

ミニ・シーケンサ(SDカード対応)(PIC18F2550)

概要

ある決められた順番や時間で、複数の電気機器のON/OFF制御を、手軽に出来るシーケンサを製作しました。

<仕様>

- 制御ポートは、8チャンネルとする。
- シーケンス(動作)は、テキストエディタで簡単に設定できることとする□(SDカードに記録)
- 時間指定は、秒単位とする。(開始時からの相対時間、時分秒で指定)
- アクティブモード(high/low)の指定を可能とする。

動作原理

1. シーケンス(動作)を指定するために、テキストエディタ(例えば、メモ帳、秀丸など)で、シーケンス設定ファイルを作成します。(ファイル名は、“sequence.txt” 固定)

<アクティブモードの設定>

出力ポートに接続されるインタフェースによっては、アクティブ・ハイ(1)で動作したり、アクティブ・ロー(0)で動作したりするものがあるので、どちらで動作するかを設定します。

```
$DEF\\ ON=0,OFF=1□□□□この例では、アクティブ・ローを指定している。 //< 初期値の設定>\\ $INIT
```

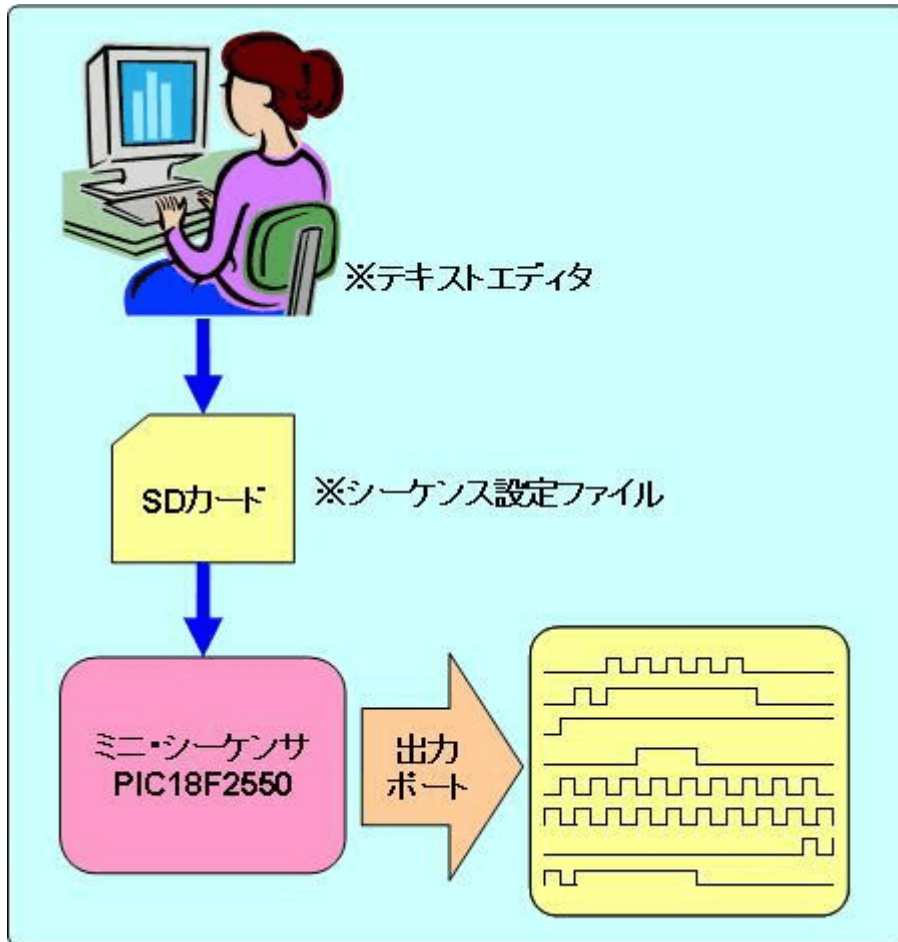
```
CH0=OFF,CH1=OFF,CH2=OFF,CH3=OFF,CH4=OFF,CH5=OFF,CH6=OFF,CH7=OFF
