

バッテリー充電ユニット(簡易多段定電流充電方式)

概要

バッテリー(充電電池含む)の充電には様々な方法があるようです。 <サイクル使用>

- $-\Delta V$ 制御充電方式
- dT/dt 制御充電方式
- ステップ制御充電方式
- 電圧制御充電方式
- Vターパ制御充電方式
- 定電圧定電流制御充電方式
- タイマー制御充電方式
- 準定電流充電方式

<スタンバイ使用>

- トリクル充電方式
- 間欠充電方式
- パルス充電方式

今回は、対象をバッテリー(鉛蓄電池)に限定した、簡易な“多段定電流充電方式”で充電するユニットを製作してみました。

<多段定電流充電方式の主な特長>

- 従来の1段による定電流充電に比べ、2-6段の定電流で段階的に減少させて強制的に充電するため、充電時間を短縮することが出来ます。
- 多段充電が電極や電解液の保護に作用するため、サイクル寿命を延ばすことが出来ます。

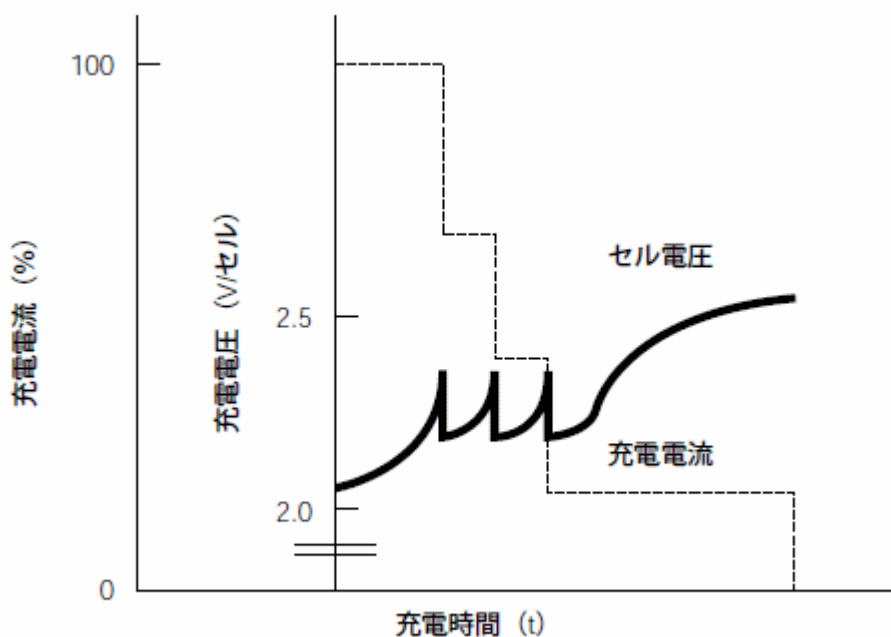


図3 多段定電流充電法 (概念図)

※NTTファシリティーズ総合研

究所で発表された論文よりグラフの引用

鉛蓄電池について 鉛蓄電池を放電したままで放置しておくと、サルフェーションと呼ばれ、充電できない状態になります。 電池寿命の大半はこのサルフェーションによるものなので、放電後は速やかに充電する事が大切です。

動作原理

充電ステップを11段階(定電流=10段、定電圧=1段)としました。 定電流充電は、1000mA~100mAまでを100mA単位で行います。 設定された電流値で定電流制御(電圧を上下)を行います。(例、ステップ0では1000mAの定電流制御)

- バッテリ電圧が、設定された電圧(定電圧)に達すると、次のステップに移ります。

定電圧充電は、13.0V~15.0Vまでを100mV単位で設定可能としました。

- 設定された電圧値で定電圧制御(電圧を上下)を行います。(電流値は無視)

充電ステップ	定電流	定電圧
0	1000mA	-
1	900mA	-
2	800mA	-
3	700mA	-
4	600mA	-
5	500mA	-
6	400mA	-
7	300mA	-
8	200mA	-
9	100mA	-
10	-	14.4V

動作原理(ハードウェア)

電源部 24V1A以上を供給できる電源をご用意ください。

電圧供給部 効率を考慮し、DCDCコンバータ(チョップパ方式の降圧タイプ)を採用しました。

電圧測定

- DCDCコンバータの出力電圧(V2)抵抗による分圧方式
- バッテリ電圧(V1)抵抗による分圧方式
- 電流検出抵抗の両端電圧(V3)微弱電圧のためオペアンプで48倍に増幅

