AN13864

Smart Battery Charger by LPC865 with SMBus Interface

Rev. 1 — 14 August 2023 Application note

Document Information

Information	Content
Keywords	Smart charger, PWM, battery
Abstract	This application note describes a smart charger solution with LPC865, including hardware and software design as well as its performance data.

1 Introduction

The application of batteries is everywhere: smartphones, notebook computers, wearable devices, handheld electronic products, smart small appliances, and so on. Information about the state of batteries is important for the user, for example, the battery temperature, voltage, current, capacity, how much time is needed to charge the battery and to get it depleted. It is important to ensure the safety of battery charging and provide a smooth and controllable charging curve. The abovementioned requirements are expected to be realized by a smart charger. A smart charging solution implemented with LPC865 is recommended.

LPC86x is an Arm Cortex-M0+ based, low-cost 32-bit MCU family operating at CPU frequencies of up to 48 MHz. LPC86x supports up to 64 KB of flash memory and 8 KB of SRAM. The peripheral complement of the LPC86x includes a CRC engine, one I2C bus interface, one I3C-MIPI bus interface, up to three USARTs, up to two SPI interfaces, one multi-rate timer, self-wake-up timer, two FlexTimers, a DMA, one 12-bit ADC, one analog comparator, function-configurable I/O ports through a switch matrix, an input pattern match engine, and up to 54 general-purpose I/O pins.

2 Overview

This section gives general information on the process of interaction between LPC865 and the smart battery.

2.1 Function block diagram

Figure 1 below shows the process of interaction between LPC865 and the smart battery.

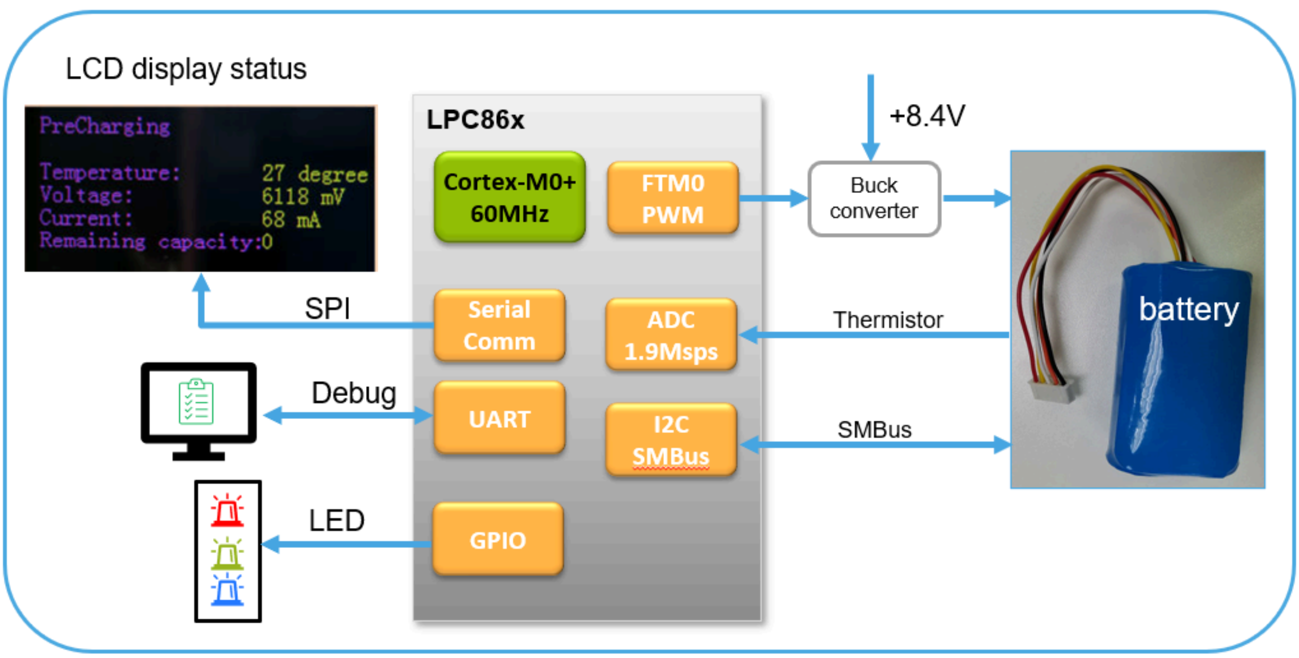

Figure 1. Function block diagram

LPC865 periodically communicates with the smart battery through the SMBUS bus to obtain battery information, and dynamically conditional PWM output to adjust charging voltage. The charging status is displayed through LEDs, and the charging information is displayed on the LCD screen.

2.2 System diagram

Figure 2 below shows the main system building blocks: NXP LPC865 battery charger board, +12 adapter, smart battery, LCD, emulator.

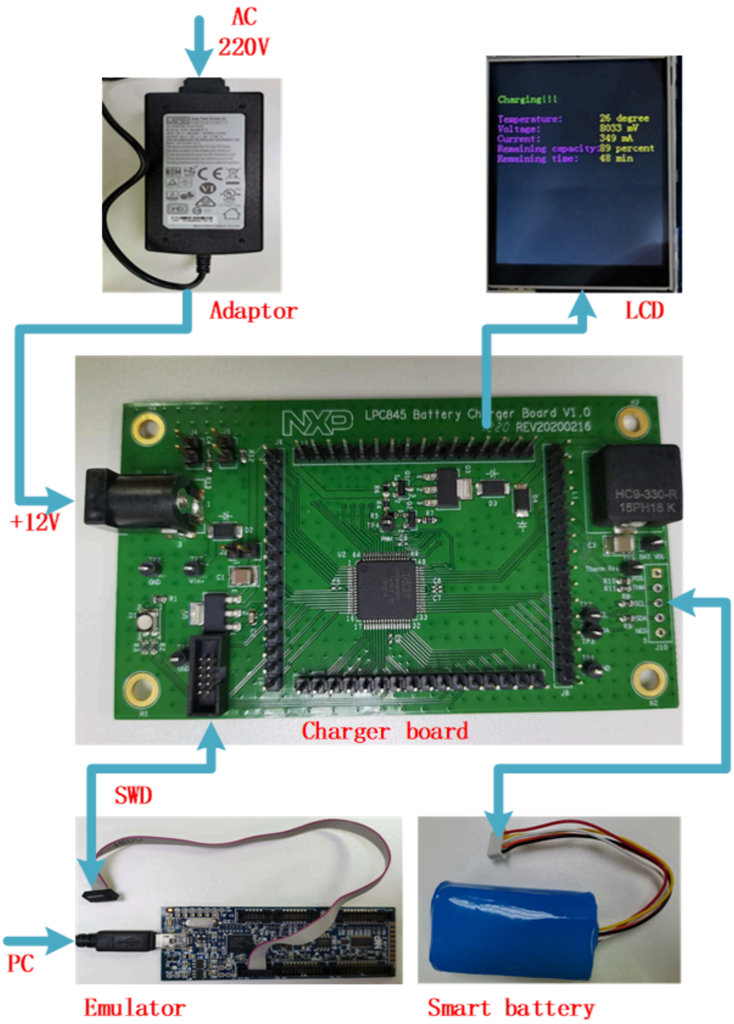

Figure 2. System diagram

Note: *The application uses the same hardware as the LPC845 smart battery charger solution. The chip LPC845 on the charger board is replaced by LPC865.*

The system can charge smart batteries with a nominal voltage of 8.4 V. During the charging process, the battery voltage, current, temperature, and charging time are displayed on the LCD screen in real time. In the process of developing the program, we use the LPC-link2 emulator to download it to the MCU. You can use any other emulators with the 1.27' 10-pin SWD connector such as J-Link, U-Link. The charging sequence can also be drawn in real time through the FreeMASTER software. During the charging process, it has gone through four stages: pre-charging, constant current charging, constant voltage charging, and charging fully.

