

簡易ラジオ受信周波数表示ユニット(7セグ4桁)

概要

今迄に多くの「周波数カウンタ」を製作しました。そして「周波数カウンタ」の機能の一部として、“ラジオ受信周波数表示←455kHz>”を実装しました。

今回は、ラジオ受信周波数表示に特化したユニットを製作してみました。

<仕様>

- 表示周波数は、ラジオの中間周波数(455kHz)を差し引いた値(ラジオ受信周波数)とします。
- 表示周波数の単位は、kHzとし□1kHz未満は、四捨五入します。
- 簡素化を図るために、クロックはPIC内蔵の8MHzを使用します。

動作原理

<メイン処理>

- 周波数をカウントします。
- 校正スイッチがOFFの場合には、測定結果の補正□-455kHz□四捨五入、7セグ表示を行います。
- 校正スイッチがONの場合には、校正值の算出、保存、7セグ表示(1MHz時の周波数)を行います。

<セグメントデータ設定処理>

- 引数で渡されたカウント値(seg1,seg2,seg3,seg4)を、表示用変数(seg_dat)にセットします。

<割り込み処理>

- TIMER2を使用して□1msecの割り込みを発生させます。
- フラグ(fc_flg)が、“-1”であれば、何もしません。
- フラグ(fc_flg)が、“0”であれば、カウント用のゲートを開けて、フラグ(fc_flg)をインクリメントします。
- フラグ(fc_flg)が、“100”であれば、カウント用のゲートを閉じて、フラグ(fc_flg)を“-1”にします。
- フラグ(fc_flg)が、上記以外であれば、フラグ(fc_flg)をインクリメントします。
- ダイナミック点灯処理を呼び出します。

<周波数カウント処理>

- カウント用のタイマ□(TIMER1)を初期化します。
- フラグ(fc_flg)を、“0”にして、カウントを開始します。
- フラグ(fc_flg)が、“-1”になると、カウントを停止します。

