



DESCRIPTION:

This circuit converts any +12 VDC switching power supply into a lead acid battery charger.

SPECIFICATION:

It operates both in constant current, CC & constant voltage, CV mode to charge a lead acid battery utilizing already made switching power supply with front portion that monitors and also works in CC mode. CV mode is done inside power supply by raising it's voltage to +15.2 VDC. The maximum charging current is limited to 1.5 A.

CIRCUIT OPERATION:

First of all a +12 VDC switching power supply is modified by raising it's voltage level from +12 VDC to +15.2 VDC by changing the two resistors values built around reference IC TL431. After Diode D3 we get around +14.8 VDC. This serves as constant voltage, CV mode part of the charger. The constant current, CC mode part of the charger is built on this circuit and limits the maximum battery charging current to 1.5 A. The microcontroller monitors the battery voltage and blinks the red LED at a rate of one second, if voltage is lower than a threshold level. When battery voltage reaches certain level, in this case +14.5 V then it turns off the red LED and turns on green LED permanently and a buzzer which sounds on every alternate interval of one second. By the position of a switch we can choose to turn on or off the buzzer in conditions if battery is fully charged. The circuit is designed in such a way that it does not draw any current from battery and thus saves it from draining in case charger plug is not connected to mains 220 VAC socket.

SOURCE CODE:

```
// PIC12F683 Configuration Bit Settings  
// 'C' source line config statements
```

// CONFIG

#pragma config FOSC = INTOSCIO // Oscillator Selection bits (INTOSCIO oscillator: I/O function on RA4/OSC2/CLKOUT pin, I/O function on RA5/OSC1/CLKIN)

#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT disabled)

#pragma config PWRTE = OFF // Power-up Timer Enable bit (PWRT disabled)

#pragma config MCLRE = OFF // MCLR Pin Function Select bit (MCLR pin function is digital input, MCLR internally tied to VDD)

#pragma config CP = OFF // Code Protection bit (Program memory code protection is disabled)

#pragma config CPD = OFF // Data Code Protection bit (Data memory code protection is disabled)

#pragma config BOREN = OFF // Brown Out Detect (BOR disabled)

#pragma config IESO = OFF // Internal External Switchover bit (Internal External Switchover mode is disabled)

#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is disabled)

// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.

#include <xc.h>

#define _XTAL_FREQ 4000000

void init_micro (void);

void init_timer (void);

void init_adc (void);

void read_adc(void);

static int adc = 0, toggle_flag = 1, timer_flag = 0, adc_flag = 0;

static int counter_a = 0;

```
#define LED_CHARGING GP4 //RED LED
#define LED_CHARGED GP5      //GREEN LED
#define BUZZER GP2           //BUZZER
#define TRUE 1
#define FALSE 0
#define PB_SWITCH GP3        //BUZZER ON/OFF SWITCH
#define POWER_GOOD GP0       //ACTIVATES THE RESISTOR DIVIDER
```

```
void main (void)
{
    init_micro();
    init_adc();
    init_timer();
    do
    {
        if(adc_flag)
        {
            //    read_adc();
            if(adc < 820)          //BATTERY NOT FULL
                toggle_flag = TRUE;
            else if(adc >= 820)     //BATTERY FULL
                toggle_flag = FALSE;
            adc_flag = FALSE;
        }
        if(timer_flag)
```

```
{  
    counter_a++;  
    timer_flag = FALSE;  
}  
if(counter_a >= 10)  
{  
    counter_a = 0;  
    if(toggle_flag)  
    {  
        LED_CHARGING ^= 1;  
        LED_CHARGED = 0;  
        BUZZER = 0;  
    }  
    else if(!toggle_flag)  
    {  
        LED_CHARGING = 0;  
        LED_CHARGED = 1;  
        if(PB_SWITCH)  
            BUZZER ^= 1;  
        else if(!PB_SWITCH)  
            BUZZER = 0;  
    }  
}  
}  
while(TRUE);  
}
```

```
void init_micro (void)
{
    OSCCON = 0b01100111;
    TRISIO = 0b00001010;
    CMCON0 = CMCON0 | 0b00000100;
    ANSEL = 0b01010010;
    GPIO = 0b00000000;
    POWER_GOOD = TRUE;
    __delay_ms(100);
}
```

```
void interrupt team_isr()
{
    if(TMR1IF)
    {
        TMR1H = 0xCF;
        TMR1L = 0x2C;
        timer_flag = TRUE;
        TMR1IF = 0;
    }

    if(ADIF)
    {
        adc_flag = TRUE;
        read_adc();
    }
}
```

```
        ADIF = 0;
        GO_nDONE = 1;
    }
}
```

void init_timer (void)

```
{
    T1CON = 0b00110100;
    TMR1H = 0xCF;
    TMR1L = 0x2C;
    TMR1IE = 1;
    PEIE = 1;
    GIE = 1;
    TMR1ON = 1;
    TMR1IF = 0;
}
```

void init_adc (void)

```
{
    ADCON0 = 0b10000100;
    ADON = 1;
    ADIF = 0;
    ADIE = 1;
    PEIE = 1;
    GIE = 1;
    GO_nDONE = 1;
}
```

```
}
```

```
void read_adc (void)
```

```
{
```

```
    adc = (((unsigned)ADRESH << 8) | ADRESL);
```

```
}
```

High Tech Power