簡易電子ボリューム(LM1972)

概要

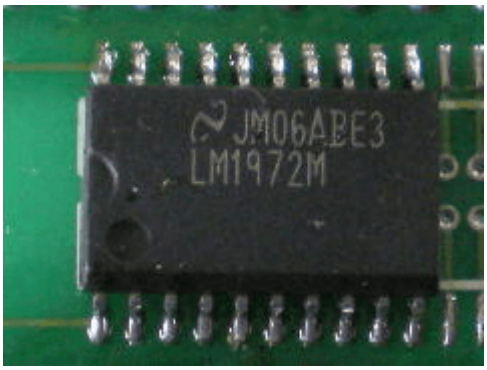
安価で高性能な電子ボリューム用のIC(LM1972)が手に入りましたので、早速、簡易な電子ボリュームを製作してみました。

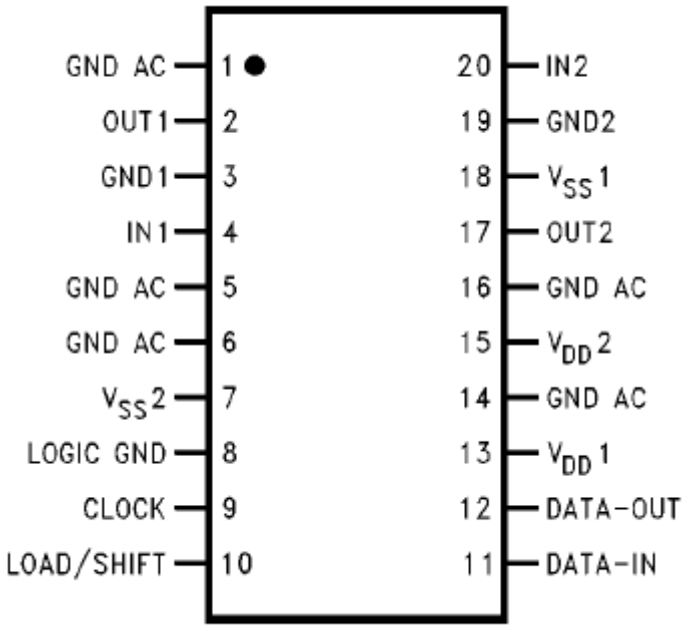
<簡易電子ボリュームの仕様>

- チャネルは、RとLの2チャネルとします。
- RとLを、単独にボリュームのアップおよびダウンを可能とします。
- 表示は、dB表示とバー表示を切り替え可能とします。
- ミュート(104dB)機能を搭載します。
- RとLの、ボリューム値はEEPROMに保存し、起動時には保存されたボリューム値を設定します。

<LM1972の主な仕様と特長>

- 全高調波歪み+ノイズ0.003%(最大)
- 周波数特性100kHz(- 3dB)(最小)
- 減衰範囲(ミュートを除く) 78dB(代表値)
- 減衰誤差± 0.25dB(最大)
- SN 比(4Vrms を基準) 110dB(最小)
- チャネル・セパレーション100dB(最小)
- 3 線シリアル・インタフェース
- デイジーチェーン接続が可能
- 減衰量104dB のミュート機能
- 減衰量変更によるポップ/ クリック音を生じない。

<LM1972の概観> 



<LM1972のピン配置>

動作原理

PICで電子ボリューム(LM1972)を制御し、その時の制御値をLCDに表示します。

動作原理(ハードウェア)

◎電子ボリューム

- LM1972を、3線シリアル・インタフェースで制御します。(ソフト制御)
- 出力側は、単なるコンデンサ出力としましたが、オペアンプのボルテージフォロアでの出力を推奨します。

◎dB表示&バー表示

- dB表示(減衰量を数値dBで表示します)
- バー表示(減衰量をLCDのバー(70点)で表示します。)

◎ボリューム値の設定

- Rチャネルのボリューム値のアップ&ダウンは、スイッチ(SW_R_UP、SW_R_DOWN)を使用します。
- Lチャネルのボリューム値のアップ&ダウンは、スイッチ(SW_L_UP、SW_L_DOWN)を使用します。
- ミュートは、スイッチ(SW_MUTE)を使用します。

