超小型OLEDと ボリウムで 時計の時刻入力の実現

今回は、制御IC SSD1306を使った 超小型OLEDにて、ソフトで 反転表示を 実現します。

反転表示とは、OLED表示の 明るい部分と暗い部分を 逆転させて表示させる事です。(図.1 参照)

反転表示を何に使うかというと、OLED上で、複数の数値項目を表示していて、そのうちの一つが、編集対象になっている事を明確に表示させるためです。

例えば、時計の時刻設定を、行う場合、時、分、秒を 順次設定する場合を考えます。

12時 34分 56秒 を 設定する事にします。
押しボタン等で編集モードにして、②の時の編集で12を入力して、③の分の編集で 34を入力して、④の秒で 56を入力して、⑤で、変更完了です。

このような編集シーケンスをプログラムで実現します。

図. 1

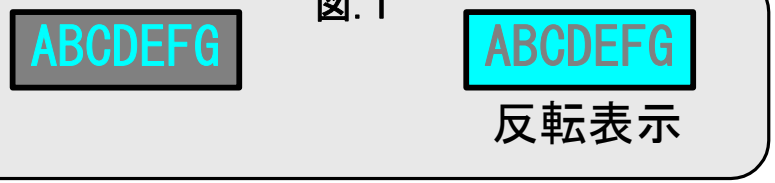


図. 2

OLEDドライバに、反転表示機能の実装

OLEDに、反転表示機能を実装するソースは半角文字FONTを読み込む dsd_OLED_sub.c です。 変更箇所は、put_char_8x16() 関数内と put_char_16x32() 関数内の ２箇所です。
このページでは、put_char_8x16() 関数内の一部分を示します。
赤い矢印で指している ~ で、bit反転しています。

void　put_char_8x16(int x, int y, char code)

```
i2c_write_7b( c_ai.adr, buf, 9 );      // 2021-08-29 変更
ptr = get_bmf_8x16_asc( code );            // ASCIIコードのフォントデータ取得
if( Rev_sw == 1 )              // ★  反転表示を行う？ 21-12-26 追加
{
        for( i=0; i<16; i++ )  // 16byte分 繰り返す
        {
                buf[i] = ~*ptr;                // bit反転して buf に格納( 反転表示のため )
                ptr++;                         // ptr更新
        }
        ptr = buf;                     // ptr を bufの先頭アドレスに変更する
}
```

```
void   put_char_16x32( int x, int y, char code )

        for( i=0; i<16; i++ )
        {
                dt = *ptr;              // ROMから 1Byte FontDataを取り出す

                if( Rev_sw == 1 )     dt = ~dt;       // ★ 21-12-26 反転表示を行う？

                Wb.w = exp2_pattern( dt );          // Byte pattern -> Word pattern変換
```

このページでは、put_char_16x32() 関数内の
一部分を示します。
赤い矢印で指している ~ で、bit反転して
ます。

要は、bit=1（明るいDot）と、bit=0（暗いDot）を、入れ替えてるだけです。

編集項目の ブリンク表示を行っている箇所です。modeが 0　の時、通常の時刻 歩進 処理です。modeが、1以上の時、編集状態で　ブリンク表示が、有効になります。　n は 1/50秒で、インクリメントされ、50に、なったら、0に戻ります。　1 秒の後半 0.5秒は 反転表示になります。

呼び出す側： OLED_Rev_Disp.c

```
    if( mode > 0 )
    {
        if( n == 0 ) set_reverse_disp( 0 );   // 正常表示
        if( n > 24 ) set_reverse_disp( 1 );   // 反転表示
    }
```

```
static  BYTE Rev_sw;   // リバース表示スイッチ ( 0=Normal 、1=Reverse )

//**************************************
//**  反転表示フラグの 設定             **
//** ---------------------------------- **
//**  sw = 0  :  ノーマル表示           **
//**  sw = 1  :  反転表示               **
//**************************************
void  set_reverse_disp( BYTE sw )
{
     Rev_sw = sw;
}
```

呼び出される側： dsd_OLED_sub.c

Rev_sw=0 で 正常表示、Rev_SW=1 で 反転表示になります。 正常表示、反転表示の設定は、set_reverse_disp() の引数で設定します。

