AVR146: Lithium-Ion Battery Charging via USB with ATmega16/32U4

Features

- **Fully Functional Design for Charging Lithium-Ion Batteries**
- **High Accuracy Measurement with 10-bit A/D Converter**
- **Modular "C" Source Code**
- **Easily Adjustable Battery and Charge Parameters**
- **Analog Inputs for Reading Battery ID and Temperature**
- **USB CDC class for user interface**

1 Introduction

This application note is based on the ATmega16/32U4 and focuses on how to use the EVK527 evaluation kit to charge Lithium-Ion (Li-Ion) batteries using USB connection as power supply.

The USB CDC class offers an easy interface to display charge parameters.

This application note is derived from:

AVR458: Charging Lithium-Ion Batteries with ATAVRBC100

The firmware is written entirely in C language (using IAR® Systems Embedded Workbench) and is easy to port to other AVR® microcontrollers.

8-bit AVR® Microcontrollers

Application Note

Rev. 7801A-AVR-06/08

2 Description

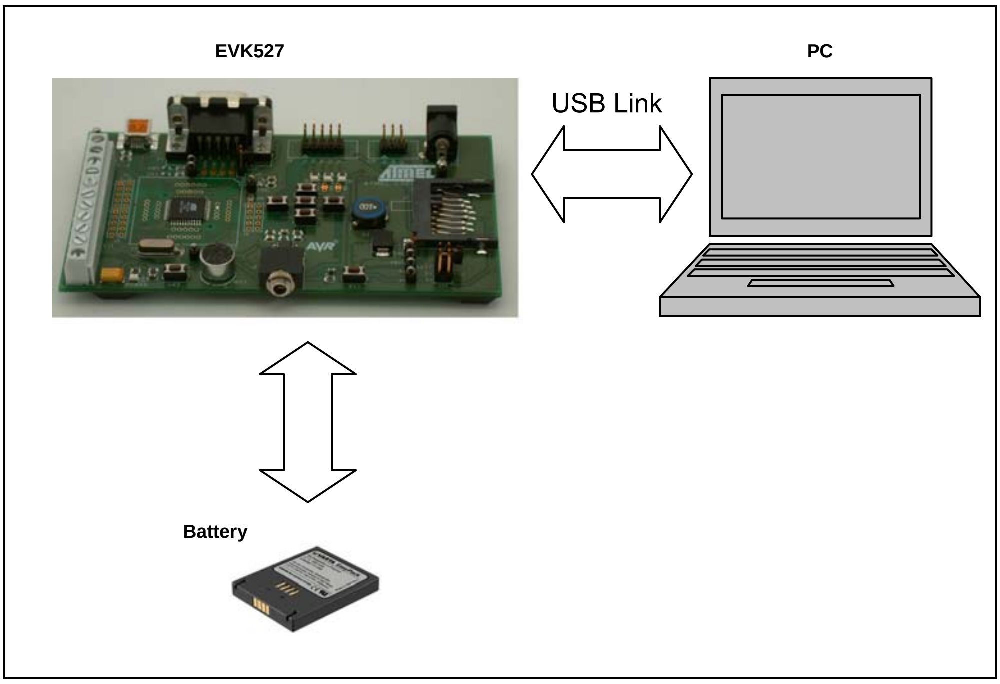
This document describes an application running on the EVK527 evaluation kit. The EVK527 is dedicated to ATmega16/32U4.

The USB offers a 5V power supply on the VBUS pin. The available current range is from 100mA to 500mA. This is enough to charge a Li-Ion battery cell.

A Li-Ion cell needs a precise control of voltage and current during charge.

ATmega16/32U4 offers a USB full speed interface, PWM channels and 10 bit-ADC channels. All these features are used to perform a Li-Ion battery charger via USB.

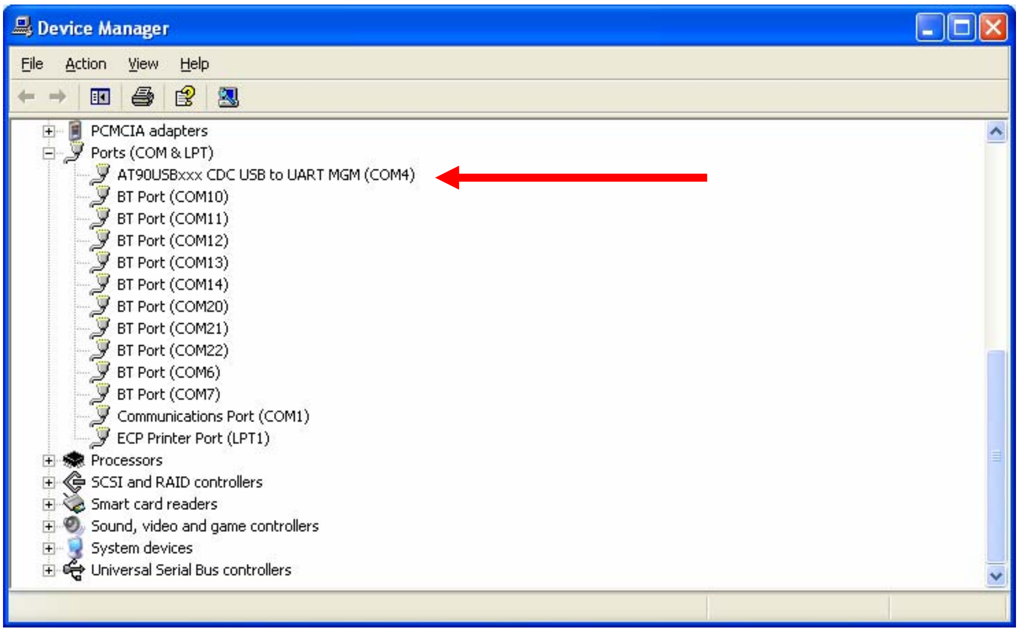
Figure 2-1. Hardware Description.



For a user friendly interface, all charging parameters (charging status, battery voltage, charge current, battery temperature…) are displayed on the PC without the use of measurement tools.

After the USB enumeration, a virtual communication port is declared (see Fig1.2). A HyperTerminal window connected to this communication port displays charging parameters. The communication port is virtual therefore HyperTerminal port settings (speed, parity…) are not taken into account. The user can select the default configuration.

Figure 2-2. Device Manager Window.



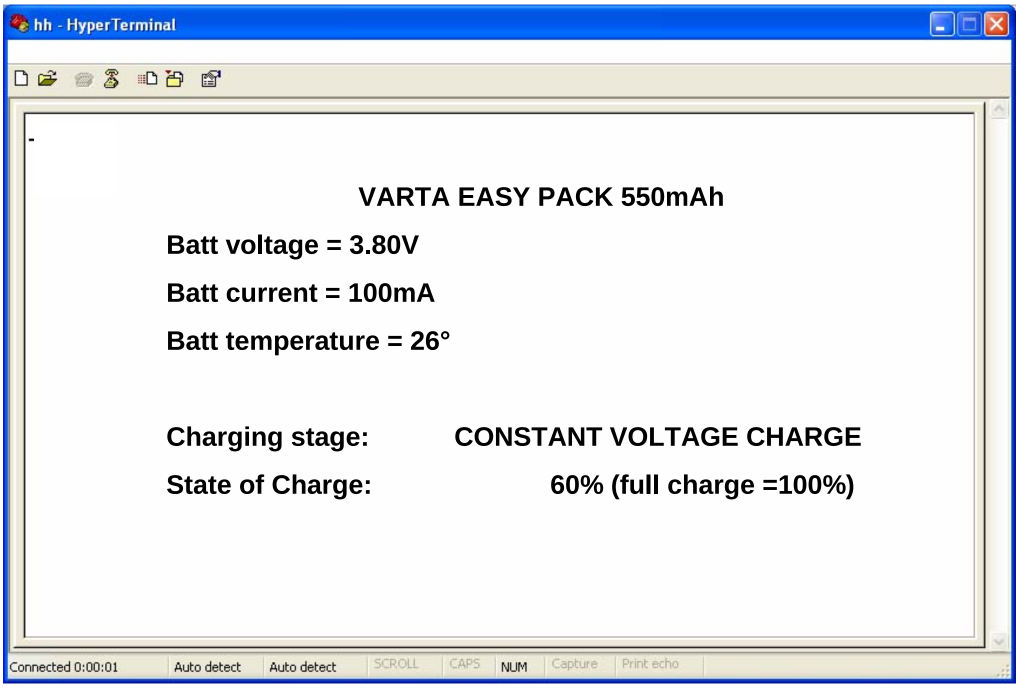
```
PCMCIA adapters
Ports (COM & LPT)
    AT90USBxxx CDC USB to UART MGM (COM4)
    BT Port (COM10)
    BT Port (COM11)
    BT Port (COM12)
    BT Port (COM13)
    BT Port (COM14)
    BT Port (COM20)
    BT Port (COM21)
    BT Port (COM22)
    BT Port (COM6)
    BT Port (COM7)
    Communications Port (COM1)
    ECP Printer Port (LPT1)
Processors
SCSI and RAID controllers
Smart card readers
Sound, video and game controllers
System devices
Universal Serial Bus controllers
```

