APPLICATION NOTE

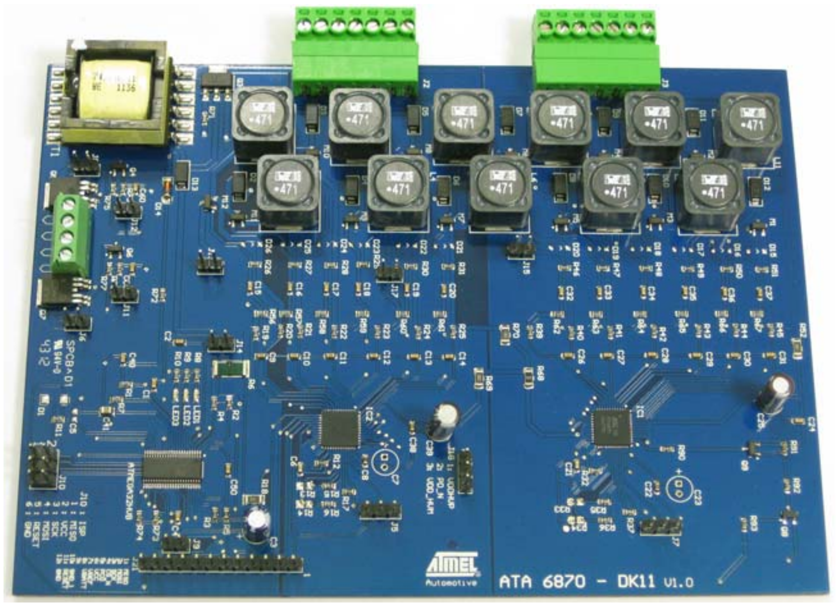
User Guide for Atmel ATA6870 and Atmel ATmega32HVB Evaluation Kit Hardware DK11

ATA6870-DK11

Features

- Evaluation of the Atmel® ATA6870
- Monitoring 12 battery cells
  - Monitoring:
    - Overvoltage (every cell)
    - Undervoltage (every cell)
    - Overheating
    - Overcurrent
- Open clamp detection
- 12-bit battery cell measurement
- 12-bit temperature measurement
- Controlling FET charge/discharge
- Status LEDs for easy evaluation
- Charge active balancing

Figure 1. Atmel ATA6870-DK11



1. Introduction

The Atmel® ATA6870-DK11 is a demonstration board for the Atmel ATA6870 and offers an easy way to start evaluation of battery applications in combination with the Atmel ATmega32HVB. The included software demonstrates implementation of a 12-cell battery management system, including the function of active balancing through inductors. The supplied code serves as an example of how to use the Atmel ATMega32HVB and Atmel ATA6870 together.

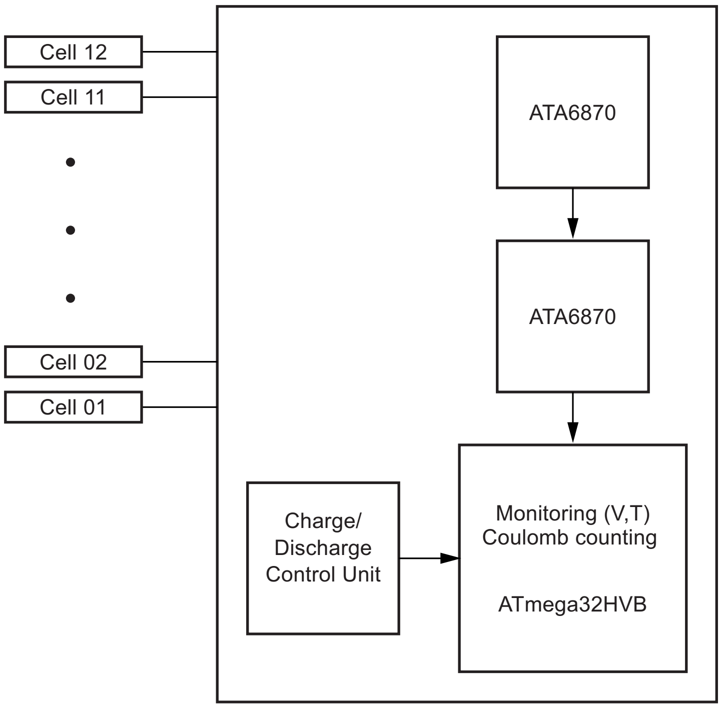
2. Safety Precautions When Using Li-Ion Batteries

Please observe the safety guidelines supplied with the batteries. If improperly used or defective, Li-ion and polymer batteries and packs may explode and cause a fire.

3. Demonstration Board

The Atmel ATA6870-DK11 is designed to enable easy evaluation of the control software for an MCU controlling multiple Atmel ATA6870 devices. The supplied sample code demonstrates a simple permanent running measurement of voltages and temperatures.

Figure 3-1. Board Concept


Cell 12
Cell 11
Cell 02
Cell 01
ATA6870
ATA6870
Charge/ Discharge Control Unit
Monitoring (V,T) Coulomb counting
ATmega32HVB

