Semiconductors

AN10338

Off-line Li-Ion battery charger with P89LPC916

Rev. 01 — 19 Nov 2004 Application note

Document information

Info	Content
Keywords	Battery charger, MCU, P89LPC916
Abstract	This application note describes how to use Philips 8-bit P89LPC916 in designing an off-line Li-ion battery charger, with the characteristics of safe charging, time efficiency and small package. Using PWM and ADC, the P89LPC916 can accurately control different charge phases, allowing the battery to become fully charged and to prevent overcharging.

PHILIPS

Revision history

Rev	Date	Description
01	20041119	Initial version

Contact information

For additional information, please visit: http://www.semiconductors.philips.com

For sales office addresses, please send an email to: sales.addresses@www.semiconductors.philips.com

1. Charging principle

1.1 Introduction

Now that many portable electrical systems and products use rechargeable batteries as their power supply, the customer has many choices of charging methods, i.e., special power management ICs, MCU controlled, or even simple logic parts. When one considers safe charging, time-efficiency and low cost factors, the MCU controlled charging method can be used as a recharge solution within many application fields.

This solution is suitable for a Li-Ion battery rated 700 mAh to 750 mAh, discharged voltage 3.6 V, limited voltage 4.2 V.

1.2 Charging phase

For most chargers, the charging procedure is divided into three main phases:

- Pre-charging phase
- Constant-current-charging phase
- Constant-voltage-charging phase

In the Pre-charging phase, it utilizes low charging current to protect the battery. However, in most recharging situations, the charged batteries still have some voltage remaining, so the charger does not need to apply this action, and just passes it on to the next charging phase.

The constant current and voltage charging phases are the main charging phase for recharging the batteries, where most of the charging energy is transformed into the battery. The maximum charging current for a battery depends on the battery's rated capacity. For example, one battery rated 700 mAh usually can be charged at 350 mAh to 400mA, for faster charging effort.

When the charged battery is Li-Ion, in the last charging procedure, the microcontroller normally maintains regular charging voltage while monitoring charging current to determine when the charging procedure should be completed.

When the battery is fully charged, most of the electrical energy is converted to thermal energy, which can result in high battery temperature. In this case, the temperature monitoring function will be added for the solution. But since most Li-Ion batteries on the market also have an over-charging protection function, the temperature detect function is rarely used.

1.3 Buck converter

The buck charging is usually used in the constant current and voltage charge phase, and the most economical way to create a tapered termination charger is to use a buck converter. A buck converter is a switching regulator that uses an inductor as an energy storage device.

1.3.1 Buck converter operation

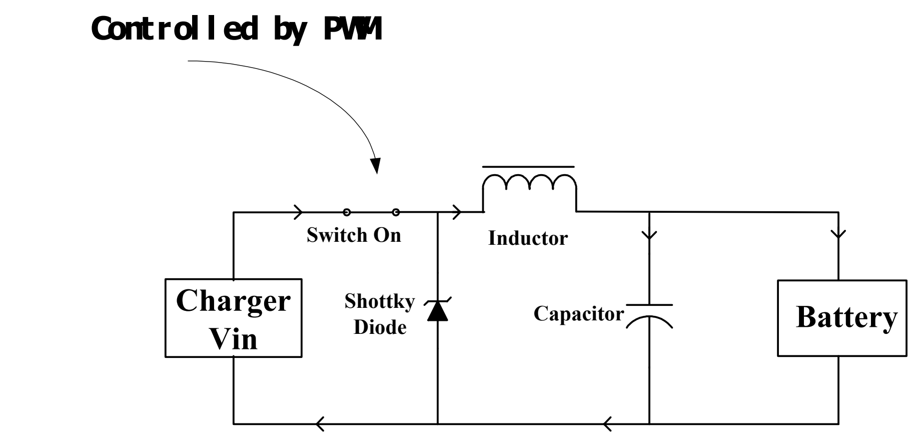


Fig 1. Buck converter switch on

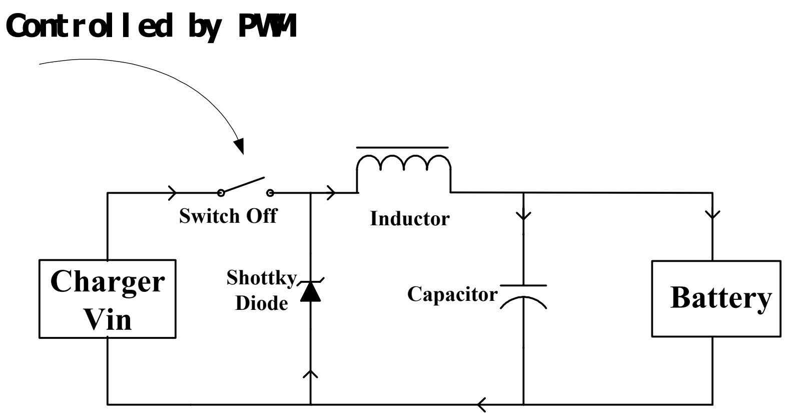


Fig 2. Buck converter switch off

The charging switch is controlled by PWM. When the switch is on, current will flow as shown in Fig 1. The capacitor is charged by the ‘Charger Vin’ through the inductor. When the switch is opened, as shown in Fig 2, the inductor will try to maintain its current flow by inducing a voltage, as the current through an inductor can’t change instantaneously. The current then flows through the diode and the inductor charges the capacitor, then the cycle repeats itself.

When we shorten switch ‘on’ time (the PWM duty cycle is decreased), the average voltage decreases and vice versa. Therefore, controlling the duty cycles allows us to regulate the charging voltage or the charging current to achieve desired output value.

1.3.2 Buck converter inductor selection

Converter inductor selection can be calculated by the following Equation:

$$L = \frac{(Vi - Vsat - Vo) * (T * DutyCycle)}{2Io} \quad (1)$$

• L:	converter inductor
• Vi:	charger voltage input to switch
• Vsat:	voltage loss on switch when switch is on
• Vo:	voltage output
• T:	the period of the PWM
• DutyCycle:	the DutyCycle of the PWM
• Io:	current output (the current for constant-current-charge)

As this Equation shows, the higher the PWM switching frequency (T smaller), the smaller the inductor, enabling lower cost.

Note that the capacitor in this circuit is simply a ripple reducer. In this case, larger is better, as ripple is inversely proportional to the value of this capacitor.

2. LPC916 charger solution

This solution uses Philips' P89LPC916 as the controller for an Off-line Li-Ion battery charger. It alternates constant current charging and constant voltage charging for fast charging, with an LED status indicator for monitoring the different work status.

2.1 P89LPC916 features

The P89LPC916 is a 16-pin microcontroller which has 2 kB of Flash memory. The Flash memory can be In-Circuit Programmed.

The P89LPC916 has 256 bytes RAM, with a 4-channel, 8-bit ADC. There are a total of four timers: two standard timers, one RTC timer, and one watchdog timer. Timer 0 has one channel which can be configured to generate PWM signals. Using an internal RC oscillator function, it's helpful for those applications (like a battery charger) that do not need high system frequency.