時刻データを、どうやって入力するか

今回のマイコンには、テンキーは、付けていませんので、数値は どうやって入力するのか という事になります。
実は、この時刻等の限定的な数値入力を行うためにボリウムを付けていたのです。

時、分、秒 のアイテム間の移動は、押しボタンを使いますが、 時、分、秒 の 数値入力は、ボリウムを、回して行います。

よって 時の場合は、ボリュームを回す事により、0～23 の値を 入力出来るようにしておきます。
分と 秒は、ボリュームを回す事により 0～59 の値を、入力出来るようにしておきます。

ボリュームには、音量調整と考えた場合、Min側に、0V , Max側に Vcc 回転する中点を、A/D入力に接続します。 R8Cマイコンの A/D入力は、10bitで、0 ～ 1023 の量子化数が、出てきます。 この量子化数を、演算処理して、0～23 または、0～59 の値が出るようにすれば、時計のデータを入力できます。
A/Dの処理は メインの OLED_Rev_Disp.c 内にあります。
演算処理は、以下のように行っています。

```
dh = ad / 43; // 時の値：0 ～ 23
ae = ad;
if( ae > 1016 )    ae = 1016;
dm = ae / 17; // 分の値：0 ～ 59
```

（参考）A/D入力の使い方

今回は、マイコン内蔵の A/Dコンバータに関して少し説明します。 A/Dコンバータとは、0～5V または、0～3.3V のアナログ電圧値を、デジタル量子化数に変換します。

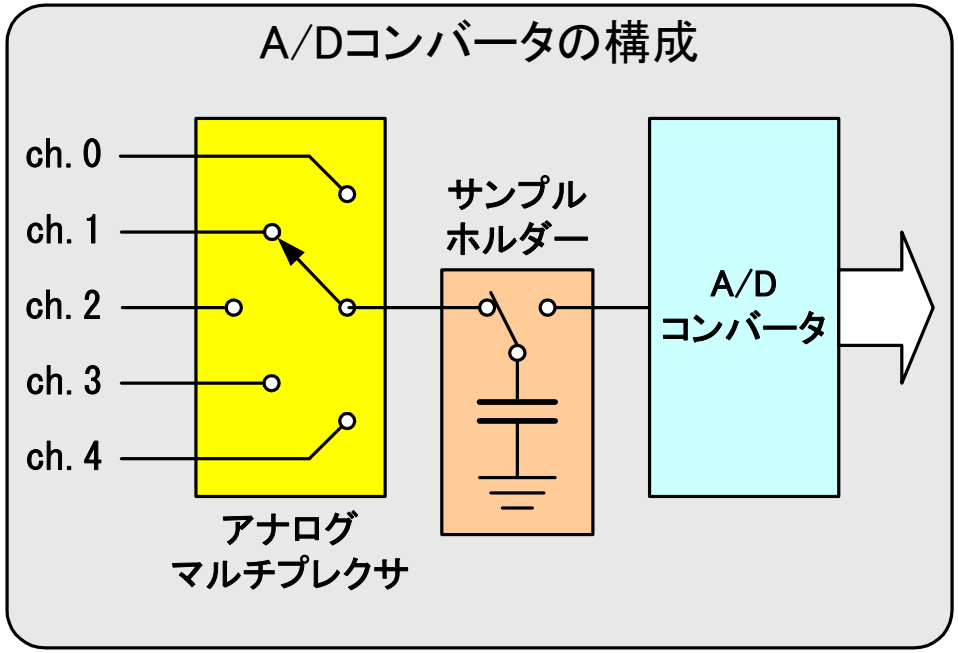
10bitの A/Dコンバータの場合 0 ～ 1023 の量子化数に変換します。 12bitの 場合は、0 ～ 4095 の 量子化数に変換します。