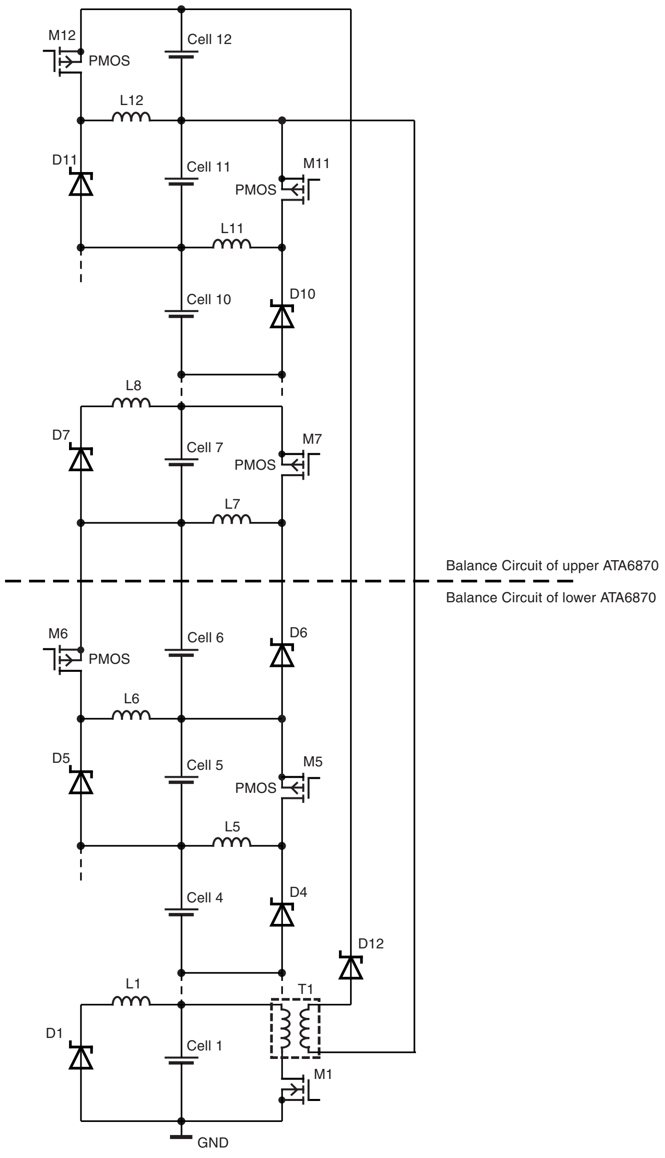
A very important characteristic of the ATA6870-DK11 is the ability to realize active balancing. Compared with the passive method, the active method is significantly more efficient with virtually lossless energy transfer between battery cells. There are usually two types of active balancing—with capacitors and with inductors. The capacitors are used for balancing currents lower than 50mA, while the inductors are for currents up to 1A or even more. On the ATA6870-DK11 the inductors with the value of 470uH are applied. Table 3-1 describes the achievable peak and average balancing currents with different inductors.

Table 3-1. **Balancing Currents Possible Using Different Inductors**

f (kHz)	Inductor Type	Inductance (µH)	Resistor of Inductor (mΩ)	Peak Balancing Current (mA)	Average Balancing Current (mA)
3	Toroidal	300	130	1240	470
3	Toroidal	470	135	790	280
3	SMD	220	380	2280	890
3	SMD	330	430	1620	610
3	SMD	470	560	1500	430

The structure is shown in Figure 3-2. It is easy to see that on the demo board the active balancing can only happen between the two neighboring cells and the balancing current flows from the higher cell to the lower cell, with the corresponding Mosfet switched on. However, if it is the lowest cell of a battery stack that needs to be discharged, a transformer concept applies and the energy is transferred from the lowest cell to the top cell in the battery stack. For more information about the active balancing, please see application note "Active Cell Balancing Methods for Li-Ion Battery Management ICs using the ATA6870."

Figure 3-2. Inductive Charge Balancing between Two Stacked ATA6870s



3.1 System Start

Follow these steps to launch the system.

3.1.1 Installing the Hardware

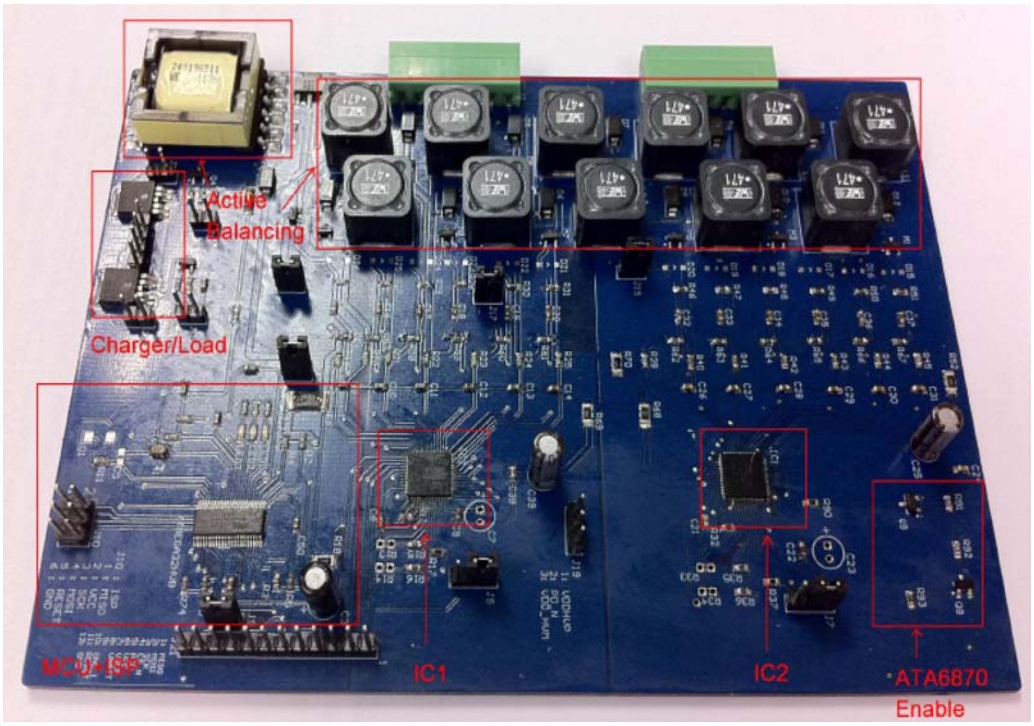
- Connect the load/charger to J1.
  - For demonstration purposes it is possible to use a resistor to simulate a load.
- Connect the 12-cell stack to the J2 and J3 screw connectors.
  - LED 1 indicates the enabled status of ATA6870-DK11 (controlled by the MCU software, see Table 4-1)
- In case of emulating cells such as a voltage divider, please apply sufficient voltage.

