

簡易WavePlayer(PIC18F2620)

概要

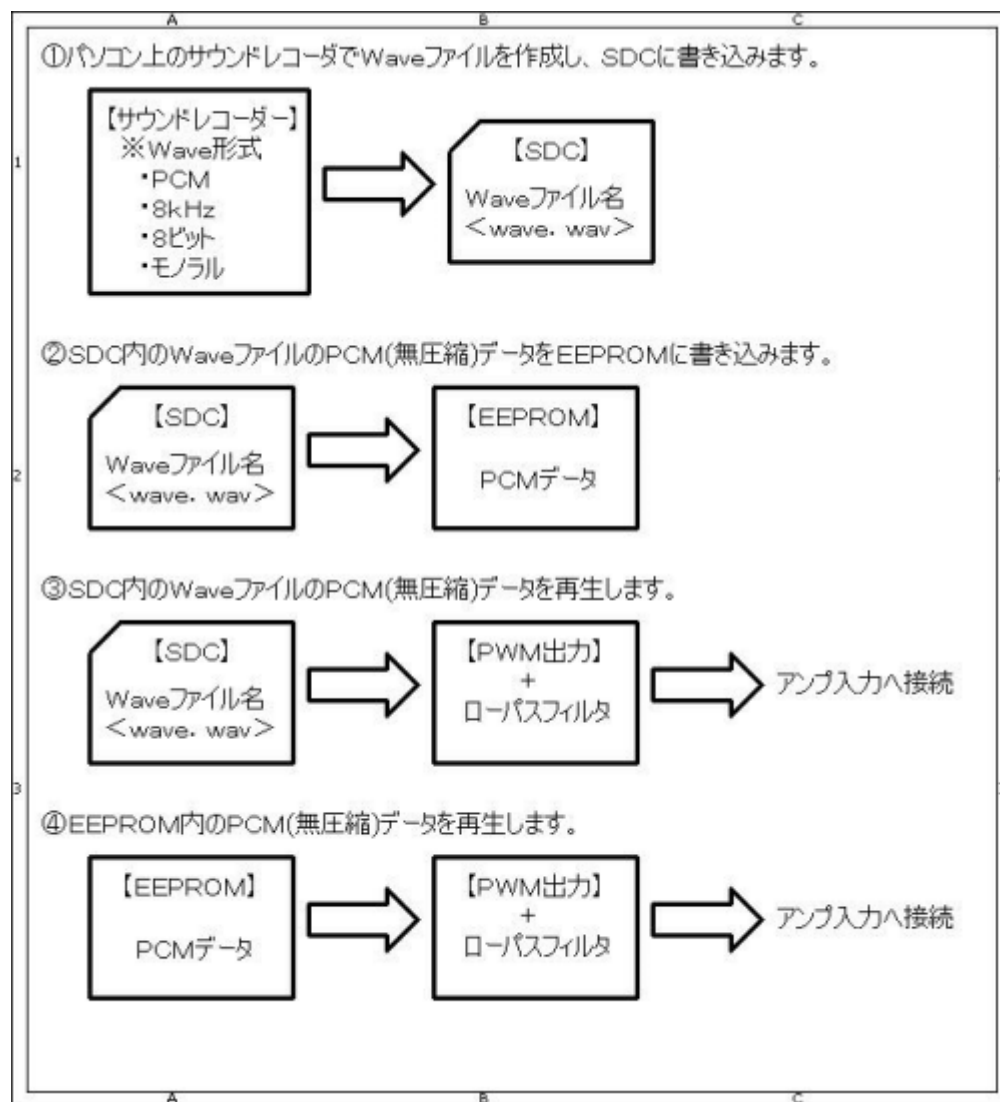
今迄、音声を記録したり、再生したりするものを製作したことがありません。そこで今回は、音声を再生することだけに注力した、“簡易WavePlayer”なるものを製作してみました。

今回製作した、“簡易WavePlayer”には、当初SDCに保存したWaveファイルを再生するだけの機能を搭載する予定でしたがSDC接続に向かないPIC(例えばPIC12F683など)では音声扱えないことになってしまいます。

