APPLICATION NOTES For

Ni-MH BATTERY CHARGER

S3F94xx-SERIES

MICROCONTROLLERS

Revision 0

SAMSUNG ELECTRONICS

Important Notice

S3F94xx-Series Microcontrollers
Application Notes, Revision 0
Publication Number:

Samsung Electronics' microcontroller business has been awarded full ISO-14001 certification (BSI Certificate No. FM24653). All semiconductor products are designed and manufactured in accordance with the highest quality standards and objectives.

Samsung Electronics Co., Ltd.
San #24 Nongseo-Lee, Kiheung-Eup
Yongin-City Kyungi-Do, Korea
C.P.O. Box #37, Suwon 449-900

TEL: (02) 760-6530, (0331) 209-6530
FAX: (02) 760-6547
Home-Page URL: Http://www.samsungsemi.com/

Printed in the Republic of Korea

Table of Contents

Overview 1
Features 1

Charging Theory 2
Ni-MH Battery 2
Charging Method 2
Theory of Operation 2
Charging Curve 2
Terminiation Methods 3

System Implementation 4
S3F94C4 Features 4
System Block Diagram & Specification 5
HardWare Implementation 5
Power Supply 5
LEDs and Switches 5
Buck Converter 5
Measurement Circuit 7
Software implementation 9
Software Flowchart 9
Source Code Files 10

Charging test 15
Test Environment 15
Test Method 15
Test Result 16

Appendix 18
S3F94C4 Features 18
Schematic 19
Source Code 20
Main.c 20
Charge.c 25
Operation.c 27
Monitor.c 31
Global_Define.h 34

List of Figures

Figure Number	Title	Page Number
1	Charging Curve of Ni-MH Battery	2
2	Diagram of Battery Charger Reference Design	4
3	Buck Converter Switch on	6
4	Buck Converter Switch off	6
5	Voltage Measurement Circuit	7
6	Charging Current Measurement Circuit	8
7	Temperature Measurement Circuit	9
8	Main Function	11
9	Fast Charge Process	12
10	Sup. Charge Process	13
11	Current Regulate Flow in Fast Charge	14
12	Test system configuration	15
13	Charging Voltage & Current Test Waveform	17
14	Pin Assignment Diagram (20-Pin DIP/SOP/SSOP Package)	18
15	Schematic of Reference Design	19

List of Tables

Table Number	Title	Page Number
1	Code File Description	10

1. OVERVIEW

Now 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 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 battery charger reference design is based on Ni-MH batteries that fully implements the latest technologies in battery charger designes. The charger can charge battery with full process control: pre-charge the new battery or low voltage battry before fast charge, fast charge Ni-MH batteries with 600mA charging current, supplementary charge after fast charge, keep trickle charge after charge finished.

This battery charger reference design used Samsung highly integrated low cost 8-bit microcontroller S3F94C4, which is ideal for battery charge with timer, PWM, 10-bit ADC. However, it can be implemented using any Samsung microcontroller with A/D converter and PWM output.

Features:

- Fast Charging Algorithm with four charging stages:
  - ✓ Pre-charge with low current when battery voltage is low
  - ✓ Fast charge with voltage and temperature control in constant current
  - ✓ Supplementary charge after fast charge for fully charge
  - ✓ Trickle charge to keep battery fully charged.
- Implements the latest technologies in battery charger designes:
  - ✓ Voltage control: 0 dv or -dv control for fast charge termination
  - ✓ Temperature control: dT/dt, Tmax control for fast charge termination
- High Accuracy measurement with 10-bit A/D converter
- Advanced features for safety and easy-to-use.
  - ✓ Automatic detection of shorted or battery inversed input
  - ✓ Configurable overvoltage, overcurrent and over temperature suspension.
  - ✓ Modular “C” source code.
- 1 bi-color LED (Red/Greed) for on battery to indicate charge status and show error messages.
- Precise power supply soure for MCU system.

2. CHARGING THEORY

2.1 NiMH Battery

Nickel Metal Hydride batteries are the most widely used battery type in new lightweight portable applicaitions(i.e., camera, camcorder, etc.). They have a higher energy density than NiCd. NiMH batteries are damaged from overcharging. It is therefore important to do accurate measurements to terminate the charging at exactly the right time(i.e.,fully charge the battery without overcharging). Like Nicd, NiMH batteries are damaged from being inversed.

NiMH has a self-discharge rate of apporximately 20% / month. NiMH batteries are charged with constant current.

2.2 Charging Method

2.2.1 Theory of operation

The charging of a battery is made possible by a reversible chemical reaction that restores energy in a chemical system. Depending on the chemicals used, the battery will have certain characteristics. When designing a charger, detailed knowledge of these characteristics is required to avoid damage inflicted by overcharging.

2.2.2 Charging Curve

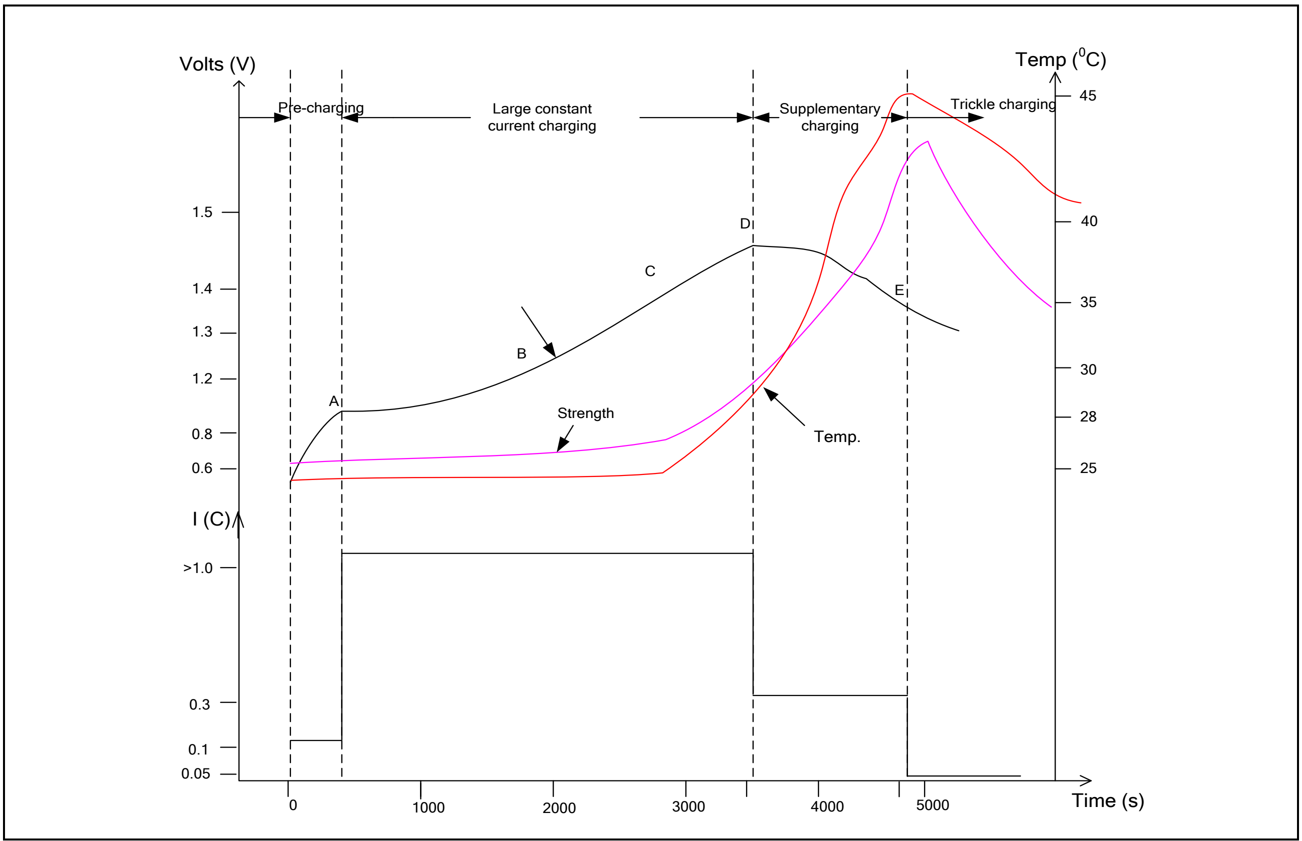

Figure 1. Charging Curve of Ni-MH Battery

If the battery is over-discharged or not used for long time, large current charge can not fully recover the nenergy capacity, so the battery need to be precharged with small current (about 1/30 ～1/20C). This stage called ***pre-charge***.

After the voltage of battery rise up, then can enter ***fast charge*** stage with large current (about 1C) to charge the battery, the charging current is depended on the capacity of the battery and the charging voltage. The charging current always keeps constant.

When match the fast charge temination condition (-△V or 0 △V), the fast charge stage terminated, but the battery is not fully charged, so need to be supplementary charged with 0.3C current. This stage called ***supplementary charge***.

When storage battery, the battery will self-discharge at a rate of C/30 to C/50, so after supplementary charge, the charger will change to ***trickle charge*** stage automatically. In trickle charge stage, charger will keep charging the battery for keep the battery in fully charged status.

2.2.3 Termination Methods

This reference design implements the use of voltage drop (-dV/dt) as primary termination method, with temperature and absolute voltage as backup. But the hareware supports all of the below mentioned methods.

Time control:

This is one of the simplest ways to measure when to terminate the charging. Normally used as backup termination when fast-charging. Also used as primary termination method in normal charging (14-16h). Applies to all batteries.

Voltage:

Charging is terminated when the voltage rises above a present upper limit. Used in combination with constant current charging. Used as backup termination.

-dV/dt—voltage Drop:

This termination method utilizes the negative derivative of voltage over time, monitoring the voltage drop occurring in some battery types if charging is continued after the battery is fully charged. Commonly used with constant current charging. It's the main termination method used in this reference design.

Temperature:

Absolute temperature can be used as termination method, but is preferred as backup termination method only. Charging should be terminated if the temperature rises above the operating termperature limit of Ni-MH batteries. It also used as backup method.

dT/dt – Temperature Rise:

The derivative of temperature over time can be used as termination method when fast charging. Normally, when the temperature increase 1℃/minute, charging should be terminated as quickly as possible.

3. SYSTEM IMPLEMENTATION

3.1 S3F94C4 Features

This reference design using Samsung S3F94C4 as main microcotorller. S3F94C4 is a 20-pin microcontroller, with 4-K bytes flash ROM, and 208 Bytes RAM. It has a 8-bit timer, 10-bit resolution ADC with 9 channels, and 8-bit PWM.

These all features makes S3F94C4 is very suitable for battery charger application: 10-bit ADC for voltage and current measurement; 8-bit PWM for charing current & voltage control , 8-bit timer for system time control. Internal RC OSC is help for those application (like battery charger) that do not need high system frequency.

3.2 System Block Diagram & Specification

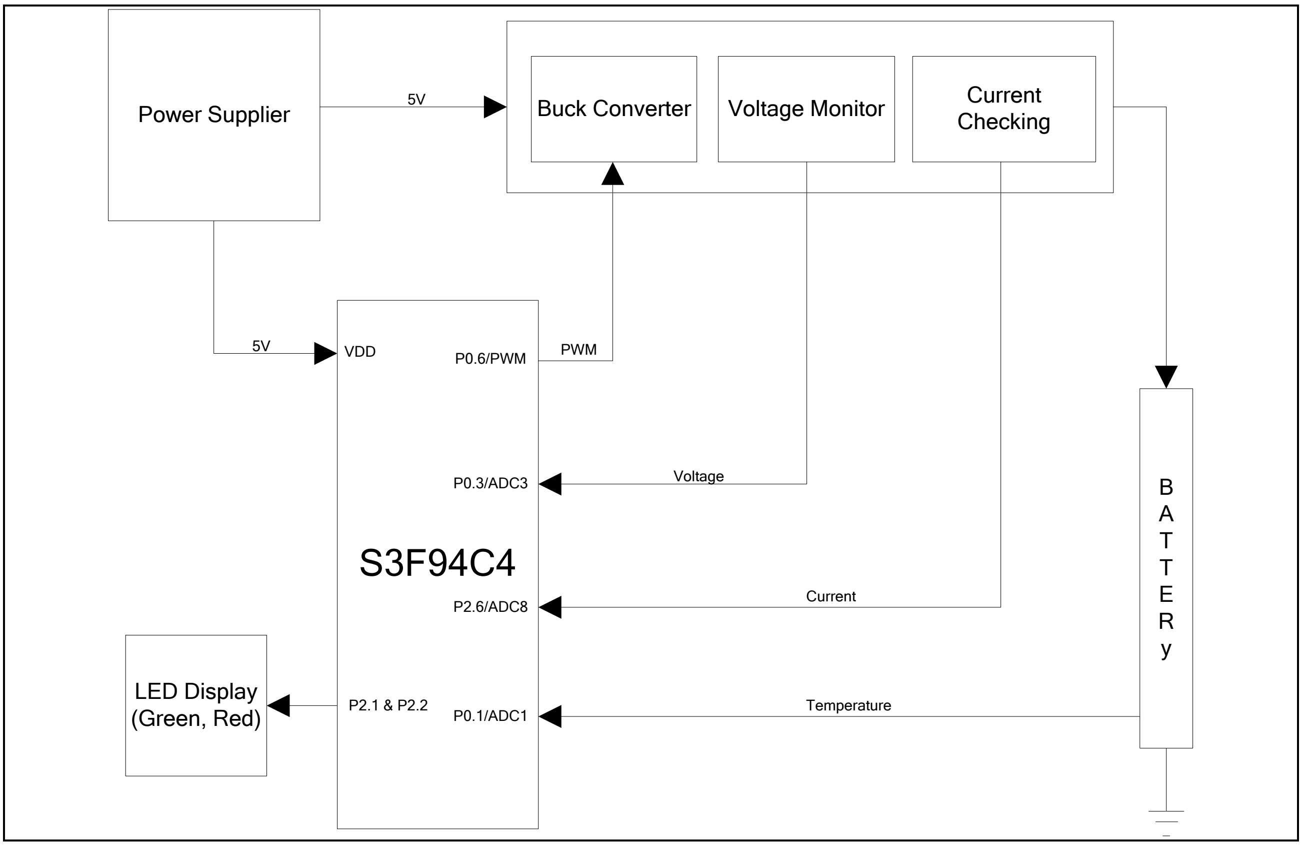

Figure 2. Diagram of Battery Charger Reference Design

- Input to MCU(three ADC input signal)
  - ✓ Voltage monitor for battery fully charged condition check and battery state check
  - ✓ Current check for constant charging current control.
  - ✓ Temperature monitor for battery temperature measurement, charge termination condition

check and battery protection.