All of these features make the P89LPC916 suitable for small applications like a battery charger.

The P89LPC916 pin configuration is shown in Fig 3.

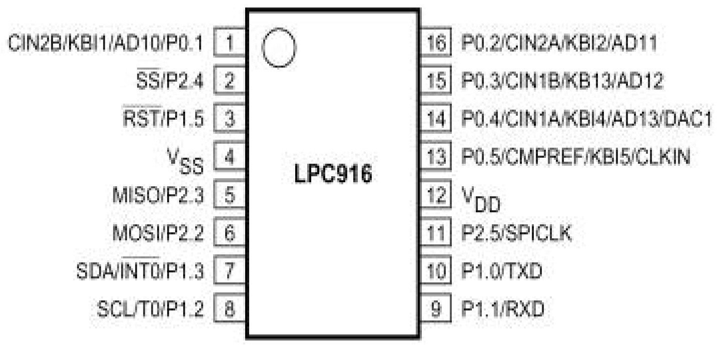


Fig 3. P89LPC916 pin configuration

2.2 Design key points for this solution

2.2.1 Precise power supply source for LPC916

Because the V_{DD} voltage for LPC916 is used as the reference voltage for the ADC, its precision is very important. A 3-terminal LDO LM1117, can provide exactly 3.31 V for the needed V_{DD} supply.

2.2.2 PWM output to control charging voltage

The LPC916 can provide PWM output as buck converter switch control, which will take advantage in charging voltage controlling. When using an on-chip RC oscillator, the PWM frequency can be set to about 14 kHz, then the buck converter switch ‘on’ time can be changed by adjusting the PWM duty cycle.

2.2.3 Apply ADC for battery charging voltage and current measurement

The LPC916 with 8-bit on chip high speed ADC can provide superior accuracy in monitoring charging voltage, which is critical to prevent overcharging in Li-Ion applications. It brings maximized effectiveness and longer battery life.

2.3 Selecting buck converter inductor

We can calculate the inductor’s value according to the following equation:

(2) $$L = \frac{(Vi - Vsat - Vo) * (T * DutyCycle)}{2Io}$$

Assuming Vi is 5.1 V, Vsat (at Io = 350 mA) is 0.5 V, the desired output voltage, Vo is 4.25 V, the desired output charge current, Io is 350 mA, 1/T is 14.7 kHz, Duty Cycle is 50%.

The inductor should be at least 10 uH, in this solution use 33 uH for buck converter inductor.

Note that if you want to use a higher input voltage, you must use a higher frequency PWM, or you must use a larger value inductor (at a greater cost), so a suitable input voltage is something that must be considered.

3. Specification

3.1 Input specifications

• Input voltage	DC 5.1 V
• Input voltage Scope	Min: 5.0 V Max: 5.5 V
• Input current	500 mA
• Input current Limit	Min: 400 mA
• Input ripple	Max: 50 mVpp

3.2 Output specifications

• Output voltage – Vout	
(At Constant-Voltage-Charging)	DC 4.23 ± 1%
• Output voltage Scope	Min: 0 V Max: 4.27 V
• Output current	
(At Constant-Current-Charging)	350 mA ± 10%
• Output current Scope	Min: 0 mA Max: 385 mA
• Output ripple	Max: 50 mVpp

4. Main Function

4.1 Charging method

- Prepare charging with small current (65 mA), meanwhile monitor the battery voltage
- Fast charging with constant current (350 mA), adjust control pulse to maintain stable current
- Fast charging with constant voltage (4.2 V), monitor the charging current

4.2 Charging current

- Pre-charge current: 65 mA
- Fast charging current: 350 mA
- End charging current: < 30 mA

4.3 Three charging phases

- Pre-charge phase, for empty battery voltage < 3 V
- Fast charging phase, constant current charge and constant voltage charge
- Timer control charging phase, last 50 minutes

4.4 Termination criteria

- Charging current detect
- Time control
- Battery temperature test (option)

4.5 LED indicators

- Low frequency red light flash indicator: Power on, charging battery
- Stable red light indicator: Charging completed
- High frequency red light flash indicator : Battery shortage, battery not on socket

4.6 Output shortage protect

If the output is being shorted, the controller will automatically identify this state, and shut down voltage output, with the LED indicator flashing at a higher frequency.

4.7 Battery out of socket indicator

At the start of charging, if the battery is not on the charge socket, then the charger will indicate this by flashing the LED at a higher frequency.

During the charging procedure, if the battery is accidentally unplugged from the charger socket, the charger will indicate this by flashing the LED at a higher frequency, until the battery is plugged back onto the charge socket. After 15 seconds, the charging procedure will be continued.

5. Charging Time

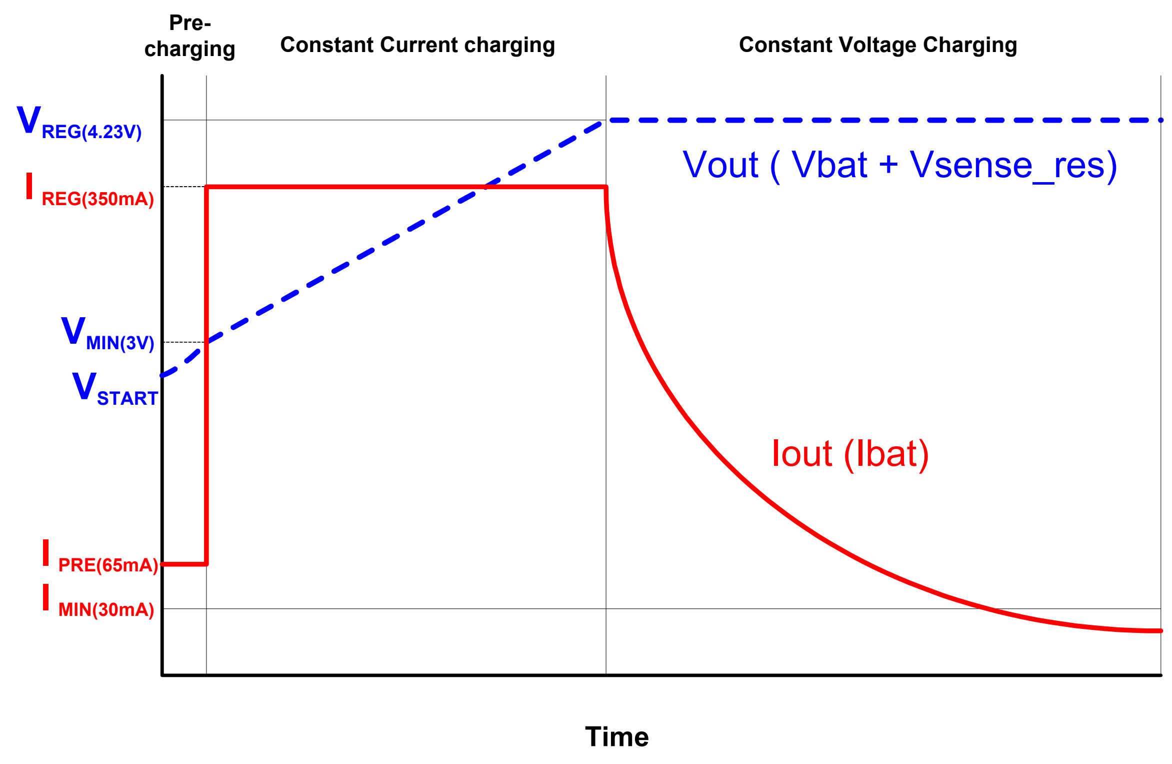


Fig 4. Three Charging Phases

Three charging phases converted criterion:

- Pre-charge phases (when needed)

  If Vbat < 3.0 ± 1% , set Iout = 10% Ireg = 65 mA
- Fast charging phases (Constant current charging)

  When Vbat <= 4.00 ± 1% V, set Iout = Ireg = 350 mA
- Fast charging phases (Constant voltage charging)

  When Vbat > 4.00 ± 1% V and Ibat >= 60 mA, set Vout = Vreg = 4.23 V
- Timer control charging phases (Constant voltage charging)

  When Ibat < 60 mA, set Vout = Vreg = 4.23 V for last 50 minutes to ensure that battery is fully charged (current is lower than 30 mA), then end charging procedure.

The whole charging process is expected to end within four hours.

6. Charging Test

6.1 Test Environments

• Temperature:	25°C
• Battery:	700mAh Li-ion Battery for Nokia 7250
	(Discharged Battery, remaining charge capacity is 10%)
• Power Supply:	Adaptor (output: 5.1 V max 2.0 A)
• Instrument:	FLUKE 187 TRUE RMS Multimeter
	VICTOR VC9806 Multimeter

6.2 Test Method

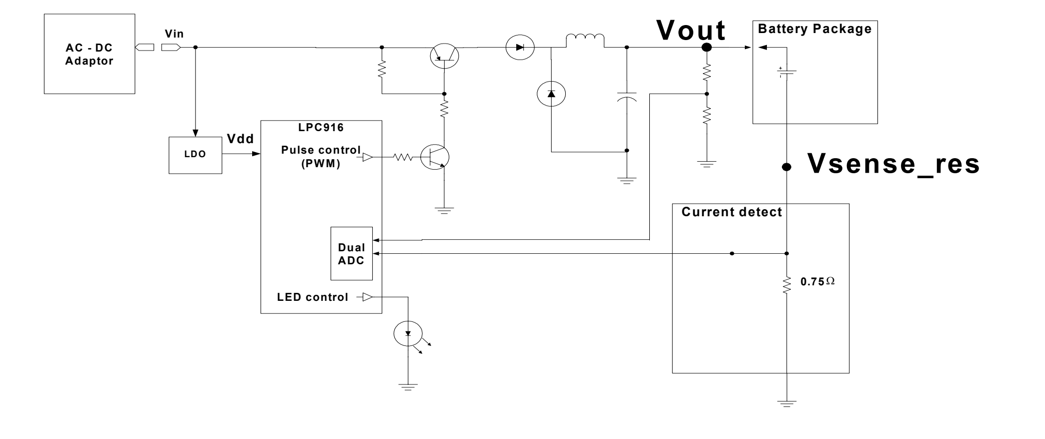


Fig 5. Function Block Diagram

Detected value:

Vout = Vbat + Vsense_res

Iout = Vsense_res /0.75

As shown with Fig 5, during the charging process, we watch the voltage at the test points of Vout and Vsense_res, using two multimeters to get the charging voltage, then calculate charging current by Vsense_res /0.75.

At the beginning of the charging procedure, we record the data every 15 seconds. When the charging current and voltage become stable, the test interval becomes longer (every 5 minutes).

6.3 Test results

• Constant current charging time:	79 minutes
• Constant current:	min: 336mA max: 353mA
• Constant current ending voltage:	Vout =4.25v, Vbat =4.02V
• End charging time:	3 hours 56 minutes (<4 hours)
• End charging current:	17mA (<30mA)
• End charging voltage:	Vout =4.22v, Vbat = 4.20V

These results may vary from battery to battery because of the variation of their physical characteristics. The original voltage of the battery also has an impact on the results. However, the specification is easily achieved. The results are shown in the following test diagrams.