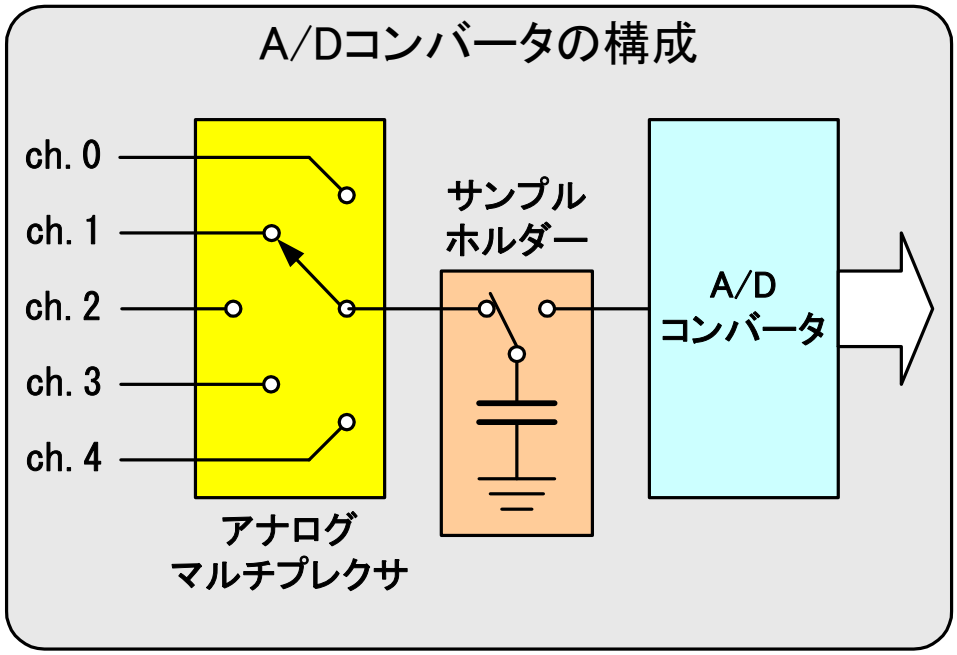
通常 10bitのA/Dコンバータを内蔵するマイコンが 多いです。 少数派ですが 12bitのA/Dコンバータを内蔵したマイコンもあります。

16bit分解能以上の A/Dコンバータは、専用の A/DコンバータICを使う事になります。

高分解能のA/Dコンバータは、ローノイズの電源や、高い安定性の基準電圧等、アナログ回路的に、難しくなります。

通常、マイコンに内蔵されている A/Dコンバータは、逐次比較型A/Dコンバータと呼ばれます。 逐次比較型A/Dコンバータの大雑把な構成を示します。 A/Dコンバータの前段にアナログマルチプレクサと、サンプルホルダーの２つの回路があります。





A/D変換には 一連のシーケンスがあります。

① アナログマルチプレクサ（ FETのアナログスイッチ ）で、入力チャネルを選択します。　この際、サンプルホルダも同時にアナログマルチプレクサ側に接続されます。

② A/D変換スタートを行います。
同時にサンプルホルダーが、A/Dコンバータ側に切り替わります。
（ A/D変換に 10us程度かかります。）

③ その後、CPUは、A/D変換が終了したか確認します。 A/D変換器から、CPUにA/D変換終了の割込みを かける事も出来ます。

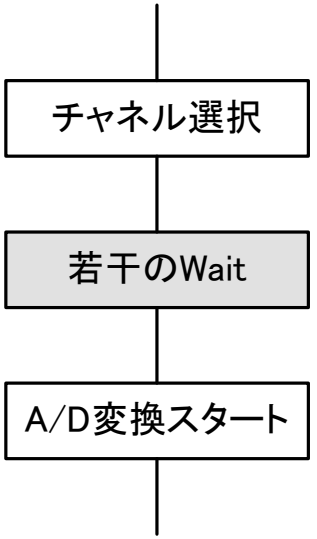
④ A/D変換器内の レジスタに入っている量子化データを CPUが 読み出します。

ここで、サンプルホルダーって 何やってるの？ と思われる方もおられるかもしれません。　理由は、②のA/D変換に 10us程度の短い時間ではありますが、時間がかかるためです。　その間に A/Dコンバータに、入力される電圧が動くと 誤差が 出るので、サンプルホルダーのキャパシタで、電圧を保持しているのです。