- Output from MCU:
  - ✓ PWM output to buck converter circuit for charging current control.
  - ✓ LED output to show charging status and error message with Green and Red LED.

System specification:

- Input voltage: DC 9.0V
- Input current: 100mA
- Output voltage: DC 1.3V
- Output current: 600mA

3.3 Hardware Implementation

3.3.1 Power Supply:

The input voltage is rectified through DC9V-DC5V and then filtered by capacitor. The rectified input voltage is supplied to both the buck converter and to LM7805 voltage regulator. The LM7805 delivers 5V for the microcontroller. The red LED marked “power on” indicates power on.

3.3.2 LEDs and Switches:

This reference design using bi-color LED to indicate the stage of the charge process. If there is no battery insert, the LED is red and blink slowly. If the charging is in processing, the LED is green and blink with different speed in different charge stage. If the battery is fully charged, the LED is green and always on. If there is some error detected, the LED is flicking red. So, from the LED displaying, all of the status of charge process will be acknowledged.

3.3.3 Buck Converter:

The buck charging is usually used in constant current charging. The most economical way to create a constant charge current is to use a buck converter. A buck converter is a switching regulator that uses an inductor as energy storage device.

The buck converter circuit is consist of one P-channel MOSFET switching transistor driven by a bipolar NPN transistor. The switching transistor is connected to an inductor, a diode and a capacitor (see Figure 3).

The charge switch is controlled by PWM. When the switch is on, current will flow as show in Figure 3. The capacitor is charged by the Vin through the inductor. When the switch is opened, as show in Figure 4, 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 cycles repeats itself.

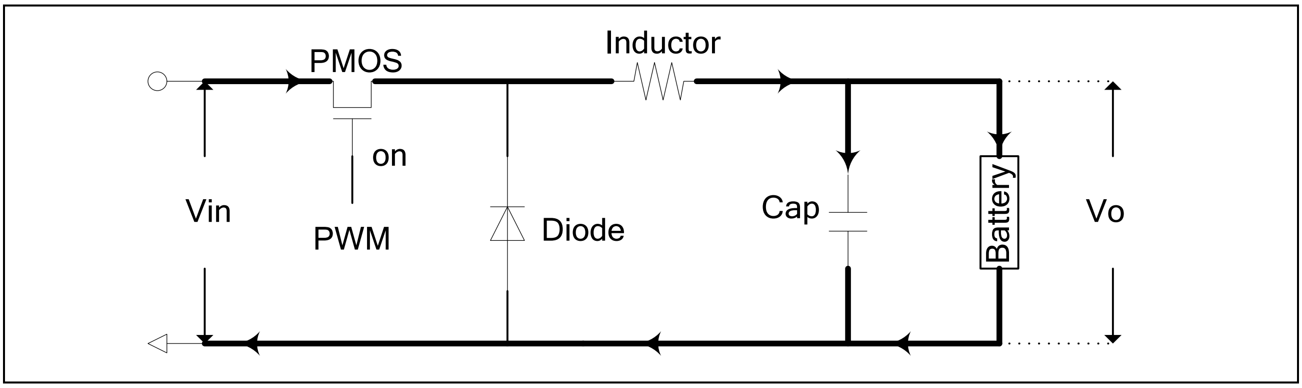

Figure 3 . Buck Converter Switch on

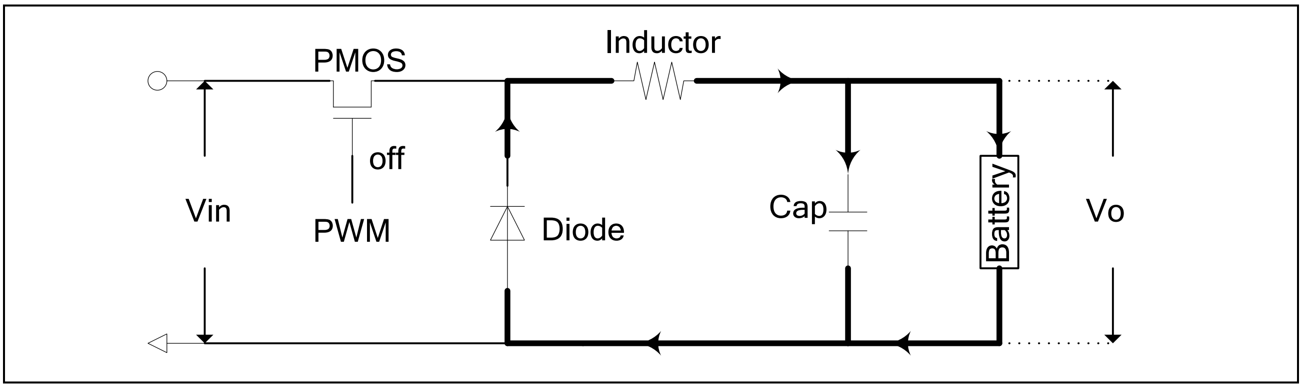

Figure 4. Buck Converter Switch off

If decreases the duty cycle of PWM by shorten the switch 'on' time, the average voltage will decrease. If increases the duty cycle of PWM by longer the switch 'on' time, the average voltage will increase. Therefore, controlling the duty cycles allows us to regulate the charging voltage or the charging current to achieve desired output value. The buck converter is most efficient running on a duty of 50%.

Inductor selection:

$$L = \frac{(V_{IN} - V_{SW} - V_O) \times D}{r \times f \times I_O}$$

Where,

- L: Conver inductor
- V_{IN}: Charger voltage input to switch
- V_{SW}: Voltage loss on switch when switch is on
- V_O: Voltage output
- V_D: Voltage drop on diode when switch is off
- I_O: Current output (the current for constant current charge)
- f: The frequency of the switch.
- D: The duty cycle of the PWM,

$$D = \frac{V_O + V_D}{V_{IN} - V_{SW} + V_D}$$

- r: Ripple of current,

$$r = \frac{\Delta I}{I_O}$$

As this equation shows, the higher the PWM switching frequency, 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.

In this reference design, we assume Vin is 5V, Vsw = 0.3V, Vo = 1.4V, Io = 600 VD is 0.5V, the frequency of switch is about 156KHz, and the ripple of current is about 10%, so the L will be 171uH, in this reference design ,we use 220uH inductor as the energy storage device.

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.

3.3.4 Measurement Circuit

Battery voltage:

The charging voltage is monitored using an op-amp to measure the voltage difference between the positive and the negative pole of the battery. The op-amp circuit for measuring the battery voltage is an ordinary differential op-amp circuit. In order to select a suitable measurement range for the charger, need to select suitable scale resistors for the voltage measurement. The voltage op-amp circuit of this reference design is shown in Figure 5. The equation for the output voltage from the op-amp circuit is shown below. The ADC is capable of measuring the voltage range from 0V to 5V, the output range from the op-amp has to be within this range:

$$V_{bat} = \frac{R_{13}}{R_{12}} V_+ + V_-$$

Where,

V_{bat}: The output voltage from op-amp to microcontroller

V_+: The positive pole of the battery

V_-: The negative pole of the battery

R_a, R_b: The resistors in the resistor network used to set the gain for the op-amp.

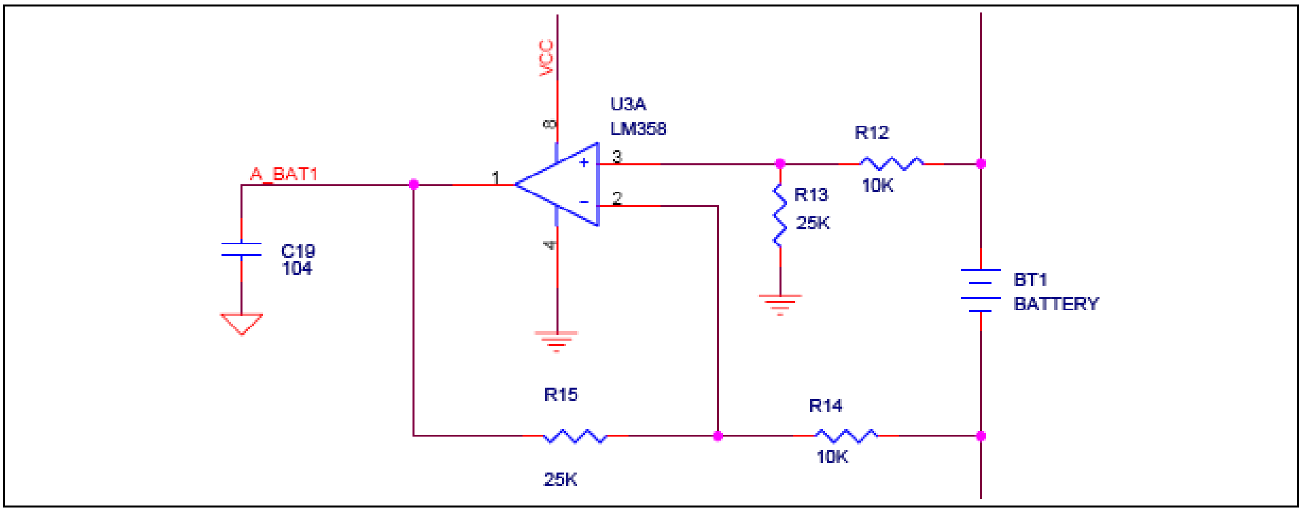

Figure 5. Voltage Measurement Circuit

Charge current:

The detail circuit of charge current measurement is shown in Figure 6.The charge current is measured by sensing the voltage over a 0.050ohm shunt-resistor. This voltage is amplified using an op-amp to improve the accuracy of the measurement before it is fed into the A/D converter.

This voltage is amplified by the factor:

$$1+\frac{R_{22}}{R_{21}}=1+\frac{50000}{470}\approx 51$$

The op-amp output voltage is therefore:

$$V_{Ibat}=\left(1+\frac{R_{22}}{R_{21}}\right)I_{charge}\,R_{20}=2.55\times I_{charge}$$

The maximum charging current that can be measured is:

$$I_{chargeMax}=\frac{V_{ref}}{2.55}=\frac{5}{2.55}=1.96\text{ A}$$

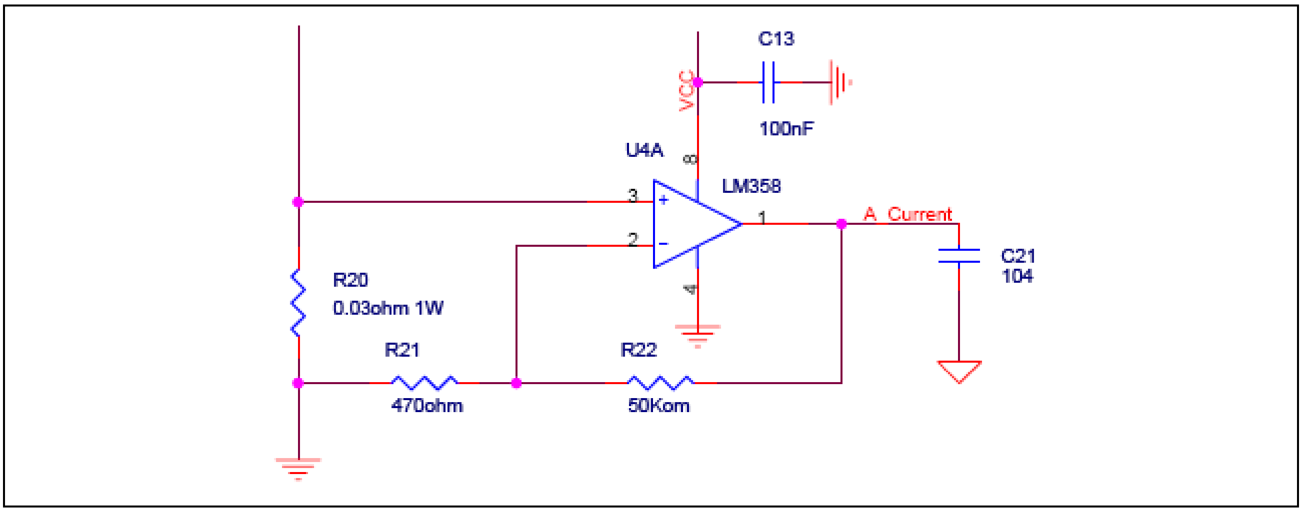

Figure 6. Charging Current Measurement Circuit

Temperature:

Temperature is measured by a negative temperature coefficient(NTC) resistor.The NTC is part of a voltage divider, which is powered by the Vdd for microcontroller. The detail circuit is shown in Figure. 7

The temperature is measured:

$$V_{temp}=V_{DD}\times\frac{R_{25}}{(R_{24}+R_{25})}$$

The resistor value is changed according to the temperature, so the V_{temp} is changed accordingly, so, can detect the temprature by check the voltage value of V_{temp} by A/D convert. But, the relationship between the temperature and resistor value is not linear, which makes it difficult to calculate the temperature from the ADC value. In fact, in the real application field,the temperature range of battery is from 10-45℃, in this temperature range, we can treat it as a linear curve apporximately.