2.1 Displaying Charging Parameters on the PC

The application performs a continuous update of parameters displayed on the PC. If no battery is detected and identified, the charger is not started. The State of Charge is only available in Constant Voltage Charge stage (see §3.2.3).

When HyperTerminal is running, the user must push the HWB button to start sending information to the PC.

Figure 2-3. User interface after a battery detected.



```
-

                VARTA EASY PACK 550mAh

        Batt voltage = 3.80V

        Batt current = 100mA

        Batt temperature = 26°


        Charging stage:        CONSTANT VOLTAGE CHARGE

        State of Charge:              60% (full charge =100%)
```

3 Theory of Operation

Battery charging 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. A detailed knowledge of these characteristics is required in order to avoid inflicting damage to the battery.

3.1 Li-Ion Battery Technology

Lithium-Ion batteries have the highest energy/weight and energy/space ratios of modern rechargeable batteries (See Reference 1 on page 34). It is currently the fastest growing battery system on the market, with end applications such as notebook computers, cell phones, portable media players, Personal Digital Assistants (PDA), power tools and medical devices.

Compared to traditional rechargeable batteries, Li-Ion batteries have low internal resistance, high cycle life, fast charge time, low self-discharge, low toxicity and no maintenance requirements. For example, lithium-ion cells with cobalt cathodes hold twice the energy of a nickel-based battery and four-times that of lead acid. Lithium-ion is a low maintenance system, an advantage that most other chemistries cannot claim. There is no memory effect with lithium-ion and the battery does not require scheduled cycling to prolong its life. Lithium-ion has a low self-discharge and is environmentally friendly. Disposal causes minimal harm.

Drawbacks of Li-Ion batteries include low tolerance of overcharge and the need for embedded protection circuitry. An electrical short can result in a large current flow, a temperature rise and thermal runaway in which flaming gases are vented.

3.1.1 Safety

Lithium-ion batteries are safe, provided certain precautions are met when charging and discharging. In addition, battery manufacturers ensure a high level of reliability by adding three layers of protection, as follows:

1. The amount of active material is limited to achieve a workable equilibrium of energy density and safety.
2. Various safety mechanisms are included within each cell.
3. An electronic protection circuit is added inside the battery pack.

Cell protection devices work as follows:

- A PTC/NTC (positive/negative temperature coefficient) device acts as a protection to inhibit high current surges.
- The CID (circuit interrupt device) opens the electrical path if an excessively high charge voltage raises the internal cell pressure.
- The safety vent allows a controlled release of gas in the event of a rapid increase in cell pressure.

The electronic protection circuit works as follows:

- A solid-state switch is opened if the charge voltage of any cell reaches a given threshold.
- A fuse cuts the current flow if the skin temperature of the cell approaches 90°C (194°F).
- The current path is cut when cell voltage drops below a given threshold. This is in order to prevent the battery from over-discharging.

Today, lithium-ion is one of the most successful and safe battery chemistries available with billions of cells being produced every year.

3.2 Charging Li-Ion Batteries

There is only one way to charge lithium-based batteries. Manufacturers of Lithium-Ion cells have very strict guidelines in charge procedures and the packs should be charged as per the manufacturers "typical" charge technique.

Li-Ion batteries are charged using constant voltage (after having reached the nominal charge voltage), with current limiter to avoid overheating in the initial stage of the charging process. Charging is terminated when the charge current drops below a threshold set by the manufacturer. Several parameters are monitored during the charge: charge time, battery temperature… The battery takes damage from overcharging and may explode if overcharged.

3.2.1 Safety

Static electricity or a faulty charger may destroy the battery's protection circuit and turn solid-state switches to a permanent ON position. This may happen without the user knowing. A battery with a faulty protection circuit may function normally but does not provide protection against abuse.

Consumer grade lithium-ion batteries cannot be charged below 0°C (32°F). If charged at cold temperatures, battery packs may appear to be charging normally but chemical reactions inside the cells may cause permanent damage and can compromise the safety of the pack.

The battery will become more vulnerable to failure if subjected to impact, crush or high rate charging.

The battery must remain cool. A battery pack that gets hot during charge should not be used.

3.2.2 Priming & Charge Intervals

Unlike many other types of rechargeable batteries, Lithium-Ion batteries do not need priming. The first charge of a Li-Ion battery is no different than the 10th or the 100th charge.

Lithium-ion batteries may be – and should be – charged often. The battery lasts longer with partial rather than full discharges. Full discharges should be avoided because of wear.

The battery loses capacity due to aging, whether used or not.

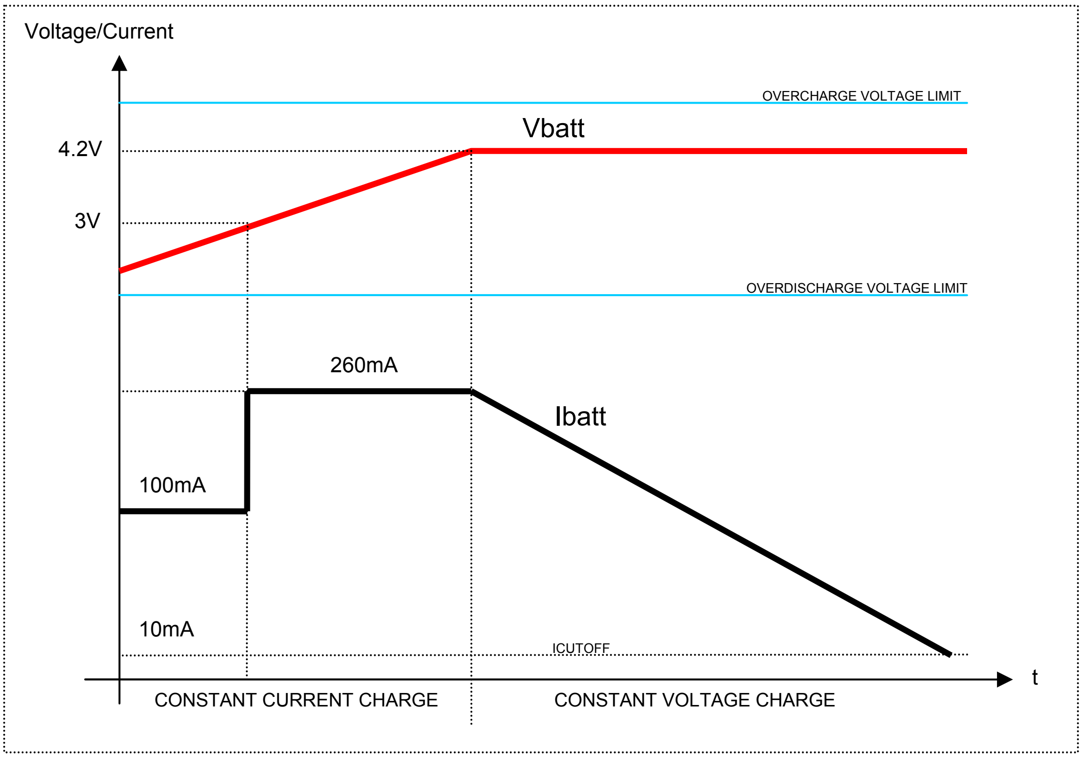
3.2.3 Charge Stages

Lithium-Ion battery charge follows three stages:

