

CWメモリーキーヤ(memory-keyer)V2

概要

前回製作した□CWメモリーキーヤ(memory-keyer)の、記録容量(時間)を増やし、且つ、記録精度を、約4倍に 向上させました。

<仕様>

- 記録チャンネルを4チャンネル搭載する。(1チャンネルあたりの記録時間は、約80秒)
- EEPROMに記録し、電源をOFFにしても記録内容が消えないようにする。
- モールス音(約1kHz)を圧電スピーカで鳴らせる。
- 単三電池2本で動作可能とする。

動作原理

記録および再生の基本的な原理は、CWメモリーキーヤ(memory-keyer)を参照してください。

<記録容量>

- PIC16F88に、EEPROM(24LC64)を、I2C方式で接続します。
- 容量が8kバイトあるので、4チャンネルに分割し、1チャンネルあたりは□2kバイト(16384ビット)とします。

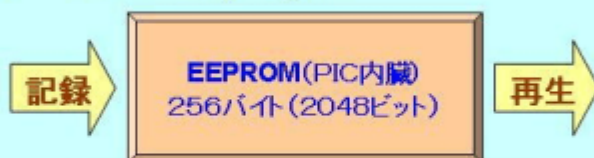
CH0	SW3=OFF	SW4=OFF
CH1	SW3=ON	SW4=OFF
CH2	SW3=OFF	SW4=ON
CH3	SW3=ON	SW4=ON

<記録精度>

- 電鍵信号のサンプリング周期を、5msecとし、1チャンネルあたりの記録時間を、約80秒とします。
80秒 16384ビット×5msec

<チャンネル数と記録容量(時間)>

■CWメモリキーヤ(memory-keyer)



チャンネル	記録容量		記録時間
CH0	256バイト	2048ビット	約40秒(※1) 約60秒(※2)

※1 サンプリング周期(20msec)
※2 サンプリング周期(30msec)