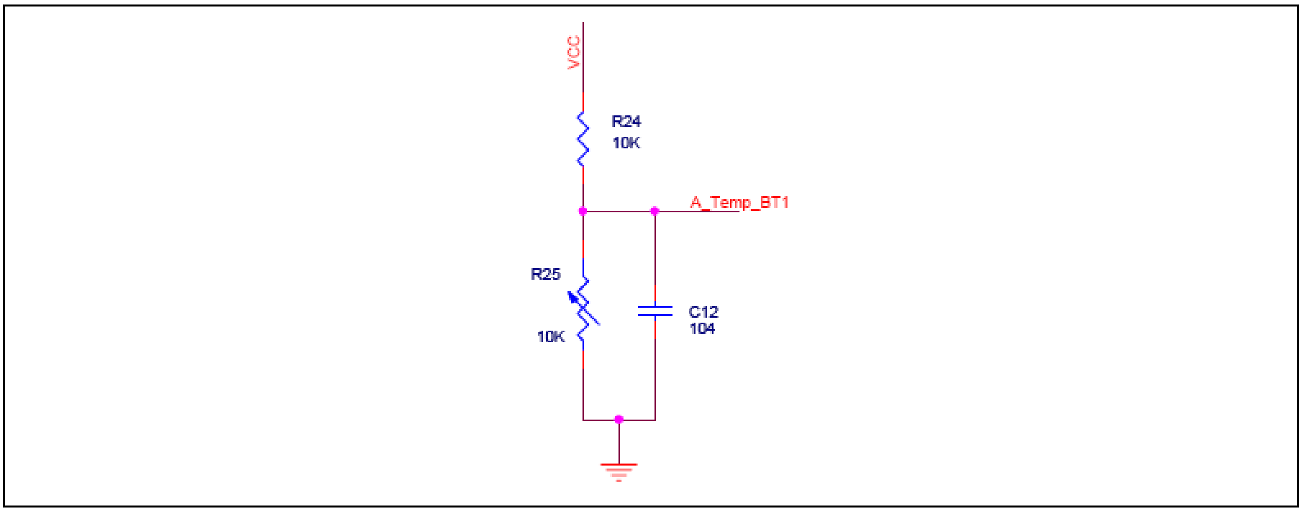

Figure 7. Temperature Measurement Circuit

3.4 Software implementation

3.4.1 Software Flowchart:

The full charge state are divided into four stage: pre-charge, fast-charge, supplementary charge and trickle charge. When a battery is inserted in, which stage is choosed is decided by the battery voltage, and the following charge stage are processed sequencely.

Charge is started if the battery voltage is within the voltage range. If the battery temprature exceed a limited value, the charge will not process. Charge is always terminated with an maximum battery voltage or maximum total-charge time expires.

The normal ways to detect that the battery is fully charged, are the Temperature Rise (dT/dt) and the voltage drop ($-dV/dt$) methods. Therefore, a sample is taken every minute for the temperature and every 2 seconds of the voltage. The values are compared to the sample taken one minute/second ago. In case the battery is fully charged, the charge status is auotomatically changed to trickle-charge.

The trickle-charge excutes in a loop when the overall charge time exceeded the large current charge time limitation, or the voltage or temprature overflow the maximum value.

In this reference design, the charger can charge two battery at the same time. These two battery have same charge mechanism and can be charged simutanenous, so, in the sofeware, there only one battery charge process for demonstration, and it can be easily expanded to support charge two batteries.

3.4.2 Source Code Files

The software is written in C langurage. The source code include following files:

Table 1. Code File Description

File Name	Description	Remark
Main.C	The main function of the code, and the system initialization function.	
Global_define.h	Global variables declaration; Constant define; Marco definition	
Charge.c	The charge function of each charge stage	
Charge.h	Head file for Charge.c; function declaration.	
Operation.c	Execution funtion of the four charge stage.	
Operation.h	Head file for operation.c; function declaration	
Monitor.c	Battery Voltage, charge current , temperature measurement function. Mianly are ADC functions	
Monitor.h	Head file for Monitor.c: function declaration	
ioS3F94C4.h	Register difinition and interrupt vectors declaration for S3F94C4.	

Main.c:

This module include the main function of the system, the system initialization function and interrupt handling routines.

In the “Sys_init” routine, all low-level initialization are done. The I/O ports and PWM, timer block are initialized. In the “System_Clear” routine, the system global variables are clear to there initial value for charge another battery.

The main function “main” is the basic function of the system, the software flowchart is realized in main function, and the major part of the main function is a dealy loop keep running in front of the software platform after chip reset, that check the battery voltage and take execution according to the battery voltage and charge state.

Global_define.h:

In this module, include the definition of the charge state, constant related to the system parameters, and the declaration of global variables. This module is included by each module for common definition and declaration.

ioS3F94C4.h:

This module include the register definintion and interrupt declaration of S3F94C4.

Main flow chart

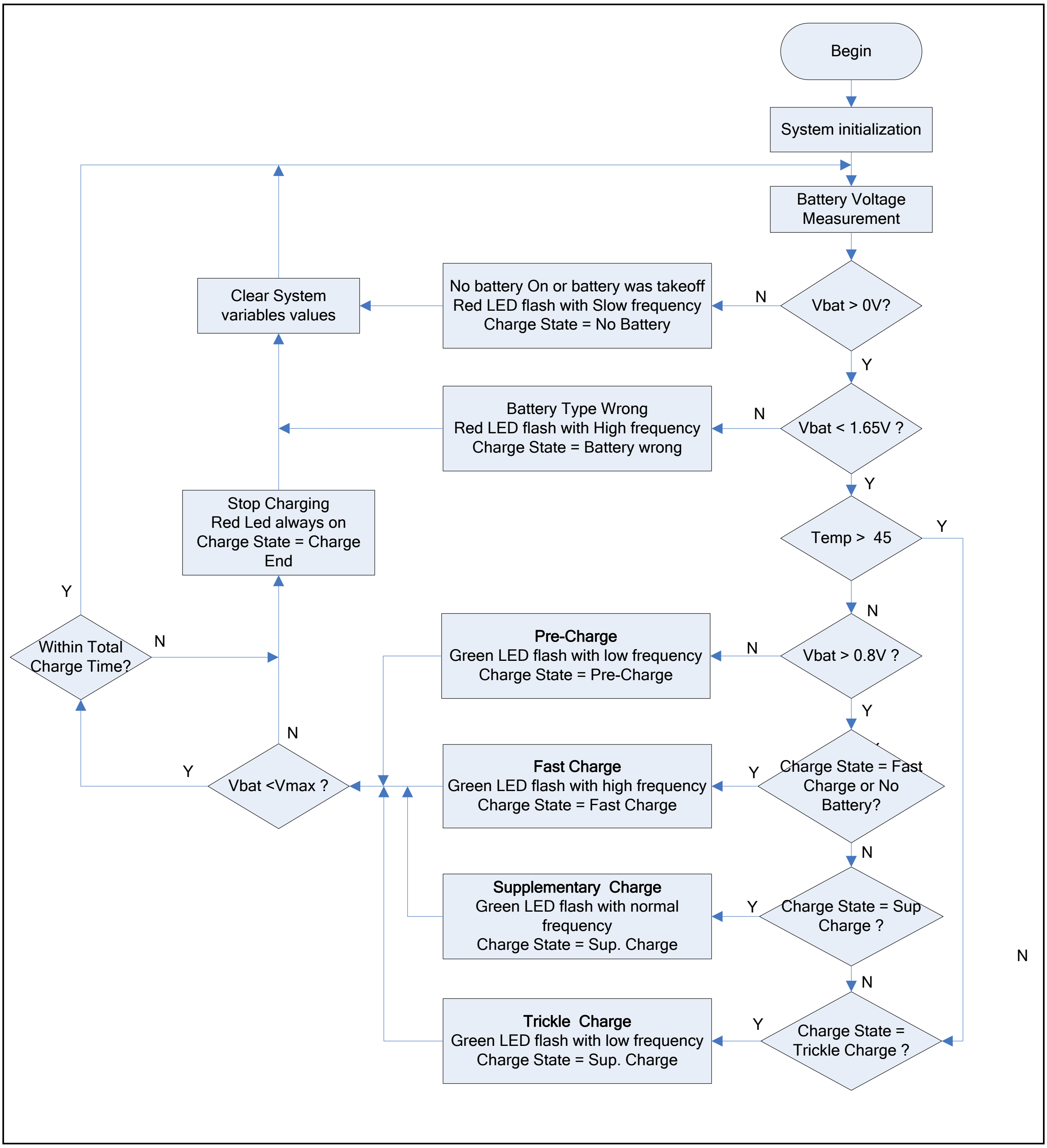

Figure 8. Main Function.

Charge.c:

This module Include the functions for each charge stage. These functions are part of the main loop, and called by main function.

Ni-MH battery is charged by constant current, in fast charge stage, the charge current is set to about 600mA. The charge is terminated by the Temperature Rise(dT/dt) and the Voltage Drop(-dV/dt) methods. Maximum charge voltage and maximum charge time are used as backup terminations.

Fast charge process:

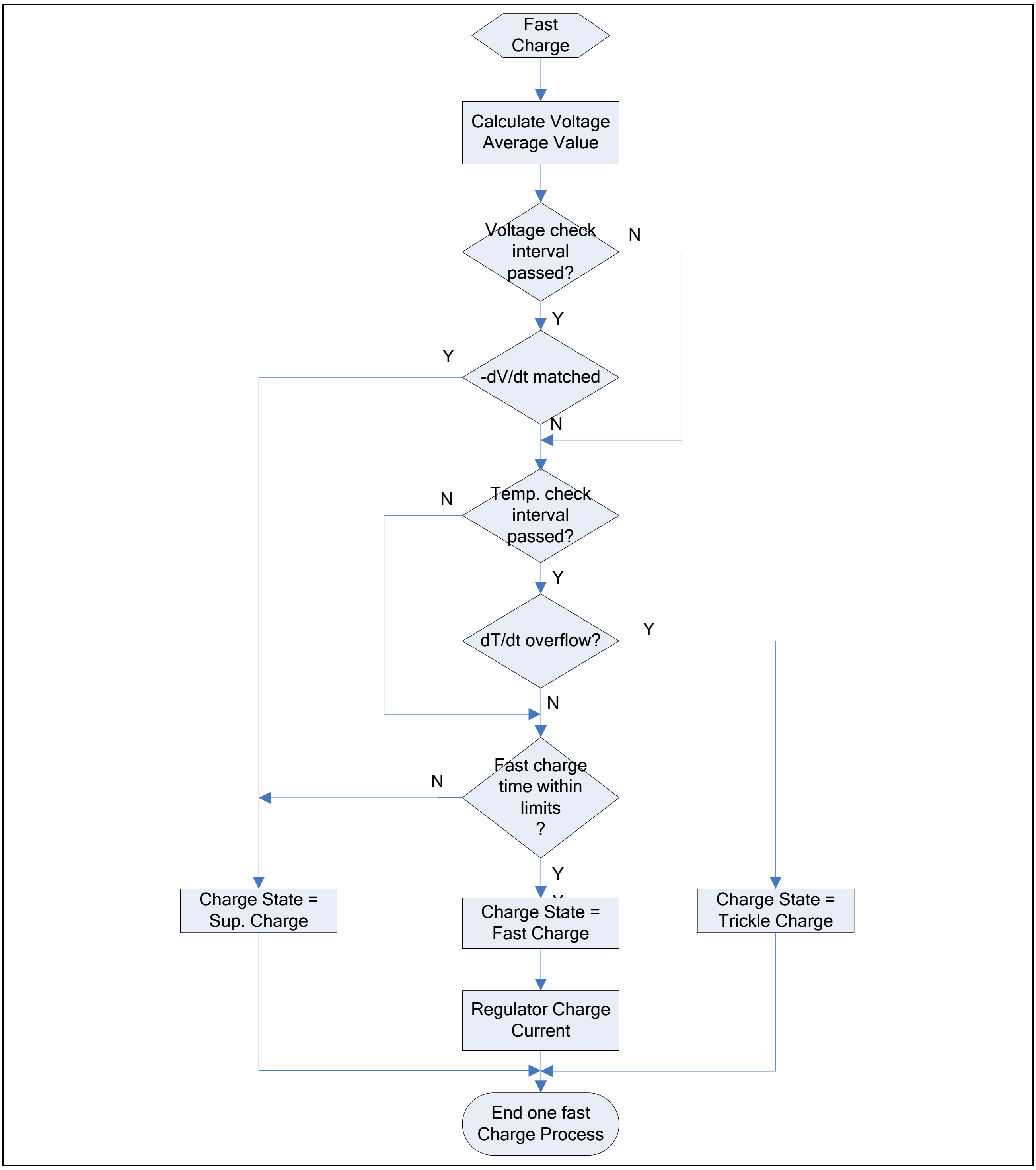

Figure 9. Fast Charge Process.