しかし、サンプルホルダーが、悪影響を及ぼす場合もあります。

ch.0 が比較的高い電圧で、次の ch.1 が、低い電圧である場合、サンプルホルダーに蓄えられた電荷が ch.1に切り替えた瞬間にch.1側の外部回路に 流れ出す場合があります。
特に、各チャネルに接続される外部回路の内部抵抗が高いと 放電に時間が かかるため、次のA/D変換に 影響を及ぼします。
よってセンサ等の出力インピーダンスが高い場合は、OPAMP等のボルテージフォロアを入れて下さい。

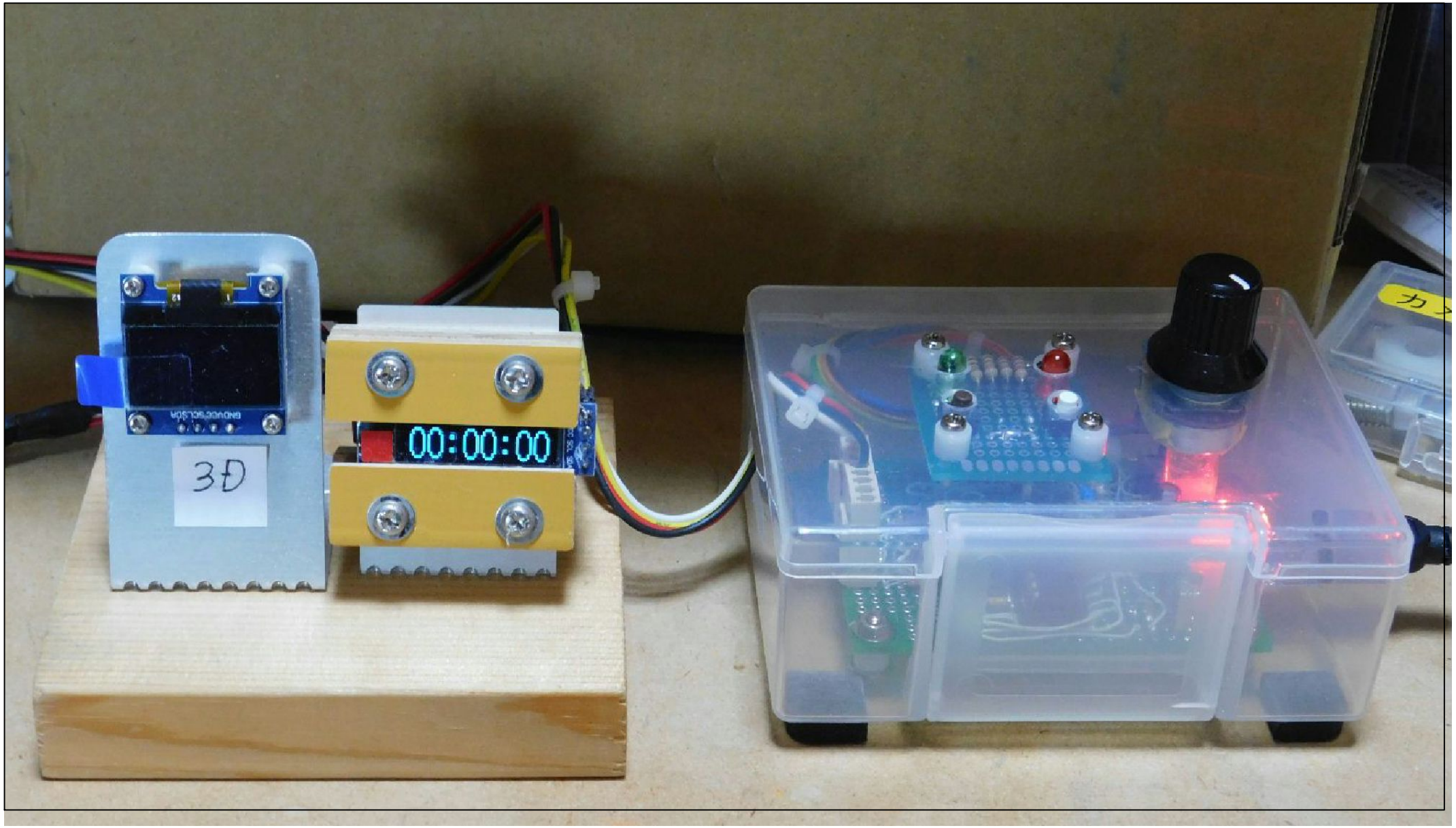
または、事後処理的に使う事が多いですがチャネル選択を行ったあと、A/D変換をスタートする前に 若干の Wait を 入れるという方法もあります。