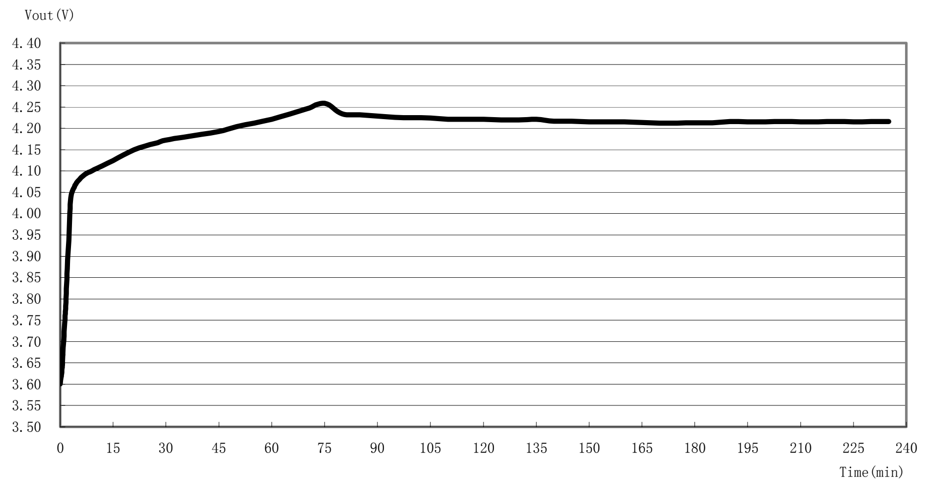


Fig 6. Charging voltage test waveform

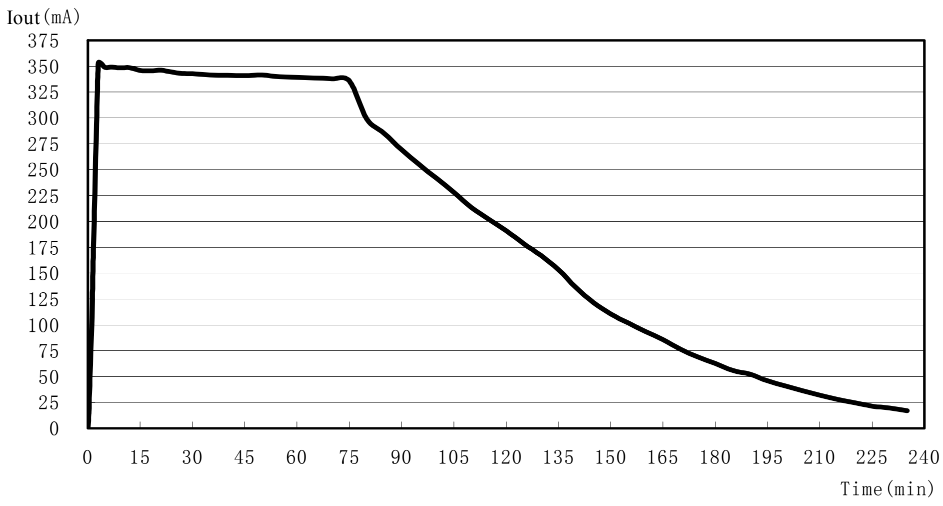


Fig 7. Charging current test waveform

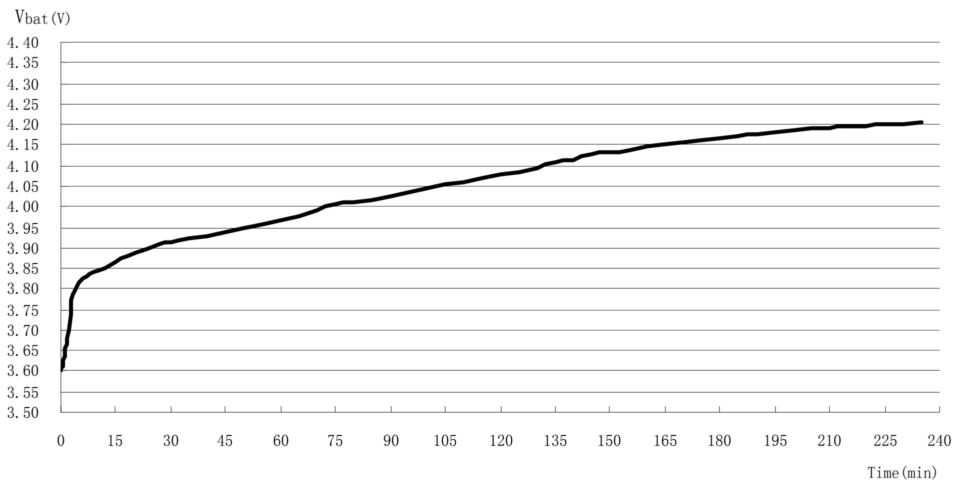
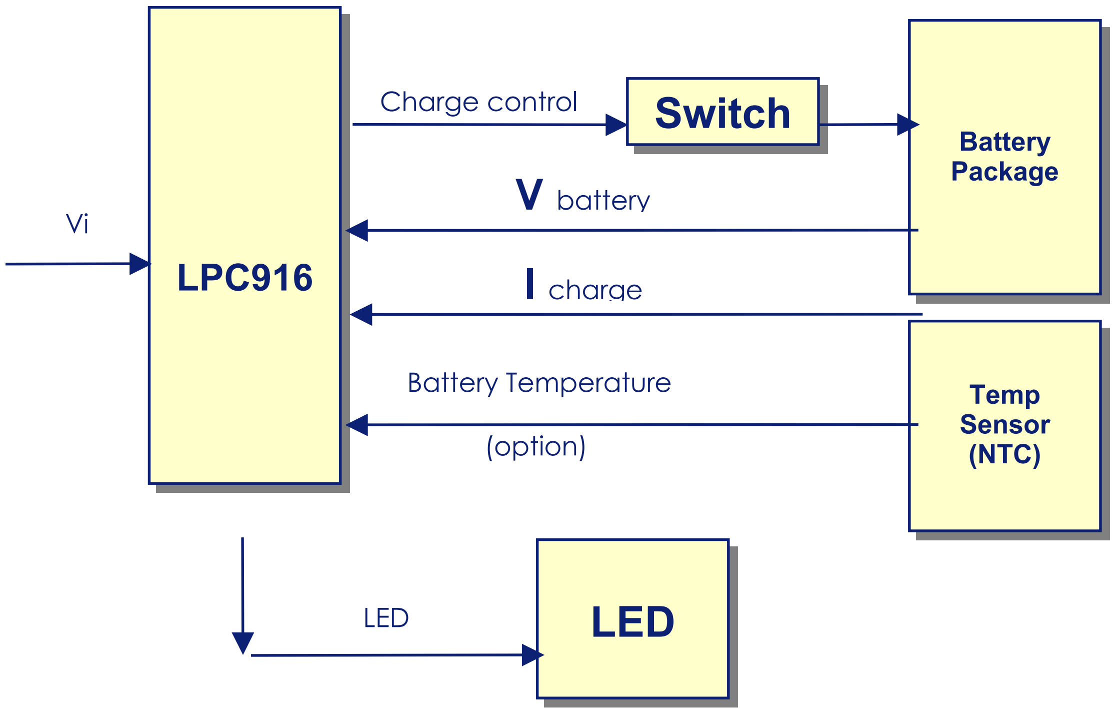
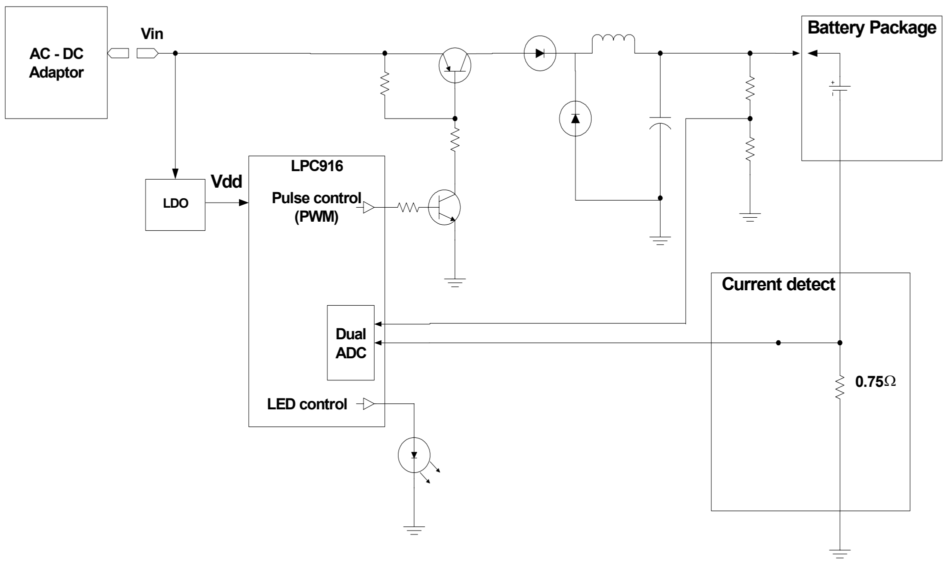