In case the battery is fully charged the charge stage is automatically changed to supplementary charge, causing the program to execute the supplementary charge function.

Supplementary charge is also charge by constant current, and the charge is terminated by Temperature Drop (dT/dt) or Maximum supplementary charge time. In case of the termination condition matched, the charge status changed to trickle charge automatically.

Supplementary charge process:

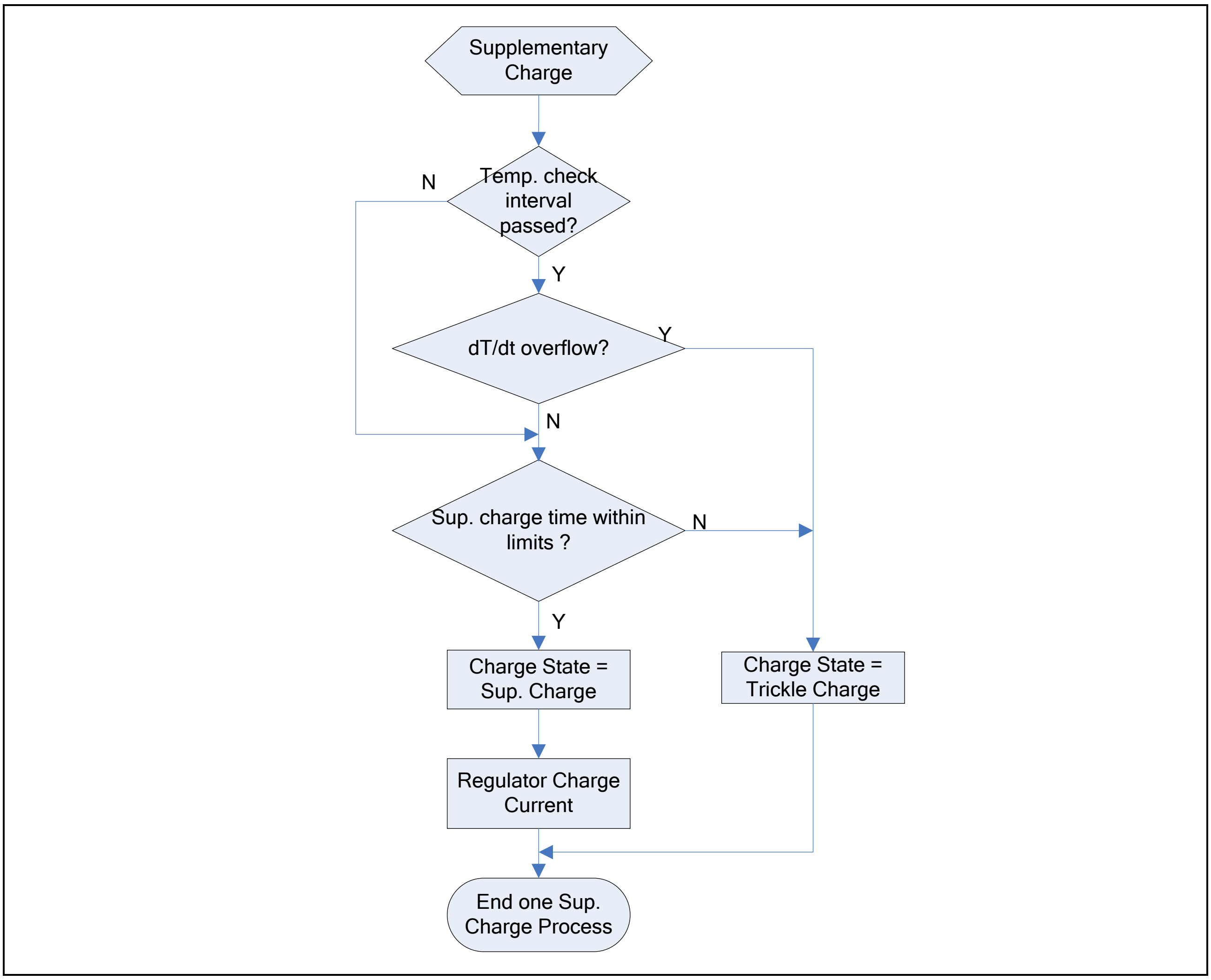

Figure 10. Sup. Charge Process.

Monitor.c:

This module mainly include the measurement functions of battery voltage, charge current and battery temperature. And the charge termination condition check functions are also included.

Operation.c:

This module include the charge current regulator of each charge state, mainly the PWM duty width control accroding the required constant charge current. And the PWM operation functions and the system message display function are also included in this module. These four stage have similar control algorithm, so, we take the fast charge as example:

Current Regulator Flow:

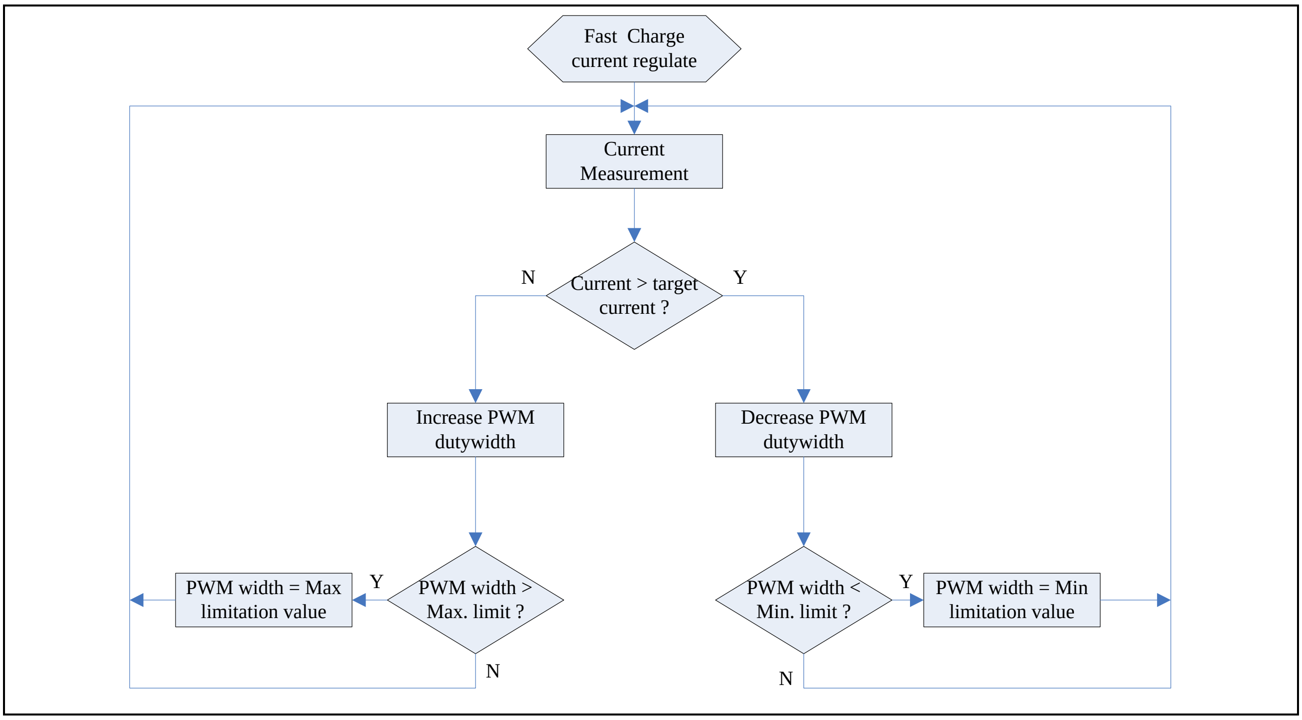

Figure 11. Current Regulate Flow in Fast Charge.

4. CHARGE TEST

Test Environment

- Temperature: 25℃
- Battery: 1300mAh Ni-MH battery
- Power Supply: Adapter (output: 9V, max 1.0A)
- Instruments: Agilent 34401A Digit Multimeter *2

Test Method

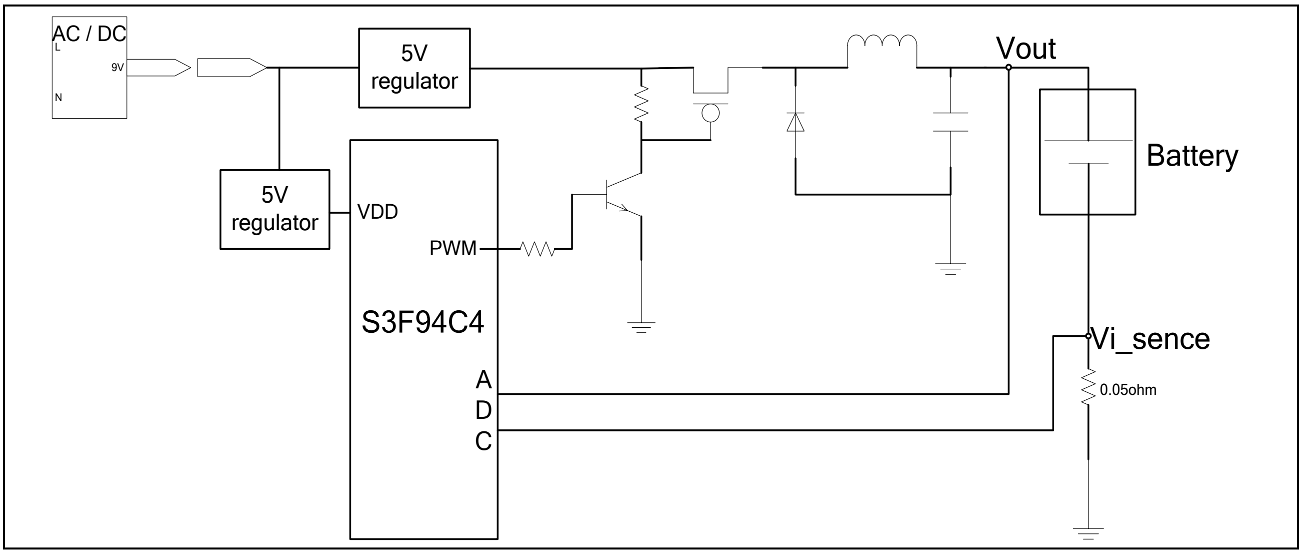

Figure 12. Test system configuration.

Detected value:

Vout = Vbat + Vi_sence

Iout = Vi_sence / 0.05

As shown in Figure 12, during the charging process, watch the voltage at the test points of Vout and Vi_sence, using two multimeters to get the charging voltage, then calculate the charging current by Vi_sence / 0.05.

At the begginning of the charging process, record the data every 60 seconds. When the charging current and voltage become stable, the test interval becomes longer (every 4 minutes).

Test Result

- Fast charging time: 56 minutes
- Constant current of fast charge: 610mA
- Fast charge end voltage: 1.408V
- Supplementary charge current: 120mA
- Supplementary charge end voltage: 1.396V
- End charge voltage:1.396V

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 following test diagrams.