1. Prequalification current: Charging of a Li-Ion battery starts with a test of battery voltage. If the voltage is under a defined threshold (PREQUAL_VOLTAGE), the charge starts with a fixed low current.
2. Constant current. The charge continues with applying constant current to the battery. The size of the charge current is battery-dependent and given by the manufacturer. This stage is complete when battery voltage has reached the threshold given by the manufacturer.
3. Constant voltage. After battery threshold voltage has been reached the charger will switch from supplying constant current to supplying constant voltage. This stage is complete when charge current has dropped below the threshold given by the manufacturer.

The below figure illustrates voltage and current of a lithium-ion battery during charging.

Figure 3-1. Charge stages and limits of a VARTA™ EasyPack 550mAh



In the figure above, “Overcharge” is the level at which cell protection circuitry cuts in and opens a solid-state switch and discontinues the charge current path. After this, battery voltage typically needs to drop several hundred millivolts before the current path is restored. “Overdischarge” is the level at which the current path is cut in order to prevent the battery from over-discharging.

3.3 VARTA™ battery

3.3.1 Typical Charge Characteristics

Battery specifications should always be verified from manufacturer's data sheets. Below is a summary of typical lithium-ion battery charge characteristics. Actual parameters may vary.

Table 3-1. Typical Charge Characteristics

Parameter	**Typical Value**
Charge time	3 hours
Charge current	1 C[1]
Charge efficiency	99.9 %
Charge current threshold	0.03 C[1]
Charge voltage	4.20 V
Charge voltage tolerance (per cell)	± 0.05 V
Temperature range	0 … +45 °C
Humidity range	65 ± 20 RH

1. C corresponds to the typical Rated capacity value (see Table 3.2)

3.3.2 Typical Battery Characteristics

The table below summarises manufacturer's data for the batteries types used in this application. Other types of batteries may be used, but may require adjustments to software and/or hardware.

Table 3-2. Manufacturer's data for VARTA™ EasyPack range of lithium-ion batteries

Parameter	**EZPack S-3.7V**	**EZPack M-3.7V**	**EZPack L-3.7V**	**EZPack XL-3.7V**	**Unit**
Rated capacity (typical)	550	750	1000	2000	mAh
Nominal voltage	3.70				V
Operating voltage range	2.75 … 4.20				V
Charge voltage	4.20				V
Charge voltage tolerance	± 50				mV
Charge current	520	720	955	955	mA
Charge cut-off time	3	3	3	4	hours
Charge cut-off current	10	14	19	38	mA
RID[1] (resistor ID)	3.9	6.8	10	24	kΩ
NTC	10				kΩ
B-value[2]	3435				K
Overcharge detection	4.35				V
Overdischarge detection	2.20				V

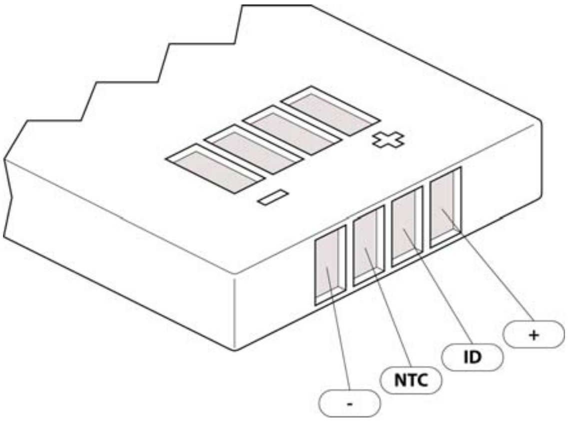
1. RID: Battery internal resistor identifies the capacity of battery connected.
2. B value is used in temperature formula.

3.3.3 Electrical pinout

This application uses a particular type of lithium-ion batteries and all configurations presented here are based on manufacturer's data. Other lithium-ion batteries may naturally be used but it is up to the user to look up battery data from manufacturer's data sheets and make sure necessary adjustments are made to firmware and hardware.

The figure below illustrates connection pads of the lithium-ion batteries used in this application.

Figure 3-2. Connection pads of a VARTA™ EasyPack cell.



The battery is connected to the battery charger as follows.

Table 3-3. Connecting battery to charger

Battery Connector	Charger Connector	Note
- (minus)	BATTERY-	
NTC	NTC/RID	Battery temperature measurement
ID	SCL	RID, Battery identification resistor
+ (plus)	BATTERY+	

3.4 VBUS Supply Voltage

USB powered applications fall into one of the three following categories:

- **Low-Power Bus** The low power bus powered functions derive all their power from VBUS and must not draw more than 1 unit load (100mA) according to the USB standard. It must also be able to work between the VBUS voltage of 4.40V and 5.25V.
- **High-Power Bus** The high power bus powered functions derive all their power from VBUS and cannot draw more than 100mA until it has been configured. Once configured, it can draw up to 5 unit loads (500mA) by requesting it in its descriptor. At full load, it must be able to work between the VBUS voltage of 4.75V and 5.25V.

- **Self-Power** Self powered functions can draw up to 100mA from VBUS and the rest from another source.

The current to power the EVK527 and to charge the battery comes from VBUS. The EVK527 must limit the charge current if needed.

An easy solution is to modify the I charge parameter in the lookup table.

For example, a 550mAh battery allows a 260mA charging current. A modification of this parameter to 90mA (for example) allows connecting the charger on a Low Power Bus, knowing that the EVK527 consumption with an 8MHz oscillator is about 10mA. In this case, the prequalification current must also be limited to 90mA.

3.5 EVK527 Revision

The EVK527 Rev1.0.0 must be updated as follows:

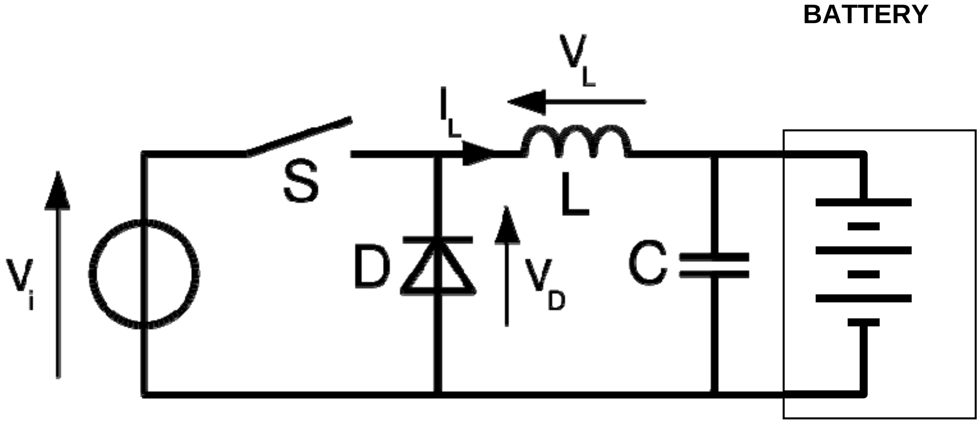
- The shunt resistor must be connected between PF0 and PF1 to use the differential input mode. PF0 must replace PD4 and SP6 must be "without solder" (different of default configuration).
- Gate pin and Source pins of Q1A must be disconnected.
- R6 and R7 new values are 13kOhms
- R3 new value is 1Ohm

Schematics given in §6 don't show these modifications.