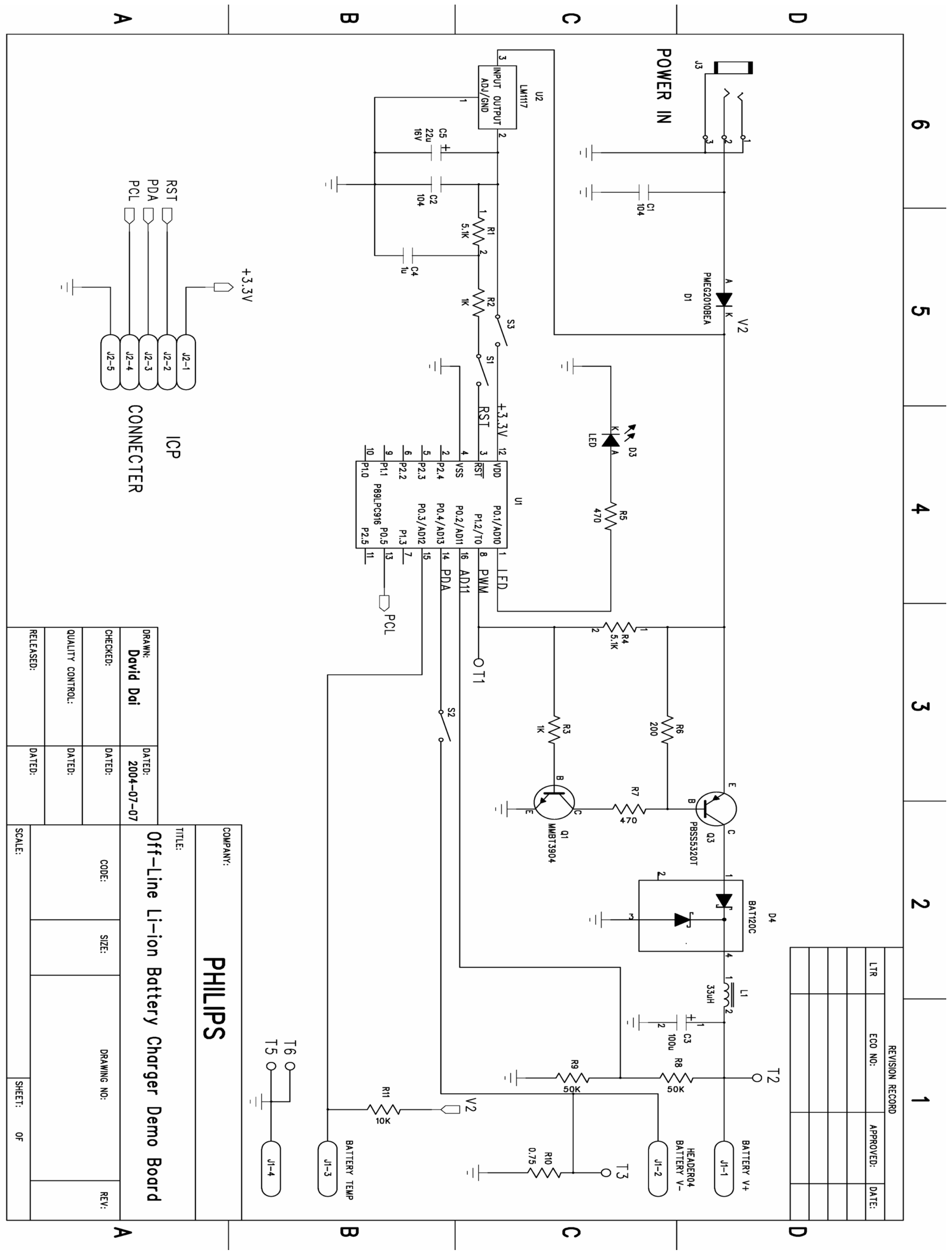
Fig 8. Battery voltage during charging

Appendix A Function Block Diagram

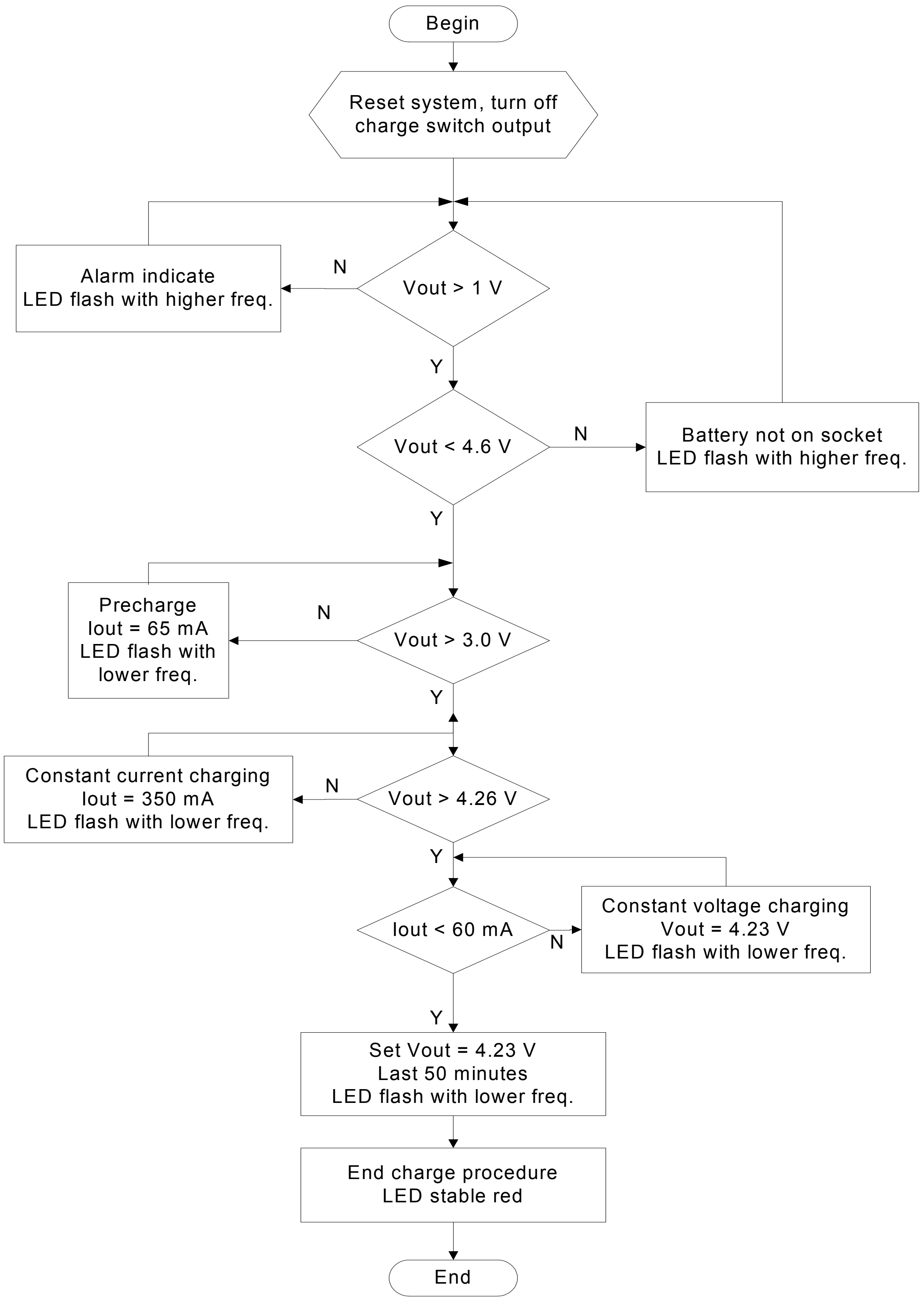




Appendix B Schematic



Appendix C Software Flowchart


Begin
Reset system, turn off charge switch output
Vout > 1 V
N
Alarm indicate LED flash with higher freq.
Y
Vout < 4.6 V
N
Battery not on socket LED flash with higher freq.
Y
Vout > 3.0 V
N
Precharge Iout = 65 mA LED flash with lower freq.
Y
Vout > 4.26 V
N
Constant current charging Iout = 350 mA LED flash with lower freq.
Y
Iout < 60 mA
N
Constant voltage charging Vout = 4.23 V LED flash with lower freq.
Y
Set Vout = 4.23 V Last 50 minutes LED flash with lower freq.
End charge procedure LED stable red
End