動作原理(ソフトウェア)

◎DCDCコンバータの出力電圧の制御 PWMのデューティ比を変化させて、出力電圧を制御します。
尚、PWMの発振周波数は、約20kHzです。

電圧(V1,V2,V3)の測定 各電圧をA/D変換(1000回)し、その平均値を求めます。

電流の測定 電流検出抵抗(0.1Ω)の両端電圧(V3)より、オームの法則で電流を求めます。

各動作の制御【動作モード0】

- 10段階の定電流で充電を行います。11段目では定電圧制御に切り替わります。(上記の表を参照)
- 開始条件=開始スイッチ押下
- 停止条件=停止スイッチ押下

【動作モード1】

- 10段階の定電流で充電を行います。
- 開始条件=バッテリーの電圧が、充電可能最低電圧を下回った時
- 停止条件=停止スイッチ押下、10段階の定電流で充電が全て完了した時

【動作モード2】

- 充電電圧を設定します□(SW_UP□SW_DOWN)
- 既定値は、14.4Vです。(設定範囲=13.0V~15.0V)

【動作モード3】

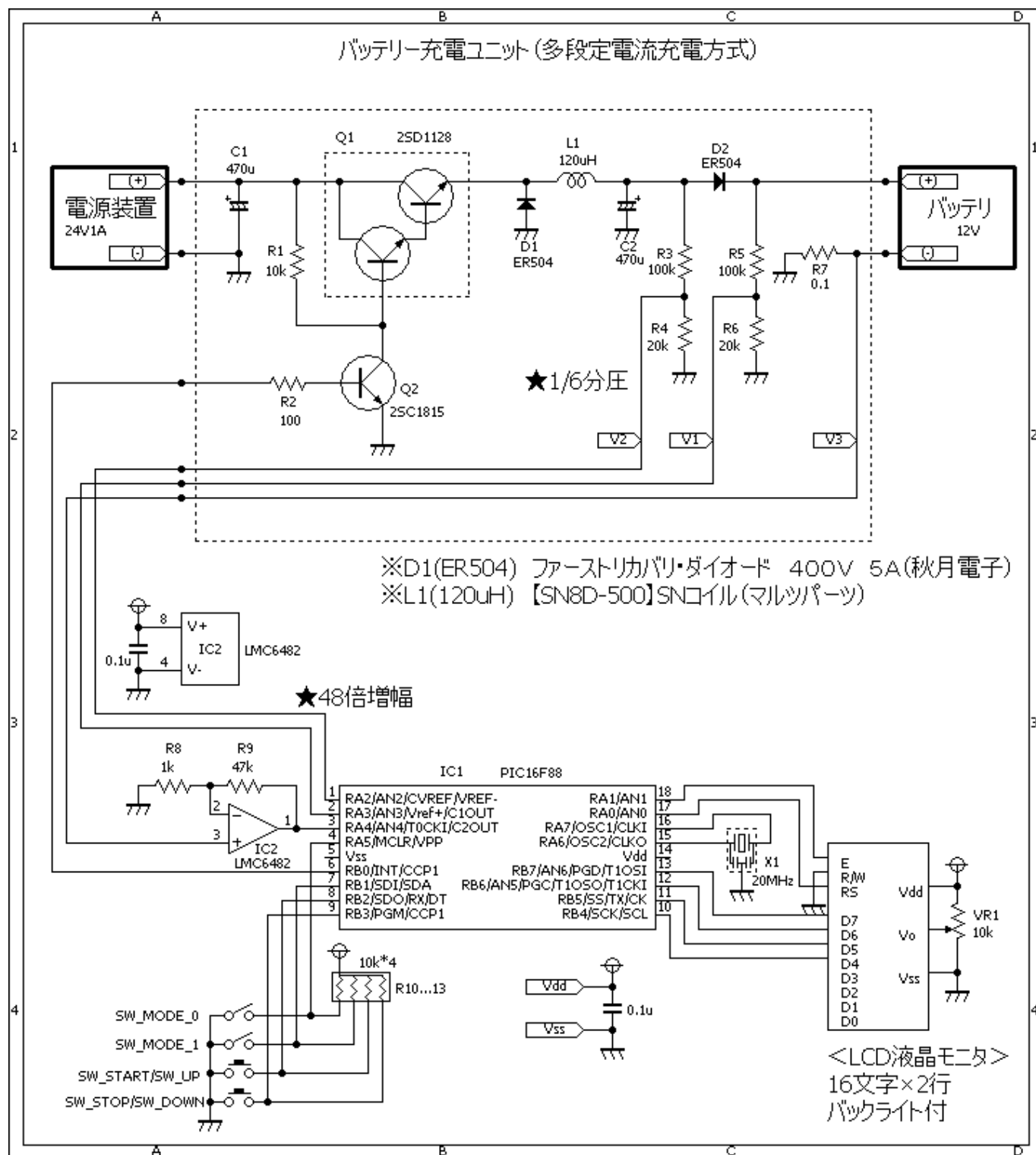
- 充電可能最低電圧を設定します□(SW_UP□SW_DOWN)
- 既定値は、10.5Vです。(設定範囲=9.0~12.0V)