そこでWaveファイルの音声データ(PCM)部だけをEEPROMに記録し、再生することが出来る機能も付け加えました。

<仕様>

- SDCに記録されたWaveファイルを再生します。
※Wave形式は、PCM(無圧縮) 8kHz 8ビット、モノラルのみとします。
- SDCに記録されたWaveファイルのPCMデータ部をEEPROMに保存します。
※EEPROMの記憶容量は、約131kバイトです。
- EEPROMに記録されたPCMデータを再生します。
- Waveファイルは、パソコン上のツール(サウンドレコーダーなど)で作成します。



動作原理(ハードウェア)

電源電圧 3.3Vの単一電源による駆動とします。

◎SDC FAT16(最大2Gバイト)に対応します□ Waveファイル名は、“wave.wav”固定とします□ Wave形式は、PCM(無圧縮)□8kHz□8ビット、モノラルのみとします。

◎EEPROM 約131kバイトの記憶容量を持つ、マイクロチップ社の、“24LC1025”を使用します。

◎LCD 16文字2行のLCDを使用します□ LCDを動作させるために□PICのPWM信号を利用して、“負電圧”を発生させ□LCDに供給します。

再生信号出力 PICのPWMとローパスフィルタを使用して、アナログ信号に変換します。微弱な信号なので、アンプを接続し、スピーカを駆動します。実験ではクリスタルイヤホンを使用しました。

動作原理(ソフトウェア)

◎SDCに記録されたWaveファイルの再生(SW1) Waveファイルのフォーマットの詳細については、公開サイトが多くありますので、其方を参考にしてください□ Waveファイルの最初の58バイトには、以下

に示すように、各種情報が記載されています。

本来は、この58バイトについてもチェックを行い、該当するデータかどうかを判断する必要があります。今回は、これらのチェックを省略(58バイト読み飛ばす)しています。59バイト以降のPCMデータを順次読み出しPWMに出力します。

尚Waveファイルの作成ツールは、フリーソフトも含めいろいろ公開されています。最も簡単な方法は、Windows標準の“サウンドレコーダー”を使って作成する方法です。

◎SDCに記録されたWaveファイルのPCMデータ部のEEPROMへの保存(SW3) Waveファイルの59バイト以降のデータを、EEPROMに書き込みます。EEPROMの容量を超えるデータは無視されます。

◎EEPROMに記録されたPCMデータの再生(SW2) EEPROMに保存されたPCMデータを順次読み出しPWMに出力します。

停止(SW4) 上記の処理を停止する場合にはSW4を押下します。

<WAVEファイルのフォーマット(例)>

```
0 : 52 49 46 46 B2 38 01 00 57 41 56 45 66 6D 74 20
1 : 12 00 00 00 01 00 01 00 40 1F 00 00 40 1F 00 00
2 : 01 00 08 00 00 00 66 61 63 74 04 00 00 00 80 38
3 : 01 00 64 61 74 61 80 38 01 00 80 80 7F 80 7F 80
4 : 7F 80 7F 80 7F 84 85 85 85 85 85 85 85 85 85 85
5 : 85 85 85 85 85 85 85 85 85 85 85 85 85 85 85 85
6 : 85 85 85 85 85 85 85 85 85 85 85 85 85 85 85 85
7 : 85 85 85 85 85 85 85 85 85 85 85 85 85 85 85 85
8 : 85 85 85 85 85 85 85 85 85 85 85 85 85 85 85 85
```

```
4バイト:52 49 46 46:“RIFF”
4バイト:B2 38 01 00:RIFFデータサイズ
4バイト:57 41 56 45:“WAVE”
4バイト:66 6D 74 20:“fmt”
4バイト:12 00 00 00:fmtサイズ
2バイト:01 00:フォーマットID
2バイト:01 00:チャンネル数
4バイト:40 1F 00 00:サンプリングレート
4バイト:40 1F 00 00:データ速度
2バイト:01 00:ブロックサイズ
2バイト:08 00:サンプルあたりのビット数
2バイト:00 00:拡張部分のサイズ
4バイト:66 61 63 74:“fact”
4バイト:04 00 00 00:factサイズ
4バイト:80 38 01 00:factデータ
4バイト:64 61 74 61:“data”
4バイト:80 38 01 00:dataサイズ
(計58バイト)
59バイト以降がPCMデータになります。
```

