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.
Revision history

<table>
<thead>
<tr>
<th>Rev</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>20041119</td>
<td>Initial version</td>
</tr>
</tbody>
</table>

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

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:

\[
L = \frac{(V_i - V_{sat} - V_o) \times (T \times \text{DutyCycle})}{2I_o}
\]

- \(L\): converter inductor
- \(V_i\): charger voltage input to switch
- \(V_{sat}\): voltage loss on switch when switch is on
- \(V_o\): voltage output
- \(T\): the period of the PWM
- \(\text{DutyCycle}\): the DutyCycle of the PWM
- \(I_o\): 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.
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:

$$L = \frac{(V_i - V_{sat} - V_o) \times (T \times \text{DutyCycle})}{2I_o}$$

Assuming $V_i$ is 5.1 V, $V_{sat}$ (at $I_o = 350$ mA) is 0.5 V, the desired output voltage, $V_o$ is 4.25 V, the desired output charge current, $I_o$ is 350 mA, $T/I$ 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: 350 mA ± 10%
  (At Constant-Current-Charging)
- 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

Three charging phases converted criterion:

- Pre-charge phases (when needed)
  - If $V_{bat} < 3.0 \pm 1\%$, set $I_{out} = 10\% \times I_{reg} = 65 \text{ mA}$

- Fast charging phases (Constant current charging)
  - When $V_{bat} \leq 4.00 \pm 1\% \text{ V}$, set $I_{out} = I_{reg} = 350 \text{ mA}$

- Fast charging phases (Constant voltage charging)
  - When $V_{bat} > 4.00 \pm 1\% \text{ V}$ and $I_{bat} \geq 60 \text{ mA}$, set $V_{out} = V_{reg} = 4.23 \text{ V}$

- Timer control charging phases (Constant voltage charging)
  - When $I_{bat} < 60 \text{ mA}$, set $V_{out} = V_{reg} = 4.23 \text{ V}$ 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

![Function Block Diagram]

Detected value:

\[ V_{out} = V_{bat} + V_{sense\_res} \]
\[ I_{out} = V_{sense\_res} / 0.75 \]

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

LPC916

Switch

Battery Package

Temp Sensor (NTC)

Battery Temperature (option)

Charge control

Vi

V battery

I charge

LED

LPC916

AC - DC Adaptor

Vdd

LED control

Dual ADC

Pulse control (PWM)

LDO

Current detect

0.75Ω

Battery Package

© Koninklijke Philips Electronics N.V. 2004. All rights reserved.
Appendix C  Software Flowchart

Begin

Reset system, turn off charge switch output

Alarm indicate LED flash with higher freq.

Vout > 1 V

Vout < 4.6 V

Precharge Iout = 65 mA LED flash with lower freq.

Vout > 3.0 V

Constant current charging Iout = 350 mA LED flash with lower freq.

Vout > 4.26 V

Iout < 60 mA

Constant voltage charging Vout = 4.23 V LED flash with lower freq.

Set Vout = 4.23 V Last 50 minutes LED flash with lower freq.

End charge procedure LED stable red

End

Battery not on socket LED flash with higher freq.
Appendix D  Reference Software

//***************************************************************************
//* Charger.c
//* By : Anthony Xie
//* Date : June 2004
//* Description : Off-line Li-ion battery charger demo for P89LPC916
//* Revisit History: Off-line Li-ion battery charger demo for P89LPC916
//* 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

(QIcon Variable Definitions
(QIcon Global Variable Definitions

uchar data RTC_Counter_LED, LED_Flash_Speed;  //RTC count for LED delay
uint data RTC_Counter_AD;        //RT C 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 delay
bit data LED Flag, AD1 Flag, AD_Flag, Battery On_Socket;
bit data Pre Charge Status, Constant Current Charge, Constant Voltage Charge;
uchar data AD1, AD1_2, AD1_3;     //AD1.1-Battery V+, AD1.2-temp., AD1.3-I detect
uchar data average;         // AD average times

(QIcon Constant Definitions
(QObject 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
    P0M2 = 0xE3;    // Set P0.1 for PP, ^1 for LED drive

    //-----------------------------------------------
    // Config P1 I/O
    //-----------------------------------------------
    P1M1 = 0x24;    // ^5 Input only
    P1M2 = 0xdf;    // set ^2 open drain

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

    KB5 = 0;    // set p0.7 = 0 can forbid 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 CCLK = CCLK/4, 1.8432M, ADC MODE
ADINS = 0xE0;   //Enable AD1, AD12, AD13
ADCON1 = 0x04;   //Enable AD1 with timer no start occurs mode, no completed & boundary interrupt

ad_convert();
//------------------------------------------------------------------------------
//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 --
    {
        ADI_1 = ADI_1/10;
        ADI_2 = ADI_2/10;
        ADI_3 = ADI_3/10;
        average = 10;
    }
    //AD11 for battery voltage detect
    if( ADI_1 > 176 )      // If Vout > 4.6v
    {
        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( ADI_1 <= 38 )      // If Vout <= 4V, <=(1/3.31*2)*256 = 256/6.66
    {
        SCL=0;
        LED_Flash_Speed = LED_FLASH_FAST; // indicate error status
        ADI_1 = 0;      //digital LED display "000"
    }
}
else if( AD1_1 <= 115)       // battery shortage or battery not in socket
{
    SCL=1;          // Enable P1.2 PWM
    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 < 4.2V, 3 x 256/6.66
{
    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 4.2V < 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
//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( 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
            }
        }
    }
}
{ 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 ) {  // Charge current is < 60mA
    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 (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 = 0xf3; //Clear RTCON.7-RTCF