動作原理(ソフトウェア)

◎LM1972の制御(初期化)

- ポートの向き(I/O)を設定します。
- LOAD信号を、high(1)に設定します。

- CLOCK信号を、low(0)に設定します。

◎LM1972の制御(ボリューム設定)

- LOAD信号を、low(0)に設定します。
- チャネル番号(8ビット)のMSB(Most Significant Bit)から、順次クロックと同期させながら8ビット分出力します。
- ボリューム値(8ビット)のMSB(Most Significant Bit)から、順次クロックと同期させながら8ビット分出力します。
- LOAD信号を、high(1)に設定します。

◎ボリューム値→dB変換

- ボリューム値が、127以上であれば、-100dBとします。
- ボリューム値が、0~96であれば、-(ボリューム値×0.5dB)とします。
- ボリューム値が、97~126であれば、-((ボリューム値-97)+49dB)とします。

◎表示モード切替

- スイッチ(SW_MODE)が、high(1)であれば、dB表示します。
- スイッチ(SW_MODE)が、low(0)であれば、バー表示します。

◎ミュート動作

- スイッチ(SW_MUTE)が、押下されると、ミュート(127)に設定します。
- スイッチ(SW_MUTE)が、再度押下されると、ミュートを解除します。

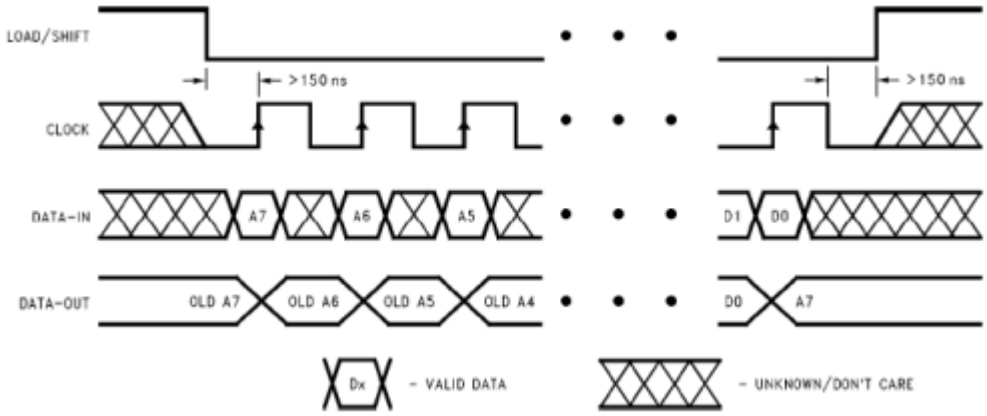
◎ボリューム値のアップ&ダウン

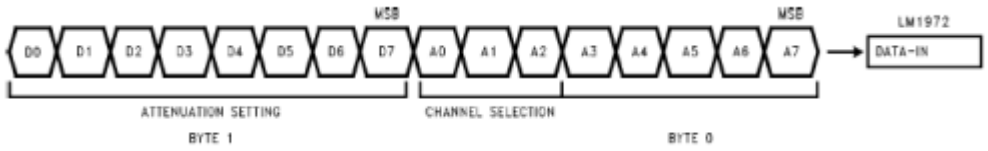
- スイッチ(SW_R_UP、SW_R_DOWN)の操作で、Rチャネルのボリューム値をアップ&ダウンさせます。
- スイッチ(SW_L_UP、SW_L_DOWN)の操作で、LRチャネルのボリューム値をアップ&ダウンさせます。