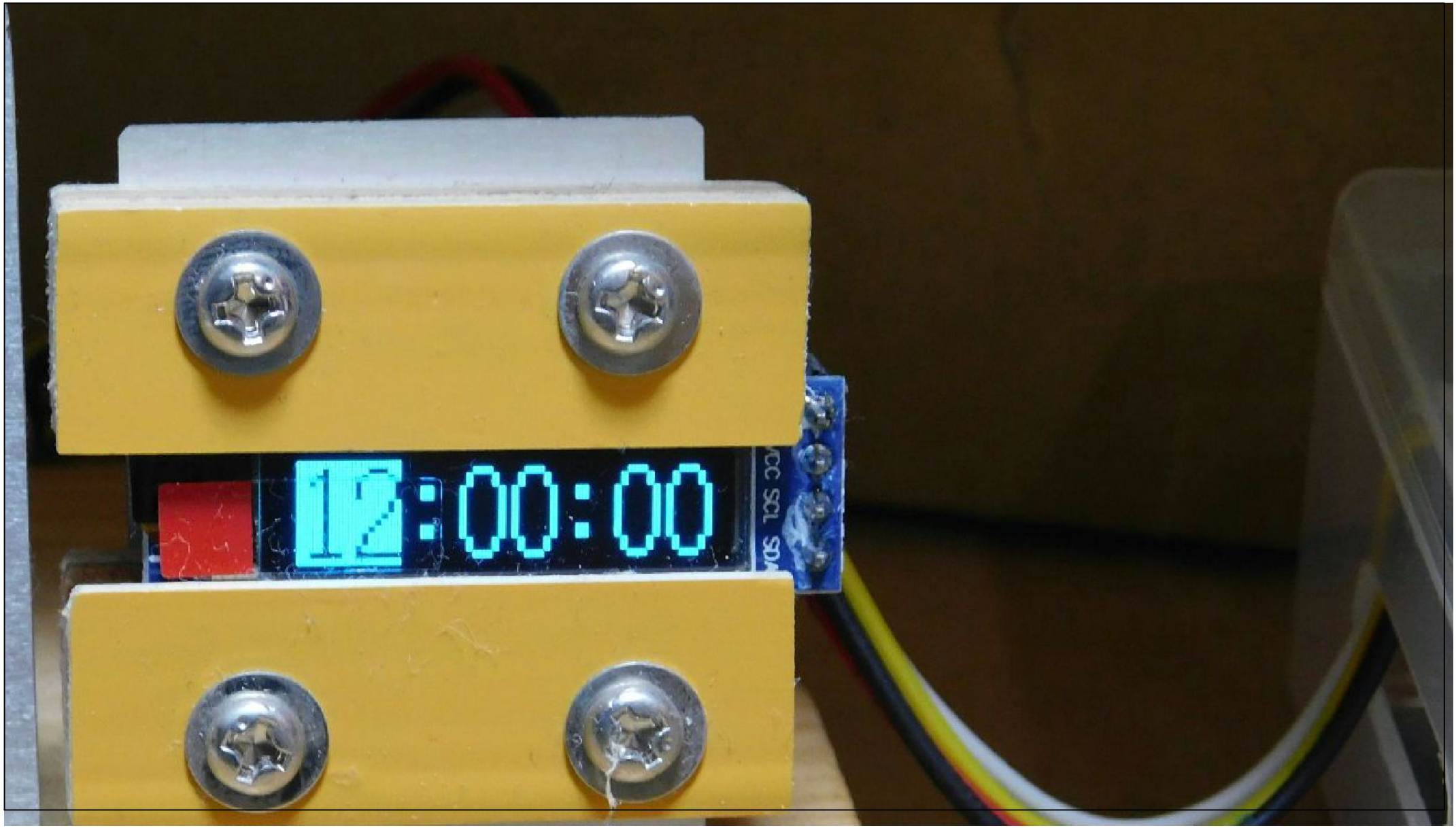


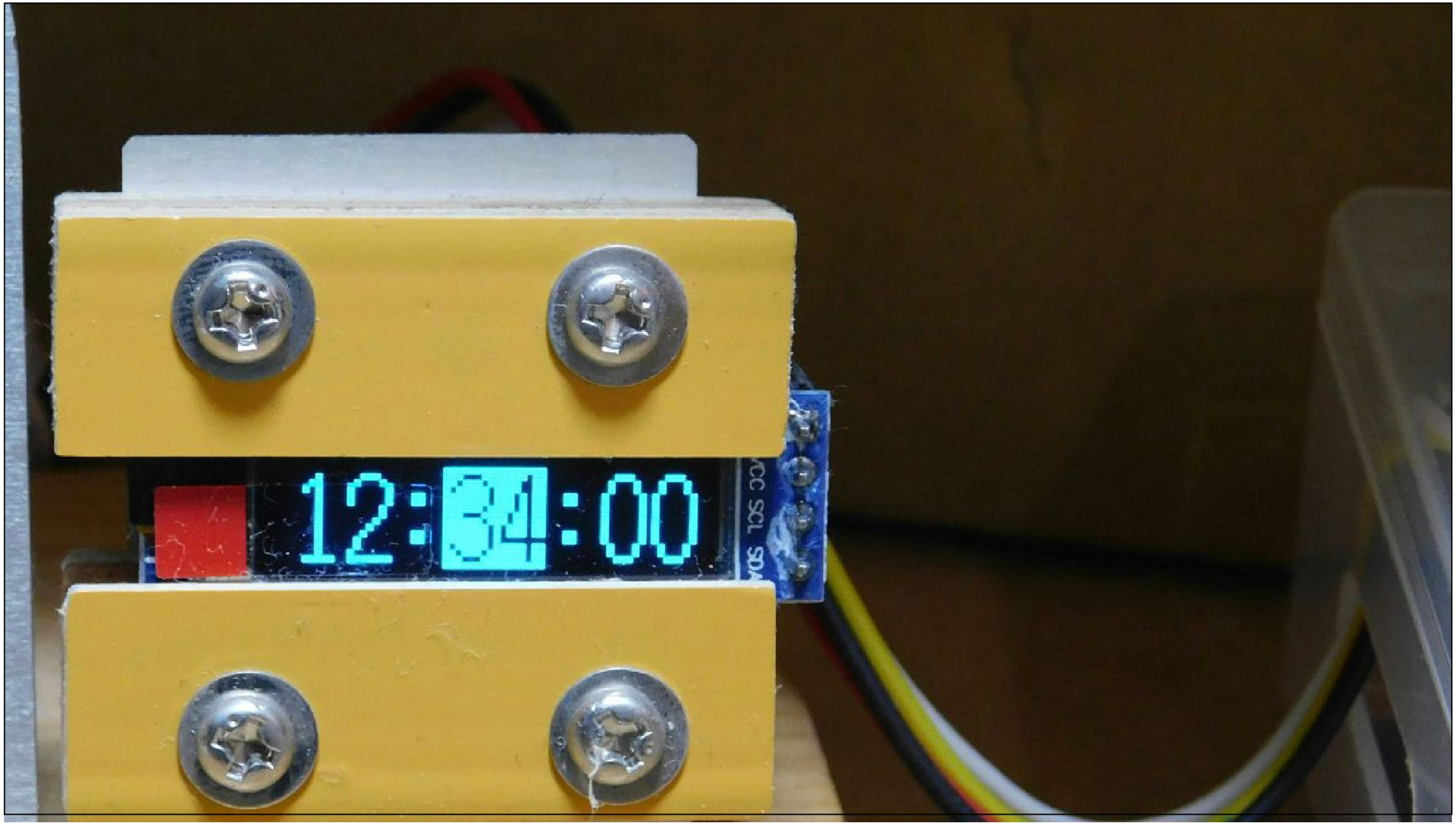
この間に、サンプルホルダーの電荷の放電が終わる。