3.1.2 Number of Cells

It is possible to run the board with a reduced number of cells. The minimum voltage for each IC is 6.9V. Cell 1 and cell 6 have to be connected. The missing cells should be short-circuited to the upper cell potential of the module. For further information, see Section 7.3 "Reduced Number of Battery Cells Configuration" of the Atmel ATA6870 datasheet. For the voltage range, see Section 3.3 "Powering the Board" on page 6. If fewer than 6 cells are used per IC, the config.h file should be adjusted (CELLSIC# under General Setting). See Section 4.1 "Supplied Code" on page 7 for further information on how to configure the supplied software correctly.

3.2 The Demonstration Board

Figure 3-3. Evaluation Board with Two Stacked Atmel ATA6870 and Atmel ATMega32HVB



3.2.1 On-Board Features

The demonstration board includes the following items:

- Two Atmel® ATA6870 QFN 7mm × 7mm
- Atmel ATMega32HVB
- 12 external P-channel MOSFETs, 11 inductors, 13 diodes and one transformer for active balancing
- Connectors
  - ISP connector for programming/debugging the Atmel ATMega32HVB
  - Screw connectors for connecting up to 12 battery cells

Table 3-2. Connector Overview

Connector	Function
J1	Connector for charger/device to be powered
J2	Bottom battery stack (cells 1-6)
J3	Upper battery stack (cells 7-12)
J10	ISP connector

Table 3-3. Jumper Overview

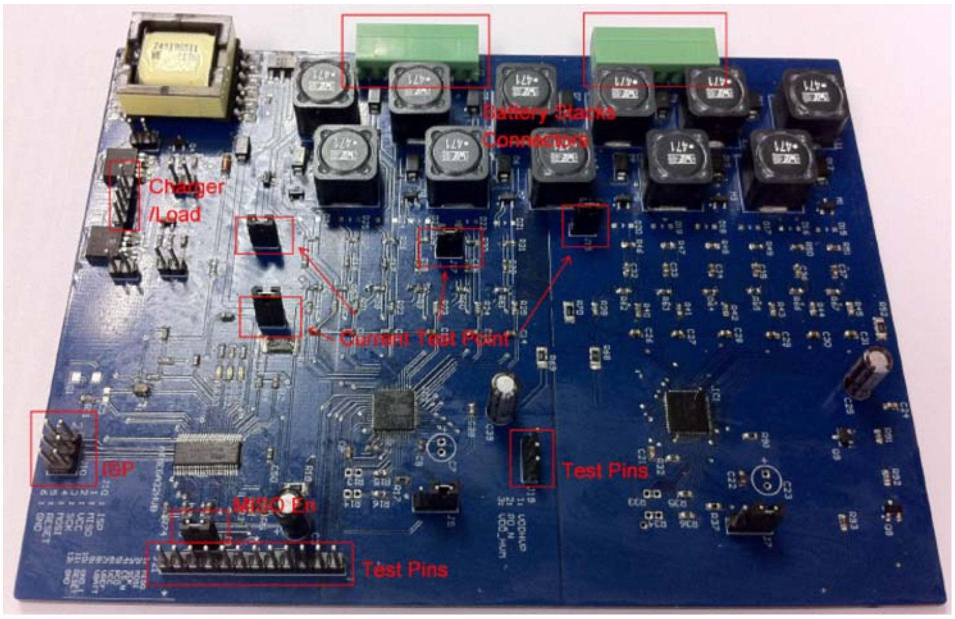
Jumper	Function
J9	Jumper to enable/disable MISO line of Atmel ATA6870 (see details below)
J13, J14, J15, J17	Current measurement for active balancing (should be closed for charge transfer)

J9 should never be set while the Atmel ATmega32HVB is being programmed or while it is entering debug mode. It can be mounted as soon as AVR Studio® prompts for additional SPI lines to be connected in debug mode or after the device has been correctly programmed.

Table 3-4. Test Point Overview

Test Point	Function
J19	IC2 work status
J21	MCU power and SPI status

Figure 3-4. Connectors, Jumpers, and Test Points



3.3 Powering the Board

3.3.1 Power Supply

The board supports supply voltages ranging from 13.8V (minimum 6.9V per Atmel ATA6870) to 60V. However, to run the board on voltages below 24V the ZDiode D14 needs to be replaced with a jumper to supply the Atmel® ATmega32HVB with sufficient voltage. If the jumper is mounted, be sure that stack voltage does not exceed 48V. The Atmel ATmega32HVB supports operating voltage from4V to 24V.

3.3.2 Emulating Cells

Battery cells can be emulated by connecting a voltage divider to the specified clamps. Section 3.1.1 “Installing the Hardware” on page 4 describes how to connect cells. The voltage limits for this setup are the same as for the real batteries. Section 3.3.1 “Power Supply” on page 6 specifies these limits.

4. Software Description: Monitoring Up to 12 Battery Cells

The supplied code is documented and easy to adjust for verifying the basic functions of the Atmel® ATA6870 and start BMS application development.

After the board is ready as described above, the microcontroller automatically starts a cyclic measurement of voltages, temperature and current. The status is indicated by the three LEDs on the board and the functions of the LEDs are programmable according to different requirements. LED1 is switched on at the beginning of the main routine and toggles with the active balancing. LED2 indicates that for some reason the charger/load MOSFETS have been disabled. The default software disables the FETs in case of these events:

- Overvoltage (at least one cell exceeds the upper default threshold of 4.2V)
- Undervoltage (at least one cell exceeds the lower default threshold of 2.5V)
- Overcurrent (the current through the shunt exceeds the default threshold of 80mA)
- Overheating (the temperature exceeds the upper threshold, default value is 80°C)
- Low temperature threshold (the default threshold is –20°C)

LED3 indicates whether the Atmel ATA6870 devices are turned on or not. An active LED indicates that the Atmel ATA6870 devices are enabled.