3 Hardware

This section gives the overview of the hardware including charger board, LCD, smart battery, power adapter.

3.1 Hardware overview

Figure 3 shows a complete demonstration system including a charger board, LCD, smart battery, power adapter.

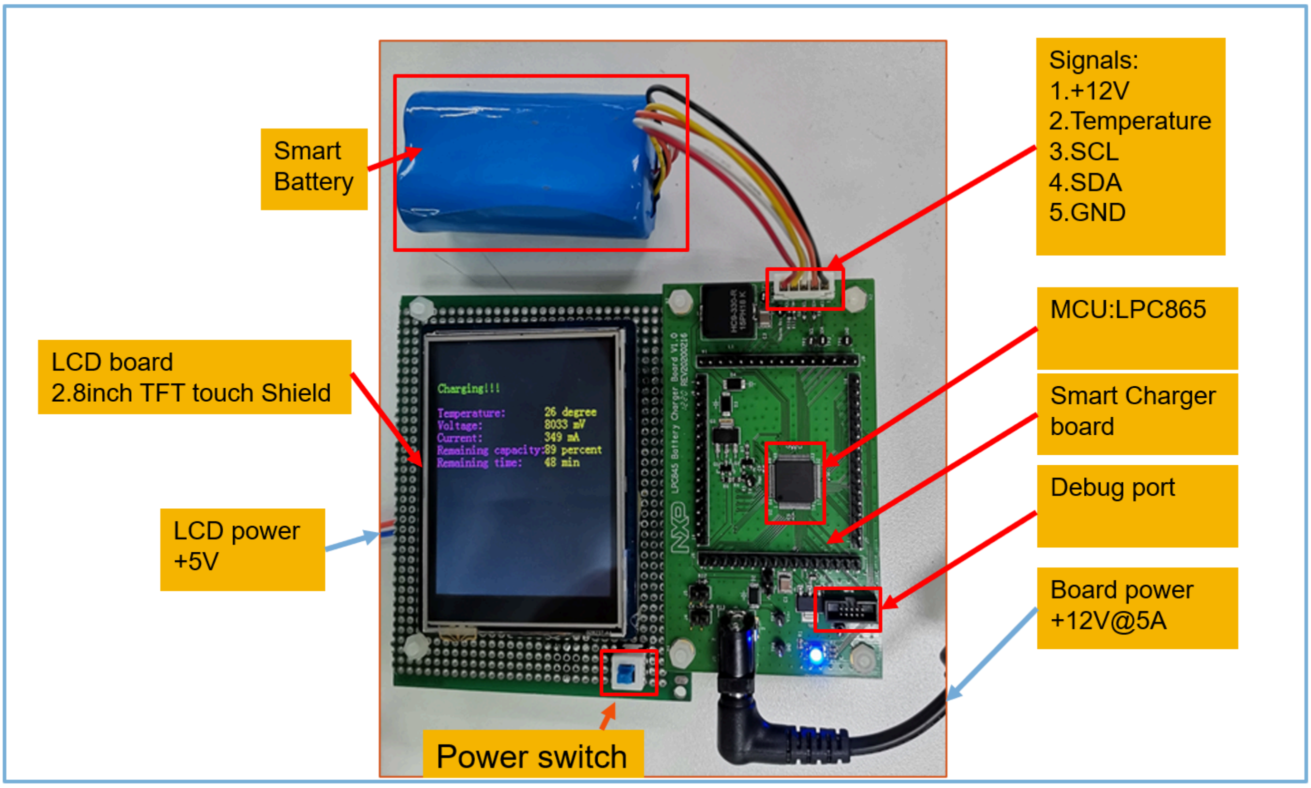

Figure 3. Hardware overview

Note: *The application uses the same hardware as the LPC845 smart battery charger solution. The chip LPC845 on the charger board is replaced by LPC865.*

Use a 12 V, 5 A adapter as a power supply that not only supplies power to the charger board but also provides voltage and current for the charging battery. The standard voltage of the smart battery is 8.4 V. The LCD is a 320x240 resolution TFT screen.

3.2 System connection

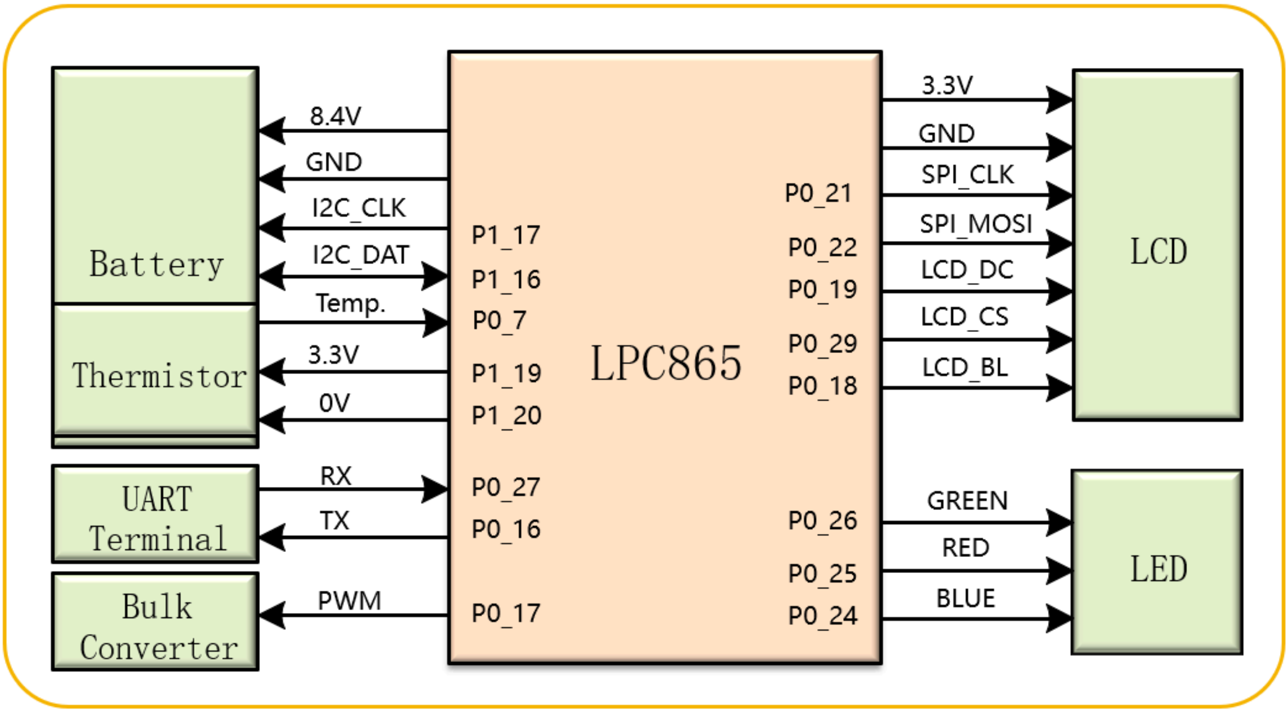

Figure 4. System connection

3.3 Charger board

Some modules in the charger board are represented on Figure 5 and listed below:

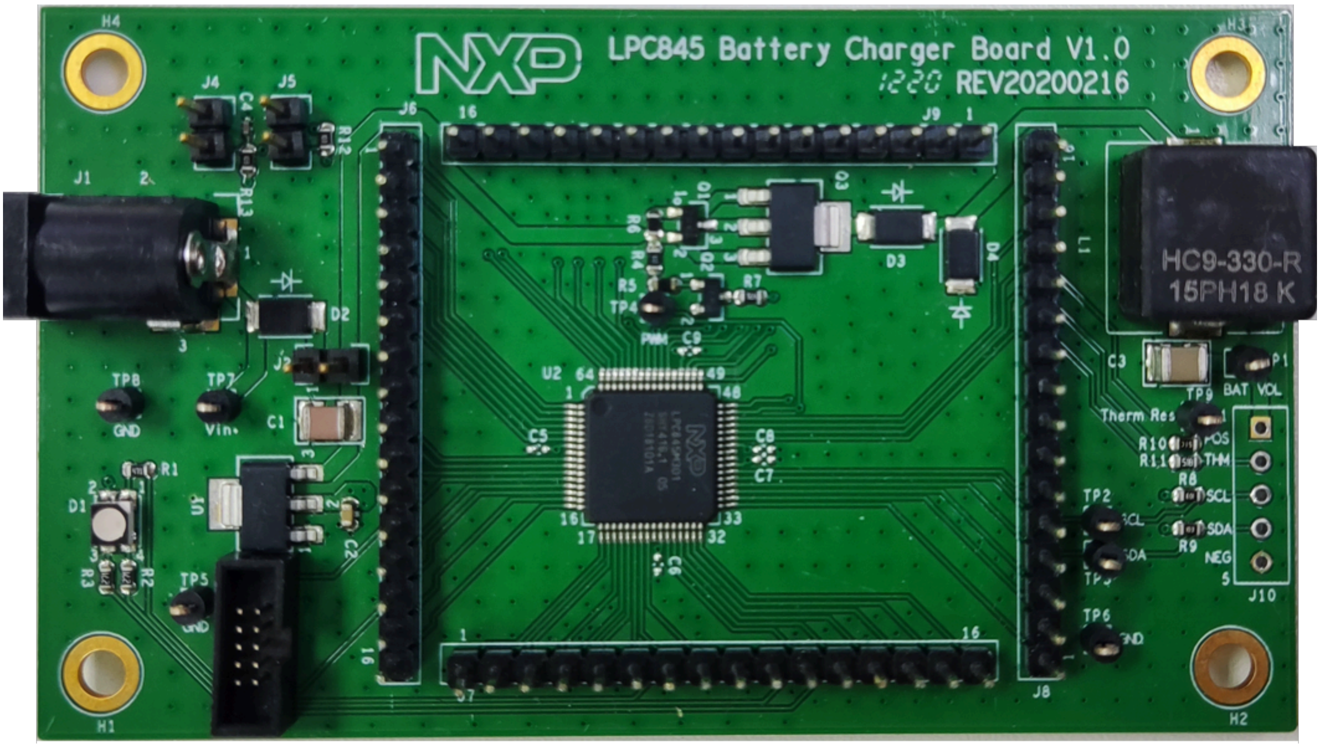

Figure 5. Charger board

Note: *The application uses the same hardware as the LPC845 smart battery charger solution. The chip LPC845 on the charger board is replaced by LPC865.*

- +12 V power socket is used to connect the 12 V power adapter.
- 64 pins of MCU are LED out providing signals such as SPI to drive the LCD screen.
- Smart battery interface is used for connecting the smart battery.
- Emulator interface is used for connecting the MCU debugger.
- Onboard LDO is used for providing 3.3 V power.
- Board bulk circuit is used for providing adjustable voltage to the battery.
- MCU controller - LPC865

