簡易可変型安定化電源(キーパッド入力)(PIC18F2620)

概要

前回製作した、簡易可変型安定化電源の操作性を向上させるために、キーパッド(keypad)を接続し、設定電圧の値を直接入力可能としました。

<仕様>

- 出力電圧範囲□0V~15Vを、0.1V単位で設定可能。
- 出力電流→約500mA
- 電圧設定方法(1)→キーパッドによる直接指定。(例:12.3Vの場合、“123*“と入力します)
- 電圧設定方法(2)□0.1V単位でアップ(キー□“B”押下)□0.1V単位でダウン(キー□“C“押下)、
- リセット機能→出力電圧を“0V”に強制設定。(キー“A“押下)
- 制御&ホールド機能(キー“D”押下で機能がフリップフロップします)
  - →設定電圧=出力電圧になるように制御します。
  - →制御機能をオフにし、現在値を維持します。
- 設定電圧保持機能→設定値をEEPROMに記録し、再起動での再設定の手間を省く。
- 表示内容→設定電圧値、出力電圧値、キー入力値、出力電圧バー

動作原理

キーパッドによるキー入力以外は、簡易可変型安定化電源と同等なので、其方を参照してください。

動作原理(ハードウェア)

PIC16F88をPIC18F2620に変更し、キーパッドを接続した以外は、簡易可変型安定化電源と同等なので、其方を参照してください。

動作原理(ソフトウェア)

キーパッドによるキー入力以外は、簡易可変型安定化電源と同等なので、其方を参照してください。 ◎キーパッドの初期化とデータ取得

- mikroCが提供する、キーパッド制御ライブラリを使用します。
- ポートBの8ビットを、16キー(4×4)のキーパッドの制御用にアサイン(割り当て)します□(Keypad_Init関数)
- キーをサーチし、クリックされたキー値を取得します□(Keypad_Key_Click関数+チャタリング除去処理)

操作方法

◎キー入力値のセット

- “０“～“９”のキーを使用してセットします。
- セット範囲は、0.0V~15.0Vです。
- “＃“キー押下により、キー入力値を“０”クリアします。

◎設定電圧値のセット

- “＊“キー押下により、キー入力値を設定電圧値にセットします。
- “B”キー押下により、設定電圧値を+100mVします。
- “C“キー押下により、設定電圧値を-100mVします。