```

<シーケンスの開始>

```
$START\\ 00:00:00,CH0=ON□□□□この例では、開始直後に、CH0をONにしている□\\
```

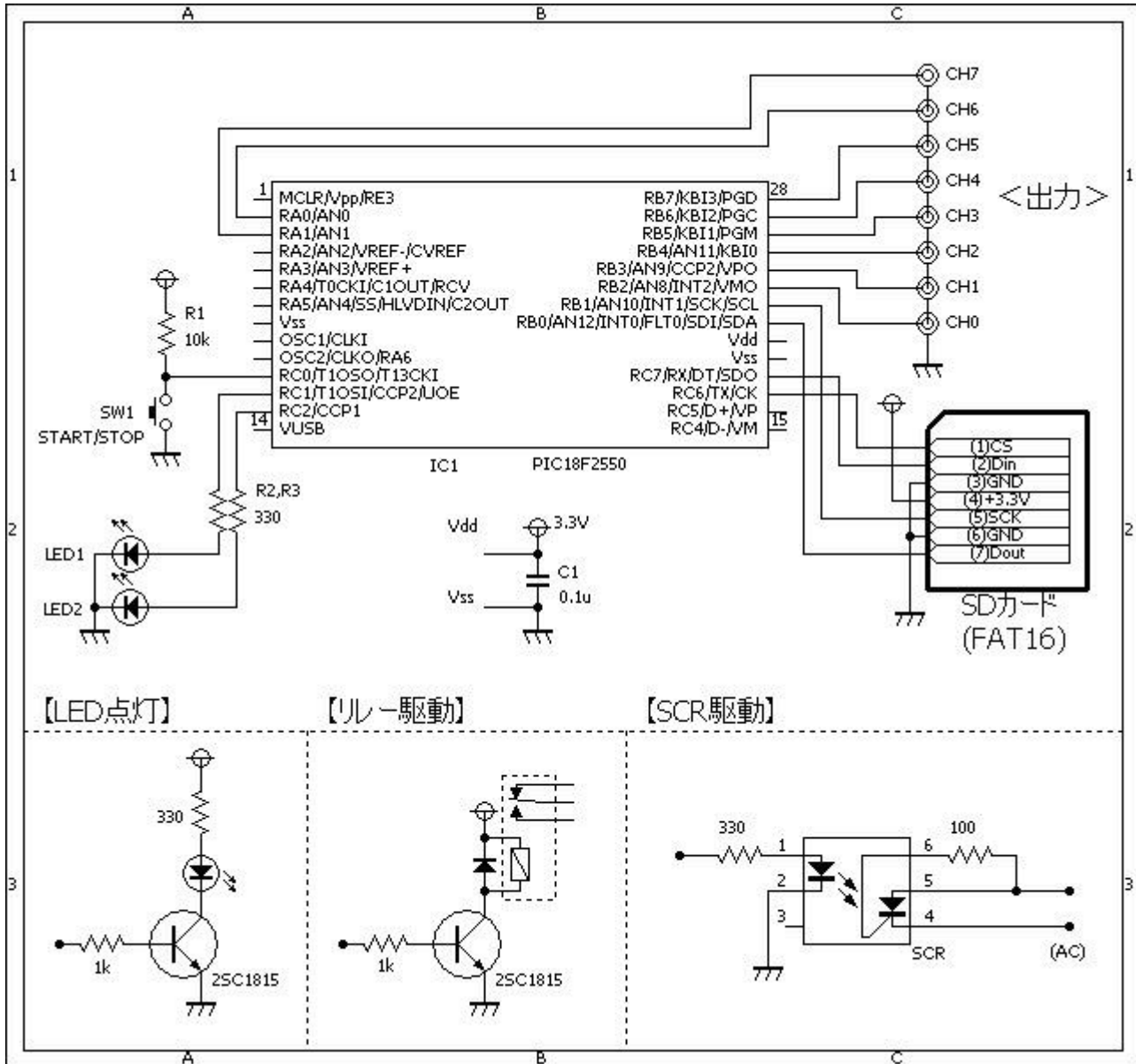
```
00:00:10,CH5=ON,CH7=ON□□□□この例では、開始10秒後に、CH5とCH7をONにしている。 //< シーケンスの停止>\\ $STOP
```

2. SW1を押下すると、シーケンス制御を開始します□(LED2点灯)
3. \$STOPで、シーケンス制御を停止します□(LED2消灯)
4. 制御途中で、シーケンスを停止する場合は、再度SW1を押下します□(LED2消灯)



<処理の流れ>

回路図



ソースコード

[mini_sequencer.c](#)

```

//*****
*
/*
『ミニ・シーケンサ(SDC対応)』

機能概要
シーケンス設定ファイルで設定した内容に応じて、出力ポートを制御する。
<設定内容>
####$DEF
    出力ポートのアクティブモードの設定 active high active low
####$INIT
    開始前の出力ポートの初期値の設定
####$START
    ポートへの出力時間と、ポートの####の設定