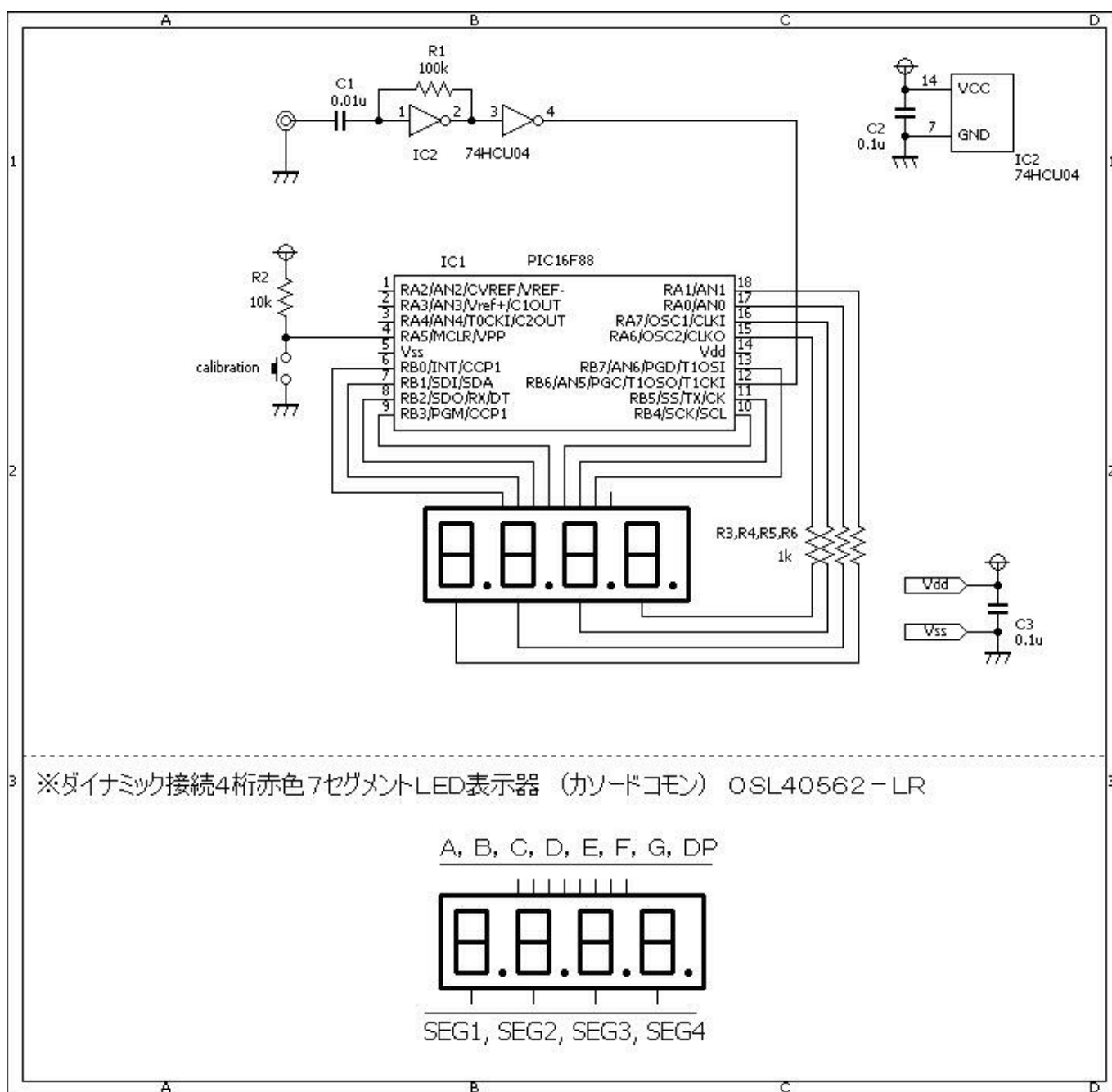
<ダイナミック点灯処理>

- 呼び出される毎に、セグメントを順次切り替えて、表示用変数(seg_dat)の値を点灯させます。

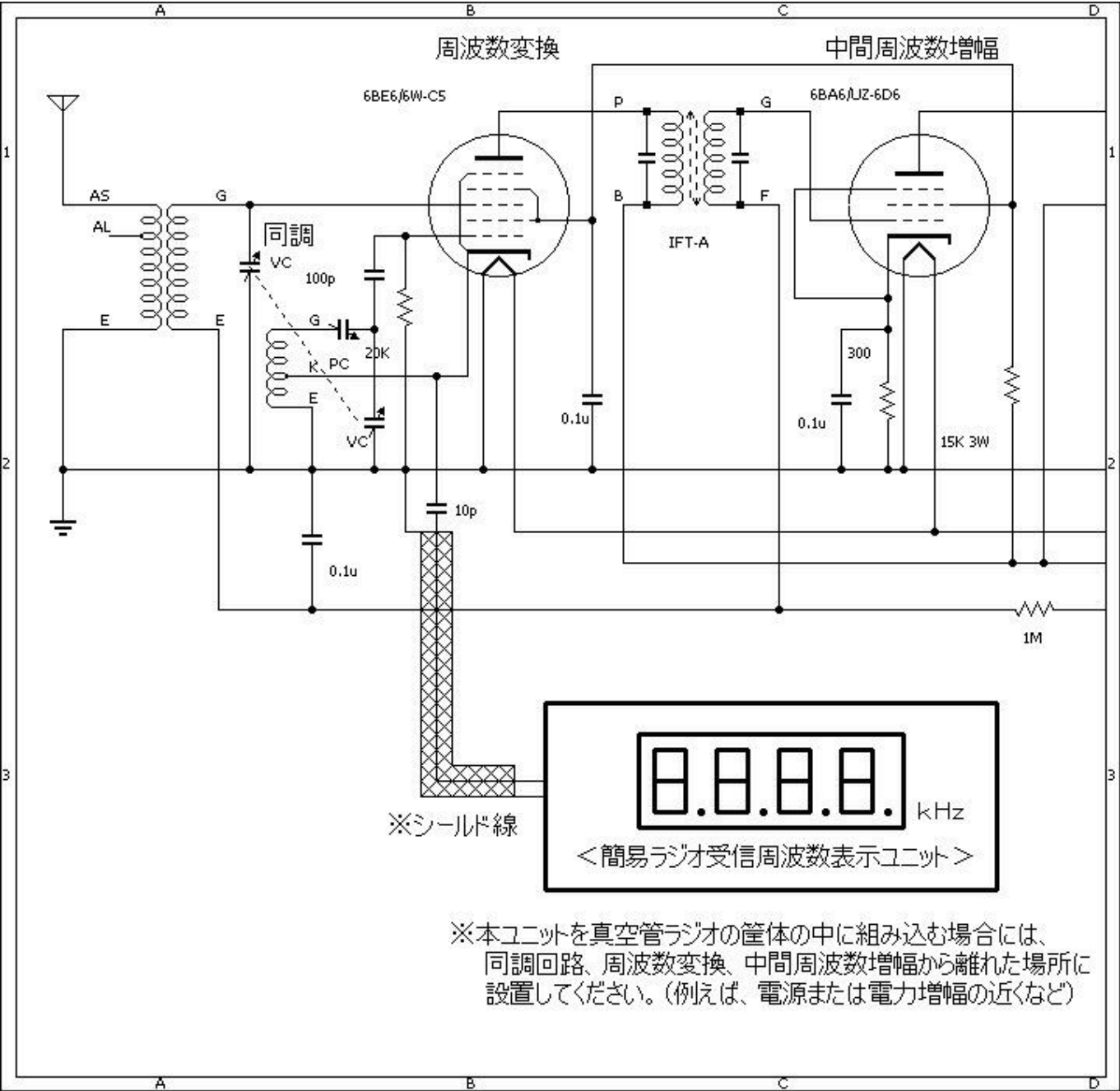
<入力アンプ>

- <校正方法> 未校正でも使用可能ですが、校正をすることにより、精度を高めることができます。

- ## 回路図



<5球スーパーラジオへの接続>



ソースコード

freq_counter_7seg_4digit.c

```
//*****  
*  
/*  
    < 簡易ラジオ受信周波数表示ユニット ( 7セグ4桁 ) >  
  
    ラジオの局発から中間周波数[455kHz]を引いた値を表示する。  
    つまり、ラジオの受信周波数を表示する。  
*/  
//*****  
*
```

```
// ■ ■ ■ 関数 & 共有データ宣言 ■ ■ ■
extern void main();
extern long measurement();
extern void init_port();
extern void init_segment();
extern void init_timer();
extern void segment_disp();
extern void segment_set_data(short seg1, short seg2, short
SEG3, short seg4);
extern void interrupt();
// *****
*
// ■ ■ ■ マクロ定義 ■ ■ ■
// 7SEG-SELECT
#define ACTIVE_LOW
// 簡易接続、標準接続□□2SA1015□□
#ifdef ACTIVE_LOW
#define SEG_ON 0
#define SEG_OFF 1
#else
// 標準接続□□2SC1815□□
#define SEG_ON 1
#define SEG_OFF 0
#endif
sbit SEG1 at PORTA.B1;
sbit SEG1_Direction at TRISA.B1;
#define SEG1_ON SEG1 = SEG_ON
#define SEG1_OFF SEG1 = SEG_OFF
sbit SEG2 at PORTA.B0;
sbit SEG2_Direction at TRISA.B0;
#define SEG2_ON SEG2 = SEG_ON
#define SEG2_OFF SEG2 = SEG_OFF
sbit SEG3 at PORTA.B7;
sbit SEG3_Direction at TRISA.B7;
#define SEG3_ON SEG3 = SEG_ON
#define SEG3_OFF SEG3 = SEG_OFF
sbit SEG4 at PORTA.B6;
sbit SEG4_Direction at TRISA.B6;
#define SEG4_ON SEG4 = SEG_ON
#define SEG4_OFF SEG4 = SEG_OFF
// 7SEG-DATA
#define SEG_DATA PORTB
#define SEG_DATA_Direction TRISB
// switch
#define SW_CAL PORTA.B5
#define SW_ON 0
#define SW_OFF 1
// other
#define INPUT_MODE 1
#define OUTPUT_MODE 0
// *****
```

```
*
// ■ ■ ■ メイン関数 ■ ■ ■
void    main()
{
    double    freq;
    long      tmp;
    char      buf[16], tmp0, tmp1, tmp2, tmp3;