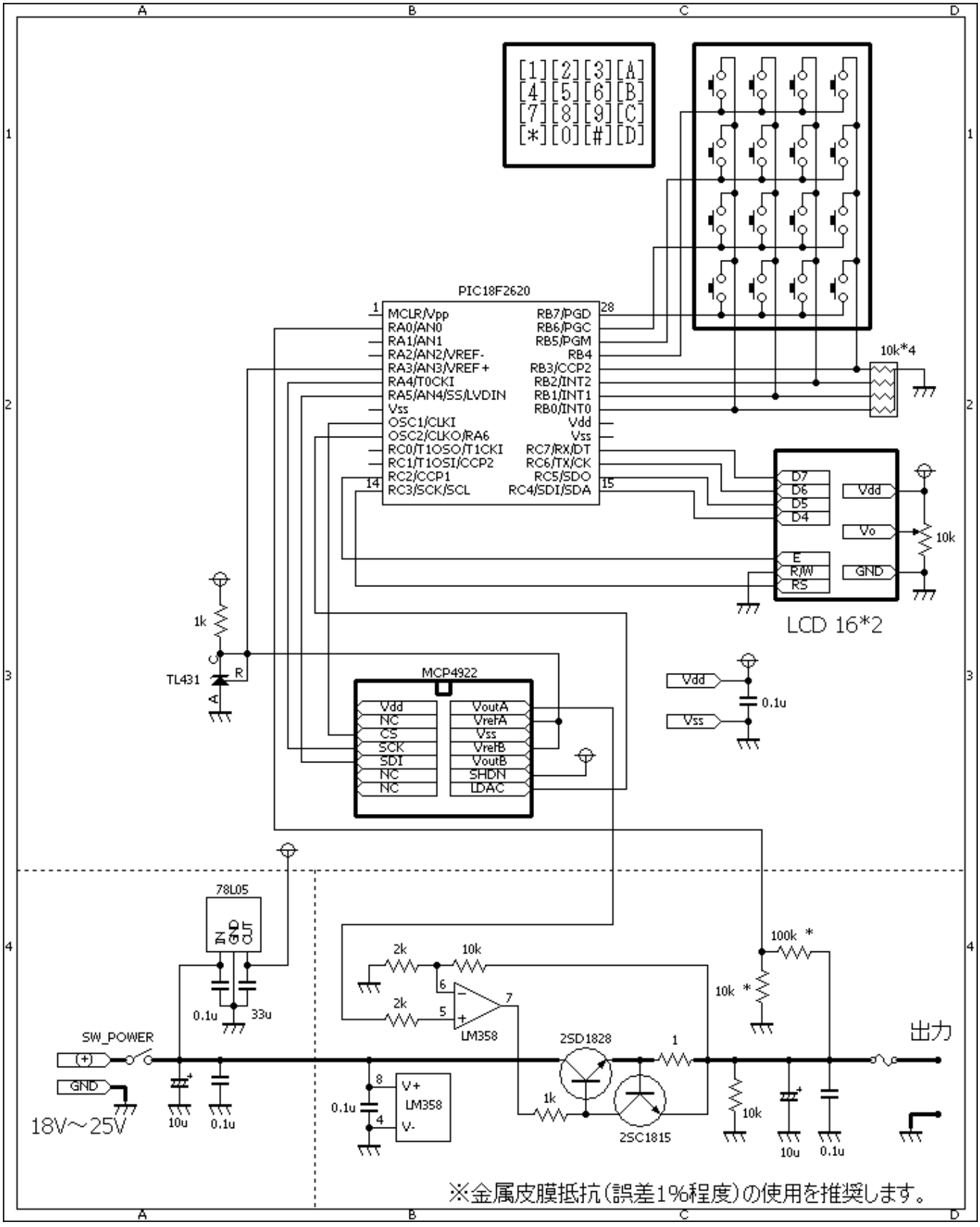
◎出力電圧の制御とホールド(キ“D”押下で機能がフリップフロップします)