【動作モード(その他)】

- 開始スイッチ(SW_START)を押下しながら、起動することにより、オペアンプのオフセット電圧をキャンセルすることが出来ます。その時には、電源装置をオフの状態、バッテリー未接続の状態にしてください。

動作モード	SW_MODE_1	SW_MODE_0
モード0	オン(0)	オン(0)
モード1	オン(0)	オフ(1)
モード2	オフ(1)	オン(0)
モード3	オフ(1)	オフ(1)
その他	起動時に開始スイッチ(SW_START)を押下	

回路図



ソースコード

[battery_charger.c](#)

```
//*****  
*  
/*  
  
    <バッテリー充電ユニット (多段定電流充電方式)>  
    ■ 動作モード 0
```

10段階の定電流で充電を行います。11段目では定電圧制御に切り替わります。

開始条件 = 開始スイッチ押下

停止条件 = 停止スイッチ押下

■ 動作モード 1

10段階の定電流で充電を行います。

開始条件 = バッテリーの電圧が、充電可能最低電圧を下回ったとき

停止条件 = 停止スイッチ押下、10段階の定電流で充電が全て完了した時

■ 動作モード 2

充電電圧を設定します。

既定値は、14.4Vです。(設定範囲 13.0V 15.0V)

■ 動作モード 3

充電可能最低電圧を設定します。

既定値は、10.5Vです。(設定範囲 9.0 12.0V)

■ その他

開始スイッチを押下しながら、起動することにより、オペアンプのオフセットをキャンセルすることが出来ます。

その時には、電源装置をオフの状態、バッテリー未接続の状態にしてください。

*/

//*****

*

//LCD

```
sbit LCD_RS at RA0_bit;
sbit LCD_EN at RA1_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 TRISA0_bit;
sbit LCD_EN_Direction at TRISA1_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;
```

//

```
sbit SW_MODE_0 at RA5_bit;
sbit SW_MODE_1 at RB1_bit;
sbit SW_UP at RB2_bit;
sbit SW_DOWN at RB3_bit;
sbit SW_START at RB2_bit;
sbit SW_STOP at RB3_bit;
```

//

```
#define MODE_0 0
#define MODE_1 1
#define MODE_2 2
#define MODE_3 3
```

//

```
#define BYTE unsigned short
#define WORD unsigned int
#define DWORD unsigned long
```

//*****

*

// ■ 関数宣言

```
extern void main();
```

```
extern void PWM1_Change_DutyEx(unsigned int duty_ratio);
extern WORD ADC_Get_Sample_Average(unsigned short channel);
extern void control();
extern void run_mode_0();
extern void run_mode_1();
extern void run_mode_2();
extern void run_mode_3();
extern short get_mode();
extern void error_disp(char num);
extern void voltage_init();
extern short voltage_set(char c);
extern void voltage_on();
extern void voltage_off();
extern void voltage_get();
//*****
*
char buf[10];
union {
    double _double;
    short _short[4];
} cal;
//*****
*
//■PWMデューティ設定関数
void PWM1_Change_DutyEx(unsigned int duty_ratio)
{
    CCP1L = duty_ratio >> 2;
    CCP1CON.CCP1Y = (duty_ratio & 0b00000001) != 0 ? 1 : 0;
    CCP1CON.CCP1X = (duty_ratio & 0b00000010) != 0 ? 1 : 0;
}
//*****
*
//■A/D変換(10000回平均)関数
WORD ADC_Get_Sample_Average(unsigned short channel)
{
    DWORD ad;
    WORD cnt;
    //
    ad = 0;
    for (cnt = 0; cnt < 1000; cnt++) {
        ad += ADC_Get_Sample(channel);
    }
    return (ad / 1000);
}
//*****
*
//■エラー表示関数
char *error_msg = "error! (?) ";
//
void error_disp(char num)
{

