    cntl_CS1 = (((cs == 1) || (cs == 0)) ? 1 : 0);
    cntl_CS2 = (((cs == 2) || (cs == 0)) ? 1 : 0);
    cntl_RW = 0;
    cntl_DI = 0;
    cntl_E = 1;
    PORTB = command;
    Delay_us(30);
    cntl_E = 0;
    cntl_CS1 = 0;
    cntl_CS2 = 0;
}

void   gLcdClr()
{
    unsigned    char    page, column;
    //
    for (page = 0xB8; page <= 0xBF; page++) {
        gLcdCommand(0, page);
        for (column = 0x40; column <= 0x7F; column++) {
            gLcdCommand(0, column);
            gLcdData(0, 0x00);
        }
    }
}

void   gLcdDotSet(unsigned char x, unsigned char y)    // x(0...127),
y(0...63)
{
    unsigned    char cs, pa, ca, dt;
    if (x < 64)
        cs = 1;
    else
        cs = 2;
    //
    switch (y / 8) {
    case 7:
        pa = 0xB8;
        break;
    case 6:
        pa = 0xB9;
        break;
    case 5:
```

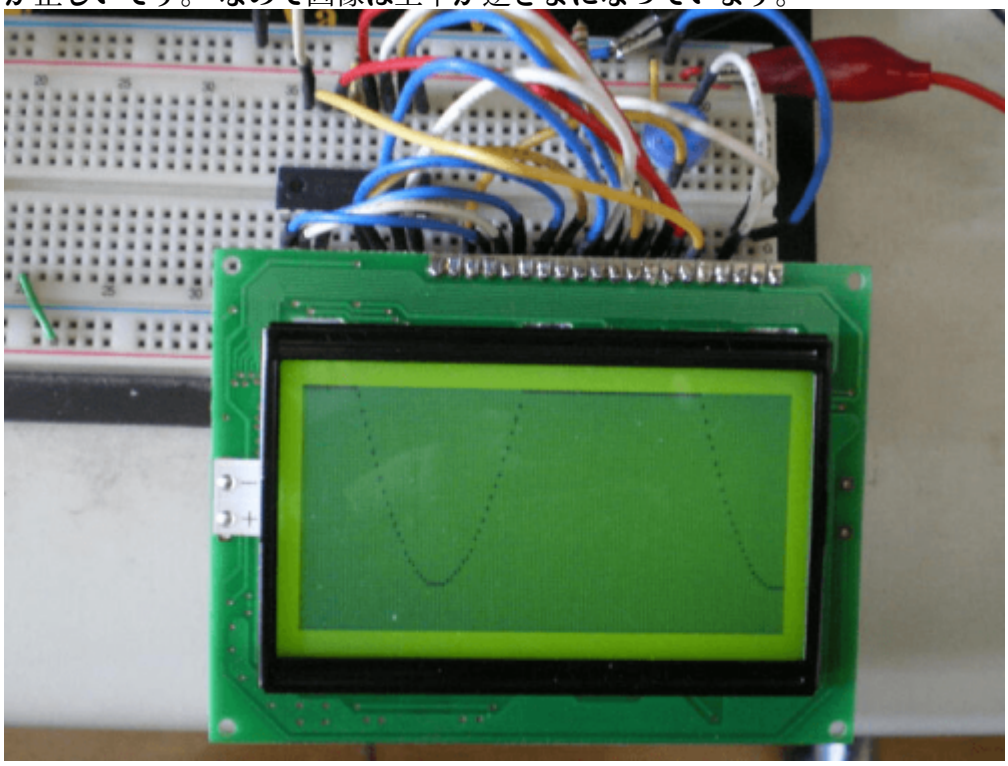
```
        pa = 0xBA;
        break;
    case 4:
        pa = 0xBB;
        break;
    case 3:
        pa = 0xBC;
        break;
    case 2:
        pa = 0xBD;
        break;
    case 1:
        pa = 0xBE;
        break;
    case 0:
        pa = 0xBF;
        break;
    }
    gLcdCommand(cs, pa);
    //
    ca = x - ((cs - 1) * 64);
    gLcdCommand(cs, ca);
    //
    switch (y / 8) {
    case 0:
        dt = y;
        break;
    case 1:
        dt = y - 8;
        break;
    case 2:
        dt = y - 16;
        break;
    case 3:
        dt = y - 24;
        break;
    case 4:
        dt = y - 32;
        break;
    case 5:
        dt = y - 40;
        break;
    case 6:
        dt = y - 48;
        break;
    case 7:
        dt = y - 56;
        break;
    }
    dt = 0b10000000 >> dt;
    gLcdData(cs, dt);
```

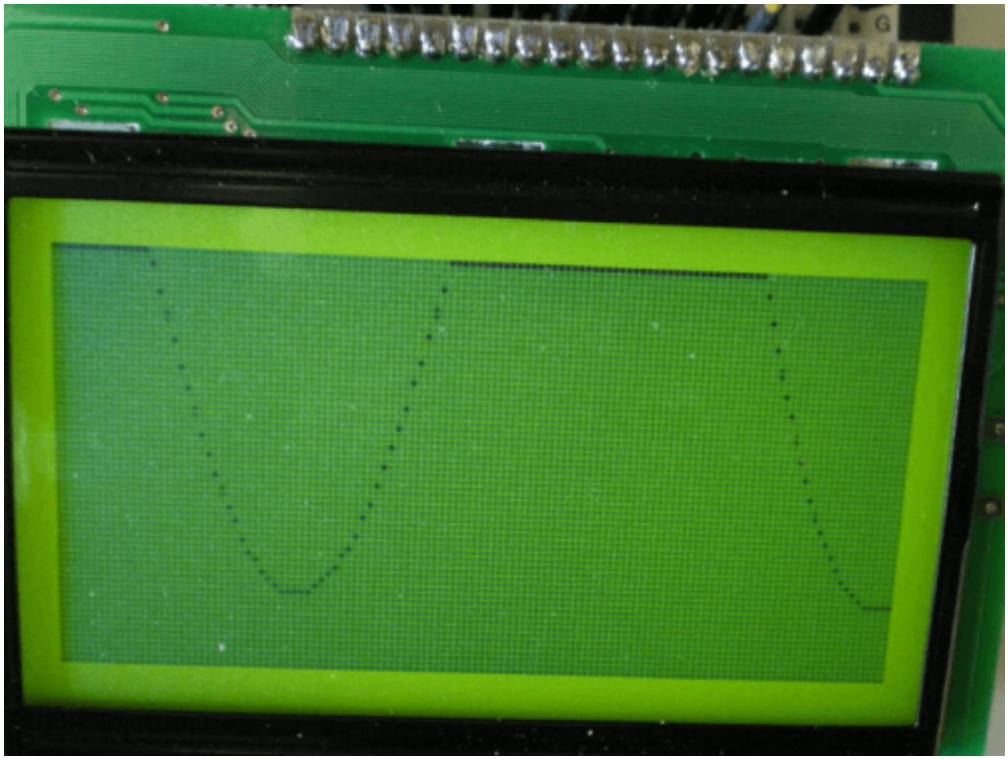
```
}