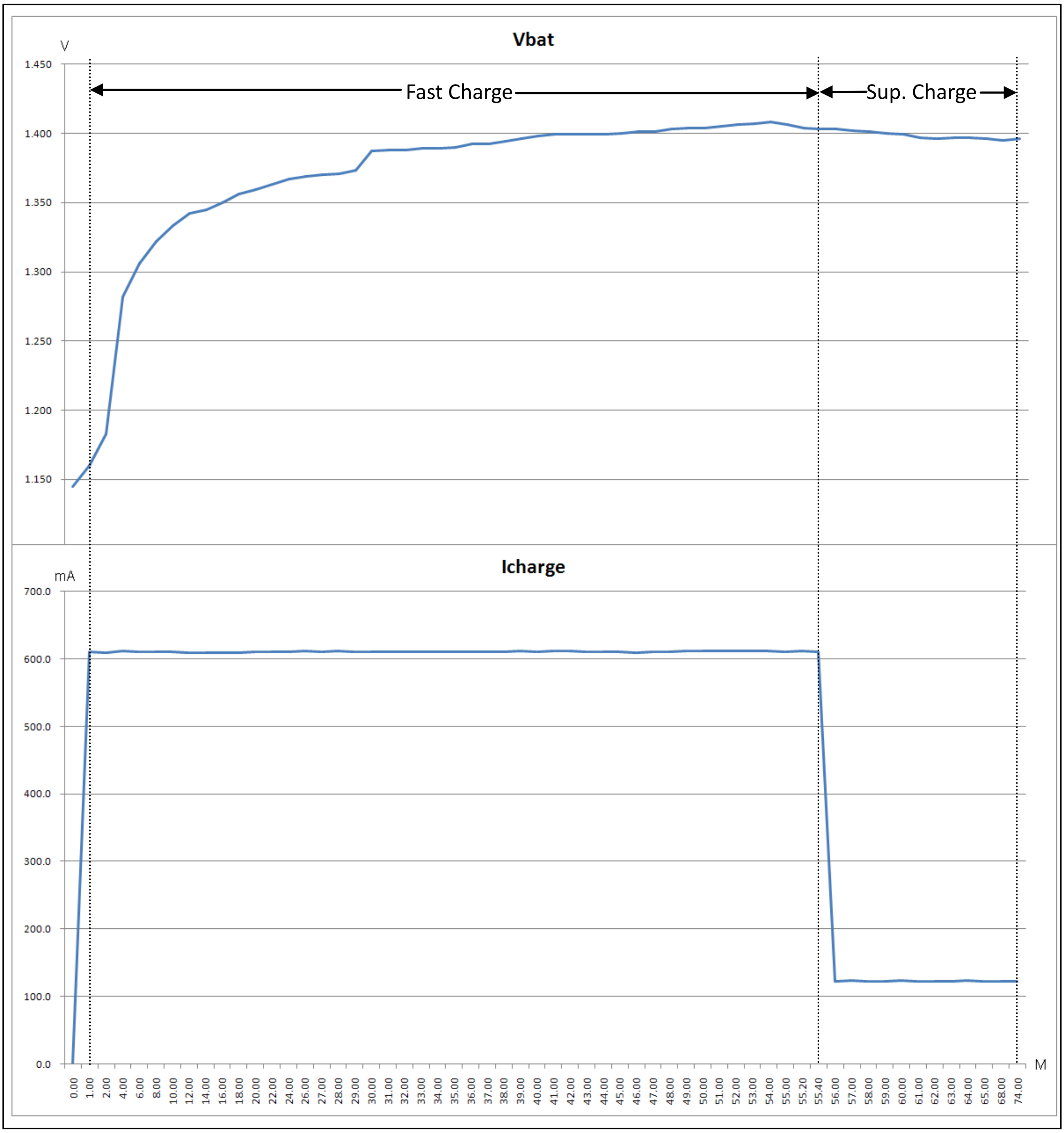

Figure 13. Charging Voltage & Current Test Waveform.

5. Appendix

S3F94C4 Features:

Memory

- 4-Kbyte internal multi-time program Full-Flash memory
- 208-byte general-purpose register area

General I/O

- Three I/O ports (Max 18 pins)
- Bit programmable ports

1-ch Three Modes High-speed PWM

- 6-bit base + 2-bit extension
- 8-bit base + 6-bit extension
- 6-bit base + 6-bit extension

Timer/Counters

- One 8-bit basic timer for watchdog function
- One 8-bit timer/counter with time interval modes

A/D Converter

- Nine analog input pins (MAX)
- 10-bit conversion resolution

Built-in RESET Circuit (LVR)

- Low-Voltage check to make system reset
- V_{LVR} = 1.9/2.3/3.0/3.6/3.9 V (by smart option)

Operating Temperature Range

- – 40°C to + 85°C

Operating Voltage Range

- 1.8 V to 5.5 V @ 1-4M Hz(LVR disable)
- LVR to 5.5V @ 1-4M Hz(LVR enable)
- 2.7 V to 5.5V @ 1-10M Hz

Package Types

- S3F94C4:
  - 20-DIP-300A
  - 20-SOP-375
  - 20-SSOP-225
  - 16-DIP-300A
  - 16-SOP-225
  - 16-TSSOP-BD44

Pin Assignment:

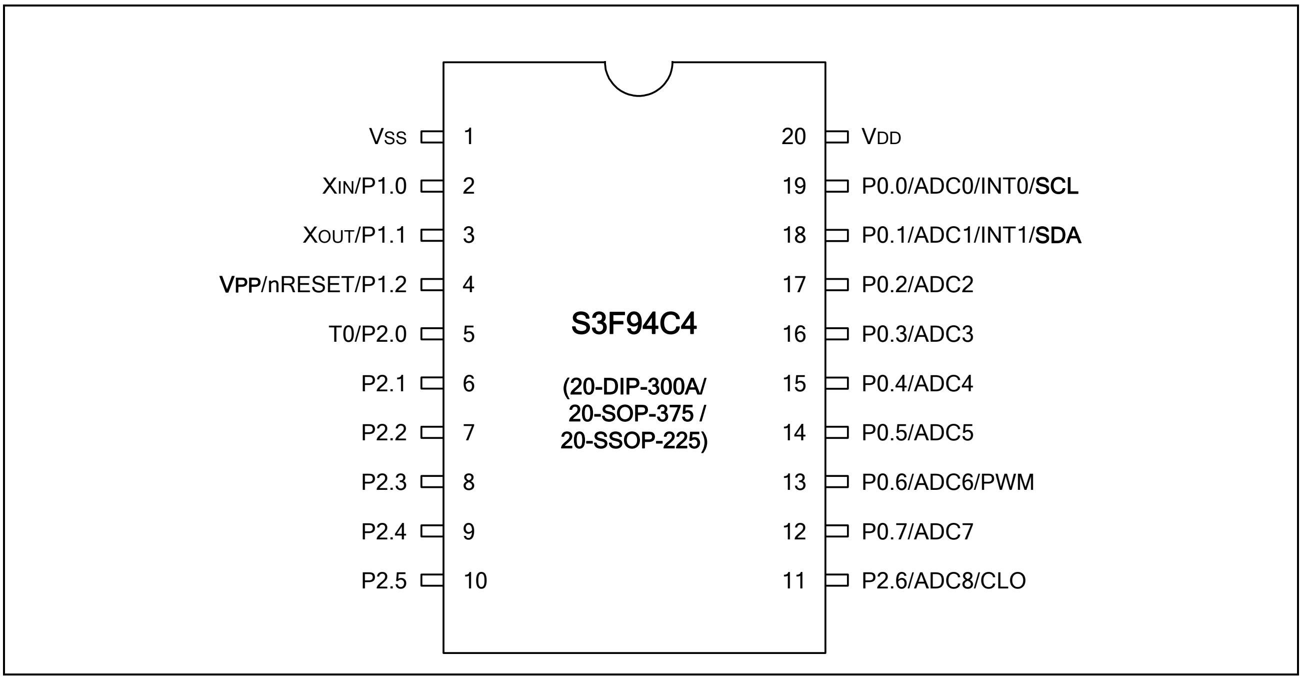

Figure 14. Pin Assignment Diagram (20-Pin DIP/SOP/SSOP Package)