Table 4-1. LED Functions

LED	Function
LED1	ON indicates access in the main routine TOGGLE indicates that the active balancing is operating
LED2	ON indicates disabled MOSFETs for one of the reasons listed above
LED3	ON indicates active Atmel ATA6870

The ATA6870 system clock should be provided externally. The Atmel ATmega32HVB has no clock divider to provide an external clock slower than 1/2 CPU clock. The requirement for the Atmel ATA6870 is $f_{CLK} > 2 \times f_{SPI}$. As a result, the clock frequency of 1MHz is mandatory to provide a 500kHz clock for the ADCs of the Atmel ATA6870 and 250kHz for SPI.

4.1 Supplied Code

4.1.1 Config.h

This section refers to the config.h file provided in the demonstration source code. Only values in the user setting paragraph should be changed and other values should not be changed in the default hardware setup. For example, the variable CELLSIC should be set to match the status of the batteries actually connected.

```
------------- GENERAL SETTING--------------------------------
CELLSIC# Selecting which cells are used Bits 0-5 -> Cells 1-6
------------- TEMPERATURE SETTING----------------------------
RES_REF# Value of the mounted reference resistor (default: 3300)

T_TLS Temperature belonging to the first value in the lookup table (index 0, default: -20)

T_TLE Temperature belonging to the last value in the lookup table (default: 80)

T_TLSZ Temperature step size used in the lookup table (default: 1)

T_LOWERTHRESHOLD Lower temperature threshold

T_UPPERTHRESHOLD Upper temperature threshold
```

```
------------- COULOMBCOUNTER SETTING-------------------------
SHUNT_RESISTANCE Value of the shunt resistor in mOhm

RCC_CONVERSIONPERIOD The cycle times for the Regular Current Check
       0x00 - 256ms (default)
0x01 - 512ms
0x02 - 1s
0x11 - 2s

RCC_DIVIDEDSZ 0x01 to enable divided Voltage (Current) stepsize

RCC_CHARGETHRESHOLD Threshold for charging current, exceeding the threshold will turn off
the Mosfets

RCC_DISCHARGETHRESHOLD Threshold for discharging current, exceeding the threshold will turn
off the Mosfets.

Balancingthreshold Threshold for the active balancing.
```

4.2 Voltage Measurements

The standard software loop measures the voltage ADC value and the offset ADC value for every cell and checks for overvoltage and undervoltage once per cycle. Further information about acquiring voltages can be found in Section 7.5.1 of the Atmel® ATA6870 datasheet. The formula for calculating the voltage is:

$$\text{Voltage (Cell)} = 4\text{V} \times \left(\frac{V_{acq} - V_{offset}}{3031 - V_{offset}}\right)$$

4.3 Temperature Measurements

The default software only measures channel 1 of chip 1. The temperature sensors are based on a resistor divider using a standard resistor and an NTC resistor. This resistor divider is connected to the reference of the ADC for temperature measuring. Because the ADC shares the same reference value, the output of temperature measurement with ADC is ratio-metric. Further information can be found in Section 7.5.3 “Temperature Channel” of the Atmel ATA6870 datasheet.

For this application Atmel recommends using Res_Ref1 = 3.3kΩ and RES_NTC1 R25 = 4.7kΩ, B = 3500. The software supplied for this board uses these values as default. The function uses a lookup table to determine the temperature. This table has to be edited if an NTC other than the recommended one is used. The values in the lookup table range from –20°C (index 0) to +80°C (index 100). These values can be edited via the config.h file in the user settings section. More Information about this file can be found in Section 4.1 “Supplied Code” on page 7. The calculation of RES_NTC is carried out based on the formula provided in Section 7.5.3 of the Atmel ATA6870 datasheet:

$$\text{adc (out)} = 2048 \times \left(1 + \frac{\text{RES\_NTC(1)}}{(\text{RES\_NTC(1)} + \text{RES\_REF(1)})} \times \frac{8}{15} - \frac{8}{10}\right)$$

When using another NTC, the LookupADC.txt has to be edited to match the NTC used.

4.4 Gas Gauging

Gas gauging is a commonly used method for estimating how much capacity and runtime are left in the battery. By combining its very accurate VADC and CCAVC the Atmel ATmega32HVB can achieve high accuracy gas gauging. Here the voltage-based gas gauge is for the remaining capacity initialization, while the coulomb-counter-based voltage is to keep track of how much current is being used to charge and discharge the battery. Because the gas gauging calculation needs characterization data for the specific battery cell, the algorithm does not fall within the scope of this application note. See the other AVR® application notes for further details.

4.5 Overcurrent Protection

The current through the shunt is calculated by measured voltage drop. The limit can be set via the CADRDC/CADRCC register. The step size depends on the settings of the CADCSRC register and the shunt used. For further information about the limiting current see Section 19.4 "Regular Current Detection Operation" of the Atmel ATmega32HVB datasheet. The supplied software allows the feature to be tested by adjusting the values in the config.h file. More Information about this file can be found in Section 4.1 "Supplied Code" on page 7. Values/part of the code should only be changed if you are aware of possible consequences. The default implementation continuously measures the current and generates an interrupt if the entered thresholds are exceeded. The thresholds are defined in the config.h file. The thresholds are written to the registers in the CCinit function in the Atmel ATA6870_func.c file. Refer to the features of the Atmel ATmega32HVB in the coulomb counter section to learn more about the time the controller waits for the values to be written.