- “D“キー押下により、設定電圧=出力電圧になるように制御します。
- “D”キー押下により、制御機能をオフにし、現在値を維持します。

◎設定電圧、出力電圧のクリア

- “A“キー押下により、設定電圧、出力電圧共に“０”クリアします。

回路図



ソースコード

power_supply_v2.c

```
//*********************************************************************
*
/*
    <簡易可変型安定化電源(キーパッド入力)>
*/
//*********************************************************************
*
#define BYTE    unsigned  short
#define WORD    unsigned  int
#define MODE_HOLD    0
#define MODE_CNTL    1
//*********************************************************************
*
extern  void    main();
extern  void    init_adc();
extern  void    init_lcd();
extern  void    init_keypad();
extern  void    init_dac();
extern  void    dac_set(char ch, WORD dt);
extern  WORD    measulement();
extern  void    BarDisp(char row, char column, short dat, short bar);
extern  void    RegistCustomChar();
extern  short   get_keypad();
//*********************************************************************
*
//■■■メイン関数■■■
void    main()
{
        int     da, tmp, v_target, v_keyin;
        double  v_out;
        char    buf[16];
        short   key, mode;
        //
        OSCCON.IRCF2 = 1;
        OSCCON.IRCF1 = 1;
        OSCCON.IRCF0 = 1;
        //
        init_adc();
        init_dac();
        init_keypad();
        init_lcd();
        Lcd_Out(1, 5, "V");
        Lcd_Out(1, 11, "V");
        Lcd_Out(2, 5, "V");
        //
        da = 0;
        v_target = EEPROM_Read(0);
        v_target <<= 8;
        v_target |= EEPROM_Read(1);
        if ((v_target < 0) || (v_target > 15000)) {
                v_target = 0;
```

```
        }
        //
        v_keyin = 0;
        mode = MODE_CNTL;
        Lcd_Chr(1, 16, '!');
        //
        while (1) {
                //出力電圧を測定します。
                v_out = measulement();
                v_out = v_out * 2.4365234375 * 11.0;
                //出力電圧、設定値、キー入力値を表示します。
                tmp = v_out;
                if ((tmp % 100) >= 50) {
                        tmp = ((tmp / 100) + 1) * 100;
                } else {
                        tmp = (tmp / 100) * 100;
                }
                WordToStr(tmp, buf);
                buf[4] = 0x00;
                buf[3] = (buf[2] == ' ') ? '0' : buf[2];
                buf[2] = '.';
                buf[1] = (buf[1] == ' ') ? '0' : buf[1];
                Lcd_Out(2, 1, buf);
                BarDisp(2, 7, v_out / 375, 40);
                //
                WordToStr(v_target, buf);
                buf[4] = 0x00;
                buf[3] = (buf[2] == ' ') ? '0' : buf[2];
                buf[2] = '.';
                buf[1] = (buf[1] == ' ') ? '0' : buf[1];
                Lcd_Out(1, 1, buf);
                //
                WordToStr(v_keyin * 100, buf);
                buf[4] = 0x00;
                buf[3] = (buf[2] == ' ') ? '0' : buf[2];
                buf[2] = '.';
                buf[1] = (buf[1] == ' ') ? '0' : buf[1];
                Lcd_Out(1, 7, buf);
                //
                key = get_keypad();
                switch (key) {
                case 'A'://設定値と出力を“0V”にします。
                        da = 0;
                        dac_set('A', da);
                        v_target = 0;
                        EEPROM_Write(0, (v_target >> 8) & 0xFF);
                        EEPROM_Write(1, v_target & 0xFF);
                        break;
                case 'B'://設定値を（+100mV単位）します。
                        v_target += 100;
                        if (v_target > 15000) {
```

```
                        v_target = 15000;
                }
                EEPROM_Write(0, (v_target >> 8) & 0xFF);
                EEPROM_Write(1, v_target & 0xFF);
                break;
        case 'C'://設定値を（-100mV単位）します。
                v_target -= 100;
                if (v_target < 0) {
                        v_target = 0;
                }
                EEPROM_Write(0, (v_target >> 8) & 0xFF);
                EEPROM_Write(1, v_target & 0xFF);
                break;
        case 'D'://制御のオン・オフを行います。（フリップフロップ）
                if (mode == MODE_CNTL) {
                        mode = MODE_HOLD;
                        Lcd_Chr(1, 16, '-');
                } else {
                        mode = MODE_CNTL;
                        Lcd_Chr(1, 16, '!');
                }
                break;
        case '#'://キー入力値をクリアする。
                v_keyin = 0;
                break;
        case '*'://キー入力値を設定値に決定する。
                v_target = v_keyin * 100;
                v_keyin = 0;
                EEPROM_Write(0, (v_target >> 8) & 0xFF);
                EEPROM_Write(1, v_target & 0xFF);
                break;
        case '0':
        case '1':
        case '2':
        case '3':
        case '4':
        case '5':
        case '6':
        case '7':
        case '8':
        case '9':
                v_keyin = v_keyin % 100;
                v_keyin = (v_keyin * 10) + (key - '0');
                if ((v_keyin < 0) || (v_keyin > 150)) {
                        v_keyin = 0;
                }
                break;
        }
        //
        if (mode == MODE_HOLD) {
```

```
                        continue;
                }
                //出力電圧が設定値になるように制御します。
                if (v_out < v_target) {
                        if ((v_target - v_out) > 2000) {
                                da += 200;
                        }
                        if ((v_target - v_out) > 1000) {
                                da += 100;
                        }
                        if ((v_target - v_out) > 100) {
                                da += 10;
                        } else {
                                da++;
                        }
                        if (da >= 4096) {
                                da = 4095;
                        }
                }
                if (v_out > v_target) {
                        if ((v_out - v_target) > 2000) {
                                da -= 200;
                        }
                        if ((v_out - v_target) > 1000) {
                                da -= 100;
                        }
                        if ((v_out - v_target) > 100) {
                                da -= 10;
                        } else {
                                da--;
                        }
                        if (da < 0) {
                                da = 0;
                        }
                }
                dac_set('A', da);
                Delay_ms(10);
        }
}
//*****************************************************************************
*
//■■■□□□初期化関数■■■
void    init_adc()
{
        ADCON1.PCFG3 = 1;
        ADCON1.PCFG2 = 0;
        ADCON1.PCFG1 = 1;
        ADCON1.PCFG0 = 1;
        ADCON1.VCFG1 = 0;
        ADCON1.VCFG0 = 1;
        ADC_Init();
```