Schematic

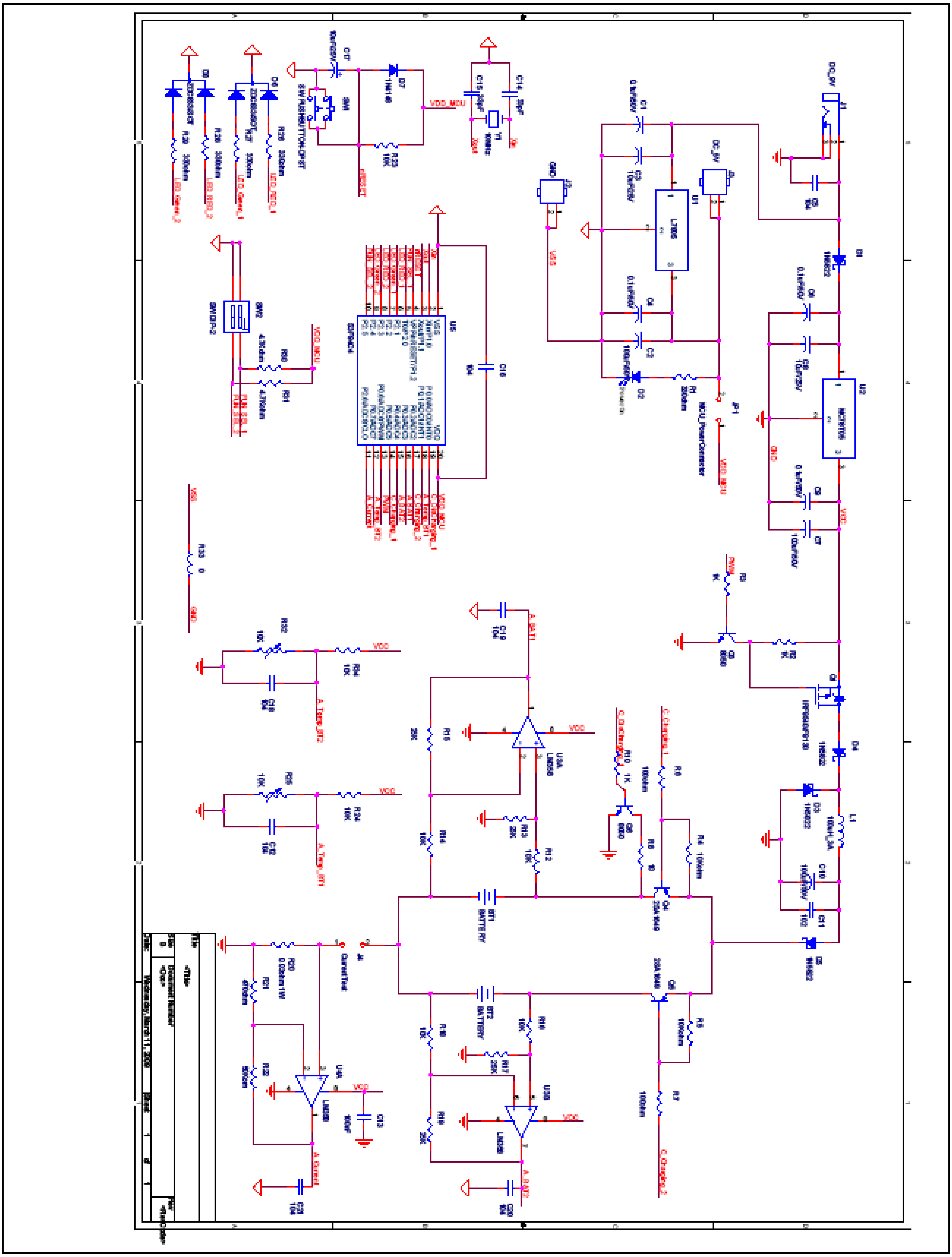

Figure 15. Schematic of Reference Design

Source Code

```
/**
* @file name    Main.c
* @description  Main functions for Battery charger code
*
*                          XTAL = 10MHz
*
* @author        Li Baoke(86-571-86726288 EXT.8103, baoke.li@samsung.com)
* @version       Preliminary 0.0
* @historyHistory type - NEW/MODify/ADD/DELete
*                 ------------------------------------------------------------------
*                 |ver type when           who                              what
*                 |---+---+----------+-------------------------+----------------------
*                 |0.0 NEW 2008-03-06    Li Baoke                         Creation
*                 ------------------------------------------------------------------
*
* @see     IAR C Compiler Tool
*/
/*******************************************************************************
***************************   I N C L U D E S   ******************************
*******************************************************************************/
#include "Globle_Define.h"
#include "Charge.h"
#include "Operation.h"
#include "Monitor.h"
/*******************************************************************************
***************************   S M A R T   O P T I O N   ********************
*******************************************************************************/
/* Smart option 3CH
     Must be initialized to 0x00 */
__root __code const unsigned char SMT1 @ 0x3C = 0x00;
/* Smart option 3DH
     Must be initialized to 0x00 */
__root __code const unsigned char SMT2 @ 0x3D = 0x00;
/* Smart option 3EH
     0xFF -> LVR Enable (default)
     0x7F -> LVR Disable              */
__root __code const unsigned char SMT3 @ 0x3E = 0x7F;
/* Smart option 3FH
     0xFF -> Internal RC 3.2MHz (default)
     0xFE -> Internal RC 0.5MHz
     0xFD -> 0.5MHzExternal RC
     0xFC -> External Crystal       */
__root __code const unsigned char SMT4 @ 0x3F = 0xFC;
/*******************************************************************************
***************************Global Variable definition************************
*******************************************************************************/
/*---------------------------------------------------------------
*                          Battery    related variables
*---------------------------------------------------------------*/
unsigned char Bat1State = 0; //battery 1 state
/* voltage monitor related */
unsigned int    Bat1Volts = 0;                            //Battery 1 voltage ADC convert result;
unsigned int    Bat1VoltsArray[9]={0,0,0,0,0,0,0,0,0};    //voltage sample array,last one is average value.
unsigned int    Bat1AvgArray[9] = {0,0,0,0,0,0,0,0,0};    //voltage average array, last one is average value.
/* temperature related */
unsigned int    Bat1TempADC = 0;                          //battery 1 temperature ADC result
unsigned int    Bat1Temp = 0;                             //battery 1 temperature
unsigned int    Bat1PreTemp = 0;                          //battery 1 pre    temp data
unsigned int    Bat1TempChkIntv    = 0;                   //battery temperature checking interval
/* time control related parameters (time = interval *    counter) */
unsigned int    Bat1TimeTotalInterval = 0;                //battery 1 total charge time interval
unsigned char Bat1TimeTotalCounter = 0;                   //battery 1 total charge time counter
unsigned int    Bat1TimeFastInterval = 0;                 //battery 1 fast charge time interval
unsigned char Bat1TimeFastCounter = 0;                    //battery 1 fast charge time counter
```

```
unsigned int    Bat1TimeSupInterval = 0;                //battery 1 Sup. charge time interval
unsigned char Bat1TimeSupCounter = 0;                   //battery 1 Sup. charing time counter
/*Termination condition check related variables*/
unsigned char Bat1VoltChkFlag =0;                       //voltage checking flag:1-start check; 0- no check
unsigned int    Bat1VoltChkIntv = 0;                    // battery 1 voltage checking interval
unsigned int    PreVolts = 0;                           //voltage check value: pre-tested value
unsigned int    PreVolts1 = 0;                          //voltage check value: pre-tested value 1
unsigned int    VoltDropCnt    = 0;                     //counter of voltage drop (Prevoltage - Vcheck >= 1)
unsigned char VoltDropCnt1 = 0;                         //counter of voltage drop every 1 minute.
unsigned int    Bat1AvgMax = 0;                         //Max value of the average voltage
unsigned int    Bat1AvgMin = 790;                       //Min value of the average voltage
unsigned int    VoltAvgDropCnt = 0;                     //counter of Vave <= Vmax-4
unsigned char VoltAvgDropCnt1 = 0;                      //counter of Vave <= Vmax-3
unsigned int    DvStartTestTime = 0;                    //-dv check delay time
/*----------------------------------------------------------------
*                          common variables
*----------------------------------------------------------------*/
unsigned char PWMWidth    = 80;                         //Fast charging    pwm duty width
unsigned char PWMRunFlag       = 0;                     //PWM run or stop flag:0 == init; 1== strat run; 2== stop run
unsigned int    ChargingCurrent = 0;                    //Charging current convert result.
unsigned char TOMatchCounter = 0;                       // TO interrupt timing counter
/******************************************************************************
****************************    F U N C T I O N S    *************************
******************************************************************************/
/*
** Main function
*/
void main(void)
{
   __enable_interrupt();
   __disable_interrupt();    //Disable globle interrupt
   SP=0xC0;    //stack point setting @ 0xC0
   Sys_init(); //System inintialization: board enviroment setting
   __enable_interrupt();    //Enable globle interrupt
   while(1)
   {
         ChargingCurrent = Charging_Current_Monitor();
         ChargingCurrent = Charging_Current_Monitor() + CURRENT_AMP_COMPENSATE;
         /*------- battery take off check--------------*/
         if( ( (Bat1State == BATTERY_FAST_CHARGING)    ||
                  (Bat1State == BATTERY_SUP_CHARGING) ||
                  (Bat1State == BATTERY_TRICKLE_CHARGING) ) &&
               (ChargingCurrent <= CHG_CURRENT_MIN)              )          //take off in charging process
         {
               Bat1State = BATTERY_CHARGING_END;
               Show_BAT1_State(BATTERY_CHARGING_END); //show message
                System_Clear();
               delay(65500); //wait for capacitor discharge
               delay(65500);
               delay(65500);
               delay(65500);
               delay(65500);
               delay(65500);
               delay(65500);
               Bat1State = NO_BATTERY;
         }
        Bat1Volts = BAT1_V_Monitor();                         //ADC result = ((2.5Vbat+) + Vbat-))*1024/5
         Bat1Volts += BAT1_V_Monitor();
         Bat1Volts += BAT1_V_Monitor();
         Bat1Volts = Bat1Volts / 3;
         if(Bat1Volts >= BAT_DETECTOR_VOLTS)
               Bat1Volts -= (ChargingCurrent/CURRENT_AMP_GAIN);
         /*------- battery on check--------------*/
         if( Bat1Volts <= BAT_DETECTOR_VOLTS )              //if Vbat <0.1V,no battery insert
         {
               Bat1State = NO_BATTERY;
               Show_BAT1_State(NO_BATTERY);                  //show message
```

```
            System_Clear();                         // clear global variables to init. values
        }
        /*-------Decide DV check delay time-----*/
        if( (Bat1Volts > BAT_DETECTOR_VOLTS) && (Bat1State == NO_BATTERY))
        {
                if(Bat1State > VOLTS_OF_INIT_DLY_1)
                   DvStartTestTime = INIT_CHECK_DLY_1;
                if(Bat1State > VOLTS_OF_INIT_DLY_2)
                   DvStartTestTime = INIT_CHECK_DLY_2;
                if(Bat1State > VOLTS_OF_INIT_DLY_2)
                   DvStartTestTime = INIT_CHECK_DLY_3;
        }
        /*------- battery type check--------------*/
        if( Bat1Volts   >= BAT_MAX_VOLTS)                    //if Vbat > 1.5V, battery tpye wrong or charging finished
        {
            if( (Bat1State != BATTERY_TYPE_ERROR) &&         //Vmax Control: if Vbat > Vmax, enter trickle charge
                (Bat1State != NO_BATTERY)            )
            {
                Bat1State = BATTERY_TRICKLE_CHARGING;
            } else {
                Bat1State = BATTERY_TYPE_ERROR;
            }
        }
        /*------- battery temperature monitor--------------*/
        Max_Temp_Detect();
        /********** Pre-charge ***************/
        if( (Bat1Volts > BAT_DETECTOR_VOLTS) &&                   //if   0.1V < Vbat <0.8V, pre-charging
            (Bat1Volts <= BAT_PREEND_VOLTS)   &&
            (Bat1State <= BATTERY_PRE_CHARGING ) )
        {
              Bat1State = BATTERY_PRE_CHARGING;
              Battery_Pre_Charge();
        }
        /********** Fast charge ***************/
        if( ((Bat1Volts > BAT_PREEND_VOLTS) &&                     //if   1.2V < Vbat <1.6V, charging...
             (Bat1Volts <= BAT_MAX_VOLTS))   &&
            (Bat1State <= BATTERY_FAST_CHARGING) )
        {
                Fast_Charge();
        }
        /********** supplementary charge ***************/
        if( ((Bat1Volts > BAT_PREEND_VOLTS) &&
             (Bat1Volts <= BAT_MAX_VOLTS))   &&
            (Bat1State == BATTERY_SUP_CHARGING) )
        {
                Sup_Charge();
        }
        /********** trickle   charge ***************/
        if(Bat1State == BATTERY_TRICKLE_CHARGING)
        {
            Bat1State = BATTERY_TRICKLE_CHARGING;
            Battery_TRK_Charge();
        }
        /*------- total charging time check --------------*/
        if( (PWMRunFlag == CHARGING_RUN) && (Bat1State != BATTERY_TRICKLE_CHARGING))
        {
           Max_ChargeTime_Detect();
        }
        Show_BAT1_State(Bat1State);
      }
}
/*
** System and peripheral registers initializtion.
*/
void Sys_init(void)
{
```

```
    /*System Control Registers Initialization*/
    BTCON = 0xA3;                    //disbale WacthDog, clear basic timer couter
    CLKCON = 0x0C;                   //enable IRQ wake up; Fcpu = Fosc/1
    /*I/O Ports Control Registers Initialization*/
    P0CONH = 0xDB;                   //11011011b
                                     //P0.7 ADC input --- battery 2 temperature monitor;
                                     //P0.6 PWM       --- Bulk circuit control signal; set as output in init stage.
                                     //P0.5 Output    --- Battery 1 charing control;
                                     //P0.4 ADC input --- Battery 2 voltage monitor;
    P0CONL = 0xEE;                   // 11101110b
                                     //P0.3 ADC input --- battery 1 voltage monitor;
                                     //P0.2 Output    --- Battery 2 charging control;
                                     //P0.1 ADC input --- Battery 1 temperature monitor;
                                     //P0.0 Oupput    --- Battery 1 Discharging control;
    P0PND = 0x00;                    //no external interrupt --- disable external interrput
    P0 = 0x00;                       //Port 0 no output;
    P1CON = 0x0A;                    //00001010b
                                     // P1.1-0 set to output to prevent current consumption
    P1 = 0x00;                       //Port1 not used.
    P2CONH = 0x32;                   //00110010b
                                     //P2.6 ADC input --- Charing current moniotr;
                                     //P2.5 Input     --- Function selection signal 2
                                     //P2.4 Output    --- Green Led for battery 2
    P2CONL = 0xA8;                   //10101000b
                                     //P2.3 Output    --- Red Led for battery 2
                                     //P2.2 output    --- Green Led for battery 1
                                     //P2.1 output    --- Red Led for battery 1
                                     //P2.0 input     --- Function selection signal 1
    P2 = 0x00;                       // P2.4-.1 output low (LED trun off).
    /* Peripheral Control Registers Initialization */
    T0CON = 0x02;                    //00000010b
                                     //Clock = fosc/4096
                                     //clear counter;enable interrupt; clear pending bit;
    T0DATA = 122;                    //inteval: 122 cycles.   50ms@10MHz system clock.
    PWMCON = 0xD0;                   //PWM initialize: 11010000b
                                     //Clock = fosc/1;
                                     //Stop run at first;
                                     //disable interrupt; clear pending bit;
    PWMDATA = 0x00;                  //base mode
    ADCON = 0x94;                    //ADC module initialize: 10010100
                                     //channel select: connect to GND
                                     //clock = fosc/4 = 2.5MHz@ fosc = 10MHz
                                     //Stop convert
    /* Global Variable initialize */
    Bat1State =   NO_BATTERY;        //default: no battery after start run...
}
void System_Clear()
{
    unsigned char i;
    PWMCON &= 0xFB;
    //set P0.6 as output :
    P0CONH   &= 0xCF;                //&11101111B (bit5 = 0)
    P0CONH   |= 0x20;                //|00100000B (bit4 = 1)
    P0_bit.b6 = 0;                   //output low to stop charging   Bat1TimeTotalInterval = 0;
    Bat1TimeTotalCounter = 0;
    Bat1TimeFastInterval = 0;
    Bat1TimeFastCounter = 0;
    Bat1TimeSupInterval = 0;
    Bat1TimeSupCounter = 0;
    Bat1VoltChkIntv = 0;
    Bat1VoltChkFlag = 0;
    for(i=0; i<8;i++)
        Bat1VoltsArray[i] = 0;
    for(i=0; i<8;i++)
        Bat1AvgArray[i] = 0;
    PreVolts = 0;
    PreVolts1 = 0;
```

```
   VoltDropCnt    = 0;
   VoltDropCnt1 = 0;
   Bat1AvgMax = 0;
   Bat1AvgMin = 790;
   VoltAvgDropCnt = 0;
   VoltAvgDropCnt1 = 0;
   DvStartTestTime = 0;

   Bat1TempADC = 0;
   Bat1Temp = 0;
   Bat1PreTemp = 0;
   Bat1TempChkIntv    = 0;

   PWMWidth = 0;
   PWMRunFlag = 0;
   ChargingCurrent = 0;
}
/*
** Delay function
*/
void delay(unsigned int nLoop_CNT)
{
   int i;
   for(i=0;i<=nLoop_CNT;i++)
      __no_operation();
}
/*******************************************************************************
****************************Interrupt service routine************************
*******************************************************************************/
/*
** Interrupt service routine (software polling sequence decide interrupt priority.)
*/
#pragma vector=__P00_vector
__interrupt void ISR_Processing(void)
{
   if(T0CON_bit.PND == 1)
   {
         TOMatchCounter ++;      //match interval: 50ms