3.4 Smart battery

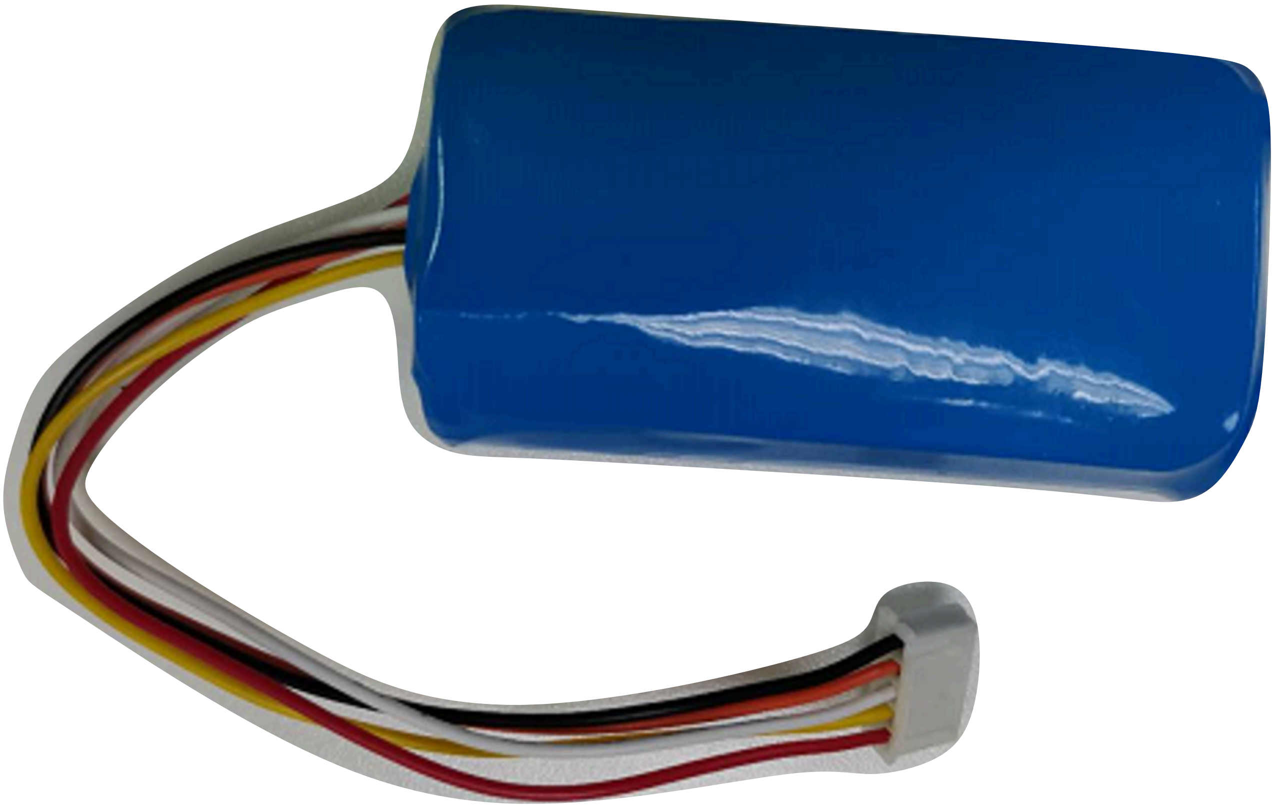

Figure 6. Smart battery

The battery pack is a smart battery that consists of two Li-ion batteries with a standard voltage of 8.4 V. The battery package contains a Li-ion battery pack manager chip named bq40z50. The battery pack supports Two-Wire SMBus v 1.1 with the interface through which the MCU can interact with it. In addition, the battery pack contains a PTC thermistor of 10 k value. The MCU can sample the voltage through the ADC to calculate its temperature.

3.5 LCD display board

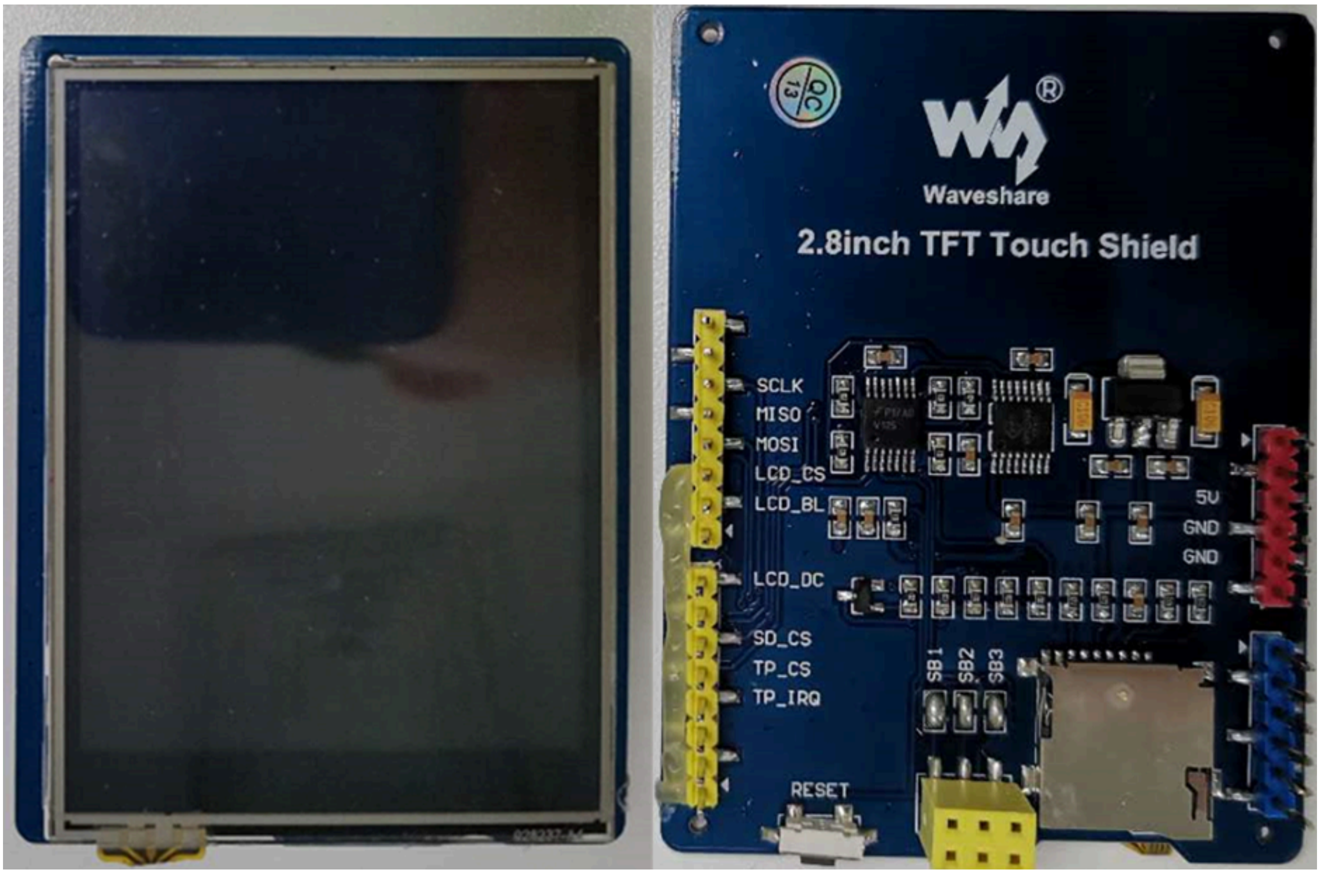

Figure 7. LCD module

The LCD board is from Waveshare, a 2.8-inch Touch LCD Shield for Arduino.

It has the following features:

- 2.8 inch TFT LCD resistive touch screen, 320x240 resolution
- Arduino standard interface, compatible with development boards such as Arduino UNO
- Controlled via SPI, several Arduino pins are used

3.6 Emulator

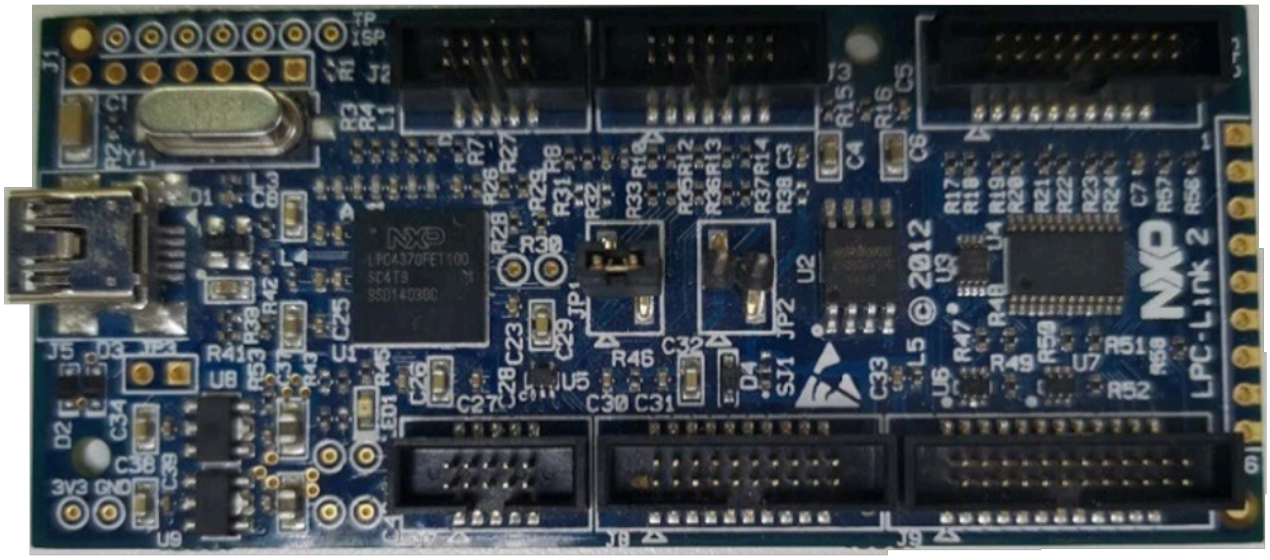

Figure 8. Emulator

LPC-Link 2, developed by NXP and Embedded Artists, is an extensible standalone debug probe that can be configured to support development tools and IDEs using various downloadable firmware images. It can be used as a standalone evaluation board for the NXP LPC4370 triple-core MCU.