    union {
        double    _double;
        short      _short[4];
    }          cal;
    //
    OSCCON = 0b01110000;           //クロックを8MHz に設定します。
    ANSEL   = 0b00000000;         //A/D変換モジュールは使用しません。
    TRISA   = 0b11111111;
    TRISB   = 0b11111111;
    //
    init_port();
    init_segment();
    init_timer();
    // 校正データを読み込む
    tmp0 = Eeprom_Read(0);
    Delay_ms(20);
    tmp1 = Eeprom_Read(1);
    Delay_ms(20);
    tmp2 = Eeprom_Read(2);
    Delay_ms(20);
    tmp3 = Eeprom_Read(3);
    Delay_ms(20);
    cal._short[0] = tmp0;
    cal._short[1] = tmp1;
    cal._short[2] = tmp2;
    cal._short[3] = tmp3;
    cal._double = ((cal._double < 0.5) || (cal._double > 1.5)) ?
1.0 : cal._double;
    // 割り込みを許可します。
    INTCON.PEIE = 1;
    INTCON.GIE = 1;
    //
    while (1) {
        //測定
        freq = measurement();
        //
        if (SW_CAL == SW_OFF) {
            //測定結果の補正、四捨五入、表示
            tmp = freq / cal._double;
            tmp -= 455000.0;
            //
            if ((tmp % 1000) >= 500) {
                tmp = (tmp / 1000) + 1;
            } else {
```

```
        tmp = tmp / 1000;
    }
    //
    LongWordToStr(tmp, buf);
    buf[6] = (buf[6] == ' ') ? 0 : buf[6] - '0';
    buf[7] = (buf[7] == ' ') ? 0 : buf[7] - '0';
    buf[8] = (buf[8] == ' ') ? 0 : buf[8] - '0';
    buf[9] = (buf[9] == ' ') ? 0 : buf[9] - '0';
    segment_set_data(buf[6], buf[7], buf[8],
buf[9]);

    } else {
        //校正値の算出、保存、表示(1MHz時の周波数)
        cal._double = freq / 1000000.0;
        //
        Eeprom_Write(0, cal._short[0]);
        Delay_ms(20);
        Eeprom_Write(1, cal._short[1]);
        Delay_ms(20);
        Eeprom_Write(2, cal._short[2]);
        Delay_ms(20);