```

```
□□□□$STOP
    制御の終了の設定

    シーケンス設定ファイルのフォーマット (設定例)
□□$DEF
□□ON=0,OFF=1
□□$INIT
□□CH0=OFF,CH1=OFF,CH2=OFF,CH3=OFF,CH4=OFF,CH5=OFF,CH6=OFF,CH7=OFF
□□$START
□□00:00:00,CH0=ON
□□00:00:01,CH1=ON
□□00:00:02,CH2=ON
□□00:00:03,CH3=ON
□□00:00:04,CH4=ON
□□00:00:05,CH5=ON
□□00:00:06,CH6=ON
□□00:00:07,CH7=ON
□□00:00:08,CH7=OFF
□□00:00:09,CH6=OFF
□□00:00:10,CH5=OFF
□□00:00:11,CH4=OFF
□□00:00:12,CH3=OFF
□□00:00:13,CH2=OFF
□□00:00:14,CH1=OFF
□□00:00:15,CH0=OFF
□□00:00:16,CH0=ON
□□00:00:17,CH1=ON,CH0=OFF
□□00:00:18,CH2=ON,CH1=OFF
□□00:00:19,CH3=ON,CH2=OFF
□□00:00:20,CH4=ON,CH3=OFF
□□00:00:21,CH5=ON,CH4=OFF
□□00:00:22,CH6=ON,CH5=OFF
□□00:00:23,CH7=ON,CH6=OFF
□□00:00:24,CH7=OFF
□□00:00:25,CH7=ON
□□00:00:26,CH7=OFF,CH6=ON
□□00:00:27,CH6=OFF,CH5=ON
□□00:00:28,CH5=OFF,CH4=ON
□□00:00:29,CH4=OFF,CH3=ON
□□00:00:30,CH3=OFF,CH2=ON
□□00:00:31,CH2=OFF,CH1=ON
□□00:00:32,CH1=OFF,CH0=ON
□□00:00:33,CH0=OFF
□□00:00:34,CH0=ON,CH1=ON,CH2=ON,CH3=ON,CH4=ON,CH5=ON,CH6=ON,CH7=ON
□00:00:35,CH0=OFF,CH1=OFF,CH2=OFF,CH3=OFF,CH4=OFF,CH5=OFF,CH6=OFF,CH7=OFF
□□$STOP
*/
//*****
*
```