Table 4-2. C Code Example

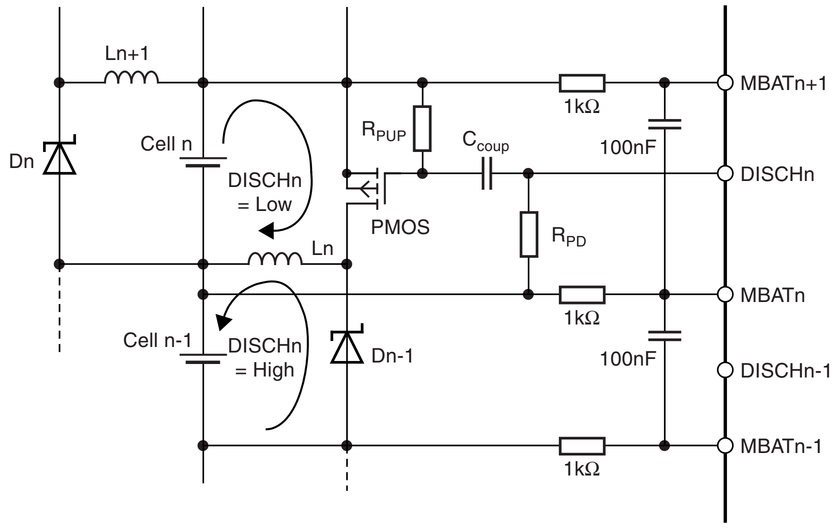
C Code Example

```
       CADRCC = RCC_CADRCC;                    // Charge Threshold
       while(CADCSRA & (1 << CADUB));        // Wait values to be written
       CADRDC = RDC_CADRDC;                    // Discharge Threshold
       while(CADCSRA & (1 << CADUB));        // Wait values to be written
```

4.6 Active Balancing

To realize the function of actively balancing the PMOS only needs to be switched on and off to charge and discharge the inductors (see Figure 4-1). The PWM signal would be an ideal switching signal on the DISCH pins to control the PMOS. In ATA6870 there are no PWM-generating modules implemented. Therefore, to simulate the function of the PWM signal in the demonstration software the continuous on and off commands can be delivered via the SPI communication from the ATmega32HVB to the ATA6870 devices.

Figure 4-1. Charging and Discharging Inductors



4.6.1 Single Balancing Mode

In the demo software two routines of the active balancing are implemented. ActiveBalance_SingleCell() is used to monitor current with a current probe by replacing jumpers J15, J17 and J13, J14 with a short cable. Under no condition is active balancing triggered in this routine and 5 PWMs are performed. The following code example is part of the ActiveBalance_SingleCell() routine, which performs a balancing operation between cell 5 and cell 4 on the IC2.

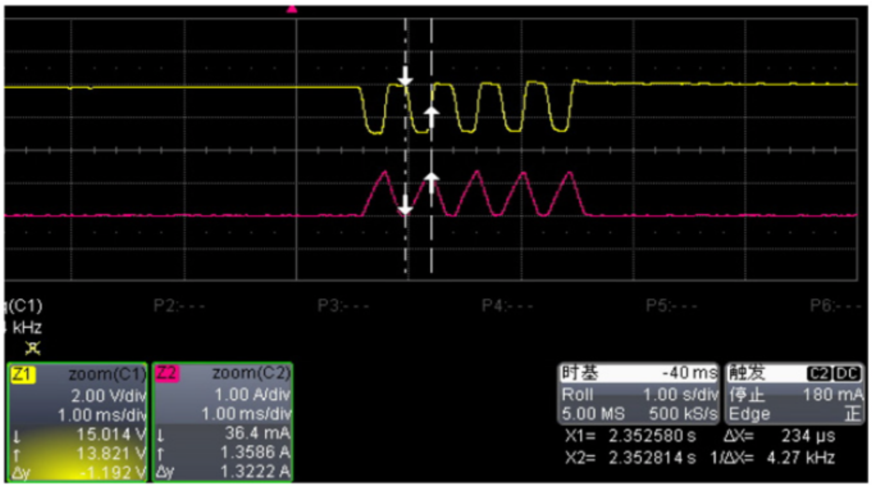
Table 4-3. C Code Example

C Code Example

```
        Discharge(0xEE);      //Switch the PMOS between the cell4 and cell5 of IC2 on
        Discharge(0xFE);      //Switch the PMOS between the cell4 and cell5 of IC2 off

        Discharge(0xEE);
        Discharge(0xFE);

        Discharge(0xEE);
        Discharge(0xFE);

        Discharge(0xEE);
        Discharge(0xFE);

        Discharge(0xEE);
        Discharge(0xFE);
```

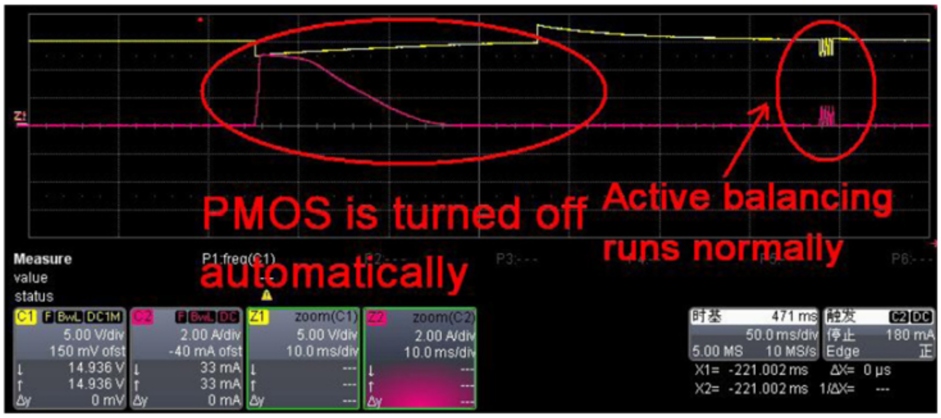
The code above generates the PWM signal by delivering “high” and “low” to the DISCH3 pin via SPI, enabling active balancing between cell 4 and cell 5 of the IC2. In Figure 4-2 the balancing current and the related switching signal are shown. The peak value of the balancing current is around 1.3A and the switching frequency is around 3.0kHz, which is almost the maximum reachable frequency when the SPI runs at around 250kHz.