        Eeprom_Write(3, cal._short[3]);
        Delay_ms(20);
        //
        LongWordToStr(cal._double * 1000.0, buf);
        buf[6] = (buf[6] == ' ') ? 0 : buf[6] - '0';
        buf[7] = (buf[7] == ' ') ? 0 : buf[7] - '0';
        buf[8] = (buf[8] == ' ') ? 0 : buf[8] - '0';
        buf[9] = (buf[9] == ' ') ? 0 : buf[9] - '0';
        segment_set_data(buf[6], buf[7], buf[8],
buf[9]);

    }
    Delay_ms(400);
}

}

//*****
*
//■■■セグメント初期化関数■■■
void    init_segment()
{
    SEG1_Direction = OUTPUT_MODE;
    SEG1_OFF;
    SEG2_Direction = OUTPUT_MODE;
    SEG2_OFF;
    SEG3_Direction = OUTPUT_MODE;
    SEG3_OFF;
    SEG4_Direction = OUTPUT_MODE;
    SEG4_OFF;
    //
    SEG_DATA_Direction = 0b01000000;
    SEG_DATA = 0x00;
}
```

```
//*****
*
//■■■ 入出力ポート初期化関数 ■■■
void    init_port()
{
}
//*****
*
//■■■ タイマー初期化関数 ■■■
void    init_timer()
{
    T2CON.T2CKPS1 = 0;
    T2CON.T2CKPS0 = 0;
    T2CON.TOUTPS3 = 1;
    T2CON.TOUTPS2 = 1;
    T2CON.TOUTPS1 = 1;
    T2CON.TOUTPS0 = 1;
    TMR2 = 0;
    PIE1.TMR2IE = 1;
    PIR1.TMR2IF = 0;
    PR2 = 125;          //125=1msec/((1sec/8MHz)*4*16PS)
    T2CON.TMR2ON = 1;
}
//*****
*
//■■■ セグメントデータ設定関数 ■■■
short   seg_dat[4] = {0, 0, 0, 0};
//
void     segment_set_data(short seg1, short seg2, short seg3, short
seg4)
{
    seg_dat[0] = seg1;
    seg_dat[1] = seg2;
    seg_dat[2] = seg3;
    seg_dat[3] = seg4;
}
//*****
*
//■■■ 割り込み関数 ■■■
short   fc_flg = -1;
void     interrupt()
{
    if (PIR1.TMR2IF == 1) {
        PIR1.TMR2IF = 0;
        //周波数カウンタ処理
        switch (fc_flg) {
            case -1:
                break;
            case 0:
                T1CON.TMR1ON = 1;          //ゲートを開ける。
                fc_flg++;
                break;
        }
    }
}
```