```

```
    short    cnt;
    //
    for (cnt = 0; cnt < 10; cnt++) {
        error_msg[8] = num + '0';
        Lcd_Out(2, 1, error_msg);
        Delay_ms(100);
        error_msg[8] = ' ';
        Lcd_Out(2, 1, error_msg);
        Delay_ms(100);
    }
}

//*****
*
// ■ 電圧制御 (初期化) 関数
int    duty = 1023;
//
void    voltage_init()
{
    duty = 1023;
    PWM1_Init(20000);          //20kHz
    PR2 = 0xFF;
    PWM1_Stop();
    TRISB.B0 = 0;
    PORTB.B0 = 1;
}

//*****
*
// ■ 電圧制御 (設定) 関数
short    voltage_set(char c)
{
    switch (c) {
        case '0':
            duty = 1023;
            PWM1_Change_DutyEx(duty);
            break;
        case '-':
            if (duty < 1023) {
                duty++;
                PWM1_Change_DutyEx(duty);
            } else {
                error_disp(1);
                return(-1);
            }
            break;
        case '+':
            if (duty > 10) {
                duty--;
                PWM1_Change_DutyEx(duty);
            } else {
                error_disp(2);
                return(-1);
            }
    }
}
```

```
        }
        break;
    }
    return(0);
}

//*****
*
// ■ 電圧制御 ( オン ) 関数
void    voltage_on()
{
    PWM1_Change_DutyEx(duty);
    PWM1_Start();
}
//*****
*
// ■ 電圧制御 ( オフ ) 関数
void    voltage_off()
{
    PWM1_Stop();
    TRISB.B0 = 0;
    PORTB.B0 = 1;
}
//*****
*
// ■ 電圧測定関数
double  v1, v2, v3, offset = 0.0;
//
void    voltage_get()
{
    //電圧を測定します。
    v1 = ADC_Get_Sample_Average(3);
    v2 = ADC_Get_Sample_Average(2);
    v3 = ADC_Get_Sample_Average(4);
    //電圧値の補正および電流値を求めます。
    v1 = v1 * 4.8828125 * 6; //分圧□100kΩ□20kΩ□
    v2 = v2 * 4.8828125 * 6; //分圧□100kΩ□20kΩ□
    v3 = (v3 * 4.8828125) / 48; //増幅□47kΩ□1kΩ□
    v3 = v3 - offset;
    v1 = v1 - v3;
}
//*****
*
// ■ 多段定電流充電制御関数
short   step = 0;
int      minpoint_v = 10500;
int      setpoint_v = 14400;
int      setpoint_i[] = {1000, 900, 800, 700, 600, 500, 400, 300, 200,
100};
//
short    control()
{