Figure 4-2. Active Balancing Diagram



There is one final consideration: What happens when the PMOS is switched on and cannot be turned off in time, for example, an interrupt happens while the PMOS is on. Does this mean a short circuit occurs between the cathode and anode of the battery cell? This is prevented from happening by the hardware design of the ATA6870-DK11, which automatically turns off the PMOS through capacitive coupling after a period of time elapses as defined by R/C. Figure 4-3 shows this effect. A delay function is inserted after the first switching on signal. After the current reaches the peak current of around 5A, it is possible to see that the PMOS is switched off on its own and the following active balancing can still run normally.

Figure 4-3. PMOS is Switched Off Automatically



4.6.2 Automatic Mode

In the other ActiveBalance() routine active balancing happens between two cells according to the voltage difference. If the voltage of the higher cell is greater than the lower cell by “Balancing threshold,” as defined in config.h, the balancing PWMs are performed to balance the difference until the differences between all the cells are beneath the threshold.

As shown in Figure 3-2 on page 3, balancing between the lowest cell and the highest cell is carried out through a transformer, which should work according to your application (current, frequency and other related characteristics should be considered).

5. Features of the Atmel ATmega32HVB

Because the Atmel® ATmega32HVB is a part of the Atmel AVR® family of products dedicated to battery management, there are several special features included such as coulomb counting and the control of the two charge/discharge MOSFETs.

5.1 Coulomb Counter

The coulomb counter ADC runs on a different clock than the CPU. This clock is slower and therefore several things have to be kept in mind before using it. Writing several registers in sequence takes a long time depending on the delays between each write cycle. The update busy (CADUB) bit in CADSRA is cleared and written by hardware. A possible solution is given in the supplied software example:

Table 5-1. C Code Example

C Code Example

```
void CCinit(){
CADRCC = RCC_CADRCC;                              // Charge Threshold
while(CADCSRA & (1 << CADUB));
CADRDC = RDC_CADRDC;                              // Discharge Threshold
while(CADCSRA & (1 << CADUB));
SETBIT(CADCSRB,1<<CADRCIE);                       // Interrupt Enable
while(CADCSRA & (1 << CADUB));
SETBIT(CADCSRC,RCC_DIVIDEDSZ<<CADVSE);            // Voltage Scaling
while(CADCSRA & (1 << CADUB));
SETBIT(CADCSRA,((1<<CADEN)|(1<<CADSE)|(RCC_CONVERSIONPERIOD<<1)));
// ADC Enable, RCC Mode, Sampling Interval
while(CADCSRA & (1 << CADUB));}
```

5.2 Charging/Discharging FETs

The two FETs are controlled by an N-channel FET driver. The pins (OC and OD) are designed for outputting a high voltage of approx. 13V. The status of the pins is controlled by software via the FCSR-FET control and the status registers.

Table 5-2. C Code Example

C Code Example

```
void Configure_Fet(unsigned char Fet){
if(Fet&0x01) SETBIT(FCSR, (1<<DFE));
else CLEARBIT(FCSR,(1<<DFE));

if(Fet&0x02) SETBIT(FCSR,(1<<CFE));
else CLEARBIT(FCSR,(1<<CFE));}
```

The example above implements an easy method to enable or disable the two FETs independently of each other. For more information, see page 148 of the Atmel ATmega32HVB datasheet. In cases where the battery string voltage exceeds the maximum voltage of the ATmega32HVB, two GPIOs can be used to switch on the NMOS as well; for example, PB0 and PA3 are responsible for controlling the charging and discharging MOSFETs in the demo software.

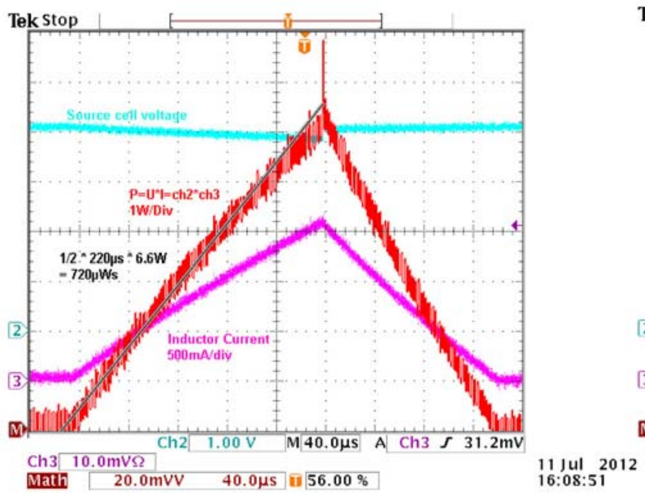
6. Power Consumption

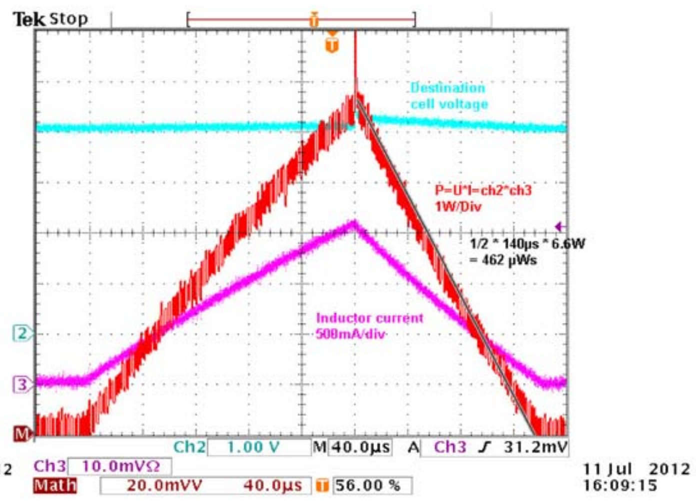
There are several ways to reduce the power consumption of the Atmel ATA6870 and the Atmel ATmega32HVB. Sleep modes are documented in Section 7.1.1 of the Atmel ATA6870 datasheet and in Section 10 of the Atmel ATmega32HVB datasheet. This board allows the Atmel ATA6870 to be enabled/disabled using the Atmel ATmega32HVB software. The PB2 pin is used to control a transistor for activating/deactivating the Atmel ATA6870. Other options which are not implemented are the use of interrupts and a timer (sleep between cycles).