/*--------------------------------------------------------------------------
REG916.H
Header file for Philips 89LPC916
 *--------------------------------------------------------------------------*/
#endif __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 FMADDRH = 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 IENO    = 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 PTOAD   = 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 TLO     = 0xA8;
sfr TLL     = 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 P0   = PSW^5;
sbit RS1  = PSW^4;
sbit RS0  = PSW^3;
sbit OV   = PSW^2;
sbit P1   = 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   = IEN0^3;
sbit ET1   = IEN0^1;
sbit EX0   = IEN0^0;

/* IEN1 */
sbit EAD   = IEN1^7;
sbit EST   = IEN1^6;
sbit ESP1  = 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 PE      = 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

**Life support** — These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips Semiconductors customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors for any damages resulting from such application.

**Right to make changes** — Philips Semiconductors reserves the right to make changes in the products - including circuits, standard cells, and/or software - described or contained herein in order to improve design and/or performance. When the product is in full production (status 'Production'), relevant changes will be communicated via a Customer Product/Process Change Notification (CPCN). Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no licence or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

**Application information** — Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.
# 8. Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Charging principle ........................................... 3</td>
</tr>
<tr>
<td>1.1</td>
<td>Introduction .................................................. 3</td>
</tr>
<tr>
<td>1.2</td>
<td>Charging phase ................................................ 3</td>
</tr>
<tr>
<td>1.3</td>
<td>Buck converter ................................................ 3</td>
</tr>
<tr>
<td>1.3.1</td>
<td>Buck converter operation ..................................... 4</td>
</tr>
<tr>
<td>1.3.2</td>
<td>Buck converter inductor selection .............................. 5</td>
</tr>
<tr>
<td>2.</td>
<td>LPC916 charger solution ....................................... 5</td>
</tr>
<tr>
<td>2.1</td>
<td>P89LPC916 features ............................................ 5</td>
</tr>
<tr>
<td>2.2</td>
<td>Design key points for this solution ............................ 6</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Precise power supply source for LPC916 ....................... 6</td>
</tr>
<tr>
<td>2.2.2</td>
<td>PWM output to control charging voltage ....................... 6</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Apply ADC for battery charging voltage and current measurement ........................................... 6</td>
</tr>
<tr>
<td>2.3</td>
<td>Selecting buck converter inductor .............................. 6</td>
</tr>
<tr>
<td>3.</td>
<td>Specification ..................................................... 7</td>
</tr>
<tr>
<td>3.1</td>
<td>Input specifications ............................................. 7</td>
</tr>
<tr>
<td>3.2</td>
<td>Output specifications ........................................... 7</td>
</tr>
<tr>
<td>4.</td>
<td>Main Function .................................................... 8</td>
</tr>
<tr>
<td>4.1</td>
<td>Charging method ................................................ 8</td>
</tr>
<tr>
<td>4.2</td>
<td>Charging current ................................................ 8</td>
</tr>
<tr>
<td>4.3</td>
<td>Three charging phases ......................................... 8</td>
</tr>
<tr>
<td>4.4</td>
<td>Termination criteria ........................................... 8</td>
</tr>
<tr>
<td>4.5</td>
<td>LED indicators .................................................. 8</td>
</tr>
<tr>
<td>4.6</td>
<td>Output shortage protect ........................................ 8</td>
</tr>
<tr>
<td>4.7</td>
<td>Battery out of socket indicator ................................. 8</td>
</tr>
<tr>
<td>5.</td>
<td>Charging Time ..................................................... 9</td>
</tr>
<tr>
<td>6.</td>
<td>Charging Test ..................................................... 10</td>
</tr>
<tr>
<td>6.1</td>
<td>Test Environments ............................................... 10</td>
</tr>
<tr>
<td>6.2</td>
<td>Test Method ....................................................... 10</td>
</tr>
<tr>
<td>6.3</td>
<td>Test results ...................................................... 10</td>
</tr>
<tr>
<td>7.</td>
<td>Disclaimers ....................................................... 25</td>
</tr>
<tr>
<td>8.</td>
<td>Contents .......................................................... 26</td>
</tr>
</tbody>
</table>

---

Appendix A Function Block Diagram ........................................... 13
Appendix B Schematic ...................................................... 14
Appendix C Software Flowchart ............................................. 15
Appendix D Reference Software ............................................. 16
Appendix E Reference Software ............................................. 16