3.6 Buck converter

A buck converter is integrated on EVK527 to control the battery voltage and the battery current. The switch is controlled by the High speed PWM output.

Figure 3-2. Buck converter schematic.



3.6.1 PWM frequency

The PWM speed for the PWM is programmed to the maximum (64MHz). The source clock is the PLL output (96MHz) used both by USB and PWM.

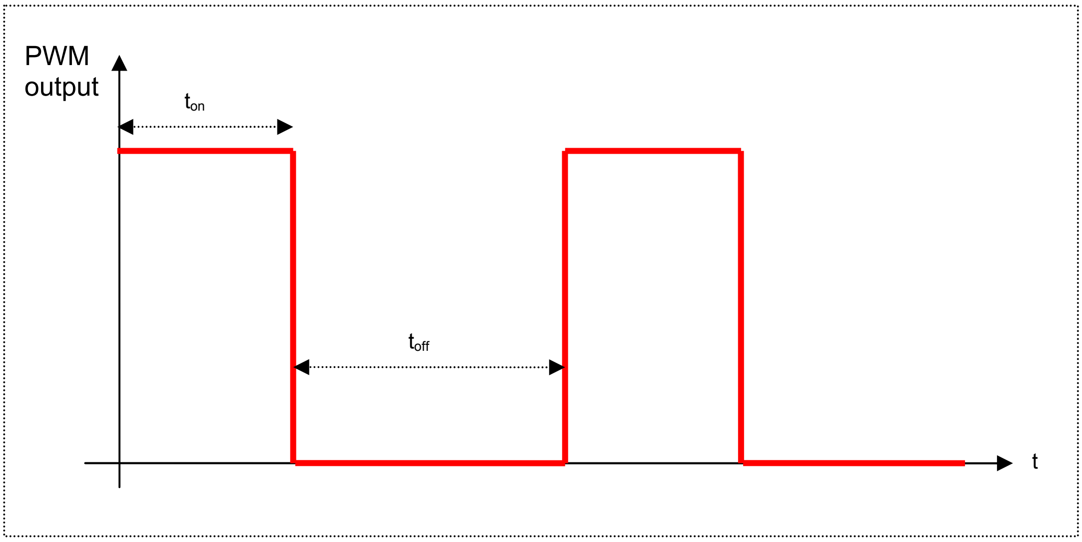
A postscaler offers a 1.5 division for the PLL signal: 96MHz/1.5 = 64MHz (see PLLTM1 and PLLTM0 in PLLFREQ register).

The result on the PWM output signal is a 250kHz frequency:

64MHz / 256 = 250kHz

Where 256 is the size in bit of the in OCR4A compare register used in Timer 4.

Figure 3-2. PWM output signal.



The software controls the battery voltage/current in modifying the duty cycle of PWM output. If ton increases, the battery voltage/current receives a more important energy load.

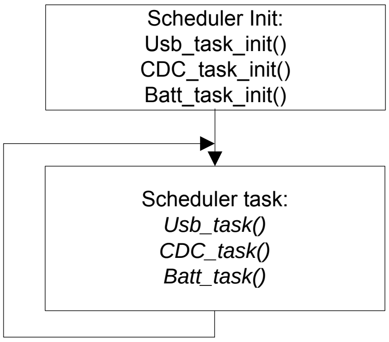
4 Battery Charger Software

4.1 Scheduler

A scheduler is implemented to call indefinitely defined tasks. Before starting this infinite loop, init functions are called.

There are three tasks. Each task is called after the end of the previous one (no pre-emption).

Figure 4-1. Scheduler



4.2 List of files

The firmware is written in C language using IAR Systems Embedded Workbench®, version 5.10. Since the firmware has been written entirely in C, it should not be a difficult task to port it to other AVR C-compilers. Some compiler specific details may, however, need to be rewritten.

In the table below are listed the files that are relevant to the compiler project.

Table 4-1. Project files for CDC application (see IAR EW workspace file)

File	Type	Note
cdc_task.c	C source code	CDC task and CDC task init functions
cdc_task.h	Header file	
main.c	C source code	Main program / Program entry point
main.h	Header file	
power_drv.c	C source code	Power management low level driver
power_drv.h	Header file	
scheduler.c	C source code	Scheduler routines
scheduler.h	Header file	
start_boot.c	C source code	Boot functions
start_boot.h	Header file	
time.c	C source code	Functions for timing
time.h	Header file	

File	Type	Note
cdc_task.c	C source code	CDC task and CDC task init functions
cdc_task.h	Header file	
main.c	C source code	Main program / Program entry point
main.h	Header file	
uart_lib.c	C source code	This file provides a minimal VT100 terminal access
uart_lib.h	Header file	
uart_usb_lib.c	C source code	UART USB functions
uart_usb_lib.h	Header file	
usb_descriptor.c	C source code	USB parameters that identify the application
usb_descriptor.h	Header file	
usb_device_task.c	C source code	USB device controller
usb_device_task.h	Header file	
usb_drv.c	C source code	USB driver routines
usb_drv.h	Header file	
usb_standard_request.c	C source code	USB device enumeration requests
usb_standard_request.h	Header file	
usb_specific_request.c	C source code	User call-back functions
usb_specific_request.h	Header file	
usb_task.c	C source code	Usb task and Usb init task functions
usb_task.h	Header file	

Table 4-2. Project files for battery module (see IAR EW workspace file)

File	Type	Note
ADC.c	C source code	Functions related to A/D converter
ADC.h	Header file	
Batt_task.c	C source code	Batt task and Batt init task functions
Batt_task.h	Header file	
battery.c	C source code	Battery-specific definitions and functions related to battery control & data acquisition
battery.h	Header file	
chargefunc.c	C source code	Charge functions
chargefunc.h	Header file	
LIIONcharge.c	C source code	Charge state function for Li-Ion batteries
LIIONcharge.h	Header file	
menu.c	C source code	State machine definitions
menu.h	Header file	
PWM.c	C source code	Functions related to generating pulse-width modulated output
PWM.h	Header file	

File	Type	Note
statefunc.c	C source code	Functions related to the states defined in menu file
statefunc.h	Header file	

4.3 Overview

The firmware integrates all functions required to charge a lithium-ion battery.

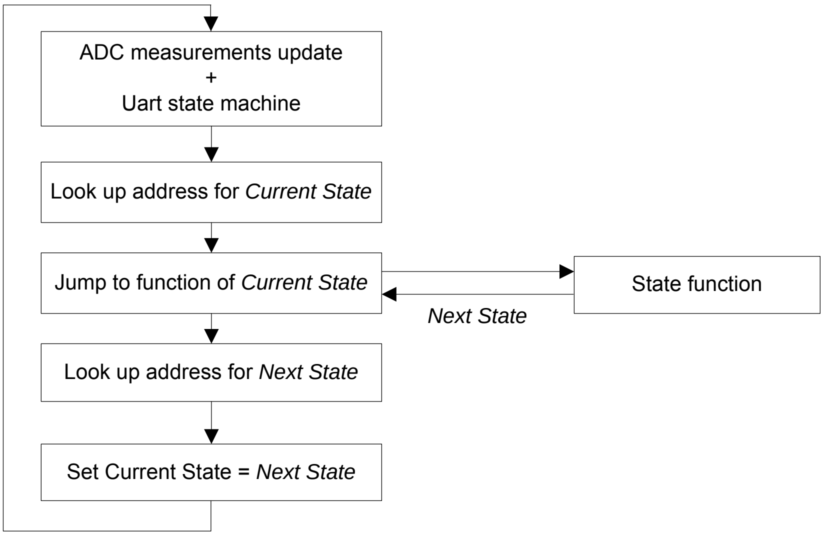
Table 4-3. Memory requirements of firmware (IAR without optimization)

Build option	Memory	Approximate value
Debug	CODE (Flash)	13900 bytes
	DATA (SRAM)	1109 bytes
	XDATA (EEPROM)	2 bytes

4.4 State Machine

A state machine is implemented in battery task. This state machine is rather simple and uses function pointers. It simply looks up the address of the next function to execute and then jumps to that function. The flow chart of the state machine is illustrated in the figure below.