7. Efficiency

Figure 8-1 shows the schematic of the active balancing circuit. The tests are based on the WE749196511 transformer. The maximum PWM switching frequency of the system is 3kHz. As a result, the transformer selected needs to be large enough to store the energy. In the following figures the individual inductor and transformer efficiency are calculated.

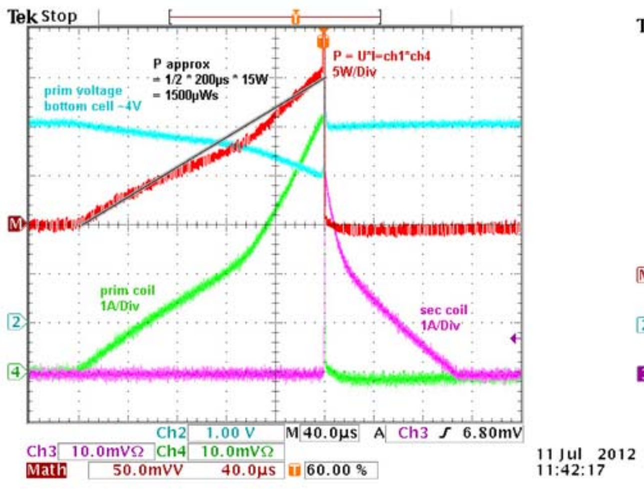
Figure 7-1. Inductor Efficiency

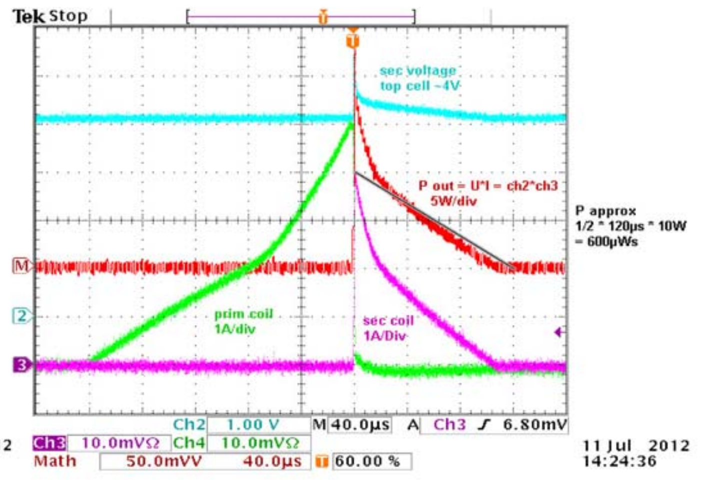




Inductor efficiency = 460µWs/720µWs = 64%

Figure 7-2. Transformer Efficiency





Transformer efficiency = 600µWs/1500µWs = 40%

The efficiency of the inductor to the next cell is around 64% and the efficiency of the transformer from the lowest cell to the highest one is 40%. Based on these values the efficiency simulation is carried out under the following conditions:

- The number of runs is 100,000
- Min/max is ±10% cell voltage difference
- Distribution across three δ
- The assumed average balance current is 1A
- 12 cells

Figure 7-3. Simulation Result with Active Balancing

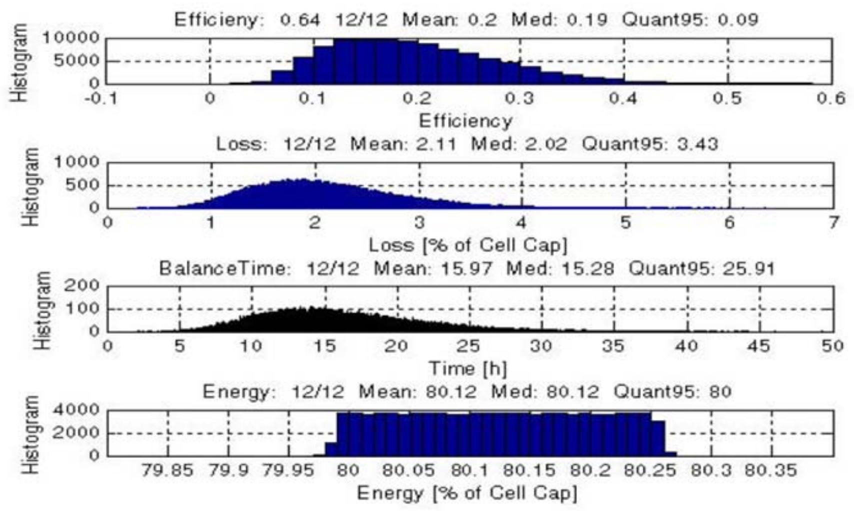
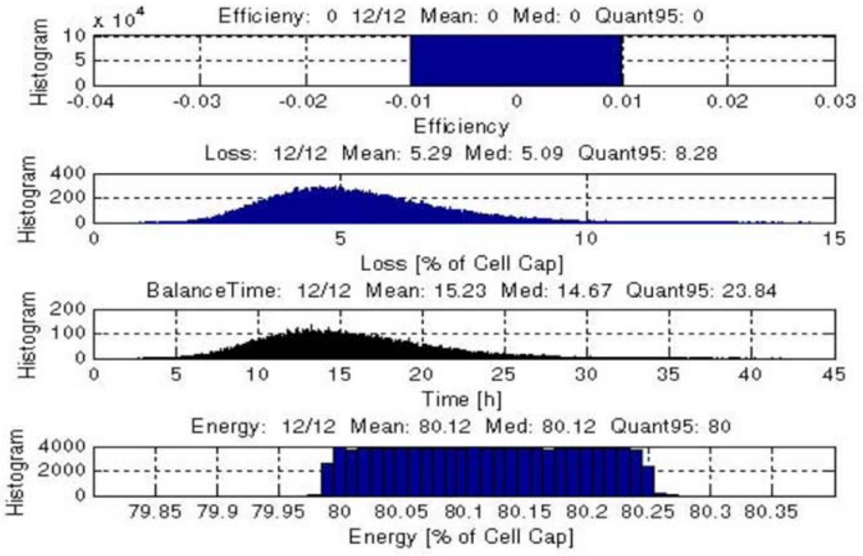