Appendix D Reference Software

```
//****************************************************************************
//*  Charger.c
//*  By : Anthony Xie
//*  Date : June 2004
//*  Description : Off-line Li-ion battery charger demo for P89LPC916
//*
//********************Revise History*****************************************
//* By : David Dai
//* Date : July 2004
//* Description : TO port the code into P89LPC916
//****************************************************************************
#include <reg916.h>                              // SFR definitions for LPC916

#define uchar unsigned char
#define uint unsigned int

//-------------------------------------------------
//Global Variable Definitions
//-----------------------------------------------
uchar data RTC_Counter_LED, LED_Flash_Speed;  //RTC count for LED delay
uint data RTC_Counter_AD;                      //RTC count for AD convert
uint data AD_Convert_Speed;                    //AD convert interval
uint data RTC_Counter;                         //RTC count for delay
uchar data RTC_Counter_Minute;                 // minutes counter for dalay
bit data LED_Flag, AD1_Flag, AD_Flag, Battery_On_Socket;
bit data Pre_Charge_Status, Constant_Current_Charge, Constant_Voltage_Charge;
uint data AD1_1,AD1_2,AD1_3;                   //AD1.1-Battery V+, AD1.2-temp., AD1.3-I detect
uchar data average;                            // AD average times

//-------------------------------------------------
//Constant Definitions
//-------------------------------------------------
#define LED_FLASH_FAST 10                      // define led flash with high fre.
#define LED_FLASH_SLOW 200                     // define led flash with low fre.
#define STOP_CHARGE_DELAY 50                   // define delay time when charge I <60mA, x Minutes

//****************************************************************************
//* Functions
//****************************************************************************
void main(void);                               // main loop
void init(void);                               // part initialization
void ad_convert(void);

//****************************************************************************
//* main()
//* Input(s) : none.
//* Returns : none.
//* Description : main loop
//****************************************************************************
void main ()
{

  EA = 0;
  init();
  EA = 1;

  for( RTC_Counter = 1; RTC_Counter < 1700; );
  RTC_Counter = 0;

  for(;;)
     {
    if(LED_Flag == 0)                   // control LED flash
     {
       KB1 = 0;
     }
    else
     {
       KB1 = 1;
     }
```

```
        if( ( AD_Flag == 1 ) && ( Battery_On_Socket == 1 ))
                                        // start AD convert and PWM control, with pre-set AD convert speed
          {                             // AD_Flag control AD convert speed
           ad_convert();
           AD_Flag = 0;
          }

        if( RTC_Counter_Minute == STOP_CHARGE_DELAY ) // If delay ended, stop charge
          {
           EA = 0;                      //shut down interrupt, so AD_Flag can not be set
           P1 = 0;                      //Shut down PWM
           LED_Flag = 1;                //LED stable RED
          }
      }
    }
//****************************************************************************
//* init()
//* Input(s) : none.
//* Returns : none.
//* Description : initialization of P89LPC916
//****************************************************************************
void init(void)
{
//SP = 0x30;          //Set SP from 50H to 7FH

//------------------------------------------------
//Config P0 I/O
//------------------------------------------------

  P0M1 = 0x1C;        // Set P0.2/3/4 for Input Only,       0001 1100
  P0M2 = 0xE3;        // Set P0.1 for PP, ^1 for led drive, 1110 0011, others all pp

//------------------------------------------------
//Config P1 I/O
//------------------------------------------------

  P1M1 = 0x24;        // ^5 Input only            0010 0100
  P1M2 = 0xdf;        // set ^2 open drain        1101 1111  others all pp

//------------------------------------------------
//Config P2 I/O (P2 not used)
//------------------------------------------------
  P2M1 = 0x00;        // All PP
  P2M2 = 0xFF;

  KB5 = 0;            //set p0.7 = 0 can forfid T1 PWM
  P1 = 0x00;          //set p1.2 = 0 can forbid TO PWM
  P2 = 0x00;

  RTC_Counter = 0x00;
  RTC_Counter_LED = 0x00;
  LED_Flag = 0;
  LED_Flash_Speed = LED_FLASH_SLOW;
  AD1_1 = 0;
  AD1_2 = 0;
  AD1_3 = 0;
  AD1_Flag = 0;

  RTC_Counter_LED= 0;
  AD_Flag = 0;
  average = 10;
  Pre_Charge_Status = 0;
  Constant_Current_Charge = 0;
  Constant_Voltage_Charge = 0;

  RTC_Counter_Minute = 0;
  Battery_On_Socket = 1;
  AD_Convert_Speed = 10;
```

```
//-----------------------------------------------
//Initialize RTC for time delay counter
//-----------------------------------------------
  WDCON &=0xE0;         //Shut down WDT
  RTCH = 0x01;          //Set 16bit counter for RTC, about 8.9ms
  RTCL = 0xFF;
  RTCCON = 0x63;        //RTC Enable, CCLK use Internal RC oscillator
  EWDRT = 1;            //Enable RTC/WD Interrupt

//-----------------------------------------------
//Initialize T0 for PWM
//-----------------------------------------------

  TMOD |= 0x02;         //Set T0 mode 6
  TAMOD |= 0x01;        //For PWM, 0x01
  TH0 = 255;            //Duty cycle = 256 - TH0, for 5v input, set initial charge current very small
  AUXR1 |= 0x10;        //Enable toggling pin P1.2
  TR0 = 1;              //Enable T0