```

```
double i;
//電圧を測定します。
voltage_get();
//電流を求めます。(オームの法則)
i = v3 / 0.1;
//□□□に表示します。
WordToStr(v1, buf);
Lcd_Out(1, 1, buf);
WordToStr(i, buf);
Lcd_Out(1, 10, buf);
WordToStr(v2, buf);
Lcd_Out(2, 1, buf);
WordToStr(duty, buf);
Lcd_Out(2, 12, buf);
//
if (step == 11) {
    Lcd_Chrc(1, 9, '-');
    return(0);
}
//
if (step == 10) {
    //定電圧制御を行います。
    Lcd_Chrc(1, 9, '*');
    if (v1 > setpoint_v) {
        if (voltage_set('-') == -1) {
            return(-1);
        }
    } else {
        if (voltage_set('+') == -1) {
            return(-1);
        }
    }
    return(0);
}
//多段定電流制御を行います。
Lcd_Chrc(1, 9, step + '0');
if (i > setpoint_i[step]) {
    if (voltage_set('-') == -1) {
        return(-1);
    }
} else {
    if (voltage_set('+') == -1) {
        return(-1);
    }
}
//設定電圧を超過すると電圧を下げ、電流の設定値を下げます。
if (v1 > setpoint_v) {
    step++;
    voltage_set('0');
    Delay_ms(1000);
}
```

```
        return(0);
    }
//*****
*
// ■ モード0 関数
void    run_mode_0()
{
    if (SW_START == 1) {
        step = 11;
        control();
        return;
    }
    //
    step = 0;
    voltage_on();
    voltage_set('0');
    //
    while (SW_STOP == 1) {
        if (control() == -1) {
            break;
        }
    }
    voltage_off();
}
//*****
*
// ■ モード1 関数
void    run_mode_1()
{
    voltage_get();
    if (v1 > minpoint_v) {
        step = 11;
        control();
        return;
    }
    //
    step = 0;
    voltage_on();
    voltage_set('0');
    //
    while (SW_STOP == 1) {
        if (control() == -1) {
            break;
        }
        if (step == 10) {
            break;
        }
    }
    voltage_off();
}
//*****
```

```
*
// ■ モード2 関数
void run_mode_2()
{
    WordToStr(setpoint_v, buf);
    Lcd_Out(1, 1, buf);
    //
    if (SW_UP == 0) {
        while (SW_UP == 0) {
            Delay_ms(100);
        }
        //
        if (setpoint_v < 15000) {
            setpoint_v += 100;
        }
        //
        EEPROM_Write(0, (setpoint_v >> 8) & 0xFF);
        EEPROM_Write(1, setpoint_v & 0xFF);
        //
        return;
    }
    if (SW_DOWN == 0) {
        while (SW_DOWN == 0) {
            Delay_ms(100);
        }
        //
        if (setpoint_v > 13000) {
            setpoint_v -= 100;
        }
        //
        EEPROM_Write(0, (setpoint_v >> 8) & 0xFF);
        EEPROM_Write(1, setpoint_v & 0xFF);
        //
        return;
    }
}

//*****
*
// ■ モード3 関数
void run_mode_3()
{
    WordToStr(minpoint_v, buf);
    Lcd_Out(1, 1, buf);
    //
    if (SW_UP == 0) {
        while (SW_UP == 0) {
            Delay_ms(100);
        }
        //
        if (minpoint_v < 12000) {
            minpoint_v += 100;
        }
    }
}
```

```
    }
    //
    EEPROM_Write(2, (minpoint_v >> 8) & 0xFF);
    EEPROM_Write(3, minpoint_v & 0xFF);
    //
    return;
}
if (SW_DOWN == 0) {
    while (SW_DOWN == 0) {
        Delay_ms(100);
    }
    //
    if (minpoint_v > 9000) {
        minpoint_v -= 100;
    }
    //
    EEPROM_Write(2, (minpoint_v >> 8) & 0xFF);
    EEPROM_Write(3, minpoint_v & 0xFF);
    //
    return;
}
}
//*****
*
// ■ モード取得関数
short flag = -1;
//
short get_mode()
{
    if ((SW_MODE_1 == 0) && (SW_MODE_0 == 0)) {
        if (flag != 0) {
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1, 6, "mV");
            Lcd_Out(2, 6, "mV");
            Lcd_Out(1, 15, "mA");
        }
        flag = 0;
        return (MODE_0);
    }
    if ((SW_MODE_1 == 0) && (SW_MODE_0 == 1)) {
        if (flag != 1) {
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1, 6, "mV");
            Lcd_Out(2, 6, "mV");
            Lcd_Out(1, 15, "mA");
        }
        flag = 1;
        return (MODE_1);
    }
    if ((SW_MODE_1 == 1) && (SW_MODE_0 == 0)) {
        if (flag != 2) {
```