3.7 AC adapter

The adapter converts 220 V AC to 12 V 5A DC power to provide power for the board and battery. Any other DC power supply can be used.

Figure 9. AC adapter

4 Software

This section gives detailed information on the software.

4.1 Source code

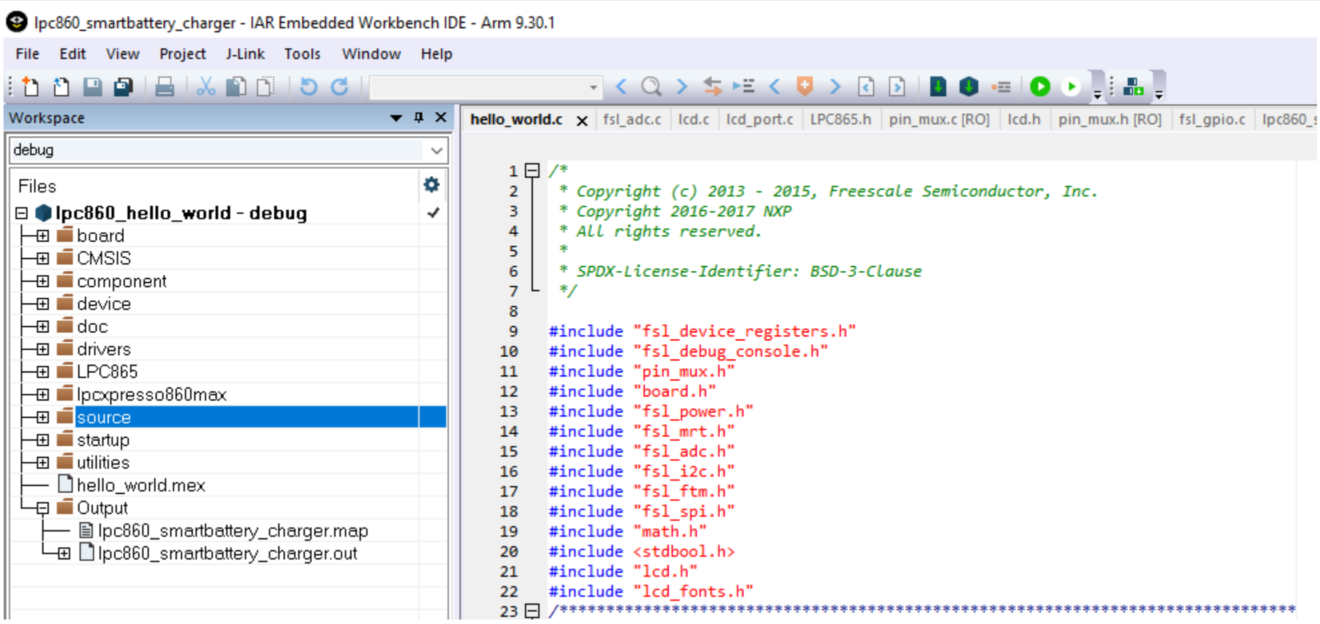

Figure 10. Code project

The code is developed on the IAR 9.30.1 IDE platform and uses the latest official SDK software development kit (SDK2.12.0). According to functional requirements, the peripherals including UART, SPI, I2C, FTM, Multimer, ADC are used.

Their functions are as below:

- UART is used to drive serial output.
- SPI is used to drive the LCD screen.
- I2C is responsible for communicating with the battery pack.
- FTM timer generates a PWM wave to control the bulk circuit.
- Multimer generates periodic interrupts for periodic sampling and modulation.
- ADC is used to collect the thermistor voltage and calculate the temperature.
- RAM 4.9 KB is used.
- Flash 26 KB is used.

4.2 Program flowchart

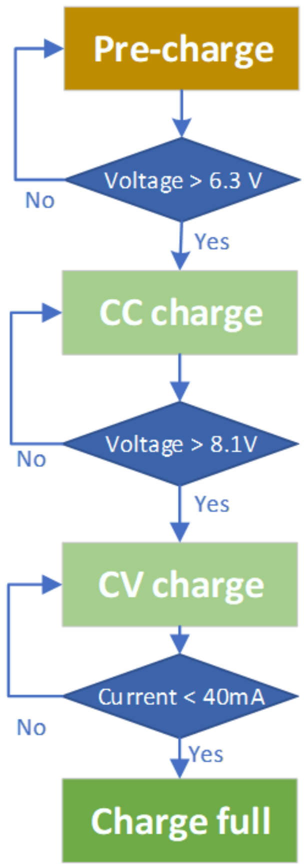

Figure 11. Program flowchart

The battery charging process mainly includes four stages: the pre-charging stage, constant current stage, constant voltage stage, and charging fully. The pre-charging phase is mainly to prevent damage to the battery due to a sinking large current when the voltage is too low. Only when it exceeds a certain voltage, it switches to constant current charging. Constant current charging is the main stage of the battery. The battery gets most of the energy in the constant current stage. At this stage, the battery voltage continues to rise. When a relatively high voltage is reached, it enters the constant voltage charging stage. The constant voltage stage is the end work. The battery is close to full capacity. At this stage, it is not appropriate to provide a large current, and instead provide stable voltage. The charging current continues to decrease. When the current reaches a small value, the battery can be considered fully charged.