//--------------------------------------------------------------------
//Enable the A/D Converter and use the CPU clock as the A/D clock.
//--------------------------------------------------------------------
  ADMODA = 0x10;        //AD1 single mode no boundary interrupt
  ADMODB = 0x40;        //AD "40" for CCLK/3, 7.3728/3=2.4576M. "60" for CLK = CCLK/4, 1.8432M, ADC MODE
  ADINS = 0xE0;         //Enable AD11,AD12,AD13
  ADCON1 = 0x04;        //Enable AD1 with timer no start occurs mode, no completed & boundary interrupt


}

//***************************************************************************
//* ad_convert()
//* Input(s) :
//* Returns :
//* Description : ad_convert & control action
//***************************************************************************
void ad_convert ()
{
  uchar temp = 0;

  ADCON1 |= 0x01;                //Enable AD convert

  while( !(ADCON1 & 0x08) );     //waiting for AD to end

  ADCON1 &= ~0x08;

  if( average-- == 0)            // every x(10) times AD output one result, first judge, second --
    {
   AD1_1 = AD1_1/10;
   AD1_2 = AD1_2/10;
   AD1_3 = AD1_3/10;
   average = 10;


//-----------------------------------------------
//AD11 for battery voltage detect
//-----------------------------------------------
   if( AD1_1 > 176 )                     // If Vout > 4.6v
    {                                    // Prevent battery plug out during charging
     SCL=0;
     Battery_On_Socket = 0;              // forbid AD convert, skip <115, <162, <176, until <38
        RTC_Counter = 5000;              //Delay 15's, then enable AD convert in RTC_Interrupt()
     LED_Flash_Speed = LED_FLASH_FAST;   // indicate error status-battery not in socket
    }
   else if( AD1_1 <= 38 )                // If Vout <= 1V, <=(1/3.31*2)*256 = 256/6.66
    {
     SCL=0;                              //Set p1.2 = 0, forbid PWM, shut down output
     LED_Flash_Speed = LED_FLASH_FAST;   // indicate error status
     AD1_1 = 0;                          // digital LED display "000"
```

```
      }                                 // battery shortage or battery not in socket
     else if( AD1_1 <= 115)             // If Vout<= 3V, 3 x 256/6.66
       {
        SCL=1;                          // Enable P1.2 PWN
        TH0 = 250;                      // preset charge current, for 5v input, about 30-65mA
        LED_Flash_Speed = LED_FLASH_SLOW;                    // indicate charging status
        if( Constant_Current_Charge == 0 ) Pre_Charge_Status = 1; // Set Precharge status
      }
     else if( AD1_1 < 162)      // detect Vout = battery voltage + 0.35*0.75
                                // for fullly charged bat,bat vol=4.24,I limit 20mA, 4.24+0.02*0.75=4.26v
                                // for protect battery, set Vout = 4.26v (162)
       {                        // if bat vol=4.2, I will be limited 80mA, 4.2+0.08*0.75=4.26v
                                // if bat vol=4.15, I will be limited 150mA, 4.15 + 0.15*0.75 = 4.26v
                                // when cons-current-charging,4.26-0.35*0.75=4.00v, so Vbat>4.00v,enter cons-vol
                                // when cons-vol-charging, charge I= min 20mA,Vbat= 4.26-0.02*0.75=4.24v
                                // ( 161-4.252, 162-4.267, 160-4.234 )
        SCL=1;
        LED_Flash_Speed = LED_FLASH_SLOW;   // indicate charging status
        if( Constant_Voltage_Charge == 0)
         {
            Constant_Current_Charge = 1;    // Set Constant Current Charge status
            Pre_Charge_Status = 0;          // forbid Precharge status

         }

       }
     else                                   // If  4.26V < Vout <4.6V
       {
        SCL=1;
        LED_Flash_Speed = LED_FLASH_SLOW;   // indicate charging status
        Constant_Voltage_Charge = 1;        // Set Constant Voltage Charge status
        Constant_Current_Charge = 0;        // forbid Constant Current Charge status
        Pre_Charge_Status = 0;              // forbid Precharge status

       }

     temp = AD1_1;

     AD1_1 = 0;                             // reset AD1_1

 //-----------------------------------------------
 //AD12 for temperature detect
 //-----------------------------------------------
 //     AD1_2 = AD1_2*83/64;                // input x 3.31 x 100 / 256
 //     First_Bit = AD1_2/100;
 //     AD1_2 = AD1_2%100;
 //     Second_Bit = AD1_2/10;
 //     Third_Bit = AD1_2%10;
 //     DP_Enable = 0;
     AD1_2 = 0;                             // reset AD1_2

 //-----------------------------------------------
 //AD13 for charge current detect
 //-----------------------------------------------
     if( Pre_Charge_Status == 1 )
       {
        if( ( AD1_3 > 1 ) && ( TH0 < 250 ) )   // ensure Pre-charger current < 65mA
          { TH0++;}
       }
     else if( Constant_Current_Charge == 1 )
       {
        if( AD1_3 < 17 )                       //  260mv,(I charge = 350mA)
          {             // if I charge < 350mA, pwm++
            if( TH0 < 2 )
         {
           TH0 = 1;
           if( RTC_Counter == 0 )              //start 20 minutes delay to end charge
             { RTC_Counter = 1; }              //this case happened for charged battery,if Vout < 2.6v
                                               //while Ibat can't achieve 350mA, even max PWM
         }
           else
```

```
            { TH0 = TH0 - 1; }
          }
        else if( AD1_3 > 17)
          {              //if I charge > 350mA, pwm--
                 if( TH0 > 253 )
           { TH0 = 254; }
         else