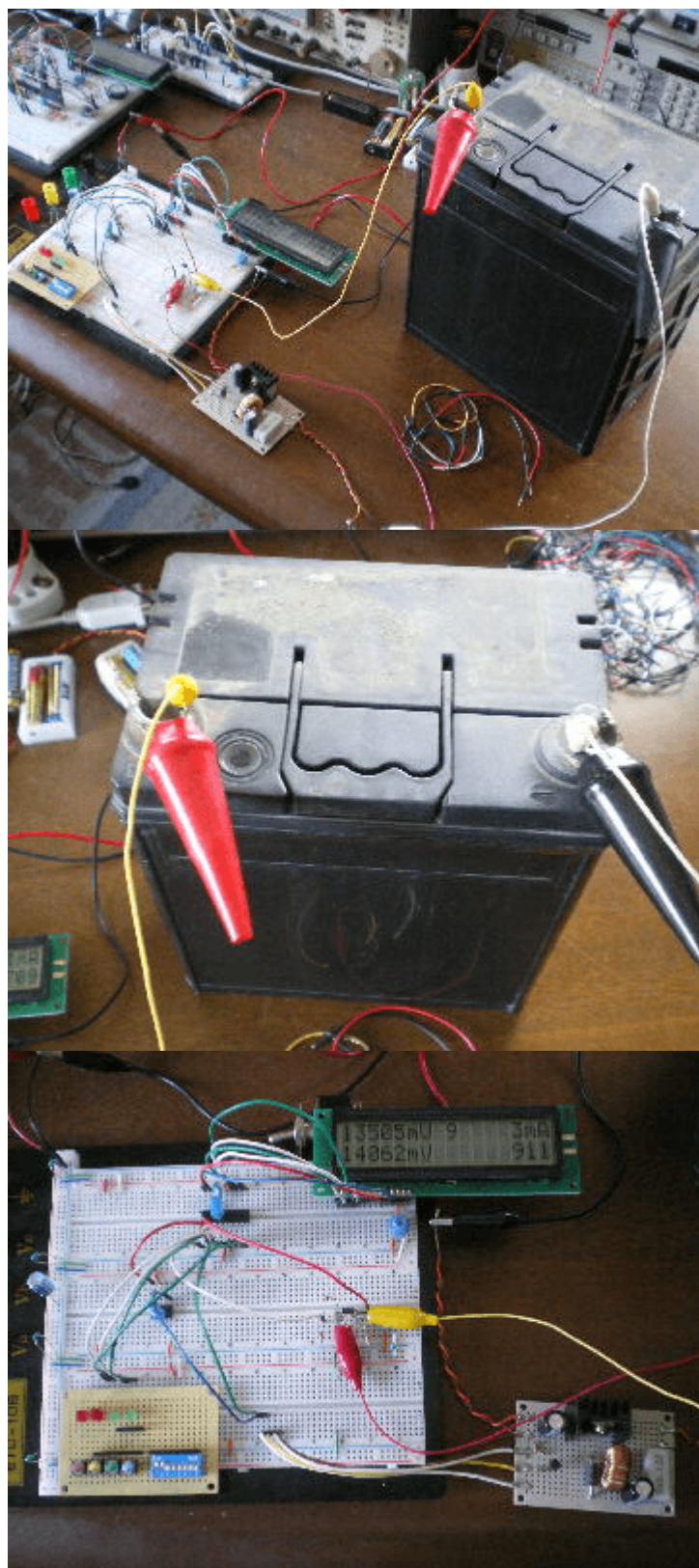
```
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1, 6, "mV");
    }
    flag = 2;
    return (MODE_2);
}
if ((SW_MODE_1 == 1) && (SW_MODE_0 == 1)) {
    if (flag != 3) {
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1, 6, "mV");
    }
    flag = 3;
    return (MODE_3);
}
}

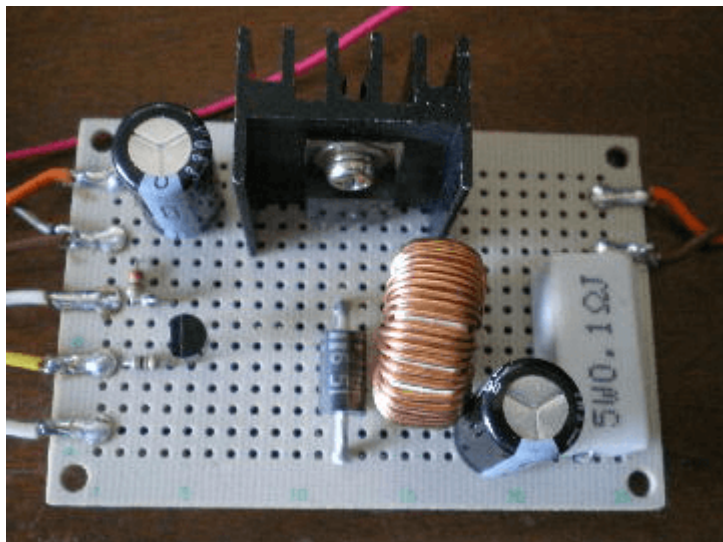
//*****
*
// ■ オペアンプオフセット電圧取得関数
void opeamp_offset_get()
{
    while (SW_START == 0) {
        v3 = ADC_Get_Sample_Average(4);
        v3 = (v3 * 4.8828125) / 48; //増幅47kΩ1kΩ
        offset = v3;
        WordToStr(v3, buf);
        Lcd_Out(1, 1, "offset=");
        Lcd_Out(1, 8, buf);
        Lcd_Out(1, 13, "mV");
    }
    cal._double = offset;
    Eeprom_Write(4, cal._short[0]);
    Eeprom_Write(5, cal._short[1]);
    Eeprom_Write(6, cal._short[2]);
    Eeprom_Write(7, cal._short[3]);
    Delay_ms(1000);
}

//*****
*
// ■ メイン関数
void main()
{
    ANSEL = 0b00001100;
    TRISA = 0b00111100;
    TRISB = 0b00001110;
    //
    ADC_Init();
    //
    Lcd_Init();
    Lcd_Cmd(_LCD_CURSOR_OFF);
    Lcd_Cmd(_LCD_CLEAR);
    Lcd_Out(1, 1, "Battery Charger");
```