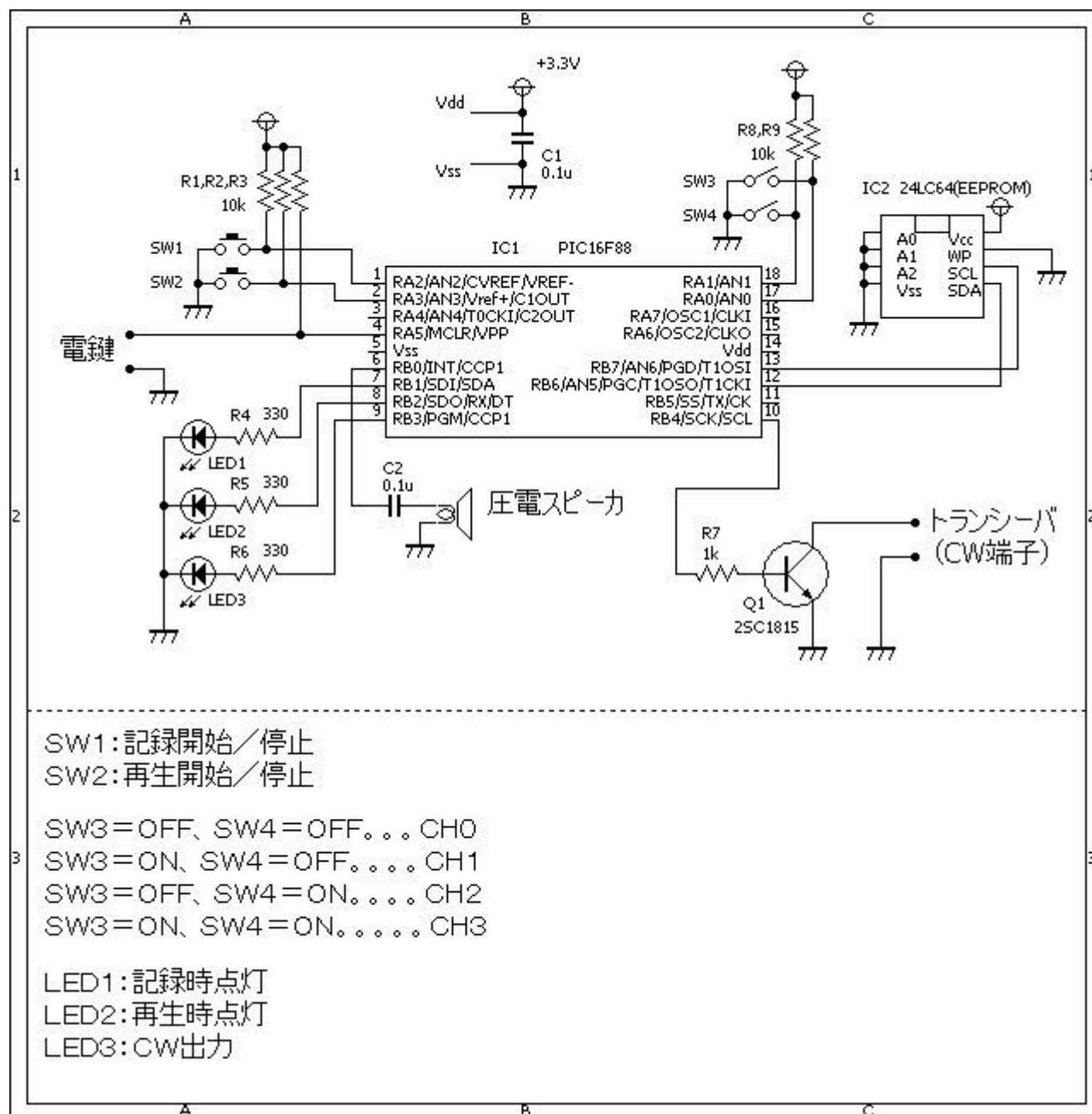
■CWメモリキーヤ(memory-keyer)V2



チャンネル	記録容量		記録時間
CH0	2048バイト	16384ビット	約80秒(※3)
CH1	2048バイト	16384ビット	約80秒(※3)
CH2	2048バイト	16384ビット	約80秒(※3)
CH3	2048バイト	16384ビット	約80秒(※3)

※3 サンプリング周期(5msec)