回路図


```
#define SW_STOP PORTB.B4

#define LED PORTB.B3

#define ACK 1
#define NO_ACK 0

#define WRITE_CYCLE_TIME 5

//*****
*
// ■ 関数および変数の宣言
extern void main();
extern void init_lcd();
extern void init_sdc();
extern void init_ccp_compare();
extern void init_i2c();
extern void EEPROM_24LC1025_Page_Write(WWORD addr, BYTE *msg, WORD len);
extern void EEPROM_24LC1025_Page_Read(WWORD addr, BYTE *msg, WORD len);
extern void play_sdc();
extern void play_eeprom();
extern void load_sdc2eeprom();
//*****
*
// ■ グローバル変数の定義
char buf[500];
//*****
*
// ■ メイン関数です。
void main()
{
    // ポートの設定
    TRISA = 0b11111111;
    TRISB = 0b11110000;
    TRISC = 0b00000000;
    ADCON1.PCFG3 = 1;
    ADCON1.PCFG2 = 1;
    ADCON1.PCFG1 = 1;
    ADCON1.PCFG0 = 1;
    // □□□の初期化□□□→□□□□再生用に使用します。
    PWM1_Init(100000); // 100kHz
    PR2 = 0xFF;
    PWM1_Set_Duty(PR2 / 2);
    PWM1_Start();
    // □□□の初期化□□□→□□□の輝度電圧（負電圧発生）に使用します。
    PWM2_Init(100000); // 100kHz
    PR2 = 0xFF;
    PWM2_Set_Duty(PR2 / 2);
    PWM2_Start();
}
```

```

Delay_ms(100);
//液晶の初期化
init_lcd();
//SDの初期化
init_sdc();
//I2Cの初期化
init_i2c();
//
while (1) {
    Lcd_Out(1, 1, "RUN MODE ?");
    //
    if (SW_PLAY_SDC == 0) {
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1, 1, "PLAY SDC");
        play_sdc();
        Lcd_Out(1, 1, "STOP      ");
        Delay_ms(500);
        Lcd_Cmd(_LCD_CLEAR);
    }
    if (SW_PLAY_EEPROM == 0) {
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1, 1, "PLAY EEPROM");
        play_eeprom();
        Lcd_Out(1, 1, "STOP          ");
        Delay_ms(500);
        Lcd_Cmd(_LCD_CLEAR);
    }
    if (SW_LOAD == 0) {
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1, 1, "LOAD SDC->EEPROM");
        load_sdc2eeprom();
        Lcd_Out(1, 1, "STOP              ");
        Delay_ms(500);
        Lcd_Cmd(_LCD_CLEAR);
    }
}
}
//*****
*
//■液晶の液晶ファイルの内容を再生する関数です。
void play_sdc()
{
    WWORD    fsize, length, cnt, addr;
    char     character;
    //
    Mmc_Fat_Assign("wave.wav", 0);
    Mmc_Fat_Reset(&fsize);
    length = 0;
    for (cnt = 0; cnt < 58; cnt++) {
        Mmc_Fat_Read(&character);
    }
}

```

```

    length += 58;
    addr = 0;
    while ((length < fsize) && (SW_STOP == 1)) {
        Mmc_Fat_Read(&character);
        PWM1_Set_Duty(character);
        length++;
        Delay_us(70);
    }
}
//*****
*
//■■■■■■■■の内容を再生する関数です。
void play_eeprom()
{
    WWORD  addr;
    //
    addr = 0;
    Soft_I2C_Start();
    Soft_I2C_Write(0xA0);
    Soft_I2C_Write((addr >> 8) & 0xFF);
    Soft_I2C_Write(addr & 0xFF);
    Soft_I2C_Start();
    Soft_I2C_Write(0xA1);
    while (addr < 0x10000) {
        PWM1_Set_Duty(Soft_I2C_Read(ACK));
        addr++;
        Delay_us(50);
        if (SW_STOP == 0) {
            break;
        }
    }
    PWM1_Set_Duty(Soft_I2C_Read(NO_ACK));
    Soft_I2C_Stop();
    //
    addr = 0;
    Soft_I2C_Start();
    Soft_I2C_Write(0xA8);
    Soft_I2C_Write((addr >> 8) & 0xFF);
    Soft_I2C_Write(addr & 0xFF);
    Soft_I2C_Start();
    Soft_I2C_Write(0xA9);
    while (addr < 0x10000) {
        PWM1_Set_Duty(Soft_I2C_Read(ACK));
        addr++;
        Delay_us(50);
        if (SW_STOP == 0) {
            break;
        }
    }
    PWM1_Set_Duty(Soft_I2C_Read(NO_ACK));
    Soft_I2C_Stop();
}