           {
             TH0 = TH0 + 1;
           }
          }
        }
      else if( Constant_Voltage_Charge == 1 )
        {
        if( AD1_3 <= 1 )                         // Charge current is < 60mA
          {
            if( RTC_Counter == 0 ) { RTC_Counter = 1; } //start 50 minutes delay to end charge
          }
        if( temp > 160 )                         //temp=AD1_1, if Vout is > 4.23v(achieve 4.23v), -- PWM
            {
            if( TH0 < 255)
              {
           TH0++;
         }
           }
         else if( temp < 160 )                   // if Vout is < 4.21v (achieve 4.21v), ++PWM
             {
             if( TH0 > 1 )
         {
             TH0--;
               }
           }
        }
     AD1_3 = 0;                                  // reset AD1_3
   }
   else
   {
      AD1_1 += AD1DAT1;
      AD1_2 += AD1DAT2;
      AD1_3 += AD1DAT3;

   }
 }
//****************************************************************************
//* RTC_Interrupt
//* Input(s) : none.
//* Returns : none.
//* Description : Interrupt from RTC
//****************************************************************************
void RTC_Interrupt (void) interrupt 10 using 1
{
   if( RTC_Counter != 0 ) { RTC_Counter++; }
   RTC_Counter_LED++;
   RTC_Counter_AD++;

   if ( RTC_Counter_LED == LED_Flash_Speed)
      {
     LED_Flag = ~LED_Flag;
     RTC_Counter_LED = 0;
   }

   if ( RTC_Counter_AD == AD_Convert_Speed )
      {
     AD_Flag = 1;
     RTC_Counter_AD = 0;
   }

   if ( RTC_Counter == 6750)          // delay for 1 minute
   {
     RTC_Counter_Minute++;
     RTC_Counter = 1;
```

```
      if( Battery_On_Socket == 0)     // complete 15's delay for battery not in socket
        {
         Battery_On_Socket = 1;
         RTC_Counter = 0;
        }
   }

   RTCCON = 0x63;                     //Clear RTCCON.7-RTCF
}
```

```
/*------------------------------------------------------------------------------
REG916.H

Header file for Philips 89LPC916
------------------------------------------------------------------------------*/

#ifndef __REG916_H__
#define __REG916_H__

/*  BYTE Registers  */
sfr P0      = 0x80;
sfr P0M1    = 0x84;
sfr P0M2    = 0x85;

sfr P1      = 0x90;
sfr P1M1    = 0x91;
sfr P1M2    = 0x92;

sfr P2      = 0xA0;
sfr P2M1    = 0xA4;
sfr P2M2    = 0xA5;
//-------------------
sfr PSW     = 0xD0;
sfr ACC     = 0xE0;
sfr B       = 0xF0;
sfr SP      = 0x81;
sfr DPL     = 0x82;
sfr DPH     = 0x83;
//-------------------
sfr ADCON1  = 0x97;
sfr ADINS   = 0xA3;
sfr ADMODA  = 0xC0;
sfr ADMODB  = 0xA1;
sfr AD1BH   = 0xC4;
sfr AD1BL   = 0xBC;
sfr AD1DAT0= 0xD5;
sfr AD1DAT1= 0xD6;
sfr AD1DAT2= 0xD7;
sfr AD1DAT3= 0xF5;

sfr AUXR1   = 0xA2;
sfr BRGR0  = 0xBE;
sfr BRGR1  = 0xBF;
sfr BRGCON = 0xBD;
sfr CMP1   = 0xAC;
sfr CMP2   = 0xAD;
sfr DIVM   = 0x95;

sfr FMADRH = 0xE7;
sfr FMADRL = 0xE6;
sfr FMCON  = 0xE4;
sfr FMDATA = 0xE5;

sfr I2ADR  = 0xDB;
sfr I2CON  = 0xD8;
sfr I2DAT  = 0xDA;
sfr I2SCLH = 0xDD;
sfr I2SCLL = 0xDC;
sfr I2STAT = 0xD9;

sfr IEN0   = 0xA8;
sfr IEN1   = 0xE8;

sfr IP0    = 0xB8;
sfr IP0H   = 0xB7;
sfr IP1    = 0xF8;
sfr IP1H   = 0xF7;

sfr KBCON  = 0x94;
sfr KBMASK = 0x86;
sfr KBPATN = 0x93;

sfr PCON   = 0x87;
sfr PCONA  = 0xB5;

sfr PT0AD  = 0xF6;
sfr RSTSRC = 0xDF;

sfr RTCCON = 0xD1;
sfr RTCH   = 0xD2;
sfr RTCL   = 0xD3;

sfr SADDR  = 0xA9;
sfr SADEN  = 0xB9;
sfr SBUF   = 0x99;
sfr SCON   = 0x98;
sfr SSTAT  = 0xBA;

sfr SPCTL  = 0xE2;
sfr SPSTAT = 0xE1;
sfr SPDAT  = 0xE3;

sfr TAMOD  = 0x8F;
sfr TCON   = 0x88;
sfr TL0    = 0x8A;
sfr TL1    = 0x8B;
sfr TH0    = 0x8C;
sfr TH1    = 0x8D;
sfr TMOD   = 0x89;
sfr TRIM   = 0x96;

sfr WDCON  = 0xA7;
sfr WDL    = 0xC1;
sfr WFEED1 = 0xC2;
sfr WFEED2 = 0xC3;

/*  BIT Registers  */
/*  PSW   */
sbit CY   = PSW^7;
sbit AC   = PSW^6;
sbit F0   = PSW^5;
sbit RS1  = PSW^4;
sbit RS0  = PSW^3;
sbit OV   = PSW^2;
sbit F1   = PSW^1;
sbit P    = PSW^0;

/*  TCON  */
sbit TF1    = TCON^7;
sbit TR1    = TCON^6;
sbit TF0    = TCON^5;
sbit TR0    = TCON^4;
sbit IE0    = TCON^1;
sbit IT0    = TCON^0;

/* ADMODA */
sbit BNDI1  = ADMODA^7;
sbit BURST1 = ADMODA^6;
sbit SCC1   = ADMODA^5;
sbit SCAN1  = ADMODA^4;
sbit BNDI0  = ADMODA^3;
sbit BURST0 = ADMODA^2;
sbit SCC0   = ADMODA^1;
sbit SCAN0  = ADMODA^0;

/*  IEN0   */
sbit EA     = IEN0^7;
sbit EWDRT  = IEN0^6;
sbit EBO    = IEN0^5;
sbit ES     = IEN0^4;
sbit ESR    = IEN0^4;
sbit ET1    = IEN0^3;
sbit ET0    = IEN0^1;
sbit EX0    = IEN0^0;

/*  IEN1   */
sbit EAD    = IEN1^7;
sbit EST    = IEN1^6;
sbit ESPI   = IEN1^3;
sbit EC     = IEN1^2;
sbit EKBI   = IEN1^1;
sbit EI2C   = IEN1^0;
```

```
/*  IP0   */
sbit PWDRT  = IP0^6;
sbit PB0    = IP0^5;
sbit PS     = IP0^4;
sbit PSR    = IP0^4;
sbit PT1    = IP0^3;
sbit PT0    = IP0^1;
sbit PX0    = IP0^0;

/*  P0  */
sbit KB5     = P0^5;
sbit CMPREF  = P0^5;
sbit KB4     = P0^4;
sbit CIN1A   = P0^4;
sbit KB3     = P0^3;
sbit CIN1B   = P0^3;
sbit KB2     = P0^2;
sbit CIN2A   = P0^2;
sbit KB1     = P0^1;
sbit CIN2B   = P0^1;

/*  P1  */
sbit RST     = P1^5;
sbit INT0    = P1^3;
sbit SDA     = P1^3;
sbit T0      = P1^2;
sbit SCL     = P1^2;
sbit RxD     = P1^1;
sbit TxD     = P1^0;

/*  P2  */
sbit SPICLK  = P2^5;
sbit SS      = P2^4;
sbit MISO    = P2^3;
sbit MOSI    = P2^2;


/*  SCON  */
sbit SM0     = SCON^7;
sbit FE      = SCON^7;
sbit SM1     = SCON^6;
sbit SM2     = SCON^5;
sbit REN     = SCON^4;
sbit TB8     = SCON^3;
sbit RB8     = SCON^2;
sbit TI      = SCON^1;
sbit RI      = SCON^0;
#endif
```

7. Disclaimers

8. Contents

Date of release:19 Nov 2004
Document number: 9397 750 14287

Published in U.S.A.