◎ボリューム値の書き込みと読み出し

- ボリューム値が、変更されるとその時の内容をEEPROMに書き込みます。
- 起動時に、EEPROMからボリューム値を読み出し、ボリューム値を設定します。

<LM1972のタイミングチャート>

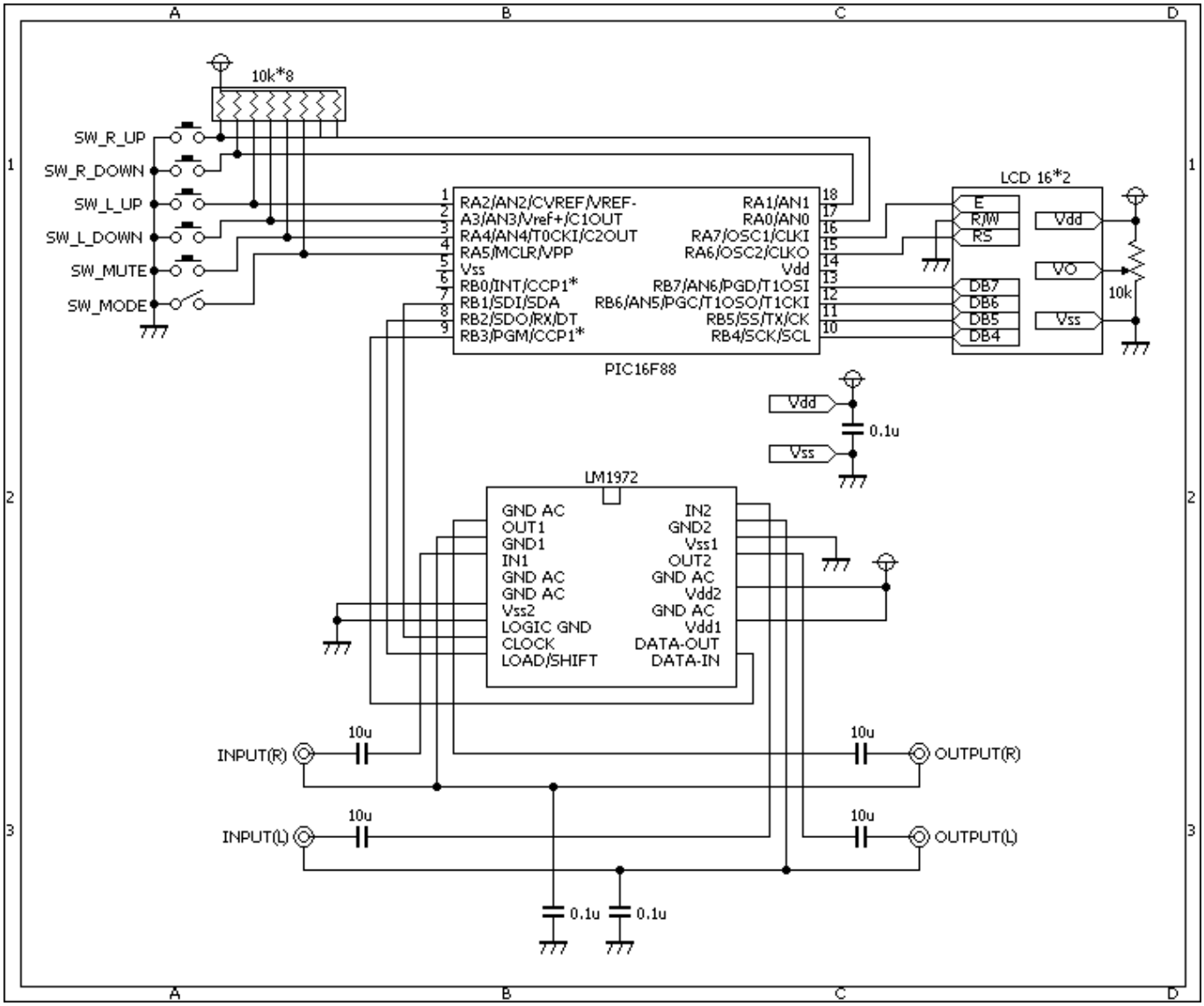




MSB LSB	
Address Register (Byte 0)	
0000 0000	Channel 1
0000 0001	Channel 2
0000 0010	Channel 3
Data Register (Byte 1)	
Contents	Attenuation Level dB
0000 0000	0.0
0000 0001	0.5
0000 0010	1.0
0000 0011	1.5
: : : : :	: :
0001 1110	15.0
0001 1111	15.5
0010 0000	16.0
0010 0001	16.5
0010 0010	17.0
: : : : :	: :
0101 1110	47.0
0101 1111	47.5
0110 0000	48.0
0110 0001	49.0
0110 0010	50.0
: : : : :	: :
0111 1100	76.0
0111 1101	77.0
0111 1110	78.0
0111 1111	100.0 (Mute)
1000 0000	100.0 (Mute)
: : : : :	: :
1111 1110	100.0 (Mute)
1111 1111	100.0 (Mute)

<LM1972のデータと減衰量の関係>

回路図



ソースコード

electronic_volume_v2.c

```
//*****************************************************************************
*
/*
    ＜電子ボリュームV2
*/
//*****************************************************************************
*
#define BYTE      unsigned short
#define WORD      unsigned int
#define DWORD     unsigned long
//
#define SW_R_UP   PORTA.B0
#define SW_R_DOWN PORTA.B1
#define SW_L_UP   PORTA.B2
#define SW_L_DOWN PORTA.B3
```

```
#define SW_MUTE   PORTA.B4
#define SW_MODE   PORTA.B5
//
#define CH_R      0
#define CH_L      1
//*****************************************************************
*
const char character0[] = { 0, 0, 0, 0, 0, 0, 0, 0};
const char character1[] = { 0,16,16,16,16,16, 0, 0};
const char character2[] = { 0,24,24,24,24,24, 0, 0};
const char character3[] = { 0,28,28,28,28,28, 0, 0};
const char character4[] = { 0,30,30,30,30,30, 0, 0};
const char character5[] = { 0,31,31,31,31,31, 0, 0};
//
void    RegistCustomChar()
{
        char    i;
        //
        LCD_Cmd(64);
        for (i = 0; i<=7; i++) {
                LCD_Chr_Cp(character0[i]);
        }