Figure 4-2. Flow chart of main function, including the state machine



Upon return, the state machine expects the function to indicate the next state as a return argument. The recognised return codes are described in the table below.

Table 4-4. State machine codes (see source code, menu.h)

Label	Related Function	Description
INIT	Initialize()	Entry state
BATCON	BatteryControl()	Check hardware and batteries

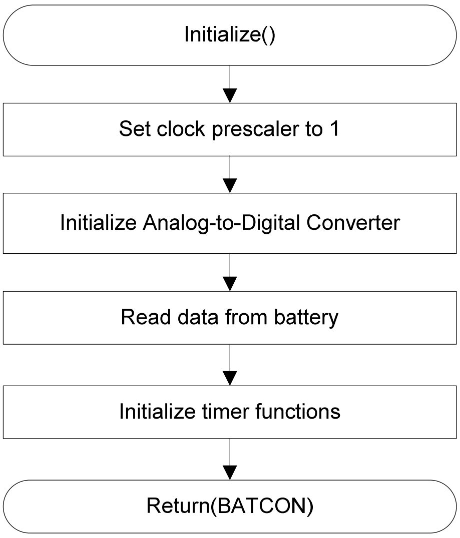
Label	Related Function	Description
PREQUAL	Charge()	Raise battery voltage, safety check
PREQUAL_CTRL	Charge()	Waiting end of PREQUAL
SLEEP	Sleep()	Low power consumption mode
CCURRENT	Charge()	Charge with constant current
CCURRENT_CTRL	Charge()	Waiting end of CCURRENT
CVOLTAGE	Charge()	Charge with constant voltage
CVOLTAGE_CTRL	Charge()	Waiting end of CVOLTAGE
ENDCHARGE	Charge()	End of successful charge
DISCHARGE	Discharge()	Go to BATCON state (ready for further implementation)
ERROR	Error()	Resolve error, if possible

State functions are described in the following sections.

4.4.1 Initialize()

The initialisation function is the first state function that will be executed after device reset. The flow chart of the function is shown in the figure below.

Figure 4-3. Flow chart of initialisation function

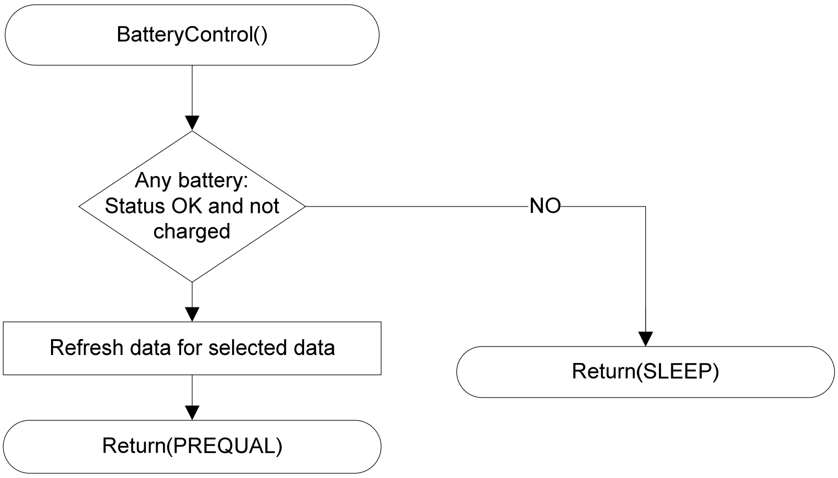


The initialisation function always exits with the same return code, pointing to the state function for battery control.

4.4.2 BatteryControl()

The battery control function verifies that jumpers are set correctly and then checks to see if there are any enabled batteries present that require charging. The program flow is illustrated in the figure below.

Figure 4-4. Flow chart of battery control function



4.4.3 Charge()

The charge function contains the charging algorithm divided into stages. For this application, it has four stages:

- Prequalification - during which the battery is charged with a constant current until a sufficient charge voltage is reached. If this happens within a given time limit, the battery is considered good and the charger may continue on the next stage. If time runs out before the voltage is reached, or battery temperature goes out of limits, the battery is considered bad and charging is halted.
- Constant current charge - during which the battery is charged with a higher, battery-specific current until the battery voltage reaches its maximum. If this happens within the battery's maximum charge time limit, the charger goes to the next stage. If the time limit expires, or battery temperature goes out of limits, the battery is considered bad and charging is halted.
- Constant voltage charge – during which the battery is charged at the maximum battery voltage until the charge current drops below a battery-specific cut-off limit, or the maximum charge time limit expires. Here too, charging is halted if battery temperature goes out of limits.
- End charge – in which the charger decides whether to go into the sleep state, or to attempt a charge of the other battery.

ChargeParameters and HaltParameters are key variables of this function. The program flow of this state function is illustrated in the figure below.

Figure 4-5. Flow chart of the charge state function

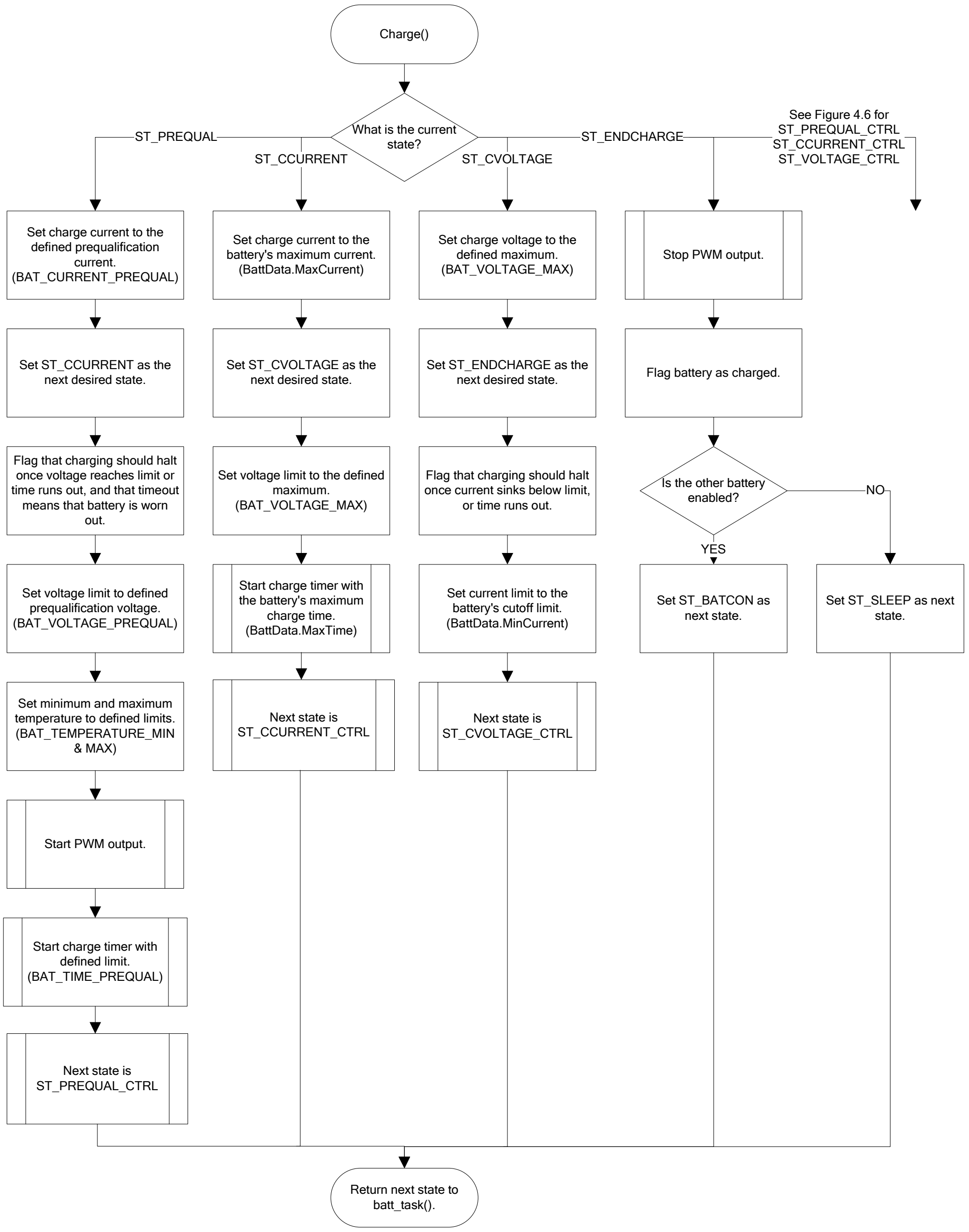
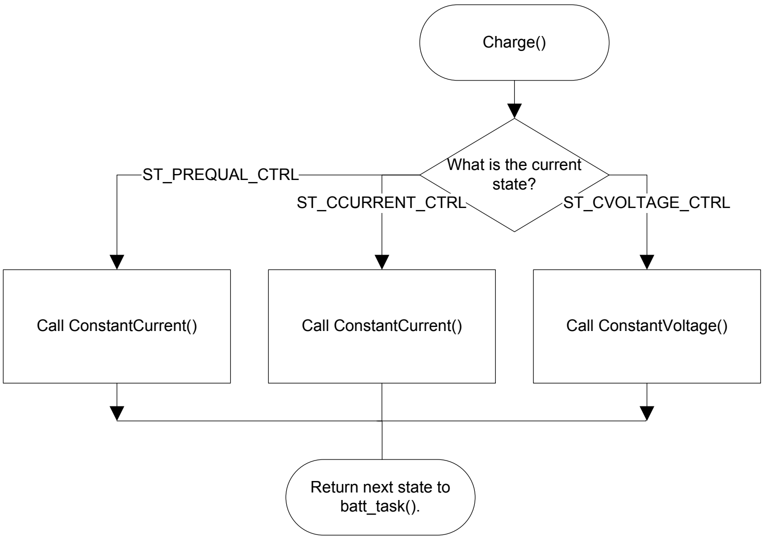