回路図



ソースコード

[cw_memory_keyer_v2.c](#)

```
//*****
*
/*
   □□□メモリーヤ□□□
*/
//*****
*

#define SW_REC          PORTA.F2
#define SW_PLAY        PORTA.F3
```

```
#define      SW_CH1      PORTA.F0
#define      SW_CH2      PORTA.F1

#define      LED_REC      PORTB.F1
#define      LED_PLAY      PORTB.F2
#define      LED_OUTPUT      PORTB.F3

#define      CW_INPUT      PORTA.F5
#define      CW_OUTPUT      PORTB.F4

#define      DELAY_TIME      5

#define      ACK      1
#define      NO_ACK      0

//*****
*

void    Pwm_Change_DutyEx(unsigned int duty_ratio)
{
    CCP1L = duty_ratio >> 2;
    CCP1CON.F6 = duty_ratio & 0b00000001;
    CCP1CON.F7 = (duty_ratio & 0b00000010) >> 1;
}

//*****
*

void    Delay_ex()
{
    Delay_ms(DELAY_TIME);
}

//*****
*

void    EEPROM_24C64_Init()
{
    Soft_I2C_Config(&PORTB, 6, 7);    //SDA=3 SCL=2
}

//*****
*

unsigned short    EEPROM_24C64_Read(unsigned int addr)
{
    unsigned    short    tmp;
    //
    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);
tmp = Soft_I2C_Read(NO_ACK);    //not acknowledge
Soft_I2C_Stop();
return (tmp);
}

//*****
*

void EEPROM_24C64_Write(unsigned int addr, unsigned short dat)
{
    unsigned short cnt;
    //
    Soft_I2C_Start();
    Soft_I2C_Write(0xA0);
    Soft_I2C_Write((addr >> 8) & 0xFF);
    Soft_I2C_Write(addr & 0xFF);
    Soft_I2C_Write(dat);
    Soft_I2C_Stop();
    Delay_ms(5);
}

//*****
*

void recProc()
{
    static unsigned int cnt1, addr_offset;
    static unsigned short cnt2, tmp;
    //
    if ((SW_CH1 == 1) && (SW_CH2 == 1)) {
        addr_offset = 0;
    }
    if ((SW_CH1 == 0) && (SW_CH2 == 1)) {
        addr_offset = 2048;
    }
    if ((SW_CH1 == 1) && (SW_CH2 == 0)) {
        addr_offset = 2048 * 2;
    }
    if ((SW_CH1 == 0) && (SW_CH2 == 0)) {
        addr_offset = 2048 * 3;
    }
    //
    LED_REC = 1;
    for (cnt1 = 0; cnt1 < 2048; cnt1++) {
        tmp = 0x00;
        for (cnt2 = 0; cnt2 < 8; cnt2++) {
            if (CW_INPUT == 0) {
```

```
        tmp |= 0x01;
        Pwm_Start();
        LED_OUTPUT = 1;
        CW_OUTPUT = 1;
    } else {
        Pwm_Stop();
        LED_OUTPUT = 0;
        CW_OUTPUT = 0;
    }
    if (cnt2 < 7) {
        tmp = tmp << 1;
        Delay_ex();
    }
}
EEPROM_24C64_Write(addr_offset + cnt1, tmp);
Delay_ex();
//
if (SW_REC == 0)
    break;
}
LED_REC = 0;
Pwm_Stop();
LED_OUTPUT = 0;
CW_OUTPUT = 0;
}

//*****
*

void playProc()
{
    static unsigned int cnt1, addr_offset;
    static unsigned short cnt2, tmp;
    //
    if ((SW_CH1 == 1) && (SW_CH2 == 1)) {
        addr_offset = 0;
    }
    if ((SW_CH1 == 0) && (SW_CH2 == 1)) {
        addr_offset = 2048;
    }
    if ((SW_CH1 == 1) && (SW_CH2 == 0)) {
        addr_offset = 2048 * 2;
    }
    if ((SW_CH1 == 0) && (SW_CH2 == 0)) {
        addr_offset = 2048 * 3;
    }
    //
    LED_PLAY = 1;
    for (cnt1 = 0; cnt1 < 2048; cnt1++) {
        tmp = EEPROM_24C64_Read(addr_offset + cnt1);
```

```
        for (cnt2 = 0; cnt2 < 8; cnt2++) {
            if ((tmp & 0x80) != 0) {
                Pwm_Start();
                LED_OUTPUT = 1;
                CW_OUTPUT = 1;
            } else {
                Pwm_Stop();
                LED_OUTPUT = 0;
                CW_OUTPUT = 0;
            }
            tmp = tmp << 1;
            Delay_ex();
        }
        //
        if (SW_PLAY == 0)
            break;
    }
    LED_PLAY = 0;
    Pwm_Stop();
    LED_OUTPUT = 0;
    CW_OUTPUT = 0;
}

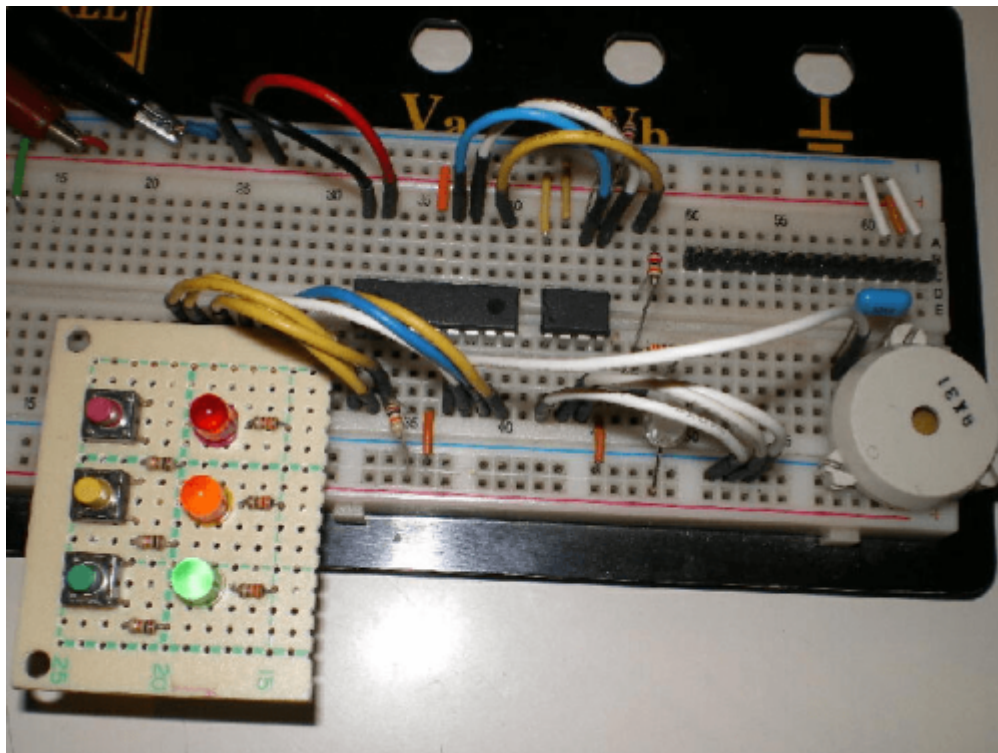
//*****
*

void main()
{
    static    short    cnt;
    //
    //ポート関連の設定
    TRISA = 0b11111111;
    TRISB = 0b00000000;
    OSCCON = 0b01100000;    // クロックを4Mhzに設定する。
    ANSEL = 0b00000000;    // □□□変換は使用しない。
    //
    EEPROM_24C64_Init();
    //PWMの設定□1kHz□
    Pwm_Init(1000);
    Pwm_Change_DutyEx((PR2 * 4) / 2);
    Pwm_Stop();
    //
    for (cnt = 0; cnt < 5; cnt++) {
        LED_REC = 1;
        LED_PLAY = 1;
        LED_OUTPUT = 1;
        Pwm_Start();
        Delay_ms(200);
        Pwm_Stop();
        LED_REC = 0;
        LED_PLAY = 0;
    }
}
```