        for (i = 0; i<=7; i++) {
                LCD_Chr_Cp(character1[i]);
        }
        for (i = 0; i<=7; i++) {
                LCD_Chr_Cp(character2[i]);
        }
        for (i = 0; i<=7; i++) {
                LCD_Chr_Cp(character3[i]);
        }
        for (i = 0; i<=7; i++) {
                LCD_Chr_Cp(character4[i]);
        }
        for (i = 0; i<=7; i++) {
                LCD_Chr_Cp(character5[i]);
        }
        LCD_Cmd(_LCD_RETURN_HOME);
}
//*****************************************************************
*
void    BarDisp(char row, char column, short mode, unsigned int dat)
{
        short   i, j, k, cnt;
        //
        if (mode == 0) {
                i = dat / 1.79;
        } else {
                i = dat;
        }
        j = i / 5;
```

```
        k = i - (j * 5);
        //
        if (row == 1)
                Lcd_Cmd(_LCD_FIRST_ROW);
        else
                Lcd_Cmd(_LCD_SECOND_ROW);
        //
        for (cnt = 1; cnt < column; cnt++) {
                Lcd_Cmd(_LCD_MOVE_CURSOR_RIGHT);
        }
        //
        for (cnt = 0; cnt < j; cnt++) {
                Lcd_Chr_Cp(5);
        }
        Lcd_Chr_Cp(k);
        for (cnt++; cnt < 10; cnt++) {
                Lcd_Chr_Cp(' ');
        }
}
//*****************************************************************************
*
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;
//
void    init_lcd()
{
        short   cnt;
        //
        Lcd_Init();
        RegistCustomChar();
        Lcd_Cmd(_LCD_CURSOR_OFF);
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1, 1, "e-volume V2.0");
        for (cnt = 0; cnt <= 80; cnt++) {
                BarDisp(2, 1, 1, cnt);
                Delay_ms(10);
        }
        Lcd_Cmd(_LCD_CLEAR);
}
//*****************************************************************************
*
```