```

```

}
//*****
*
//■■■■の■■■■ファイルの内容を■■■■■■に書き込む関数です。
void    load_sdc2eprom()
{
    WWORD    fsize, length, cnt, addr;
    char      character;
    //
    Mmc_Fat_Assign("wave.wav", 0);
    Mmc_Fat_Reset(&fsize);
    length = 0;
    for (cnt = 0; cnt < 58; cnt++) {
        Mmc_Fat_Read(&character);
    }
    length += 58;
    addr = 0;
    while ((length < fsize) && (addr < 0x20000) && (SW_STOP == 1))
    {
        for (cnt = 0; cnt < 128; cnt++) {
            Mmc_Fat_Read(&character); //1個データを飛ばします。
            Mmc_Fat_Read(&character);
            buf[cnt] = character;
        }
        EEPROM_24LC1025_Page_Write(addr, buf, 128);
        addr += 128;
        fsize += 128;
        LongWordToStr(addr, buf);
        Lcd_Out(2, 1, buf);
    }
    Lcd_Cmd(_LCD_CLEAR);
}
//*****
*
//■■■■を初期化する関数です。
sbit LCD_RS at RA5_bit;
sbit LCD_EN at RA4_bit;
sbit LCD_D7 at RA0_bit;
sbit LCD_D6 at RA1_bit;
sbit LCD_D5 at RA2_bit;
sbit LCD_D4 at RA3_bit;
sbit LCD_RS_Direction at TRISA5_bit;
sbit LCD_EN_Direction at TRISA4_bit;
sbit LCD_D7_Direction at TRISA0_bit;
sbit LCD_D6_Direction at TRISA1_bit;
sbit LCD_D5_Direction at TRISA2_bit;
sbit LCD_D4_Direction at TRISA3_bit;

void    init_lcd()
{
    short    cnt;

```

```

//
Lcd_Init();
Lcd_Cmd(_LCD_CLEAR);
Lcd_Cmd(_LCD_CURSOR_OFF);
Lcd_Out(1, 1, "wave player v1.0");
for (cnt = 0; cnt < 16; cnt++) {
    Lcd_Chrc(2, cnt + 1, 0xFF);
    Delay_ms(100);
}
Lcd_Cmd(_LCD_CLEAR);
}
//*****
*
//■■■■を初期化する関数です。
sfr sbit Mmc_Chip_Select          at RC6_bit;
sfr sbit Mmc_Chip_Select_Direction at TRISC6_bit;

void    init_sdc()
{
    SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV64,
        _SPI_DATA_SAMPLE_MIDDLE, _SPI_CLK_IDLE_LOW, _SPI_LOW_2_HIGH);
    if (Mmc_Fat_Init()) {
        while (1) {
            Lcd_Out(2, 1, "SDC error!");
            Delay_ms(500);
            Lcd_Out(2, 1, "          ");
            Delay_ms(500);
        }
    }
    SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV4,
        _SPI_DATA_SAMPLE_MIDDLE, _SPI_CLK_IDLE_LOW, _SPI_LOW_2_HIGH);
    Lcd_Out(2, 1, "SDC complete!");
    Delay_ms(500);
    Lcd_Cmd(_LCD_CLEAR);
}
//*****
*
//■■■■を初期化する関数です。
sbit  Soft_I2C_Scl          at RC0_bit;
sbit  Soft_I2C_Sda          at RC1_bit;
sbit  Soft_I2C_Scl_Direction at TRISC0_bit;
sbit  Soft_I2C_Sda_Direction at TRISC1_bit;

void    init_i2c()
{
    Soft_I2C_Init();
}
//*****
*
//■■■■■■にページ単位でデータを書き込む関数です。
void    EEPROM_24LC1025_Page_Write(WWORD addr, BYTE *msg, WORD len)