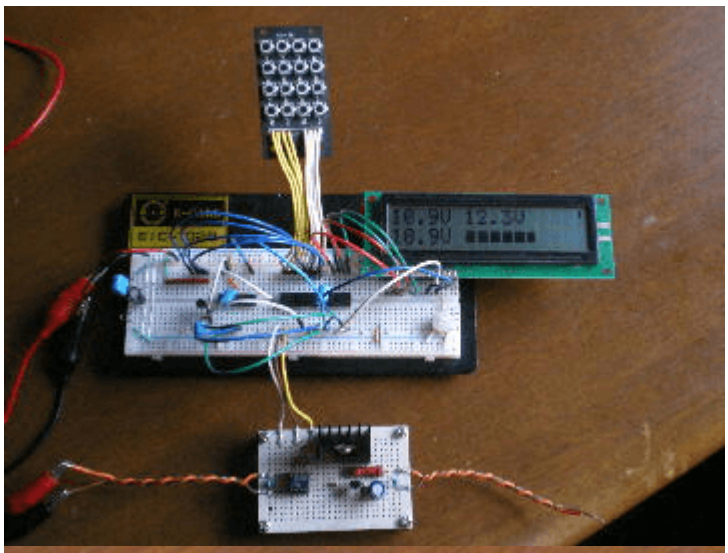
```
}
//*********************************************************************
*
//■■■キーパッド初期化関数■■■
char keypadPort at PORTB;
//
void    init_keypad()
{
        Keypad_Init();
}
//*********************************************************************
*
//■■■キー取得関数■■■
short   kp_tbl[16] =
{'1','2','3','A','4','5','6','B','7','8','9','C','*','0','#','D',};
short   get_keypad()
{
        char    kp;
        //
        while (1) {
                //kp = Keypad_Key_Press();
                kp = Keypad_Key_Click();
                if (kp == 0) {
                        return (-1);
                }
                       Delay_ms(100);
                while (Keypad_Key_Click() != 0) {
                        Delay_ms(10);
                }
                kp = kp_tbl[kp - 1];
                return (kp);
        }
}
//*********************************************************************
*
//■■■出力電圧測定関数■■■
WORD    measulement()
{
        WORD    ad;
        short   cnt;
        //
        ad = 0;
        for (cnt = 0; cnt < 50; cnt++) {
                ad += ADC_Get_Sample(0);
        }
        return (ad / 50);
}
//*********************************************************************
*
//■■■□□□初期化関数■■■
sbit CS at RA7_bit;
```