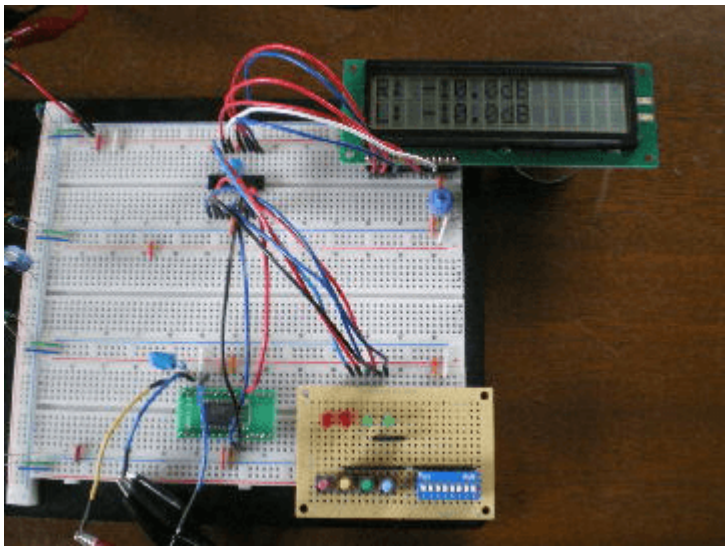
```
sbit LM1972_CLOCK at RB1_bit;
sbit LM1972_LOAD at RB2_bit;
sbit LM1972_DATA at RB3_bit;
sbit LM1972_CLOCK_Direction at TRISB1_bit;
sbit LM1972_LOAD_Direction at TRISB2_bit;
sbit LM1972_DATA_Direction at TRISB3_bit;
//
void    init_volume()
{
        LM1972_CLOCK_Direction = 0;
        LM1972_LOAD_Direction = 0;
        LM1972_DATA_Direction = 0;
        //
        LM1972_LOAD = 1;
        LM1972_CLOCK = 0;
}
//*********************************************************************
*
void    set_volume(BYTE channel, BYTE attenuation)
{
        short   cnt;
        //
        LM1972_LOAD = 0;
        for (cnt = 0; cnt < 8; cnt++) {
                LM1972_DATA = (channel & 0x80) != 0 ? 1 : 0;
                LM1972_CLOCK = 1;
                LM1972_CLOCK = 0;
                channel <<= 1;
        }
        for (cnt = 0; cnt < 8; cnt++) {
                LM1972_DATA = (attenuation & 0x80) != 0 ? 1 : 0;
                LM1972_CLOCK = 1;
                LM1972_CLOCK = 0;
                attenuation <<= 1;
        }
        LM1972_LOAD = 1;
}
//*********************************************************************
*
void    db_cnv(BYTE dt, char* result)
{
        int     tmp;
        //
        tmp = dt;
        if (tmp >= 127) {
                IntToStr(-1000, result);
                result[9] = 0x00;
                result[8] = 'B';
                result[7] = 'd';
                result[6] = result[5];
                result[5] = '.';
```