         if( (PWMRunFlag ==CHARGING_RUN) && (Bat1State != BATTERY_TRICKLE_CHARGING) )
         {
               Bat1TimeTotalInterval++;
         }
         if( (PWMRunFlag ==CHARGING_RUN) && ( Bat1State == BATTERY_FAST_CHARGING) )
         {
               Bat1TimeFastInterval++;
               //Bat1TempChkIntv++;
               if(Bat1VoltChkFlag == 1)
               {
                     Bat1VoltChkIntv++;
               }
         }
         if( (PWMRunFlag ==CHARGING_RUN) && (Bat1State == BATTERY_SUP_CHARGING))
         {
               Bat1TimeSupInterval++;
               //Bat1TempChkIntv++;
         }
   }
   PWMCON_bit.PND = 0;
   T0CON_bit.PND = 0;      //clear timer0 pending bit.
   P0PND_bit.INT0_PND = 0;    // Clear    pending bit
}
```

```
/**
* @file name     Charge.c
* @description  charge function for fast charge and supplementary charge
* @author       Li Baoke(86-571-86726288 EXT.8103, baoke.li@samsung.com)
* @version          Preliminary 0.0
* @history       |----------------------------------------------------------------
*                |ver type when            who                                   what
*                |---+---+----------+------------------------+---------------------
*                |0.0 NEW 2008-03-06    Li Baoke                          Creation
*/
#include "Globle_Define.h"
#include "Charge.h"
#include "Monitor.h"
#include "Operation.h"
/*Fast charge process*/
void Fast_Charge(void)
{
     unsigned char i;
     /**********calculate the average voltage***************/
    if(Bat1VoltsArray[0] == 0)                                   //in the inita state,set the first as Bat1Volts
    {
        for(i=0; i<8;i++)
            Bat1VoltsArray[i] = Bat1Volts;
    }
    for(i = 8; i>0; i--)                                         //array data rotate right one.
        Bat1VoltsArray[i] = Bat1VoltsArray[i-1];
    Bat1VoltsArray[0] = Bat1Volts;                               //set the first the data as the newest voltage sample value
    Bat1VoltsArray[8] = 0;                                       //the last one set as 0
    for(i = 0; i<8; i++)                                         //get sum of the 8 data.
       Bat1VoltsArray[8] += Bat1VoltsArray[i];
    Bat1VoltsArray[8] = Bat1VoltsArray[8] / 8;                   //the last one is the average of the 8 sample values.
    /**********0dv and -dv control***************/
    if ( (Bat1Volts >= START_CHECKING_VOLTAGE) ||                //0 dv and -dv control: when Vbat > 1.3V,
          (Bat1TimeTotalInterval >= DV_STARTTEST_TIME_LMT) )     //start charing time limit
    {
        Bat1VoltChkFlag = 1;
    }
    if (Bat1VoltChkIntv >= VOLT_CHK_INTV)                        // 0 dv and -dv control:
    {
        if(Bat1AvgArray[0] == 0)
        {
            for(i=0; i<9;i++)
               Bat1AvgArray[i] = Bat1VoltsArray[8];
        }
        for(i = 8; i>0; i--)                                     //array data rotate right one.
            Bat1AvgArray[i] = Bat1AvgArray[i-1];
        Bat1AvgArray[0] = Bat1VoltsArray[8];                     //set the first data as the newest sample value
        Bat1AvgArray[8] = 0;                                     //the last one set as 0
        for(i = 0; i<8; i++)                                     //get sum of the 8 data.
           Bat1AvgArray[8] += Bat1AvgArray[i];
        Bat1AvgArray[8] = Bat1AvgArray[8] / 8;                   //the last one is the average of the 8 sample.
        if(Bat1AvgArray[8] > Bat1AvgMax)
        {
            Bat1AvgMax = Bat1AvgArray[8];
        }else if(Bat1AvgArray[8] <= Bat1AvgMin)
        {
            Bat1AvgMin = Bat1AvgArray[8];
        }
        if(Bat1AvgMax >= (Bat1AvgArray[8] + 4))
        {
            VoltAvgDropCnt ++;
        }
        if(Bat1AvgMax >= (Bat1AvgArray[8] + 3))
        {
            VoltAvgDropCnt1 ++;
        }
        if(PreVolts == 0)
```

```
        {
            PreVolts = Bat1AvgArray[8];
        }
        if( PreVolts > (Bat1AvgArray[8] ))
        {
            VoltDropCnt ++;
        }
        if( (Bat1TimeFastInterval % 1200) == 0)
        {
            if(PreVolts1 == 0)
            {
                PreVolts1 = Bat1AvgArray[8];
            }
            if( PreVolts1 > (Bat1AvgArray[8] +1) )
            {
                VoltDropCnt1 ++;
            }
            PreVolts1 = Bat1AvgArray[8];
        }
        PreVolts = Bat1AvgArray[8];
        Bat1VoltChkIntv = 0;            // recounter
    }
    if((VoltAvgDropCnt >= 10) || (VoltDropCnt >= 10) || (VoltAvgDropCnt1 >= 50) ||(VoltDropCnt1 >= 1))
    {
        Bat1State = BATTERY_SUP_CHARGING;
        VoltDropCnt = 0;
    }
    /* dT/dt check */
    DT_Dt_Detect();

    if(Bat1TimeFastInterval >= MAX_FAST_INTEVEL)
    {
        Bat1TimeFastInterval = 0;
        Bat1TimeFastCounter ++;
    }
    if( (Bat1TimeFastCounter >= MAX_FAST_COUNTER) ||            //fast charging time control
        (Bat1State == BATTERY_SUP_CHARGING)    )
    {
        Bat1State = BATTERY_SUP_CHARGING;          //exceed fast charge limit,then enter supplymentary charge
    } else {
        Battery_Fast_Charge();
        if (Bat1State == NO_BATTERY)
            delay(2000);
        Bat1State = BATTERY_FAST_CHARGING;
    }
}
/*
**Supplementary charge process
*/
void Sup_Charge(void)
{
    /* dT/dt check */
    DT_Dt_Detect();

    if(Bat1TimeSupInterval >= MAX_SUP_INTEVEL)
    {
        Bat1TimeSupInterval = 0;
        Bat1TimeSupCounter ++;
    }
    if ((Bat1TimeSupCounter >= MAX_SUP_COUNTER)) //Supplementary charging time control
    {
        Bat1State = BATTERY_TRICKLE_CHARGING;
    } else {
        Bat1State = BATTERY_SUP_CHARGING;
        Battery_Sup_Charge();
    }
}
```

```
/*
* @file name     operation.c
* @description  Battery charger operation of each mode
* @author          Li Baoke(86-571-86726288 EXT.8103, baoke.li@samsung.com)
* @version        Preliminary 0.0
* @history    History type - NEW/MODify/ADD/DELete
*                      |-------------------------------------------------------------------
*                      |ver type when            who                                    what
*                      |---+---+----------+-------------------------+----------------------
*                      |0.0 NEW 2008-03-06    Li Baoke                            Creation
*/
#include "Globle_Define.h"
#include "Operation.h"
/*******************************************************************************
****************************Charge operation functions********************
*******************************************************************************/
/*
** Pre charing function
  */
void Battery_Pre_Charge(void)
{
	if (ChargingCurrent <= PRE_CHG_CURRENT_LMT)
	{
		PWMWidth++;
		if (PWMWidth >= PRE_PWM_MAX)
			PWMWidth = PRE_PWM_MAX;
		PWM_Operation(PWMWidth);
	}else {
		PWMWidth--;
		if (PWMWidth <= PRE_PWM_MIN)
			PWMWidth = PRE_PWM_MIN;
		PWM_Operation(PWMWidth);
	}
}
/*
** fast charing function
  */
void Battery_Fast_Charge(void)
{
	if (ChargingCurrent <= FAST_CHG_CURRENT_LMT)
	{
		PWMWidth++;
		if (PWMWidth >= FAST_PWM_MAX)
			PWMWidth = FAST_PWM_MAX;
		PWM_Operation(PWMWidth);
	}else {
		PWMWidth--;
		if (PWMWidth <= FAST_PWM_MIN)
			PWMWidth = FAST_PWM_MIN;
		PWM_Operation(PWMWidth);
	}
}
/*
** supplymentary charging function
  */
void Battery_Sup_Charge(void)
{
	if (ChargingCurrent <= SUP_CHG_CURRENT_LMT)
	{
		PWMWidth++;
		if (PWMWidth >= SUP_PWM_MAX)
			PWMWidth = SUP_PWM_MAX;
		PWM_Operation(PWMWidth);
	}else {
		PWMWidth--;
		if (PWMWidth <= SUP_PWM_MIN)
			PWMWidth = SUP_PWM_MIN;
```

```
        PWM_Operation(PWMWidth);
    }
}
/*
** trickle charing function
 */
void Battery_TRK_Charge(void)
{
    if (ChargingCurrent <= TRK_CHG_CURRENT_LMT)
    {
        PWMWidth++;
        if (PWMWidth >= TRK_PWM_MAX)
            PWMWidth = TRK_PWM_MAX;
        PWM_Operation(PWMWidth);
    }else {
        PWMWidth--;
        if (PWMWidth <= TRK_PWM_MIN)
            PWMWidth = TRK_PWM_MIN;
        PWM_Operation(PWMWidth);
    }
}
/*******************************************************************************
****************************   PWM operation functions*************************
*******************************************************************************/
void PWM_Duty_Set(unsigned char dutywidth)          //set PWM dutywidth
{
    PWMDATA = dutywidth <<2;                         //set PWMDATA.5-2 = dutywidth
}
void PWM_Exten_Set(unsigned char ext)           //set PWM extension bit
{
    PWMDATA |= ext;                             //set PWMDATA.1-0
}
void PWM_Start_Run(void)                        //PWM start counter
{
    PWMCON |= 0x04;
}
void PWM_Stop_counter(void)                     //PWM stop counter
{
    PWMCON &= 0xFB;                              // &11111011B (bit2 = 0);
}
void PWM_Enable_Interrupt(void)                 //PWM interrpt enable
{
    PWMCON |= 0x02;
}
void PWM_Clock_Select_64(void)
{
    PWMCON &= 0x3F;                              // &00111111B (bit7.-6 = 00)
}
void PWM_Clock_Select_8(void)
{
    PWMCON &= 0x7F;                              // &01111111B (bit7 = 0)
    PWMCON |= 0x40;                              // |01000000B (bit6 = 1)
}
void PWM_Clock_Select_2(void)
{
    PWMCON &= 0xBF;                              // &10111111B (bit6 = 0)
    PWMCON |= 0x80;                              // |10000000B (bit7 = 1)
}
void PWM_Clock_Select_1(void)
{
    PWMCON |= 0xC0;                              // |11000000B (bit6 = 1)
}
void PWM_Operation(unsigned char width)
{
    unsigned char Temp, Temp2;
    //set P0.6 as PWM output:
    P0CONH   &= 0xDF;                            //&11011111B (bit5 = 0)
```

```
    P0CONH    |= 0x10;                                  //|00010000B (bit4 = 1)
    Temp = width / 4;
    Temp2 = (Temp & 0x3F)<<2;
    //PWM_Duty_Set(Temp);
    Temp = width % 4;
    Temp2 = Temp2 | (Temp & 0x03);
    //PWM_Exten_Set(Temp);
    PWMDATA = Temp2;
    PWMRunFlag = CHARGING_RUN;                          //PWM state: run
    PWM_Start_Run();
}
void PWM_Stop(void)
{
    PWM_Stop_counter();
    //set P0.6 as output :
    P0CONH    &= 0xCF;                                  //&11101111B (bit5 = 0)
    P0CONH    |= 0x20;                                  //|00100000B (bit4 = 1)
    P0_bit.b6 = 0;                                      //output low to stop charging
    PWMRunFlag = 2;                                     //PWM state: stop run

}
/*******************************************************************************
****************************    battery state Display ***********************
*******************************************************************************/
void Show_BAT1_State(unsigned char State_Flag)    //battery 1 state show
{
    if( Bat1State == NO_BATTERY)                        //no battery , Red LED blink
    {
        P2_bit.b2 = 0;                                  //Led_Green_1 turn off,
        if( TOMatchCounter == 30)
        {
            TOMatchCounter = 0;
            P2_bit.b1 = ~P2_bit.b1;                     //Led_Red_1 blinking slowly
        }
    }else if(Bat1State == BATTERY_TYPE_ERROR)           //battery type wrong , Red LED blink
    {
        P2_bit.b2 = 0;                                  //Led_Green_1 turn off,
        if( TOMatchCounter == 4)
        {
            TOMatchCounter = 0;
            P2_bit.b1 = ~P2_bit.b1;                     //Led_Red_1 blinking quickly
        }
    }else if(Bat1State == BATTERY_PRE_CHARGING)         //battery precharg, Green LED blink
    {
        P2_bit.b1 = 0;                                  //LED_Red_1 turn off
        if(TOMatchCounter == 18)
        {
            TOMatchCounter = 0;
            P2_bit.b2 = ~P2_bit.b2;                     //Led_Green_1 blinking slowly
        }
    }else if(Bat1State == BATTERY_FAST_CHARGING)        //battery fast charging , Green LED blink
    {
        P2_bit.b1 = 0;                                  //LED_Red_1 turn off
        if(TOMatchCounter == 2)
        {
            TOMatchCounter = 0;
            P2_bit.b2 = ~P2_bit.b2;                     //Led_Green_1 blinking quickly
        }
    }else if (Bat1State == BATTERY_SUP_CHARGING)        //battery supplementary charging,Green LED blink
    {
        P2_bit.b1 = 0;                                  //LED_Red_1 turn off
        if(TOMatchCounter == 10)
        {
            TOMatchCounter = 0;
            P2_bit.b2 = ~P2_bit.b2;                     //Led_Green_1 blinking normally
        }
    }else if (Bat1State == BATTERY_TRICKLE_CHARGING)    //battery trickle charging
```

```
    {
        P2_bit.b1 = 0;                                          //LED_Red_1 turn off
        if(TOMatchCounter == 26)
        {
            TOMatchCounter = 0;
            P2_bit.b2 = ~P2_bit.b2;                             //Led_Green_1 blinking very slowly
        }
    }else if (Bat1State == BATTERY_CHARGING_END)     //battery charging end. Red LED blink (same with no battery)
    {
        P2_bit.b2 = 0;                                          //Led_Green_1 turn off,
        if( TOMatchCounter == 30)
        {
            TOMatchCounter = 0;
            P2_bit.b1 = ~P2_bit.b1;                             //Led_Red_1 blinking slowly
        }
    }
}
```

```
/*
* @file name     Monitor.c
* @description  measurement functions and system abormal state protect
* @author       Li Baoke(86-571-86726288 EXT.8103, baoke.li@samsung.com)
* @version          Preliminary 0.0
* @history    History type - NEW/MODify/ADD/DELete
*             |-------------------------------------------------------------------
*                   |ver type when           who                                  what
*                   |---+---+----------+-------------------------+----------------------
*                   |0.0 NEW 2008-03-06    Li Baoke                           Creation
*/

/*******************************************************************************
***************************    I N C L U D E S    *****************************
*******************************************************************************/
#include "Globle_Define.h"
#include "monitor.h"
#include "Operation.h"
/*******************************************************************************
**************************** parameter measurement    *******************
*******************************************************************************/
/* battery 1 voltage convert (amplifier output) */
unsigned int BAT1_V_Monitor(void)
{
        unsigned int ADC_Result = 0;                    //store convert result
        __disable_interrupt();                          //disable interrupt
        ADCON = 0x34;                                        //00110100b
                                                          //ADC channel 3(P0.3, A_BAT1)
        ADC_Start_Convert();                              //Start convert
        while(ADCON_bit.EOC == 0);
        ADC_Result = ADDATAH;                                //load ADDATAH
        ADC_Result = ((ADC_Result<<2) & 0x03FC) | (ADDATAL & 0x03);            //get convert result