void   gLcdInit()
{
    unsigned    char    page, column;
    //
    cntl_CS1 = 0;
    cntl_CS2 = 0;
    cntl_E   = 0;
    cntl_RW  = 0;
    cntl_DI  = 0;
    //
    gLcdCommand(0, 0b00111111);
    gLcdCommand(0, 0b11000000);
    gLcdClr();
}

//*****************************************************************************
*

void   main()
{
    static    unsigned    char    buf1[64], buf2[64];
    unsigned    char    x, y, cnt;
    unsigned    int        ad;
    //
    OSCCON = 0b01110000;        //クロックは8Mhz
    CMCON  = 0b00000111;        //コンパレータは使用しない。
    ANSEL  = 0b00000001;
    TRISA  = 0b00100001;
    TRISB  = 0b00000000;
    //□□□□の初期化
    Delay_ms(100);
    gLcdInit();
    //  データバッファのクリア
    for (cnt = 0; cnt < 64; cnt++) {
        buf1[cnt] = 0;
        buf2[cnt] = 0;
    }
    while (1) {
        //  アナログデータの取り込み（最初の６４データ）
        for (cnt = 0; cnt < 64; cnt++) {
            ad = Adc_Read(0);
            buf1[cnt] = (ad >> 4) & 0b00111111;
        }
        //  アナログデータの取り込み（次の６４データ）
        for (cnt = 0; cnt < 64; cnt++) {
            ad = Adc_Read(0);
            buf2[cnt] = (ad >> 4) & 0b00111111;
        }
        //  データのドット表示（最初の６４データ）
```

```
        for (x = 0; x < 64; x++) {
            gLcdDotSet(x, buf1[x]);
        }
        //  データのドット表示（次の６４データ）
        for (x = 0; x < 64; x++) {
            gLcdDotSet(x + 64, buf2[x]);
        }
        //  画面クリア
        Delay_ms(100);
        gLcdClr();
    }
}

//*****************************************************************************
*
```

動作確認

ブレッドボードで確認しました。接続線が多いのでかなりごちゃごちゃしています。 信号には約100Hzの正弦波の半波を入力しました。 このLCDの上下の方向は、本来は20ピンの接続端子を下にしてみるのが正しいです。 なので画像は上下が逆さまになっています。





如何ですか? これを利用すればPICを使用した電子工作の活用範囲が大きく広がりますね。