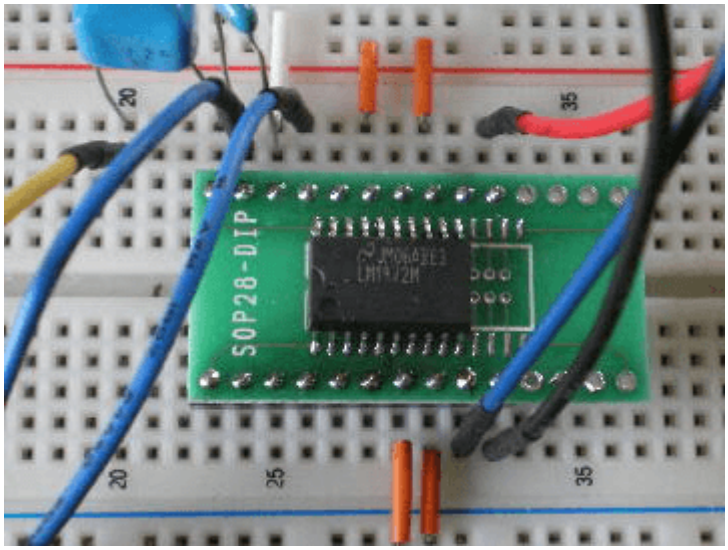
```
                return;
        }
        if (tmp <= 96) {
                tmp *= 5;
                IntToStr(tmp * -1, result);
                result[9] = 0x00;
                result[8] = 'B';
                result[7] = 'd';
                result[6] = result[5];
                result[5] = '.';
                return;
        }
        tmp = 490 + ((tmp - 97) * 10);
        IntToStr(tmp * -1, result);
        result[9] = 0x00;
        result[8] = 'B';
        result[7] = 'd';
        result[6] = result[5];
        result[5] = '.';
}
//*********************************************************************
*
void    main()
{
        BYTE   R_data, L_data;
        char   buf[16];
        short  mute_flg = 0;
        //
        OSCCON = 0b01110000;
        CMCON  = 0b00000111;
        ANSEL  = 0b00000000;
        TRISA  = 0b11111111;
        //
        init_lcd();
        Lcd_Out(1, 1, "R:");
        Lcd_Out(2, 1, "L:");
        //
        init_volume();
        //
        R_data = EEPROM_Read(0);
        R_data = R_data > 127 ? 127 : R_data;
        L_data = EEPROM_Read(1);
        L_data = L_data > 127 ? 127 : L_data;
        set_volume(CH_R, R_data);
        set_volume(CH_L, L_data);
        db_cnv(R_data, buf);
        Lcd_Out(1, 3, &buf[1]);
        db_cnv(L_data, buf);
        Lcd_Out(2, 3, &buf[1]);
        //
        while (1) {
```

```
                //□□□ボリューム□□□
                if (SW_R_UP == 0) {
                        Delay_ms(100);
                        if (R_data > 0) {
                                R_data--;
                                EEPROM_Write(0, R_data);
                        }
                        set_volume(CH_R, R_data);
                }
                //□□□ボリューム□□□□□
                if (SW_R_DOWN == 0) {
                        Delay_ms(100);
                        if (R_data < 127) {
                                R_data++;
                                EEPROM_Write(0, R_data);
                        }
                        set_volume(CH_R, R_data);
                }
                //□□□ボリューム□□□
                if (SW_L_UP == 0) {
                        Delay_ms(100);
                        if (L_data > 0) {
                                L_data--;
                                EEPROM_Write(1, L_data);
                        }
                        set_volume(CH_L, L_data);
                }
                //□□□ボリューム□□□□□
                if (SW_L_DOWN == 0) {
                        Delay_ms(100);
                        if (L_data < 127) {
                                L_data++;
                                EEPROM_Write(1, L_data);
                        }
                        set_volume(CH_L, L_data);
                }
                //表示モード切替□□□表示、バー表示）
                if (SW_MODE == 1) {
                        db_cnv(R_data, buf);
                        Lcd_Out(1, 3, &buf[1]);
                        Lcd_Out(1, 11, "      ");
                        db_cnv(L_data, buf);
                        Lcd_Out(2, 3, &buf[1]);
                        Lcd_Out(2, 11, "      ");
                } else {
                        BarDisp(1, 3, 0, 127 - R_data);
                        BarDisp(2, 3, 0, 127 - L_data);
                }
                //ミュート動作
                if (SW_MUTE == 0) {
                        Delay_ms(100);
```