```

```

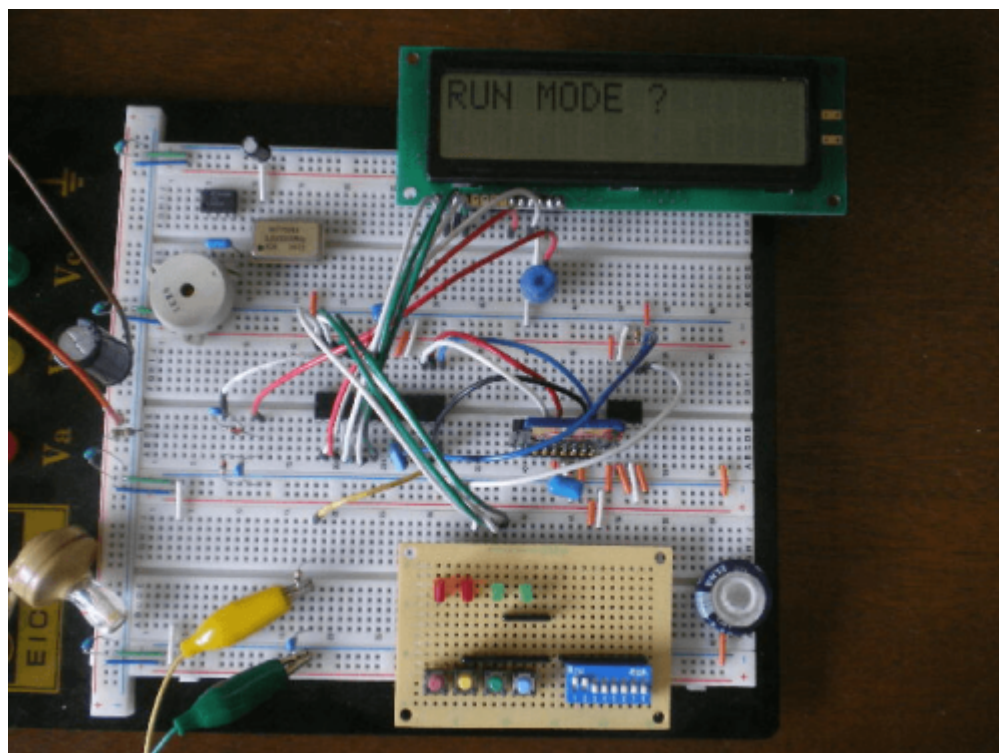
{
    int    cnt;
    //
    Soft_I2C_Start();
    if ((addr & 0x10000) == 0)
        Soft_I2C_Write(0xA0);
    else
        Soft_I2C_Write(0xA8);
    Soft_I2C_Write((addr >> 8) & 0xFF);
    Soft_I2C_Write(addr & 0xFF);
    for (cnt = 0; cnt < len; cnt++) {
        Soft_I2C_Write(msg[cnt]);
    }
    Soft_I2C_Stop();
    //
    Delay_ms(WRITE_CYCLE_TIME);
}

//*****
*
//■■■■■■■■からページ単位でデータを読み込む関数です。
void    EEPROM_24LC1025_Page_Read(WWORD addr, BYTE *msg, WORD len)
{
    int    cnt;
    //
    Soft_I2C_Start();
    if ((addr & 0x10000) == 0)
        Soft_I2C_Write(0xA0);
    else
        Soft_I2C_Write(0xA8);
    Soft_I2C_Write((addr >> 8) & 0xFF);
    Soft_I2C_Write(addr & 0xFF);
    Soft_I2C_Start();
    if ((addr & 0x10000) == 0)
        Soft_I2C_Write(0xA1);
    else
        Soft_I2C_Write(0xA9);
    for (cnt = 0; cnt < (len - 1); cnt++) {
        msg[cnt] = Soft_I2C_Read(ACK);
    }
    msg[cnt] = Soft_I2C_Read(NO_ACK);
    Soft_I2C_Stop();
    //
    Delay_ms(WRITE_CYCLE_TIME);
}

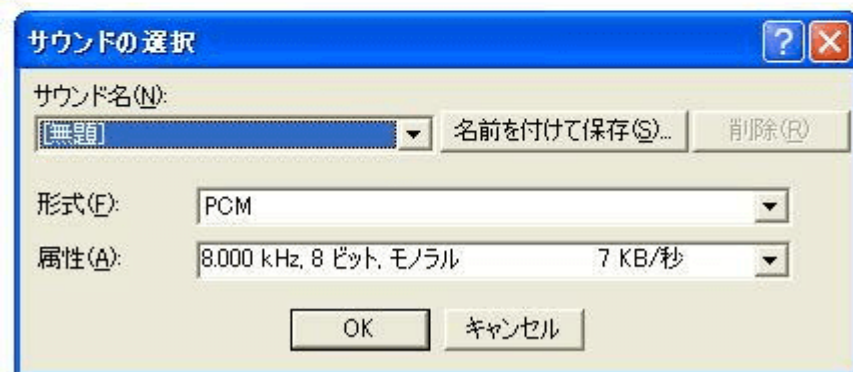
//*****
*