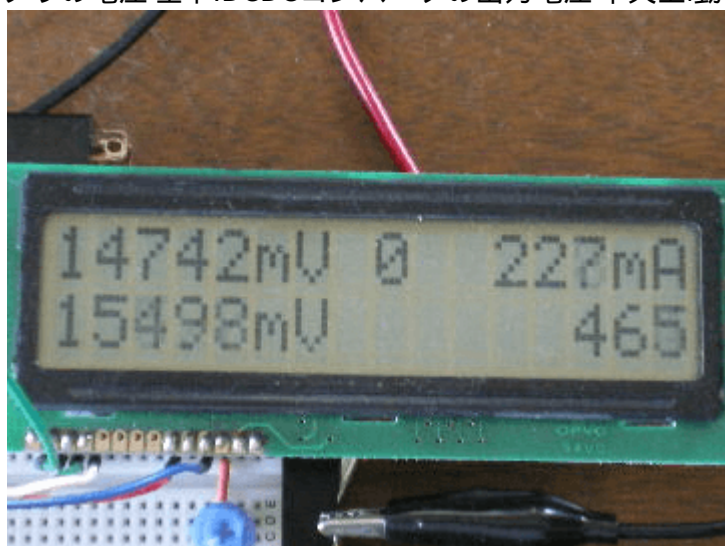
```
Delay_ms(1000);
Lcd_Cmd(_LCD_CLEAR);
//
voltage_init();
//設定値の読み込み
setpoint_v = EEPROM_Read(0);
setpoint_v <= 8;
setpoint_v = setpoint_v | EEPROM_Read(1);
if ((setpoint_v < 13000) || (setpoint_v > 15000)) {
    setpoint_v = 14400;
}
minpoint_v = EEPROM_Read(2);
minpoint_v <= 8;
minpoint_v = minpoint_v | EEPROM_Read(3);
if ((minpoint_v < 9000) || (minpoint_v > 12000)) {
    minpoint_v = 10500;
}
cal._short[0] = Eeprom_Read(4);
cal._short[1] = Eeprom_Read(5);
cal._short[2] = Eeprom_Read(6);
cal._short[3] = Eeprom_Read(7);
offset = ((cal._double < 0.0) || (cal._double > 100.0)) ? 0.0 :
cal._double;
//オペアンプオフセット電圧取得
if (SW_START == 0) {
    opeamp_offset_get();
}
//
while (1) {
    switch (get_mode()) {
    case MODE_0:
        run_mode_0();
        break;
    case MODE_1:
        run_mode_1();
        break;
    case MODE_2:
        run_mode_2();
        break;
    case MODE_3:
        run_mode_3();
        break;
    }
}
}
//*****
*
```