Figure 4-6. End of charge state function.



4.4.4 Discharge()

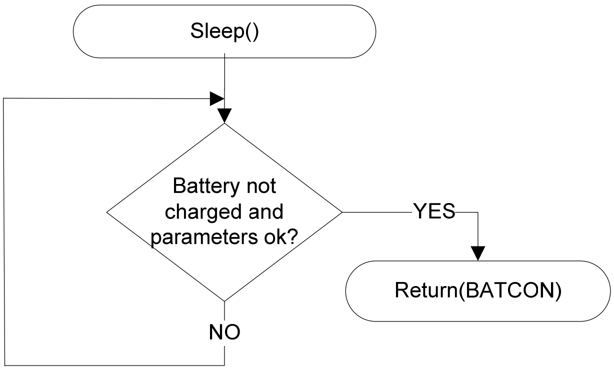
This function has not been implemented.

4.4.5 Sleep()

The application enters sleep mode when all batteries have been fully charged. It wakes up at regular intervals to check the current status of the batteries. Sleep mode is terminated as soon as any battery requires charging.

Sleep mode is illustrated in the flow chart below.

Figure 4-7. Flow chart of sleep function

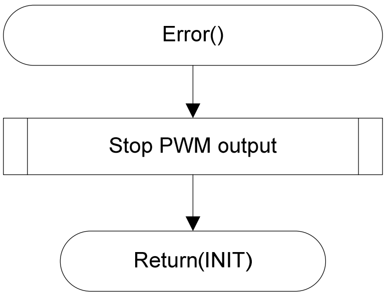


4.4.6 Error()

Program flow is diverted here when an error has occurred. Program execution will exit the error handler when all sources of error have been cleared.

The program flow is illustrated in the figure below.

Figure 4-8. Flow chart of error handler



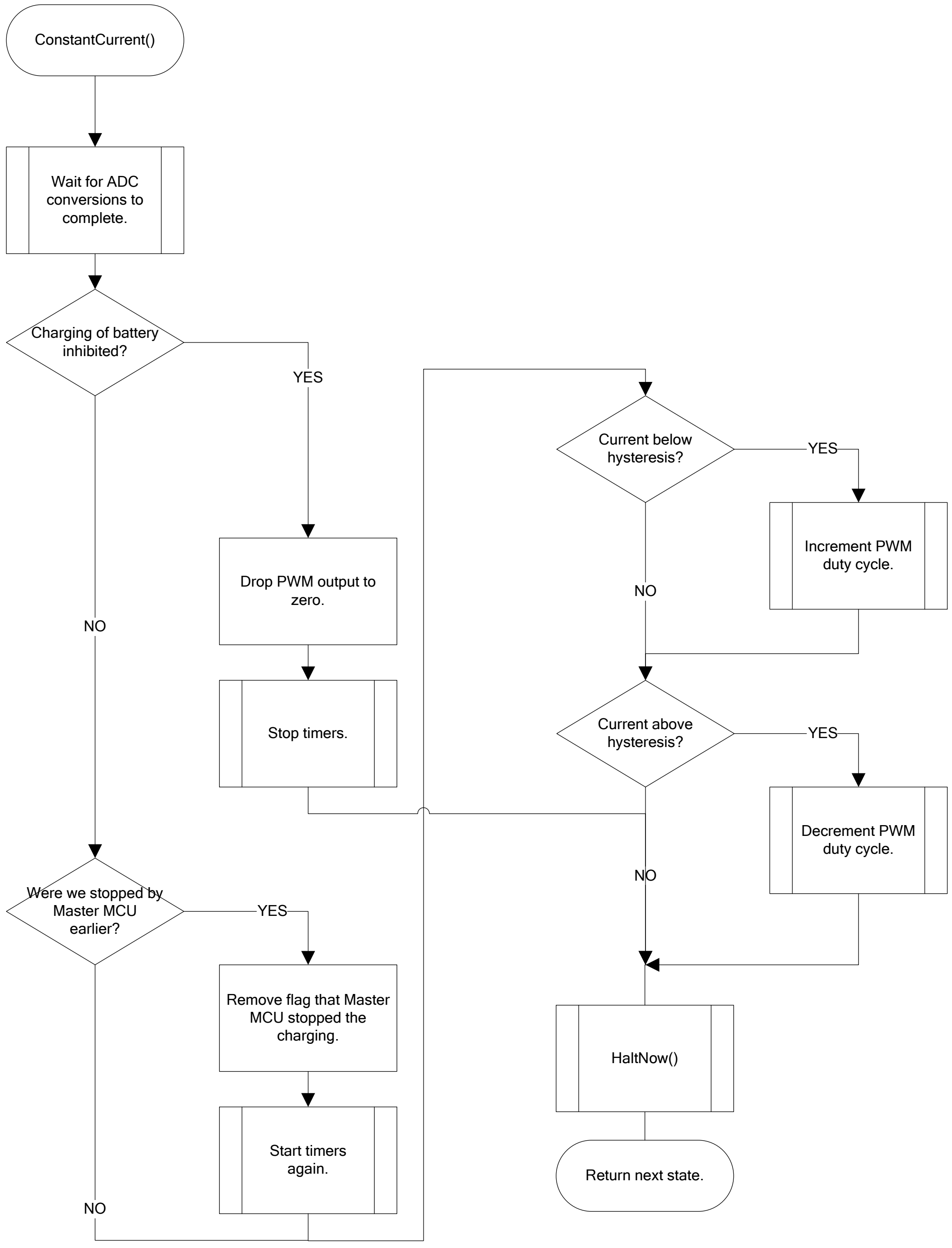
4.5 Charging Control Functions

These functions are called by Charge() after all parameters have been set.