#define SW PORTC.F0

```
#define LED1 PORTC.F1
#define LED2 PORTC.F2

#define CR 0x0d
#define LF 0x0a

#define MODE_DEF 0
#define MODE_INIT 1
#define MODE_START 2
#define MODE_STOP 3

#define CH0 PORTB.F2
#define CH1 PORTB.F3
#define CH2 PORTB.F4
#define CH3 PORTB.F5
#define CH4 PORTB.F6
#define CH5 PORTB.F7
#define CH6 PORTA.F0
#define CH7 PORTA.F1

//*****
*

static long clock;

void interrupt()
{
    if (PIR1.CCP1IF == 1) {
        PIR1.CCP1IF = 0;
        //
        clock++;
        LED1 = ~LED1;
    }
}

//*****
*

void init_sdc()
{
    static short cnt;
    //□□□□□□□□の初期化
    Spi_Init_Advanced(MASTER_OSC_DIV64, DATA_SAMPLE_MIDDLE,
CLK_IDLE_LOW, LOW_2_HIGH);
    if (Mmc_Fat_Init(&PORTC, 6)) {
        while (1) {
            LED2 = 1;
            Delay_ms(100);
            LED2 = 0;
            Delay_ms(100);
        }
    }
}
```

```
    }
    Spi_Init_Advanced(MASTER_OSC_DIV16, DATA_SAMPLE_MIDDLE,
CLK_IDLE_LOW, LOW_2_HIGH);
    for (cnt = 0; cnt < 5; cnt++) {
        LED2 = 1;
        Delay_ms(300);
        LED2 = 0;
        Delay_ms(300);
    }
}

//*****
*

void SwitchONcheck()
{
    while (Button(&PORTC, 0, 1, 0) == 0)
        ;
    while (Button(&PORTC, 0, 1, 1) == 0)
        ;
}

//*****
*

void sequenceProc()
{
    //変数の定義
    static char buf[80], *pnt, tmp[8];
    static unsigned long fsize, length;
    static unsigned short character, cnt, mode, lineCnt, hh,
mm, ss, hhTmp, mmTmp, ssTmp;
    static unsigned short onData, offData, outputData;
    //変数の初期化
    clock = 0;
    mode = MODE_STOP;
    lineCnt = 0;
    onData = 1;
    offData = 0;
    //のファイルのオープン
    Mmc_Fat_Assign("sequence.txt", 0);
    Mmc_Fat_Reset(&fsize);
    length = 0;
    //
    while (1) {
        //設定ファイルから1行分のデータを読み込む
        for (cnt = 0; cnt < 80; cnt++) {
            Mmc_Fat_Read(&character);
            length++;
            if (character == ' ') {
```

```
        cnt--;  
        continue;  
    }  
    if (character == CR) {  
        cnt--;  
        continue;  
    }  
    if (character == LF) {  
        buf[cnt] = 0x00;  
        break;  
    }  
    buf[cnt] = character;  
}  
lineCnt++;  
//定義かを判断する。  
pnt = strstr(buf, "$DEF");  
if (pnt != 0x00) {  
    mode = MODE_DEF;  
    continue;  
}  
//初期化かを判断する。  
pnt = strstr(buf, "$INIT");  
if (pnt != 0x00) {  
    mode = MODE_INIT;  
    continue;  
}  
//開始かを判断する。  
pnt = strstr(buf, "$START");  
if (pnt != 0x00) {  
    mode = MODE_START;  
    continue;  
}  
//停止かを判断する。  
pnt = strstr(buf, "$STOP");  
if (pnt != 0x00) {  
    mode = MODE_STOP;  
    return;  
}  
//定義の設定  
if (mode == MODE_DEF) {  
    pnt = strstr(buf, "ON=");  
    if (pnt != 0x00) {  
        onData = *(pnt + 3) - '0';  
    }  
    pnt = strstr(buf, "OFF=");  
    if (pnt != 0x00) {  
        offData = *(pnt + 4) - '0';  
    }  
    continue;  
}  
//初期値の設定
```

```
if (mode == MODE_INIT) {
    tmp[0] = 'C';
    tmp[1] = 'H';
    tmp[2] = '?';
    tmp[3] = '=';
    tmp[4] = 0x00;
    for (cnt = 0; cnt < 8; cnt++) {
        tmp[2] = cnt + '0';
        pnt = strstr(buf, tmp);
        if (pnt != 0x00) {
            if (strncmp((pnt + 4), "ON", 2) == 0) {
                outputData = onData;
            } else {
                outputData = offData;
            }
            switch (cnt) {