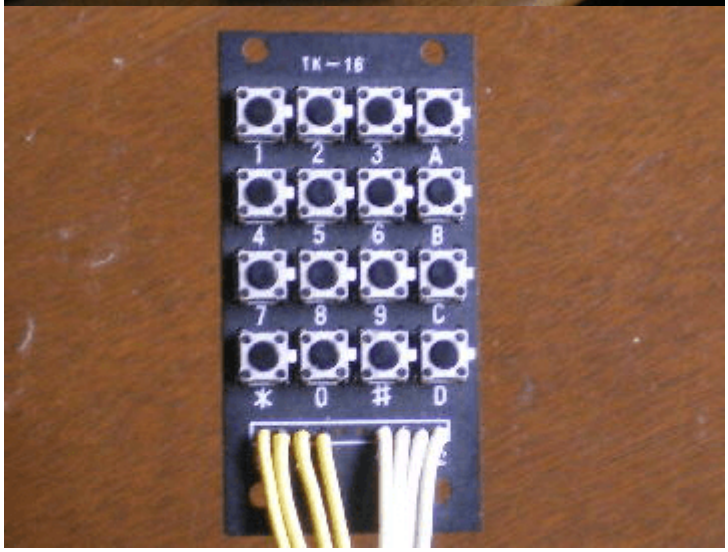
```
sbit CS_Direction at TRISA7_bit;
sbit LDAC at RA6_bit;
sbit LDAC_Direction at TRISA6_bit;
sbit SoftSpi_SDI at RC0_bit;
sbit SoftSpi_SDO at RA5_bit;
sbit SoftSpi_CLK at RA4_bit;
sbit SoftSpi_SDI_Direction at TRISC0_bit;
sbit SoftSpi_SDO_Direction at TRISA5_bit;
sbit SoftSpi_CLK_Direction at TRISA4_bit;
//
void    init_dac()
{
        CS_Direction = 0;
        CS = 1;
        LDAC_Direction = 0;
        LDAC = 1;
        //
        Soft_SPI_Init();
        //
        dac_set('A', 0);
        dac_set('B', 0);
}
//*****************************************************************************
*
//■■■□□□電圧設定関数■■■
void    dac_set(char ch, WORD dt)
{
        BYTE    tmp;
        //
        if (ch == 'A') {
                dt = 0b0111000000000000 | dt;
        } else {
                dt = 0b1111000000000000 | dt;
        }
        //
        CS = 0;
        tmp = (dt >> 8) & 0x00FF;
        Soft_SPI_Write(tmp);
        tmp = dt & 0x00FF;
        Soft_SPI_Write(tmp);
        CS = 1;
        //
        LDAC = 0;
        LDAC = 1;
}
//*****************************************************************************
*
//■■■□□□初期化関数■■■
sbit LCD_RS at RC3_bit;
sbit LCD_EN at RC2_bit;
sbit LCD_D7 at RC7_bit;
```

```
sbit LCD_D6 at RC6_bit;
sbit LCD_D5 at RC5_bit;
sbit LCD_D4 at RC4_bit;
sbit LCD_RS_Direction at TRISC3_bit;
sbit LCD_EN_Direction at TRISC2_bit;
sbit LCD_D7_Direction at TRISC7_bit;
sbit LCD_D6_Direction at TRISC6_bit;
sbit LCD_D5_Direction at TRISC5_bit;
sbit LCD_D4_Direction at TRISC4_bit;
//
void    init_lcd()
{
        short   cnt;
        //
        Lcd_Init();
        RegistCustomChar();
        Lcd_Cmd(_LCD_CURSOR_OFF);
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1, 1, "Power Supply v2");
        for (cnt = 0; cnt <= 80; cnt++) {
                BarDisp(2, 1, cnt, 80);
                Delay_ms(10);
        }
        Lcd_Cmd(_LCD_CLEAR);
}
//**********************************************************************
*
//■■■キャラクタ登録関数■■■
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);
}
//*********************************************************************
*
//■■■バー表示関数■■■
void    BarDisp(char row, char column, short dat, short bar)
{
        short   j, k, cnt;
        //
        j = dat / 5;
        k = dat - (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 < (bar / 5); cnt++) {
                Lcd_Chr_Cp(' ');
        }
}
//*********************************************************************
*
```

動作確認





左側:**キーパッドで12.3Vをキー**入力**します。** 右側: “ * “ 入力**で、キー**入力値**を**設定値**にセットし、**設定電圧=出力電圧**になるように**制御**します。**