4.3 PWM generation

PWM is used to adjust the output voltage of the bulk circuit and is generated by the FlexTimer. The Counter/timer block is designed to count cycles of the system clock or an externally supplied clock and can optionally generate a pulse width modulator via match outputs. In this application, the 60 MHz system clock is used as the clock source for FlexTimer. The frequency of PWM is 70 kHz. The minimum duty cycle number is 857. Different charging voltages and charging currents are produced by generating PWM waves with different duty cycles.

```
uint32_t FTM_Init_Config(void){

    ftm_config_t ftmInfo;
    ftm_chnl_pwm_signal_param_t ftmParam;
    ftm_pwm_level_select_t pwmLevel = FTM_PWM_ON_LEVEL;

    /* Fill in the FTM config struct with the default settings */
    FTM_GetDefaultConfig(&ftmInfo);
    /* Calculate the clock division based on the PWM frequency to be obtained */
    ftmInfo.prescale = FTM_CalculateCounterClkDiv(BOARD_FTM_BASEADDR, DEMO_PWM_FREQUENCY, FTM_SOURCE_CLOCK);
    /* Initialize FTM module */
    FTM_Init(BOARD_FTM_BASEADDR, &ftmInfo);

    /* Configure ftm params with frequency 24kHZ */
    ftmParam.chnlNumber            = BOARD_FTM_CHANNEL;
    ftmParam.level                 = pwmLevel;
    ftmParam.dutyCyclePercent      = 0;
    ftmParam.firstEdgeDelayPercent = 0U;
    ftmParam.enableComplementary   = false;
    ftmParam.enableDeadtime        = false;
    if (kStatus_Success !=
        FTM_SetupPwm(BOARD_FTM_BASEADDR, &ftmParam, 1U, kFTM_EdgeAlignedPwm, DEMO_PWM_FREQUENCY, FTM_SOURCE_CLOCK))
    {
        PRINTF("\r\nSetup PWM fail, please check the configuration parameters!\r\n");
        return -1;
    }

    FTM_StartTimer(BOARD_FTM_BASEADDR, kFTM_SystemClock);

    return kStatus_Success;
}
```

Figure 12. PWM generation code

4.4 SMbus communication

The LPC865 has one I2C module that supports SMBus. I2C is selected for the communication bus between the MCU and the battery. Read the value of the battery-related register from the given device address to get the charging information such as voltage, current, temperature.

```
uint16_t Information_Read_From_Battery(uint32_t address){

    status_t reVal  = kStatus_Fail;
    uint8_t deviceAddress;
    uint8_t  RxData[2];
    uint16_t ReturnData;
    i2c_master_transfer_t masterXfer = {0};

    deviceAddress = address;

    i2c_master_config_t masterConfig;

    /*
     * masterConfig.debugEnable = false;
     * masterConfig.ignoreAck = false;
     * masterConfig.pinConfig = kI2C_2PinOpenDrain;
     * masterConfig.baudRate_Bps = 100000U;
     * masterConfig.busIdleTimeout_ns = 0;
     * masterConfig.pinLowTimeout_ns = 0;
     * masterConfig.sdaGlitchFilterWidth_ns = 0;
     * masterConfig.sclGlitchFilterWidth_ns = 0;
     */
    I2C_MasterGetDefaultConfig(&masterConfig);

    /* Change the default baudrate configuration */
    masterConfig.baudRate_Bps = I2C_BAUDRATE;

    /* Initialize the I2C master peripheral */
    I2C_MasterInit(EXAMPLE_I2C_MASTER, &masterConfig, I2C_MASTER_CLOCK_FREQUENCY);

    /* Create the I2C handle for the non-blocking transfer */
    I2C_MasterTransferCreateHandle(EXAMPLE_I2C_MASTER, &g_m_handle, i2c_master_callback, NULL);
```

Figure 13. SMbus code

4.5 Temperature sample

GPIO outputs a high-level voltage to supply power for the thermistor and then, according to the voltage on the thermistor, to calculate the actual temperature of the smart battery. If the temperature changes, the resistor value of the thermistor changes as well. The voltage on the thermistor changes and this change can be captured by the ADC sampling on the MCU.

LPC865 has one 12-bit ADC with up to 12 input channels with sample rates of up to 1.2 Msamples/s. ADC channel 0 is responsible for sampling the thermistor voltage, and the corresponding temperature is calculated. If an over-temperature situation occurs, turn off the PWM signal immediately.

```
ADC_DoSoftwareTriggerConvSeqA(DEMO_ADC_BASE);
/* Wait for the converter to be done. */
while (!ADC_GetChannelConversionResult(DEMO_ADC_BASE, DEMO_ADC_SAMPLE_CHANNEL_NUMBER, &adcResultInfoStruct))
{
}
g_Voltage_NTC = (float)adcResultInfoStruct.result/4096*3.3;
Rt = (33200*g_Voltage_NTC)/(3.3 - g_Voltage_NTC);
g_Temperature_NTC = (1/(log(Rt/Rp)/Bx+(1/T2)))-273.15;
```

Figure 14. ADC code

4.6 Cycle interrupt

```
void MRT0_IRQHandler(void)
{
    /* Clear interrupt flag.*/
    MRT_ClearStatusFlags(MRT0, kMRT_Channel_0, kMRT_TimerInterruptFlag);
    g_mrtCountValue++;

    if((g_mrtCountValue% 2) == 0){ //every 500ms
      g_Voltage = Information_Read_From_Battery(Voltage_Add);
      g_Current = Information_Read_From_Battery(Current_Add);
```

Figure 15. Multi-Rate Timer code

LPC865 has a four-channel Multi-Rate Timer (MRT) for repetitive interrupt generation at up to four programmable fixed rates. Through the configuration, a polling task can be made every 100 ms. Check the charging status and make related PWM signal adjustments to make this behavior run through all stages of charging.

4.7 LCD driver

```
gpio_pin_config_t   gpioPinConfig;
// Select the function clock source for the master, SPI0 (slave will be clocked by the master's SCK)
SYSCON->FCLKSEL2[0] = 1;
SYSCON->SYSAHBCLKCTRL0 |= (1<<11);
SYSCON->PRESETCTRL0 &= ~(1<<11);
SYSCON->PRESETCTRL0 |= (1<<11);
EXAMPLE_SPI_MASTER->DIV = 5;
EXAMPLE_SPI_MASTER->CFG = SPI_CFG_ENABLE_MASK | SPI_CFG_MASTER_MASK |SPI_CFG_CPOL_MASK | SPI_CFG_CPHA_MASK  ;
EXAMPLE_SPI_MASTER->DLY = 0;
EXAMPLE_SPI_MASTER->TXCTL = SPI_TXCTL_EOT_MASK | SPI_TXCTL_LEN(7);
```

Figure 16. SPI code

There are two SPI modules on LPC865, and the 30 MHz clock can be used as the baud rate clock. Excluding delays introduced by the external device and PCB, the maximum supported bit rate for SPI master mode is 30 Mbit/s. Such speed can improve the refresh rate of the screen.

4.8 FreeMASTER

FreeMASTER is a user-friendly real-time debug monitor and data visualization tool that enables runtime configuration and tuning of embedded software applications. In the application note, the system information is displayed through FreeMASTER, and the setting of charging stages can be done in FreeMASTER.

The FreeMASTER install package can be obtained from the NXP website.

Display information:

- Charging voltage,
- Charging current,
- Remaining battery capacity,
- Remaining charging time.

Setting:

- Cross voltage from pre-charge to CC charge stage;
- Cross voltage from CC charge to CV charge stage;
- Cross current from CV charge to full charge stage.