動作確認

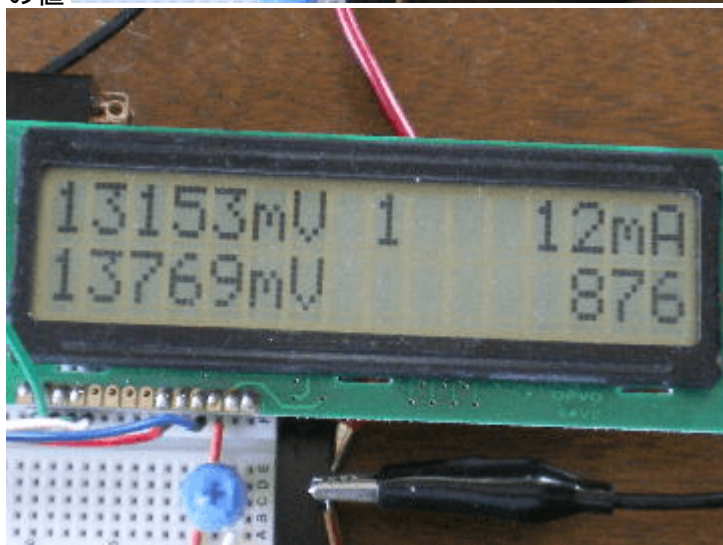


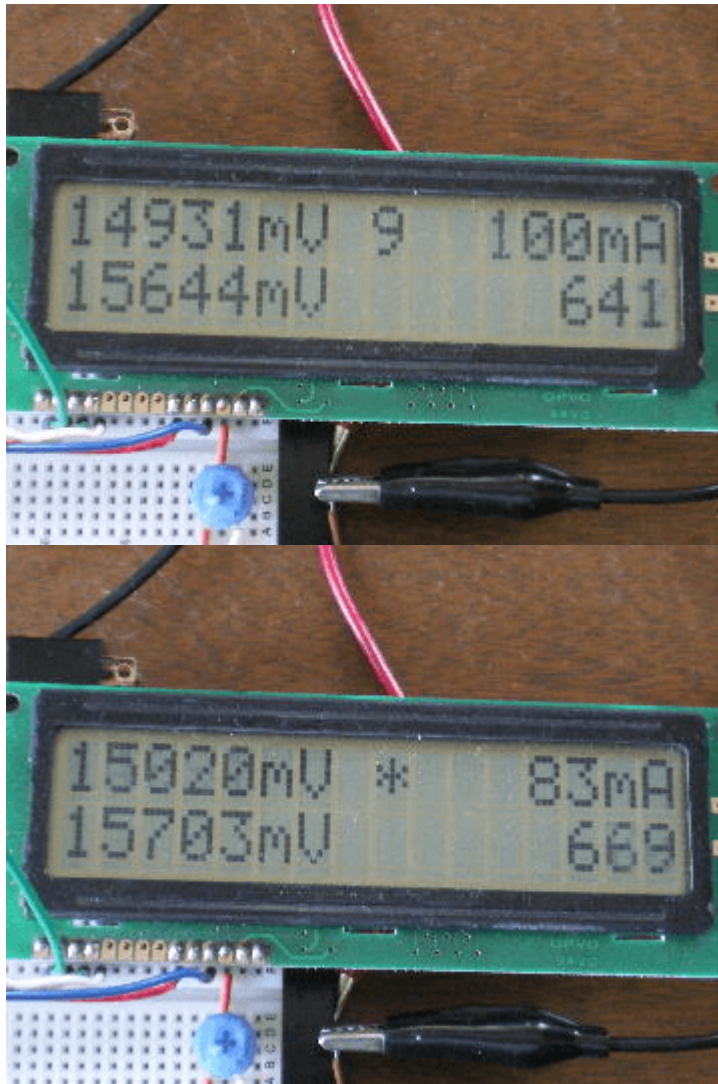


多段階定電流充電を行っているところです。(ステップ0、1.....9、定電圧充電) <LCD表示の説明> 左上: バッテリの電圧 左下:DCDCコンバータの出力電圧 中央上:動作ステップ番号 右上:充電電流値 右下:PWM

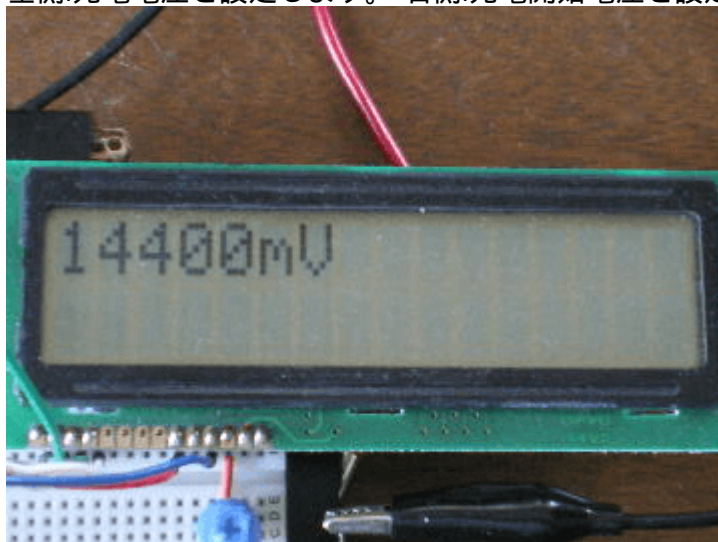


の値





左側:充電電圧を設定します。 右側:充電開始電圧を設定します。





From:
<http://deepsky.jp/wiki/> - うごくといいな

Permanent link:
<http://deepsky.jp/wiki/doku.php?id=elechobby:picdic:pic16f88:148&rev=1588227969>

Last update: **2025/10/17 14:28**