        __enable_interrupt();                            //enable interrupt

     return ADC_Result;
}
/* charging current convert (amplifier output) */
unsigned int Charging_Current_Monitor(void)
{
        unsigned int ADC_Result = 0;                    //store convert result
        __disable_interrupt();                          //disable interrupt
        ADCON = 0x84;                                        //10000100b
                                                          //ADC channel 8(P2.6, charging current)

        ADC_Start_Convert();                              //Start convert

        while(ADCON_bit.EOC == 0)
          {
                __no_operation();
          }
        ADC_Result = ADDATAH;                                //load ADDATAH
        ADC_Result = ((ADC_Result<<2) & 0x03FC) | (ADDATAL & 0x03);            //get convert result
        __enable_interrupt();                            //enable interrupt

     return ADC_Result;
}
/* battery 2 voltage convert (amplifier output) */
unsigned int BAT2_V_Monitor(void)
{
        unsigned int ADC_Result = 0;                    //store convert result
        __disable_interrupt();                          //disable interrupt
        ADCON = 0x44;                                        //01000100b
                                                          //ADC channel 4(P0.4, A_BAT2)
        ADC_Start_Convert();                              //Start convert
        while(ADCON_bit.EOC == 0);

        ADC_Result = ADDATAH;                                //load ADDATAH
```

```
        ADC_Result = ((ADC_Result<<2) & 0x03FC) | (ADDATAL & 0x03);          //get convert result
        __enable_interrupt();                                //enable interrupt

    return ADC_Result;
}
/* battery 1 temp. convert (amplifier output) */
unsigned int BAT1_Temp_Monitor(void)
{
        unsigned int ADC_Result = 0;                    //store convert result
        __disable_interrupt();                          //disable interrupt
        ADCON = 0x14;                                       //01000100b
                                                          //ADC channel 1(P0.1, A_Temp_BT1)
        ADC_Start_Convert();                              //Start convert
        while(ADCON_bit.EOC == 0);

        ADC_Result = ADDATAH;                               //load ADDATAH
        ADC_Result = ((ADC_Result<<2) & 0x03FC) | (ADDATAL & 0x03);          //get convert result
        __enable_interrupt();                                //enable interrupt

    return ADC_Result;
}
/* battery 2 temp. convert (amplifier output) */
unsigned int BAT2_Temp_Monitor(void)
{
        unsigned int ADC_Result = 0;                    //store convert result
        __disable_interrupt();                          //disable interrupt
        ADCON = 0x74;                                       //01110100b
                                                          //ADC channel 1(P0.1, A_Temp_BT1)
        ADC_Start_Convert();                              //Start convert
        while(ADCON_bit.EOC == 0);

        ADC_Result = ADDATAH;                               //load ADDATAH
        ADC_Result = ((ADC_Result<<2) & 0x03FC) | (ADDATAL & 0x03);          //get convert result
        __enable_interrupt();                                //enable interrupt

    return ADC_Result;
}
/*******************************************************************************
***************************   charge condition detector*******************
*******************************************************************************/
void Max_Temp_Detect()
{
      Bat1TempADC = BAT1_Temp_Monitor();                         //get temp signal convert result
      Bat1Temp = ( 43676 - (60 * Bat1TempADC) ) / (1024 - Bat1TempADC);     //temp calculate forum
      if (Bat1Temp >= 35)                                        //temperature control: if Temp > 45C, stop fast or supplementary charing and enter trickle charging
      {
            Bat1State = BATTERY_TRICKLE_CHARGING;
      }
}
void DT_Dt_Detect()
{
      if(Bat1TempChkIntv >= 2400)                      //dT / dt : temperature control:
      {
          if(Bat1Temp > (Bat1PreTemp +1 ) )
          {
              Bat1State = BATTERY_TRICKLE_CHARGING;
          }
          Bat1TempChkIntv = 0;
          Bat1PreTemp = Bat1Temp;
      }
}
void Max_ChargeTime_Detect()
{
      if(Bat1TimeTotalInterval >= MAX_TOTOL_INTEVEL)
      {
        Bat1TimeTotalInterval = 0;
```

```
        Bat1TimeTotalCounter ++;
    }
    if(Bat1TimeTotalCounter >= MAX_TOTOL_COUNTER)
    {
        Bat1State = BATTERY_TRICKLE_CHARGING;        //if exceed charge timing limitation, enter trickle charge
    }
}
/******************************************************************************
****************************    ADC operation    ***************************
******************************************************************************/
void ADC_Start_Convert(void)
{
    ADCON |= 0x01;                                  // |00000001B (bit0 = 1)
}
```

```
/**
* @file name    Global_Define.h
* @description  global variables and definitions.
* @author    Li Baoke(86-571-86726288 EXT.8103, baoke.li@samsung.com)
* @version       Preliminary 0.0
* @historyHistory type - NEW/MODify/ADD/DELete
*                    |-----------------------------------------------------------------
*                    |ver type when           who                                  what
*                    |---+---+----------+-------------------------+----------------------
*                    |0.0 NEW 2008-03-06    Li Baoke                          Creation
*                      -----------------------------------------------------------------
*/
#ifndef __GLOBLE_DEFINE_H
#define __GLOBLE_DEFINE_H
/* Header file including union declaration of registers. */
#include "ioS3C9454.h"
/* This header file contains some intrinsic functions. */
#include "intrinsics.h"
/*******************************************************************************
***************************   D E C L A R A T I O N   **********************
*******************************************************************************/
void Sys_init();
void System_Clear();
void delay(unsigned int nLoop_CNT);
/*******************************************************************************
***************************Charge Status define    *******************************
*******************************************************************************/
#define NO_BATTERY                                     0                         //no battery insert or battery inversed
#define BATTERY_TYPE_ERROR                     1                         //battery type not correct
#define BATTERY_PRE_CHARGING                 2                         //battery in pre-charging
#define BATTERY_FAST_CHARGING               3                         //battery in fast charging
#define BATTERY_SUP_CHARGING                 4                         //battery charging finished
#define BATTERY_TRICKLE_CHARGING           5                         //battery charging finished
#define BATTERY_CHARGING_END                 6                         //battery charging finished
/*******************************************************************************
***************************System paremeter define    ***************************
*******************************************************************************/
/*-----------------------------------------------------------------
*                         ADC convert parameters
*-----------------------------------------------------------------*/
#define CURRENT_AMP_GAIN                     46                              //current monitor amplifier gain.
#define CURRENT_AMP_COMPENSATE       14                            //compensate for current ADC convert result.
/*-----------------------------------------------------------------
*                         charging state change condition
*-----------------------------------------------------------------
*                 voltage = ((2.5Vbat+) + Vbat-))*1024/5
*     voltage < 50:                  no battery or battery was inversed
*     615 < voltage < 790:      fast charging / supplementary charging /trickle charing
*     voltage > 790:                battery type wrong or charing finished.
*     charing current < 15:    no battery or battery was token off
*/
#define BAT_DETECTOR_VOLTS           50                 //when battery voltage bigger than this value, means battery on.
#define BAT_PREEND_VOLTS              615               //when battery voltage bigger than this value, stop pre-charging.
#define BAT_MAX_VOLTS                   790               //if voltage bigger than this, stop charing
#define CHG_CURRENT_MIN               15               //if current less than this,means no battery or battery was toke off
#define START_CHECKING_VOLTAGE    790            //-dv/0dv start checking voltage
#define DV_STARTTEST_TIME_LMT      20000          //-dv/0dv start checking time limit
#define VOLTS_OF_INIT_DLY_1        630             //voltage value 1
#define INIT_CHECK_DLY_1             20000          //if voltage < 630, then delay time will be longer
#define VOLTS_OF_INIT_DLY_2        660             //voltage value 2
#define INIT_CHECK_DLY_2             10000          //if voltage < 660, then delay time will be short
#define VOLTS_OF_INIT_DLY_3        700             //voltage value 1
#define INIT_CHECK_DLY_3             0                //if voltage > 700, then no delay.
/*-----------------------------------------------------------------
*                         charging time control constents
*-----------------------------------------------------------------*/
#define CHARGING_RUN                             1          //charging runing
```

```
#define MAX_TOTOL_INTEVEL                    60000    //total charging time max interval
#define MAX_TOTOL_COUNTER                       6        //max total charging time counter
#define MAX_FAST_INTEVEL                     12000    //Fast charging time max interval
#define MAX_FAST_COUNTER                     6        //max Fast charging time counter
#define MAX_SUP_INTEVEL                      60000    //Sup. charging time max interval
#define MAX_SUP_COUNTER                      2        //max Sup. charging time counter
#define VOLT_CHK_INTV                        40       // -dv/0dv checking interval
/*-----------------------------------------------------------------
*                    charging PWM width control constents
*-----------------------------------------------------------------*/
#define PRE_PWM_MIN                          40       //pre-charge mimimun duty width
#define PRE_PWM_MAX                          60       //pre-charge maximum duty width
#define PRE_CHG_CURRENT_LMT                  30       //pre-charge constant current

#define FAST_PWM_MIN                         100      //Fast-charge mimimun duty width
#define FAST_PWM_MAX                         220      //Fast-charge maximum duty width
#define FAST_CHG_CURRENT_LMT                 340      //Fast-charge constant current

#define SUP_PWM_MIN                          60       //Sup.-charge mimimun duty width
#define SUP_PWM_MAX                          100      //Sup.-charge maximum duty width
#define SUP_CHG_CURRENT_LMT                  110      //Sup.-charge constant current

#define TRK_PWM_MIN                          24       //Trickle-charge mimimun duty width
#define TRK_PWM_MAX                          60       //Trickle-charge maximum duty width
#define TRK_CHG_CURRENT_LMT                  40       //Trickle-charge constant current
/*******************************************************************************
***************************Global Variable definition***************************
*******************************************************************************/
/*------------------- Battery related variables --------*/
extern unsigned char Bat1State;                    //battery 1 state
/* voltage monitor related */
extern unsigned int   Bat1Volts;                   //Battery 1 voltage ADC convert result;
extern unsigned int   Bat1VoltsArray[9];           //voltage sample array, the last one [8] is average value.
extern unsigned int   Bat1AvgArray[9];             //voltage average array, last one is average value.
/* temperature related */
extern unsigned int   Bat1TempADC;                 //battery 1 temperature ADC result
extern unsigned int   Bat1Temp;                    //battery 1 temperature
extern unsigned int   Bat1PreTemp;                 //battery 1 pre   temp data
extern unsigned int   Bat1TempChkIntv;             //battery temperature checking interval
/* time control related parameters (time = interval *   counter) */
extern unsigned int   Bat1TimeTotalInterval;       //battery 1 total charing time interval
extern unsigned char Bat1TimeTotalCounter;         //battery 1 total charing time counter
extern unsigned int   Bat1TimeFastInterval;        //battery 1 fast charing time interval
extern unsigned char Bat1TimeFastCounter;          //battery 1 fast charing time counter
extern unsigned int   Bat1TimeSupInterval;         //battery 1 Sup.charing time interval
extern unsigned char Bat1TimeSupCounter;           //battery 1 Sup. charing time counter
/*Termination condition check related variables*/
extern unsigned char Bat1VoltChkFlag;              //voltage checking flag:1-start check; 0- no check
extern unsigned int   Bat1VoltChkIntv;             // battery 1 voltage checking interval
extern unsigned int   PreVolts;                    //voltage check value: pre-tested value
extern unsigned int   PreVolts1;                   //voltage check value: pre-tested value 1
extern unsigned int   VoltDropCnt;                 //counter of voltage drop (Prevoltage - Vcheck >= 1)
extern unsigned char VoltDropCnt1;                 //counter of voltage drop every 1 minute.
extern unsigned int   Bat1AvgMax;                  //Max value of the average voltage
extern unsigned int   Bat1AvgMin;                  //Min value of the average voltage
extern unsigned int   VoltAvgDropCnt;              //counter of Vave <= Vmax-4
extern unsigned char VoltAvgDropCnt1;              //counter of Vave <= Vmax-3
extern unsigned int   DvStartTestTime;             //-dv check delay time
/*-----------------------------------------------------------------
*                    common variables
*-----------------------------------------------------------------*/
extern unsigned char PWMWidth;                     //Fast charging   pwm duty width
extern unsigned char PWMRunFlag;                   //PWM run or stop flag:0 == init; 1== strat run; 2== stop run
extern unsigned int   ChargingCurrent;             //Charging current convert result.
extern unsigned char TOMatchCounter;               // TO interrupt timing counter
#endif /* __GLOBLE_DEFINE_H */
```