```

動作確認



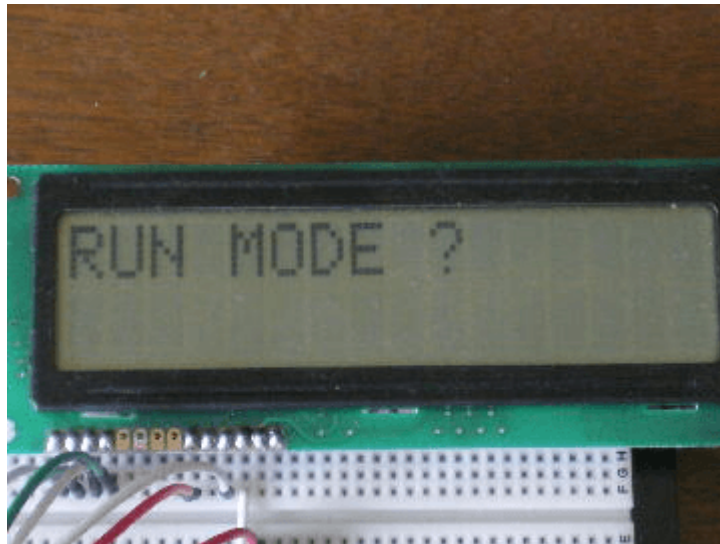
サウンドレコーダ (Windows標準付属ツール)を使用してWaveファイルを作成します。



名前 ▲	サイズ	種類	更新日時
 wave	667 KB	Wave サウンド	2010/07/23 11:30

SDCに保存したWAVEファイル(例)です。

左側:動作スイッチ押下待ちの画面です。右側:SDC内のWaveファイルに記録されているPCM(無圧縮)デー



データを再生しています。



左側:EEPROM内に記録されているPCM(無圧縮)データを再生しています。 右側:SDC内のWaveファイルのPCM(無圧縮)データをeepromに書き込んでいます。





如何ですか？

EEPROMに音声データを記録し、再生することが出来るので、今後の製作物に多いに活かせそうです。例えば、

- 電圧や電流の測定器に置いてLCDや7セグで表示するのではなく、音声で読み上げてくれる。
- 温度制御では、温度の高低を、音声で教えてくれる。
- モーター制御では、回転の異常を、音声で警告してくれる。
- 時計の目覚ましでは、音声で時間を教えてくれる。

著作権表示 copyright notice

このページは稲崎様の閉鎖したHPのコピーで、著作権は稲崎様にあります。詳細 This page is a copy of Mr. Inasaki's closed website, and the copyright is held by him. [Details](#)

From:

<http://deepsky.jp/wiki/> - うごくといいな

Permanent link:

<http://deepsky.jp/wiki/doku.php?id=elechobby:picdic:otherpic:189>

Last update: **2020/05/01 19:58**