                case 0:
                    CH0 = outputData;
                    break;
                case 1:
                    CH1 = outputData;
                    break;
                case 2:
                    CH2 = outputData;
                    break;
                case 3:
                    CH3 = outputData;
                    break;
                case 4:
                    CH4 = outputData;
                    break;
                case 5:
                    CH5 = outputData;
                    break;
                case 6:
                    CH6 = outputData;
                    break;
                case 7:
                    CH7 = outputData;
                    break;
            }
        }
        continue;
    }
}
//制御を行う。
if (mode == MODE_START) {
    //制御時間を取得する。
    if (buf[2] == ':') {
        buf[2] = 0x00;
    }
}
```

```
    hh = atoi(buf);
    buf[2] = ':';
}
if (buf[5] == ':') {
    buf[5] = 0x00;
    mm = atoi(&buf[3]);
    buf[5] = ',';
}
if (buf[8] == ',') {
    buf[8] = 0x00;
    ss = atoi(&buf[6]);
    buf[8] = ',';
}
//制御時間を確認する。
while (1) {
    hhTmp = clock / 36000;
    mmTmp = (clock - (36000 * (long)hhTmp)) / 600;
    ssTmp = (clock - (36000 * (long)hhTmp) - (600 *
(long)mmTmp)) / 10;
    if ((hh == hhTmp) && (mm == mmTmp) && (ss == ssTmp))
        break;
    if (SW == 0) {
        while (Button(&PORTC, 0, 1, 1) == 0)
            ;
        return;
    }
}
//ポートの制御を行う。
tmp[0] = 'C';
tmp[1] = 'H';
tmp[2] = '?';
tmp[3] = '=';
tmp[4] = 0x00;
for (cnt = 0; cnt < 8; cnt++) {
    tmp[2] = cnt + '0';
    pnt = strstr(buf, tmp);
    if (pnt != 0x00) {
        if (strncmp((pnt + 4), "ON", 2) == 0) {
            outputData = onData;
        } else {
            outputData = offData;
        }
    }
    switch (cnt) {
    case 0:
        CH0 = outputData;
        break;
    case 1:
        CH1 = outputData;
        break;
    case 2:
        CH2 = outputData;
```

```
        break;
    case 3:
        CH3 = outputData;
        break;
    case 4:
        CH4 = outputData;
        break;
    case 5:
        CH5 = outputData;
        break;
    case 6:
        CH6 = outputData;
        break;
    case 7:
        CH7 = outputData;
        break;
    }
}

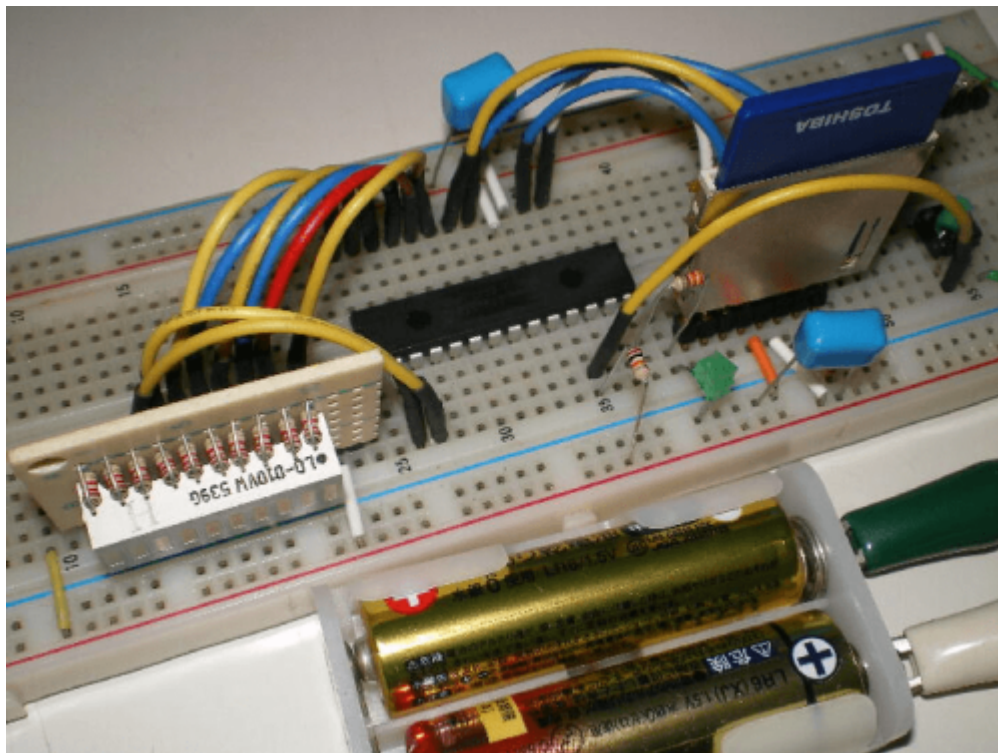
//*****
*

void main()
{
    //コンパレータは使用しない。
    CMCON = 0b00000111;
    //□□□変換の設定
    ADCON1.PCFG3 = 1;
    ADCON1.PCFG2 = 1;
    ADCON1.PCFG1 = 1;
    ADCON1.PCFG0 = 1;
    //
    OSCCON.IRCF2 = 1;
    OSCCON.IRCF1 = 1;
    OSCCON.IRCF0 = 1;
    //ポートの設定
    TRISA = 0b0000000000;
    TRISB = 0b00000001;
    TRISC = 0b00000001;
    //
    CH0 = 0;
    CH1 = 0;
    CH2 = 0;
    CH3 = 0;
    CH4 = 0;
    CH5 = 0;
}
```

```
CH6 = 0;
CH7 = 0;
//TIMER1の設定
PIE1.TMR1IE = 0;
PIR1.TMR1IF = 0;
T1CON.T1CKPS0 = 1;
T1CON.T1CKPS1 = 1;
T1CON.NOT_T3SYNC = 1;
T1CON.TMR1CS = 0;
T1CON.TMR1ON = 0;
TMR1L = 0;
TMR1H = 0;
//CCPの設定
PIE1.CCP1IE = 1;
PIR1.CCP1IF = 0;
CCP1CON.CCP1M3 = 1;
CCP1CON.CCP1M2 = 0;
CCP1CON.CCP1M1 = 1;
CCP1CON.CCP1M0 = 1;
CCPR1L = 0xA8; // 0.1sec...(1÷8000000)*4*8*25000
CCPR1H = 0x61; //
//□□□□□□□□の初期化
init_sdc();
// 割り込みを許可する。
INTCON.PEIE = 1;
INTCON.GIE = 1;
//
T1CON.TMR1ON = 1;
//
while (1) {
    SwitchONcheck();
    LED2 = 1;
    sequenceProc();
    LED2 = 0;
}
}

//*****
*
```