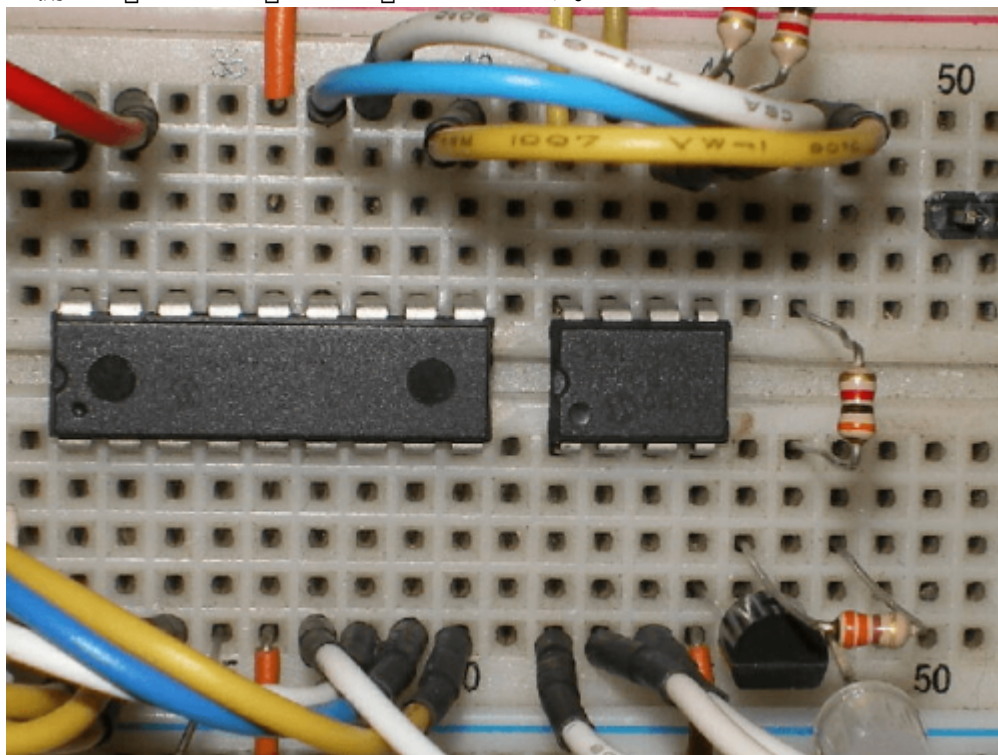
```
    LED_OUTPUT = 0;
    Delay_ms(200);
}
//
while (1) {
    if (SW_REC == 0) {
        while (Button(&PORTA, 2, 1, 1) == 0)
            ;
        //
        recProc();
        //
        while (Button(&PORTA, 2, 1, 1) == 0)
            ;
    }
    if (SW_PLAY == 0) {
        while (Button(&PORTA, 3, 1, 1) == 0)
            ;
        //
        playProc();
        //
        while (Button(&PORTA, 3, 1, 1) == 0)
            ;
    }
}
}

//*****
*
```

動作確認



左側からPIC16F88、24LC64、2SC1815です。



From:

<http://deepsky.jp/wiki/> - うごくといいな

Permanent link:

<http://deepsky.jp/wiki/doku.php?id=elechobby:picdic:pic16f88:115&rev=1588210137>

Last update: 2025/10/17 14:27