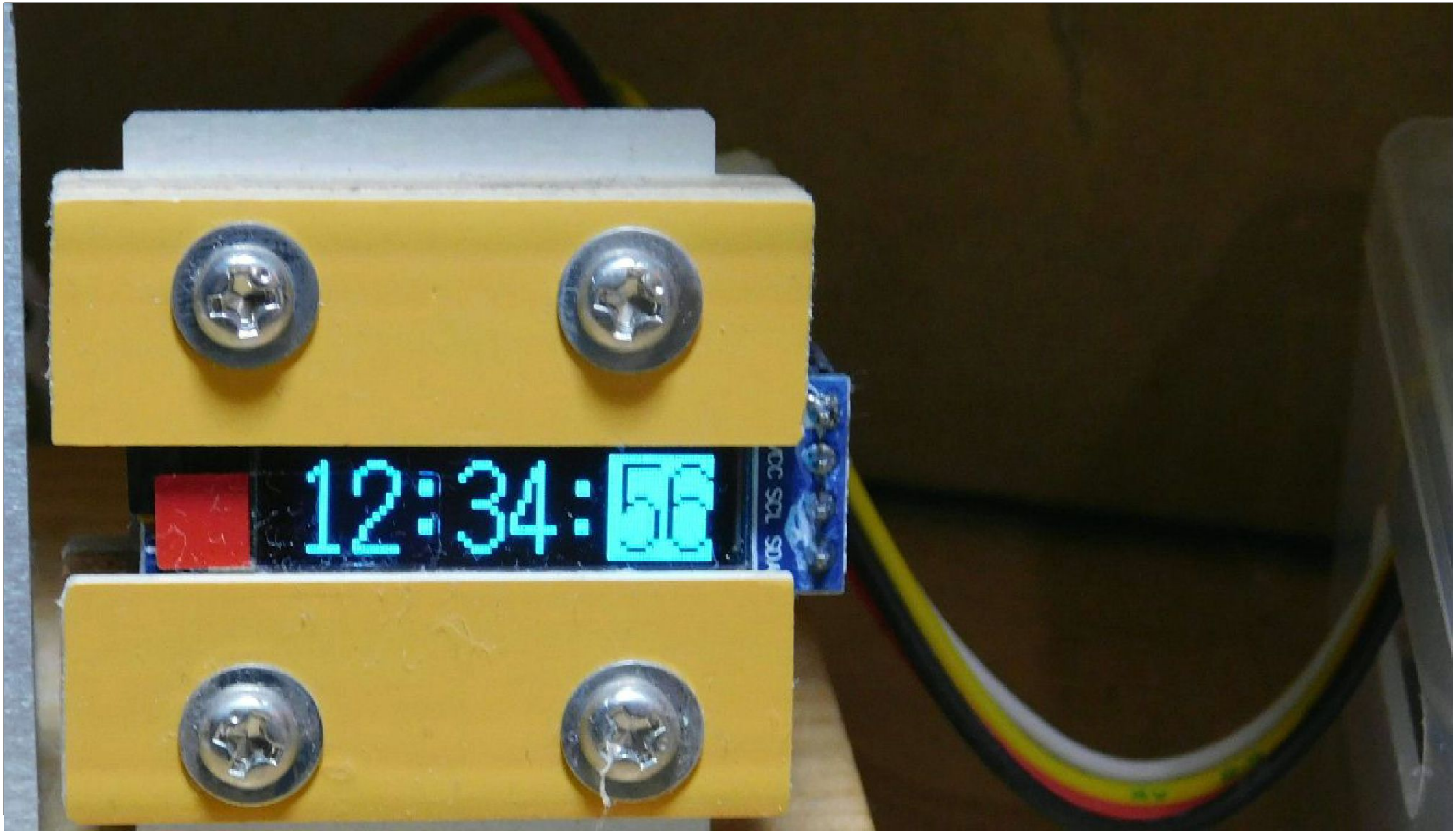
では、動画に移ります。

A/Dコンバータの 量子化数のダンプ表示と時計の時刻設定を想定したデモを行います。


3D
00:00:00


17:00:00
SCL


12:34:00
VCC SCL SDA


12:34:
SCL
SDA


12:34:56
VCC SCL