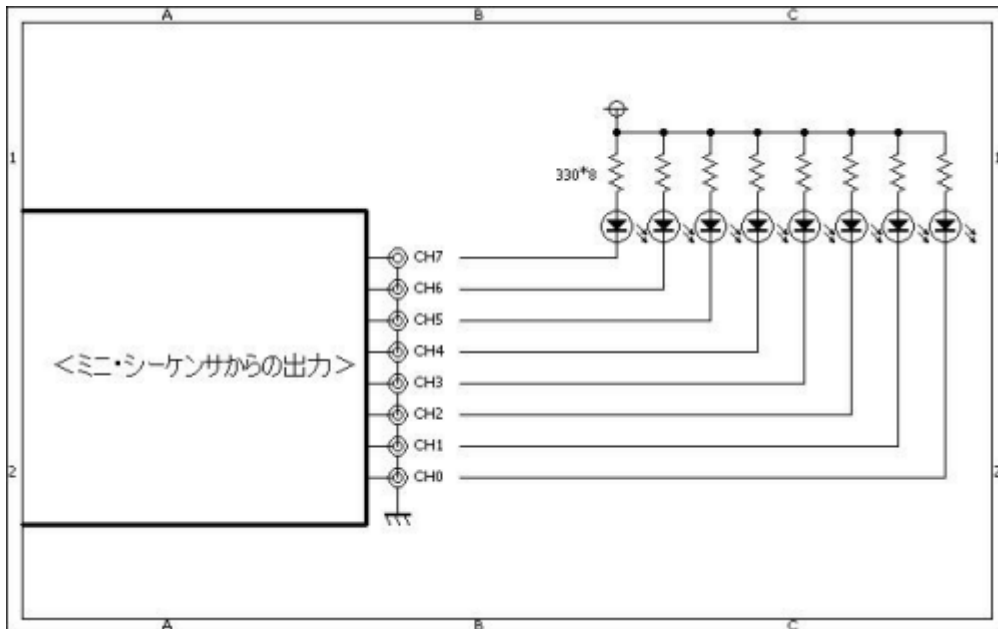
動作確認



出力ポート(CH0~CH7)に、LEDを接続して、動作を確認しました。右側から□CH0□CH1□CH2□CH3□CH4□CH5□CH6□CH7□未接続、未接続です。



LED接続時の回路図です。(参考) アクティブ・ロウ(active low)で、LEDが点灯します。



<シーケンス設定ファイルの例>

```

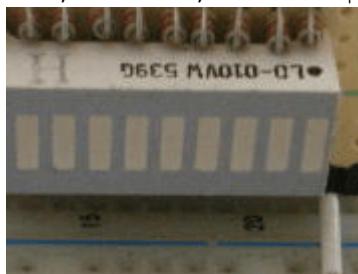
$DEF
ON=0, OFF=1
$INIT
CH0=OFF, CH1=OFF, CH2=OFF, CH3=OFF, CH4=OFF, CH5=OFF, CH6=OFF, CH7=OFF
$START
00:00:00, CH0=ON
00:00:01, CH1=ON
00:00:02, CH2=ON
00:00:03, CH3=ON
00:00:04, CH4=ON
00:00:05, CH5=ON
00:00:06, CH6=ON
00:00:07, CH7=ON
00:00:08, CH7=OFF
00:00:09, CH6=OFF
00:00:10, CH5=OFF
00:00:11, CH4=OFF
00:00:12, CH3=OFF
00:00:13, CH2=OFF
00:00:14, CH1=OFF
00:00:15, CH0=OFF
00:00:16, CH0=ON
00:00:17, CH1=ON, CH0=OFF
00:00:18, CH2=ON, CH1=OFF
00:00:19, CH3=ON, CH2=OFF
00:00:20, CH4=ON, CH3=OFF
00:00:21, CH5=ON, CH4=OFF
00:00:22, CH6=ON, CH5=OFF
00:00:23, CH7=ON, CH6=OFF
00:00:24, CH7=OFF
00:00:25, CH7=ON
00:00:26, CH7=OFF, CH6=ON

```

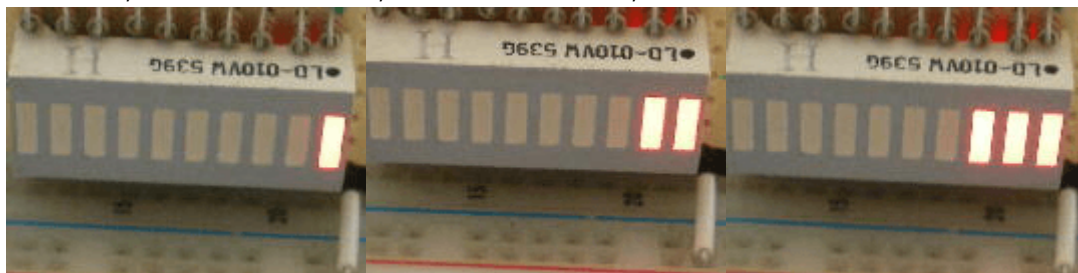
```
00:00:27,CH6=OFF,CH5=ON  
00:00:28,CH5=OFF,CH4=ON  
00:00:29,CH4=OFF,CH3=ON  
00:00:30,CH3=OFF,CH2=ON  
00:00:31,CH2=OFF,CH1=ON  
00:00:32,CH1=OFF,CH0=ON  
00:00:33,CH0=OFF  
00:00:34,CH0=ON,CH1=ON,CH2=ON,CH3=ON,CH4=ON,CH5=ON,CH6=ON,CH7=ON  
00:00:35,CH0=ON,CH1=OFF,CH2=ON,CH3=OFF,CH4=ON,CH5=OFF,CH6=ON,CH7=OFF  
00:00:36,CH0=OFF,CH1=ON,CH2=OFF,CH3=ON,CH4=OFF,CH5=ON,CH6=OFF,CH7=ON  
00:00:37,CH0=OFF,CH1=OFF,CH2=OFF,CH3=OFF,CH4=OFF,CH5=OFF,CH6=OFF,CH7=OFF  
$STOP
```

上記のシーケンス設定ファイルで動作させて見ました。

```
$DEF ON=0,OFF=1 $INIT  
CH0=OFF,CH1=OFF,CH2=OFF,CH3=OFF,CH4=OFF,CH5=OFF,CH6=OFF,CH7=OFF $START
```