Figure 7-4. Simulation Result with Passive Balancing



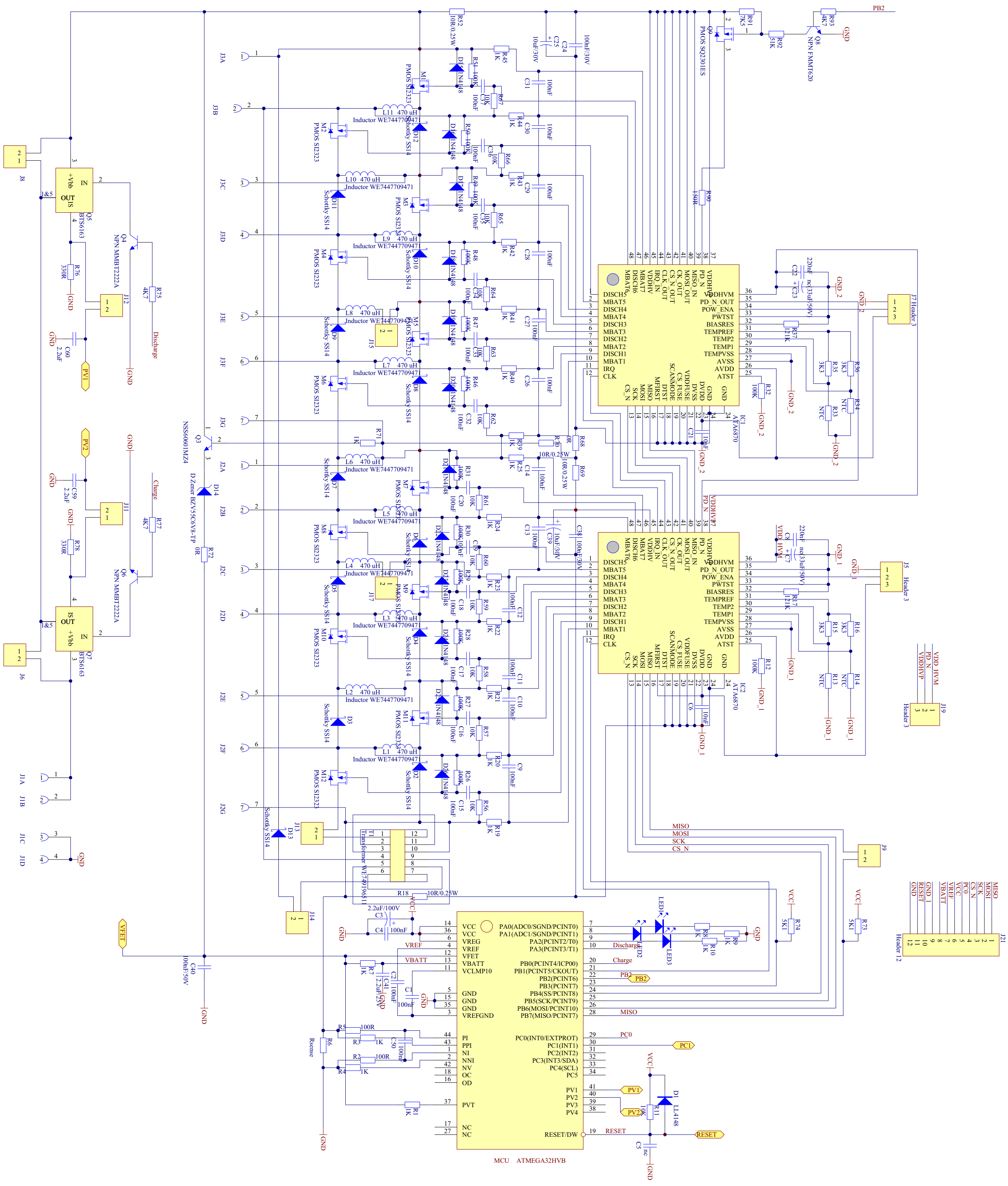
Every result consists of four diagrams:

1. Efficiency of the whole stack
2. Overall loss due to the balancing
3. Time until the balancing is finished
4. Final energy in each cell

The simulation result indicates that with passive balancing the overall loss is around 5.29% and with active balancing the overall loss is 2.11%. It can therefore be concluded that active balancing of ATA6870-DK11 reduces the balancing losses to 40% instead of 100% of passive balancing.

8. Schematic

Figure 8-1. Schematic of ATA6870-DK11



9. Revision History

Please note that the following page numbers referred to in this section refer to the specific revision mentioned, not to this document.

Revision No.	History
9302C-AUTO-04/15	• Put document in the latest template
9302B-AUTO-06/13	• Section 7 “Efficiency” on page 13 updated

Enabling Unlimited Possibilities®