The charging voltage, current and remaining capacity of the battery are shown on Figure 17:

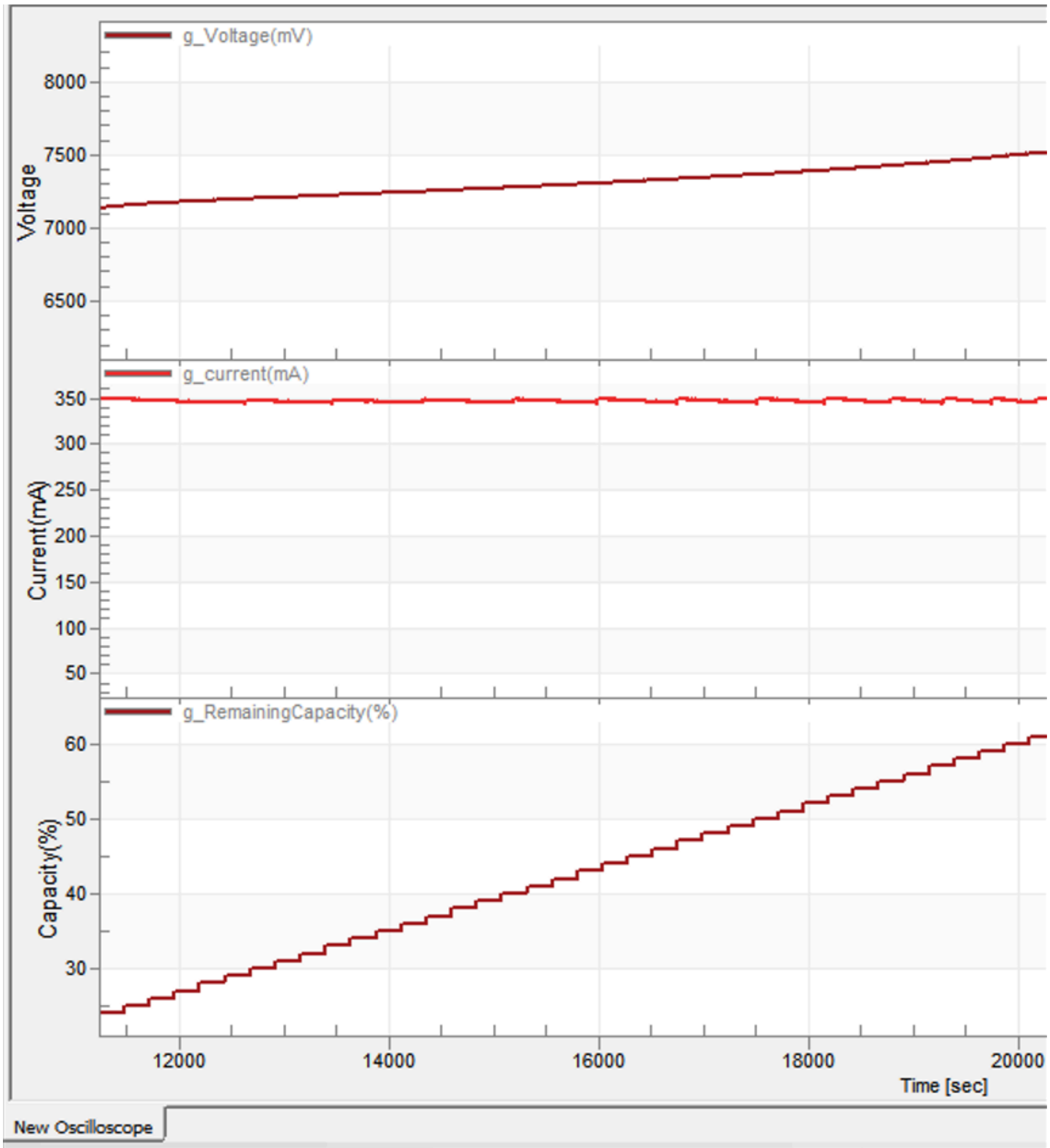

Figure 17. FreeMASTER monitor

Note: *The values of variables `g_PreChargeMaxVoltage, g_CCChargeMaxVoltage, g_CVChargeMinCurrent` can be modified by the user. The values of variables `g_Voltage, g_Current, g_RemainingCapacity` can only be read by the user.*

5 Step-by-step demo

1. Connect the adapter, LCD, emulator to the charger board.
2. Push down the power switch, comply, and download the code into MCU
3. Push up the power switch and connect the battery to the charger board
4. If the battery voltage is lower than `g_PreChargeMaxVoltage`, it enters the pre-charge mode. When the battery voltage crosses `g_PreChargeMaxVoltage`, it enters the constant current charging mode. The constant voltage charging mode starts when the battery voltage reaches `g_CCChargeMaxVoltage`. The voltage is maintained at `g_CCChargeMaxVoltage` until the current drops to `g_CVChargeMinCurrent`. Then charging is completed. The states of these four charging stages are shown on Figure 18:

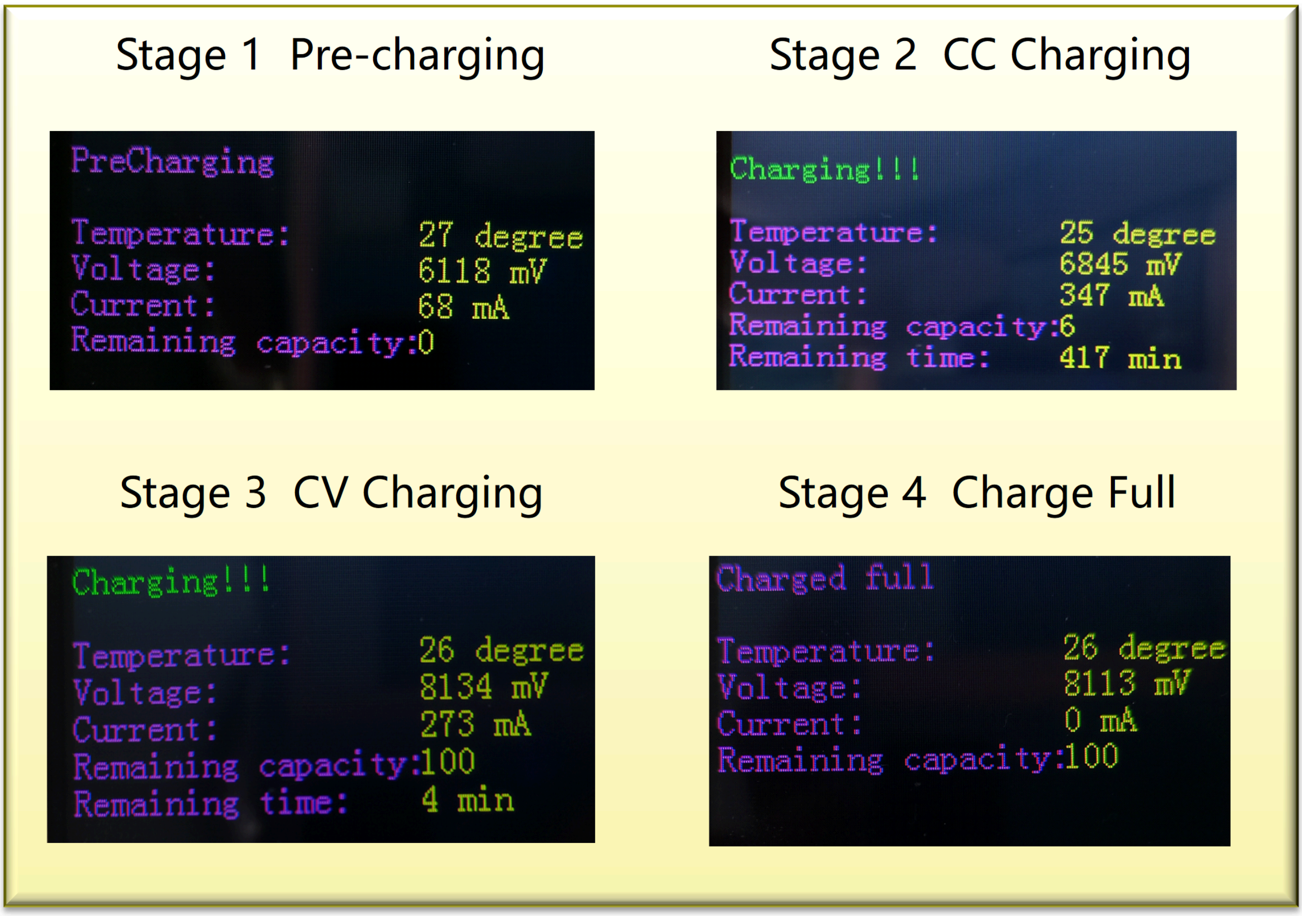

Figure 18. Four charging stages

6 Charging timing and specification

Figure 19. Charging timing

Figure 19 is the real charging voltage and current curve captured from FreeMASTER. During the entire charging process, the battery capacity ranges from 94 % to 100 %. From left to right, there are four stages including pre-charging, constant current charging, constant voltage charging, and charging completion. They are marked with four colors.

Specification:

1. The range of charging voltage: 6 V ~ 8.4 V
2. Charging voltage (CV charging mode): Set by `g_CCChargeMaxVoltage`, 8.15 V by default
3. The range of charging current: 0 mA ~ 385 mA
4. Charging current (CC charging mode): 350 mA
5. Charging time from empty to full: about 7 hours
6. Over charging voltage alert: 8.5 V
7. Over charging current alert: 500 mA
8. Over temperature alert: 50 degrees

6.1 Appendix A Schematic

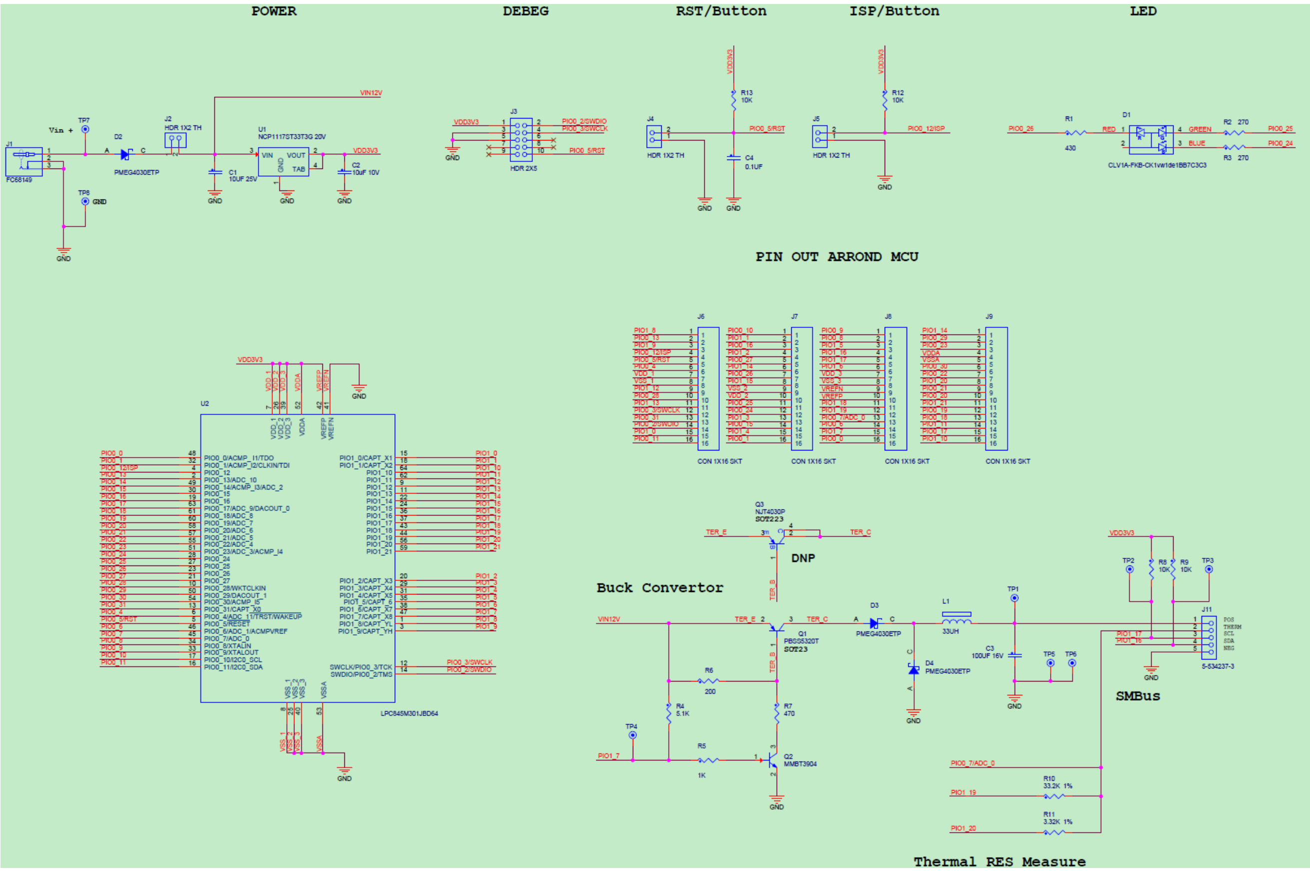

Figure 20. Schematic

Note: *The application uses the same hardware as the LPC845 smart battery charger solution. The chip LPC845 on the charger board is replaced by LPC865.*

7 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2023 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN

ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

8 Revision history

Table 1 summarizes the revisions to this document.

Table 1. Revision history

Revision number	Release date	Description
1	14 August 2023	Initial public release

9 Legal information

9.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

9.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Suitability for use in automotive applications — This NXP product has been qualified for use in automotive applications. If this product is used by customer in the development of, or for incorporation into, products or services (a) used in safety critical applications or (b) in which failure could lead to death, personal injury, or severe physical or environmental damage (such products and services hereinafter referred to as "Critical Applications"), then customer makes the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, safety, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. As such, customer assumes all risk related to use of any products in Critical Applications and NXP and its suppliers shall not be liable for any such use by customer. Accordingly, customer will indemnify and hold NXP harmless from any claims, liabilities, damages and associated costs and expenses (including attorneys' fees) that NXP may incur related to customer's incorporation of any product in a Critical Application.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. - NXP B.V. is not an operating company and it does not distribute or sell products.

9.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, μVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Contents

Date of release: 14 August 2023
Document identifier: AN13864