00:00:00,CH0=ON 00:00:01,CH1=ON 00:00:02,CH2=ON



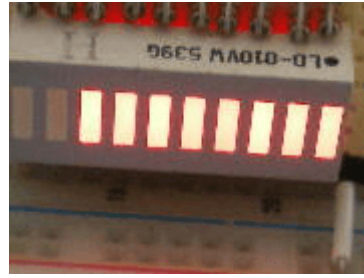
00:00:03,CH3=ON



00:00:04,CH4=ON 00:00:05,CH5=ON 00:00:06,CH6=ON



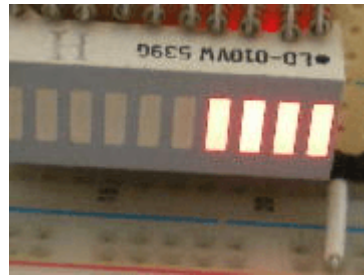
00:00:07,CH7=ON



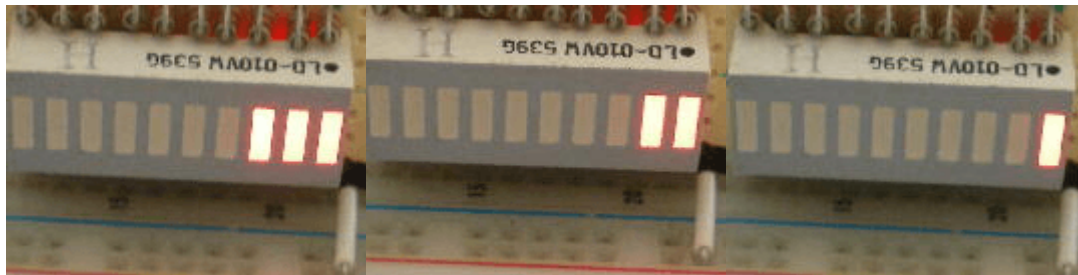
00:00:08,CH7=OFF 00:00:09,CH6=OFF 00:00:10,CH5=OFF



00:00:11,CH4=OFF

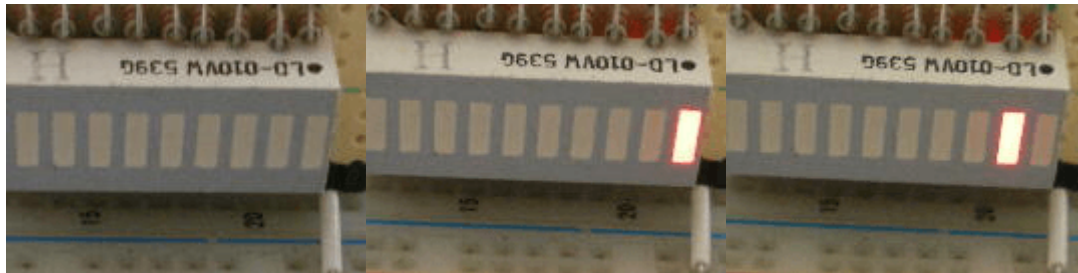


00:00:12,CH3=OFF 00:00:13,CH2=OFF 00:00:14,CH1=OFF



00:00:15,CH0=OFF

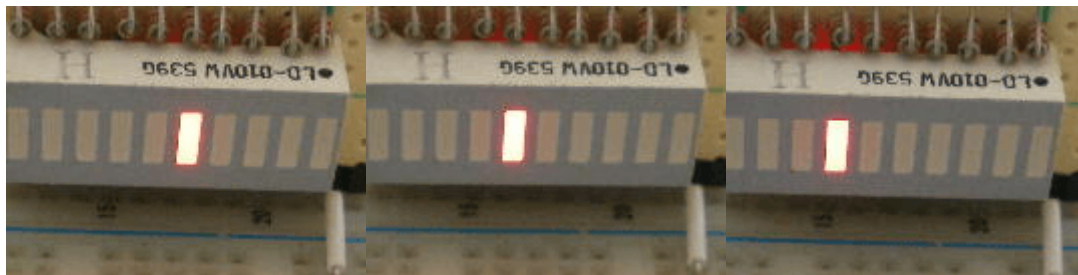
00:00:16,CH0=ON 00:00:17,CH1=ON,CH0=OFF 00:00:18,CH2=ON,CH1=OFF