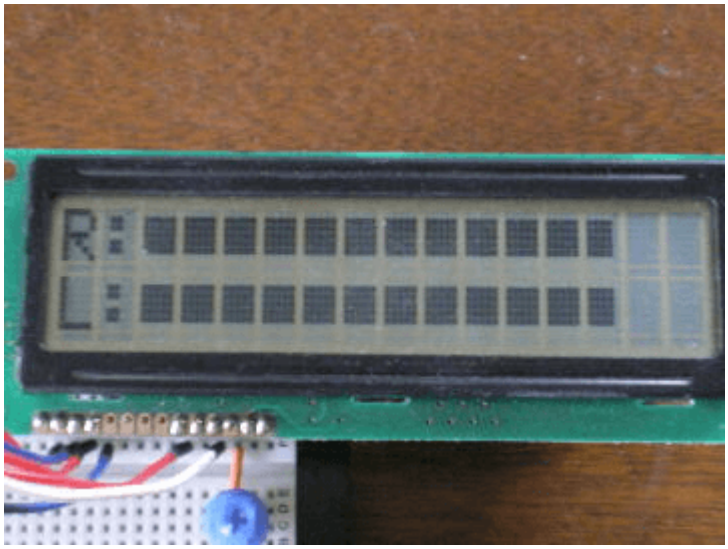
```
                        if (mute_flg == 0) {
                                mute_flg = 1;
                                Lcd_Chr(1, 2, '_');
                                Lcd_Chr(2, 2, '_');
                                set_volume(CH_R, 127);
                                set_volume(CH_L, 127);
                        } else {
                                mute_flg = 0;
                                Lcd_Chr(1, 2, ':');
                                Lcd_Chr(2, 2, ':');
                                set_volume(CH_R, R_data);
                                set_volume(CH_L, L_data);
                        }
                }
        }
}
//*****************************************************************************
*
```

動作確認

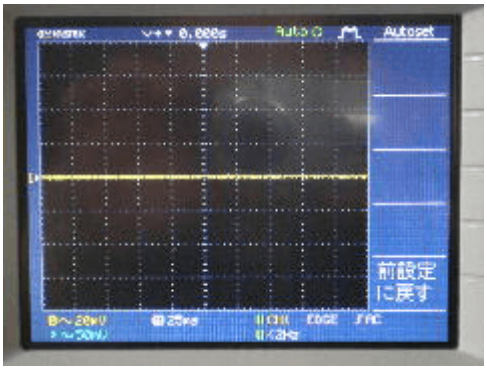




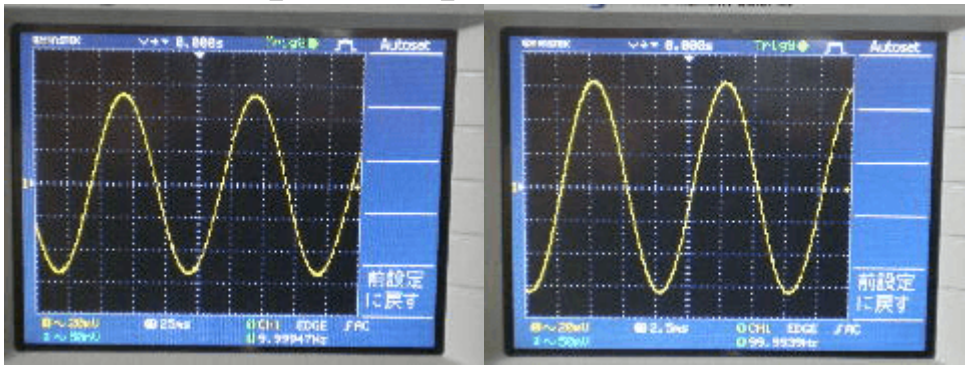
左側:dB表示、右側:**バー**表示

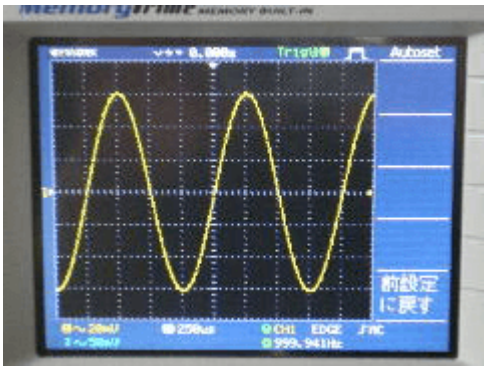


（編者注：破損**した画像しかありませんでした**）



左側から、ミュート□10Hz正弦波□100Hz正弦波





左側から□1KHz正弦波□10kHz□100KHz正弦波