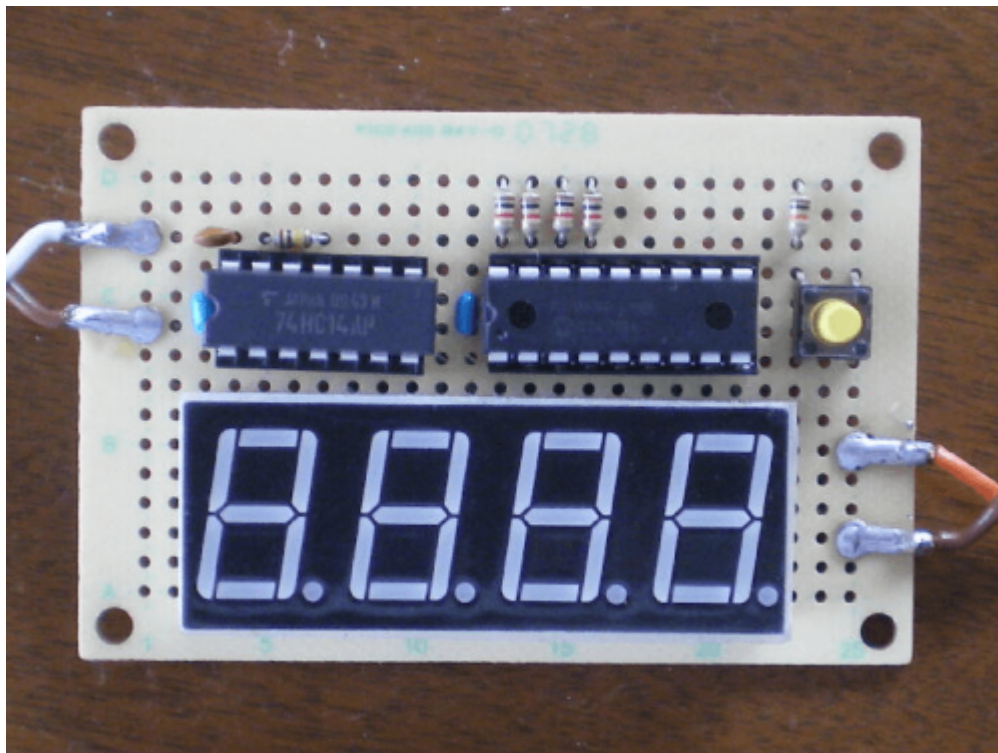
```
        case 100:
            T1CON.TMR1ON = 0;                // ゲートを閉める。
            fc_flg = -1;
            break;
        default:
            fc_flg++;
            break;
    }
    //
    segment_disp();
}

//*****
*
// ■ ■ ■ 周波数測定関数 ■ ■ ■
long measurement()
{
    long freq;
    //
    freq = 0;
    PIE1.TMR1IE = 0;
    PIR1.TMR1IF = 0;
    T1CON.T1CKPS1 = 0;
    T1CON.T1CKPS0 = 0;
    T1CON.TMR1CS = 1;
    T1CON.NOT_T1SYNC = 1;
    TMR1H = 0;
    TMR1L = 0;
    fc_flg = 0;
    while (fc_flg != -1) {
        if (PIR1.TMR1IF == 1) {
            PIR1.TMR1IF = 0;
            freq++;
        }
    }
    if (PIR1.TMR1IF == 1) {
        PIR1.TMR1IF = 0;
        freq++;
    }
    return (((freq * 65536) + ((long)TMR1H * 256) + (long)TMR1L) *
10);
}
//*****
*
// ■ ■ ■ セグメントデータ表示関数 ■ ■ ■
short seg_tbl[10] = {
    0b00111111,    //0
    0b00000110,    //1
    0b10011011,    //2
    0b10001111,    //3
    0b10100110,    //4
```

```
        0b10101101,      //5
        0b10111101,      //6
        0b00100111,      //7
        0b10111111,      //8
        0b10101111      //9
    };
short seg_cnt = 0;
//
void segment_disp()
{
    switch (seg_cnt) {
    case 0:
        SEG4_OFF;
        SEG_DATA = seg_tbl[seg_dat[0]];
        SEG1_ON;
        seg_cnt = 1;
        break;
    case 1:
        SEG1_OFF;
        SEG_DATA = seg_tbl[seg_dat[1]];
        SEG2_ON;
        seg_cnt = 2;
        break;
    case 2:
        SEG2_OFF;
        SEG_DATA = seg_tbl[seg_dat[2]];
        SEG3_ON;
        seg_cnt = 3;
        break;
    case 3:
        SEG3_OFF;
        SEG_DATA = seg_tbl[seg_dat[3]];
        SEG4_ON;
        seg_cnt = 0;
        break;
    }
}
//*****
*
```