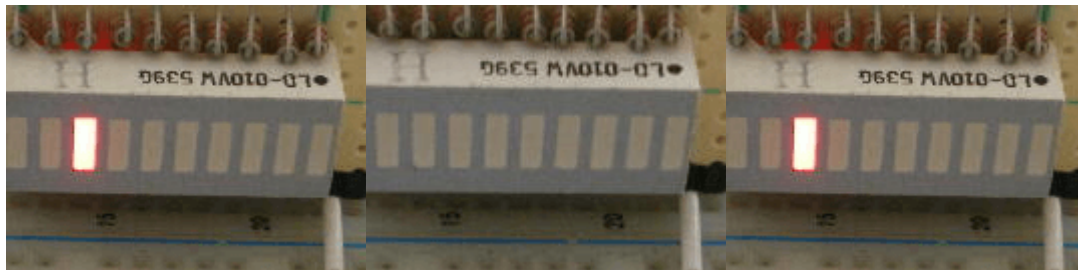
00:00:19,CH3=ON,CH2=OFF 00:00:20,CH4=ON,CH3=OFF



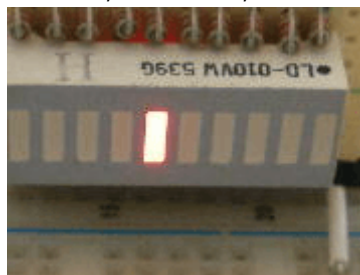
00:00:21,CH5=ON,CH4=OFF 00:00:22,CH6=ON,CH5=OFF



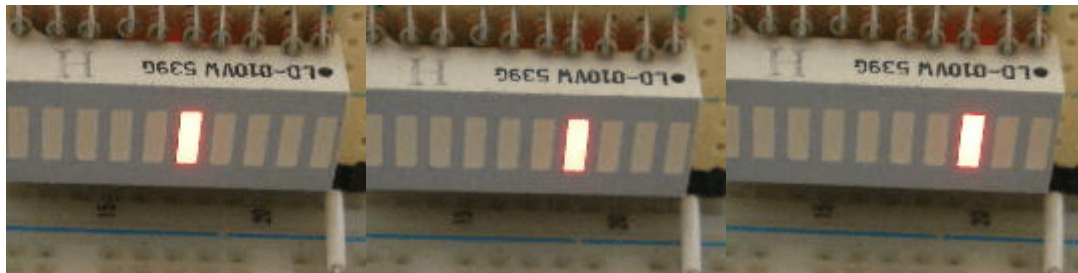
00:00:23,CH7=ON,CH6=OFF 00:00:24,CH7=OFF 00:00:25,CH7=ON 00:00:26,CH7=OFF,CH6=ON



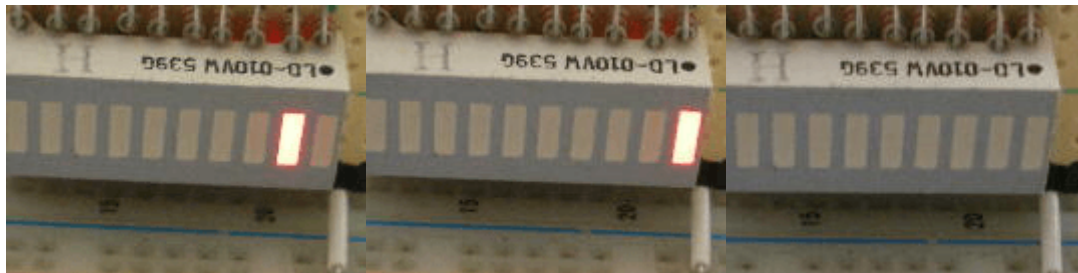
00:00:27,CH6=OFF,CH5=ON 00:00:28,CH5=OFF,CH4=ON



00:00:29,CH4=OFF,CH3=ON 00:00:30,CH3=OFF,CH2=ON



00:00:31,CH2=OFF,CH1=ON 00:00:32,CH1=OFF,CH0=ON 00:00:33,CH0=OFF
00:00:34,CH0=ON,CH1=ON,CH2=ON,CH3=ON,CH4=ON,CH5=ON,CH6=ON,CH7=ON





00:00:35,CH0=ON,CH1=OFF,CH2=ON,CH3=OFF,CH4=ON,CH5=OFF,CH6=ON,CH7=OFF
 00:00:36,CH0=OFF,CH1=ON,CH2=OFF,CH3=ON,CH4=OFF,CH5=ON,CH6=OFF,CH7=ON
 00:00:37,CH0=OFF,CH1=OFF,CH2=OFF,CH3=OFF,CH4=OFF,CH5=OFF,CH6=OFF,CH7=OFF



如何ですか? 今回は、出力機能専用のシーケンサですが、入力機能を取り入れると、更に高度なシーケンサが出来ます。例えば、入力信号(CH10)が、ONのときに出力信号(CH5とCH8)をOFFにする等。。。また、時間処理では、内臓のクロック8MHzを使用していますが、RTC(リアルタイムクロック)を実装すれば、精度も向上し、更に開始からの相対時間ではなく、絶対時間で動作も可能となります。時計ユニット(RTC)

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:otherpic:177&rev=1588242348>

Last update: 2025/10/17 14:27