動作確認

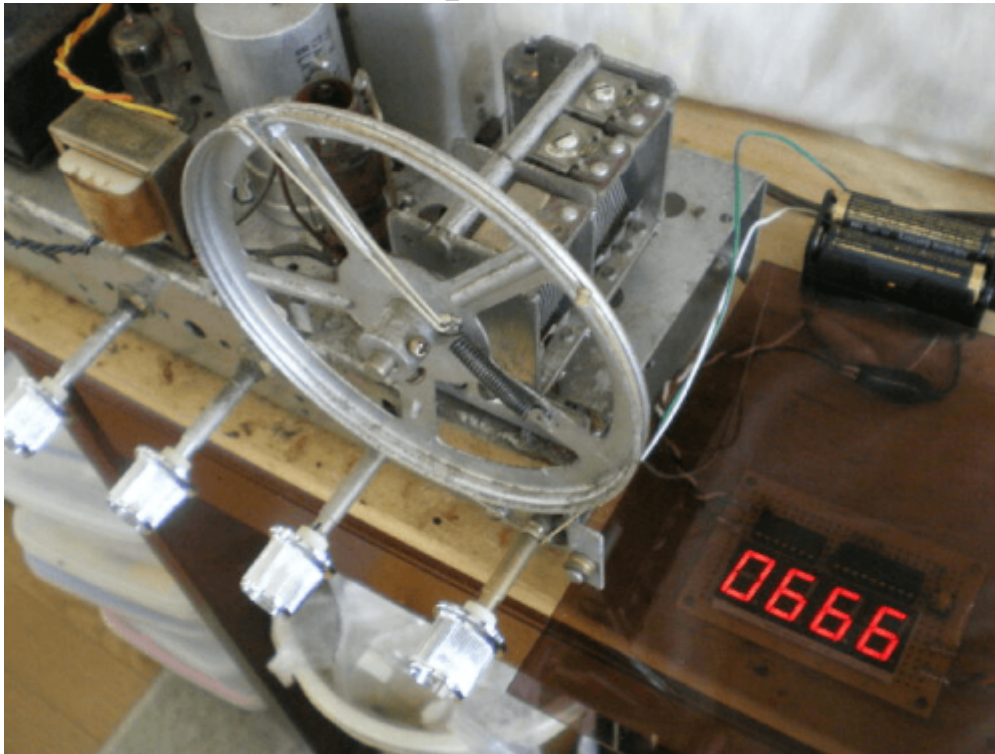
入力アンプの74HCU04は、手持ち部品の関係で、74HC14を使用しています。



1121kHzを入力して□-455kHzした値(666kHz)が表示されることを確認してみました。7セグLEDを、アクリル板で覆うことにより、表示が見やすくなります。



手持ちの真空管ラジオに接続してNHK第一放送(666kHz)を受信してみました。



From:

<http://deepsky.jp/wiki/> - うごくといいな

Permanent link:

<http://deepsky.jp/wiki/doku.php?id=elechobby:picdic:pic16f88:141&rev=1588224649>

Last update: **2025/10/17 14:27**