4.5.1 Constant Current/Voltage

These two functions are similar, apart from what ADC measurements they try to keep within limits. Therefore, only the flow chart for ConstantCurrent() is illustrated in the figure below. They both make use of the variable ChargeParameters.

Figure 5-3. Flow chart for ConstantCurrent()



4.5.2 Charge Halt Determination

Charge halt is determined by HaltNow(). This function is called by ConstantCurrent() and ConstantVoltage() every time they loop, to decide if a stage of charging is done.

With the variable HaltParameters the user can specify at what terms the charging should be halted, and if an error should be flagged if for example the time limit expires. An error flag will also result in ST_ERROR being set as the next state, thereby aborting the charge. If no errors are flagged, the next desired state, set earlier in Charge(), will apply.

Lastly, the function checks if temperature is within limits, if the battery is OK and if mains voltage is above minimum. Should any of these tests fail, the next state is set to an appropriate error handler (ST_ERROR, ST_INIT or ST_SLEEP) and charging is aborted.

Figure 5-4. Flow chart for HaltNow() part 1.

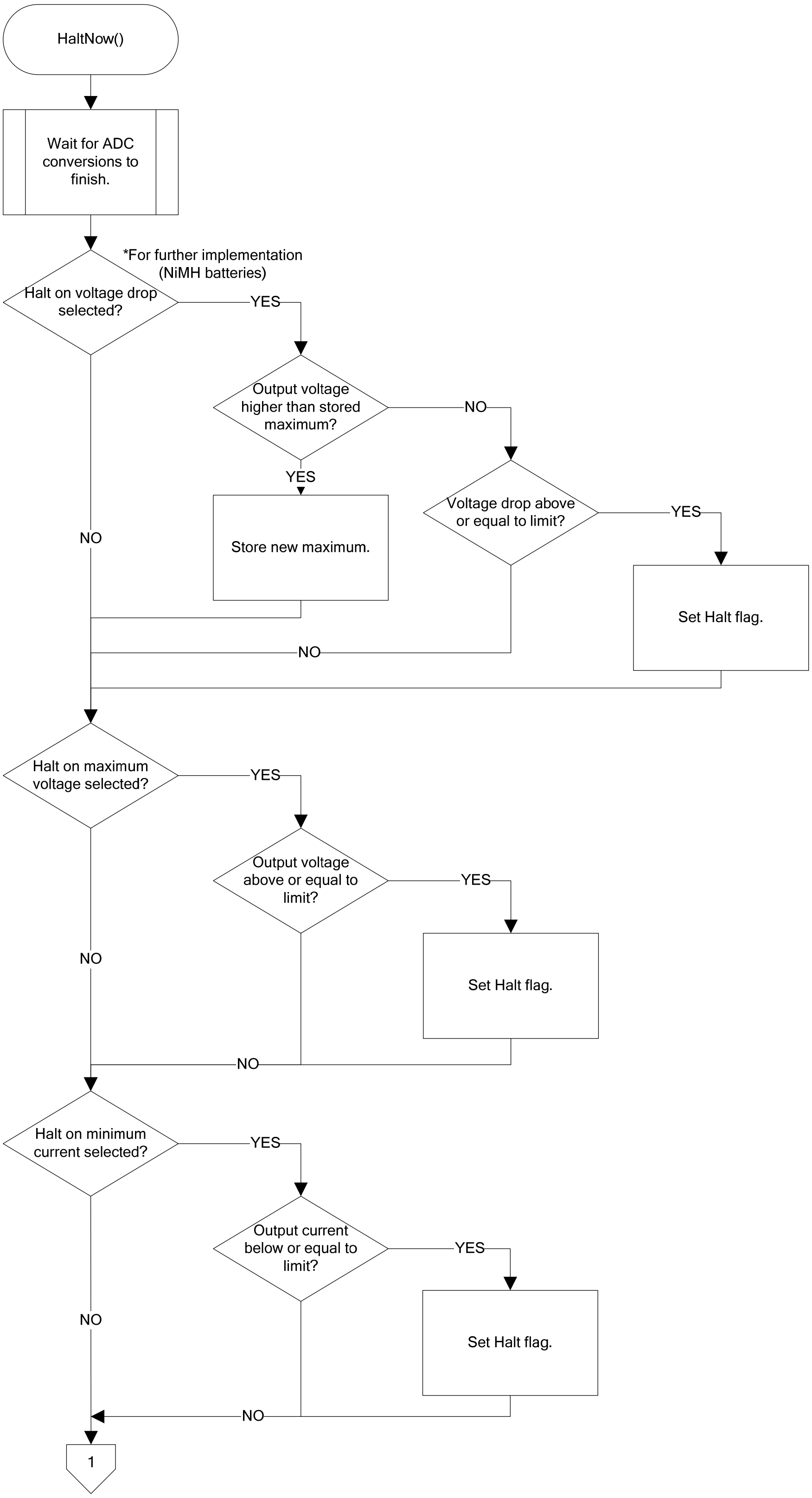


Figure 5-5. Flow chart for HaltNow() part 2

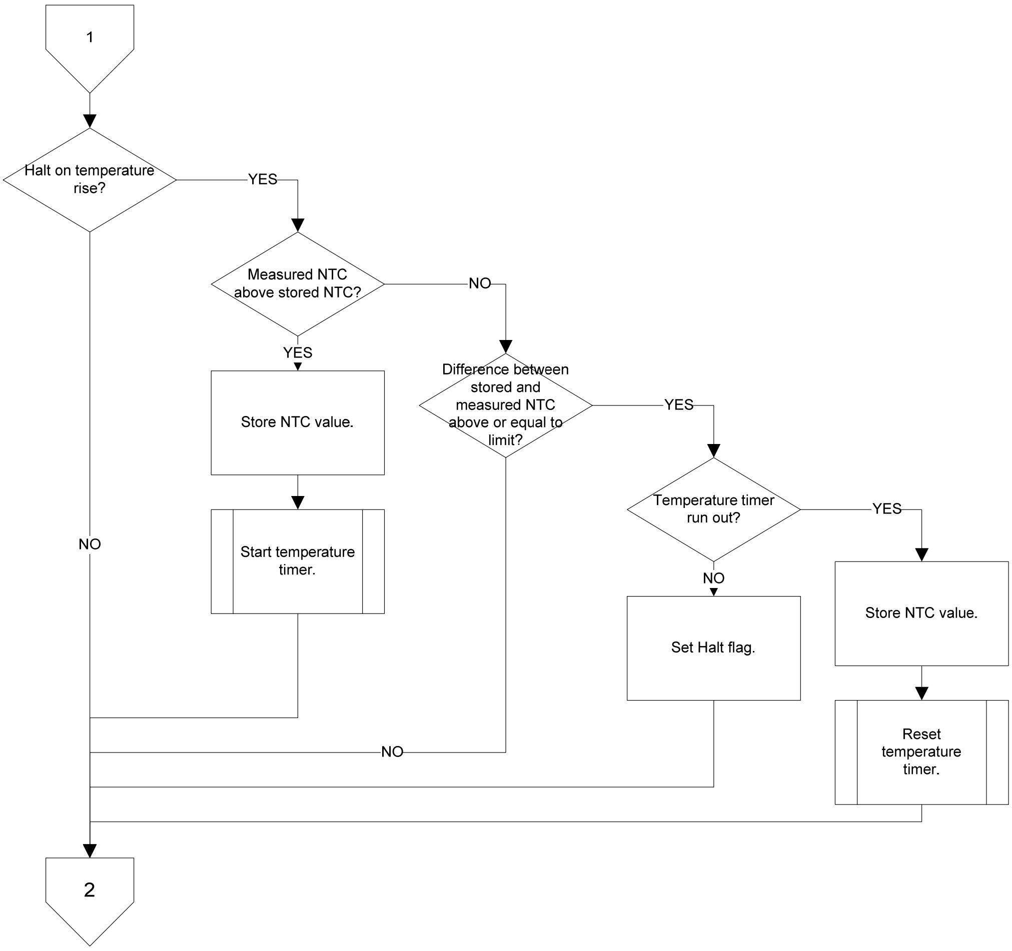


Figure 5-6. Flow chart for HaltNow() part 3

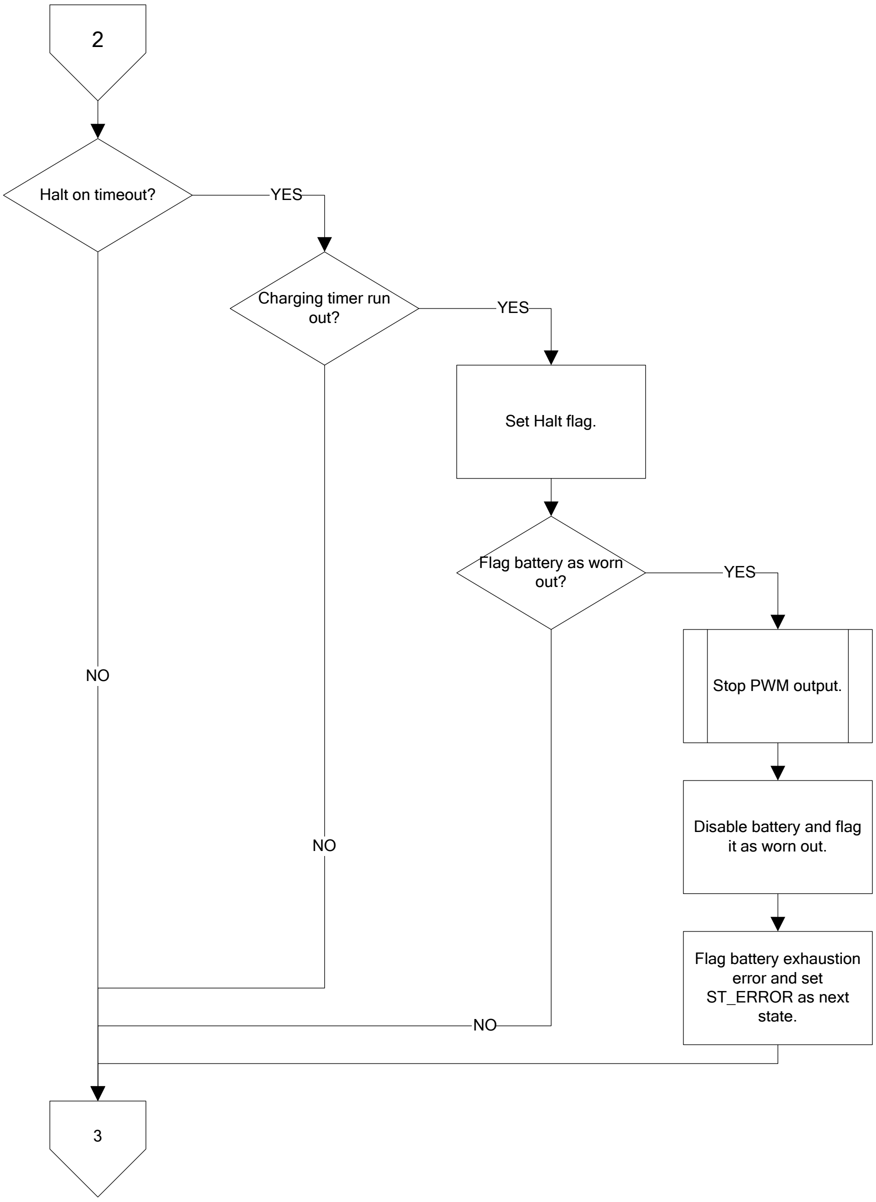
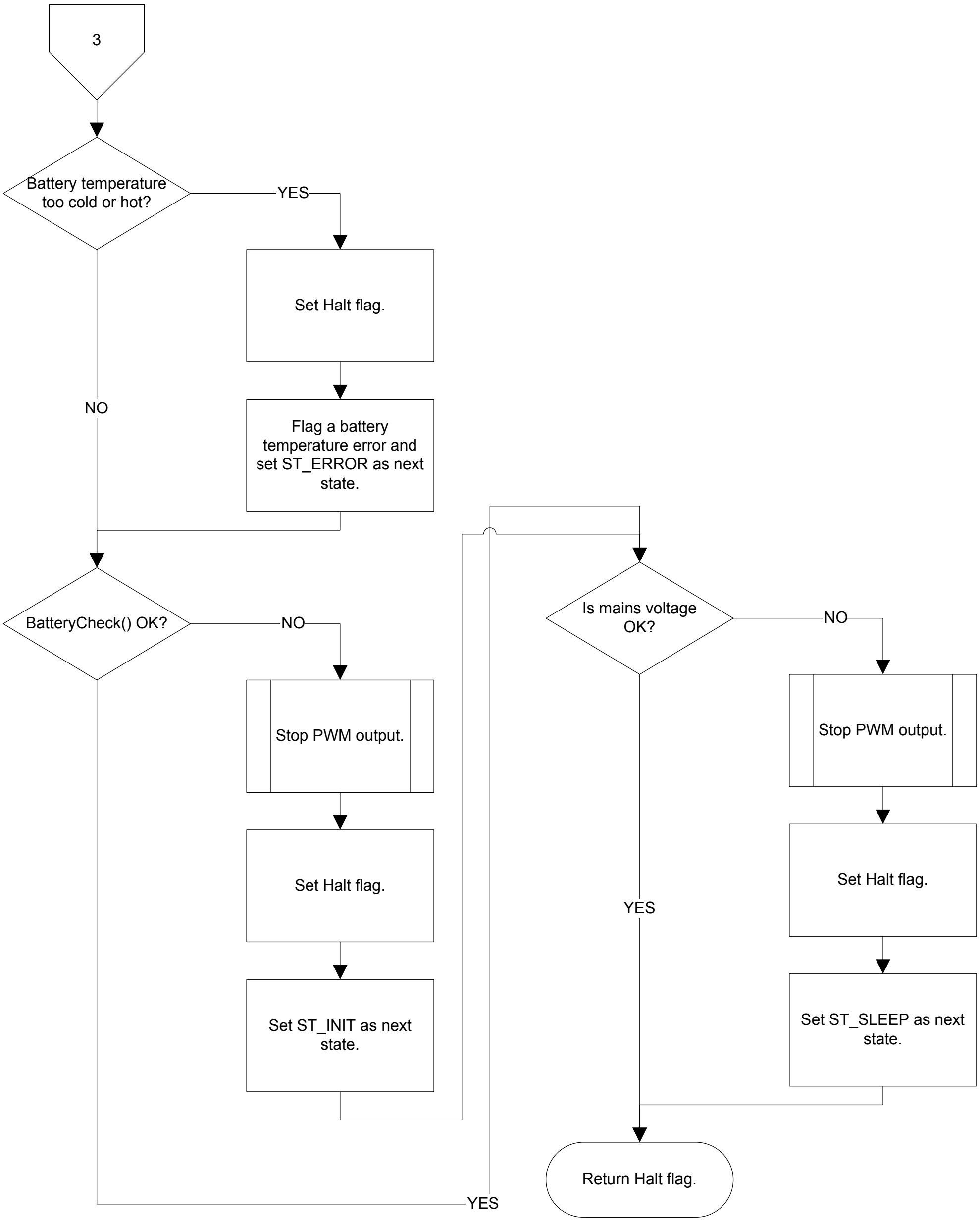


Figure 5-7. Flow chart for HaltNow() part 4

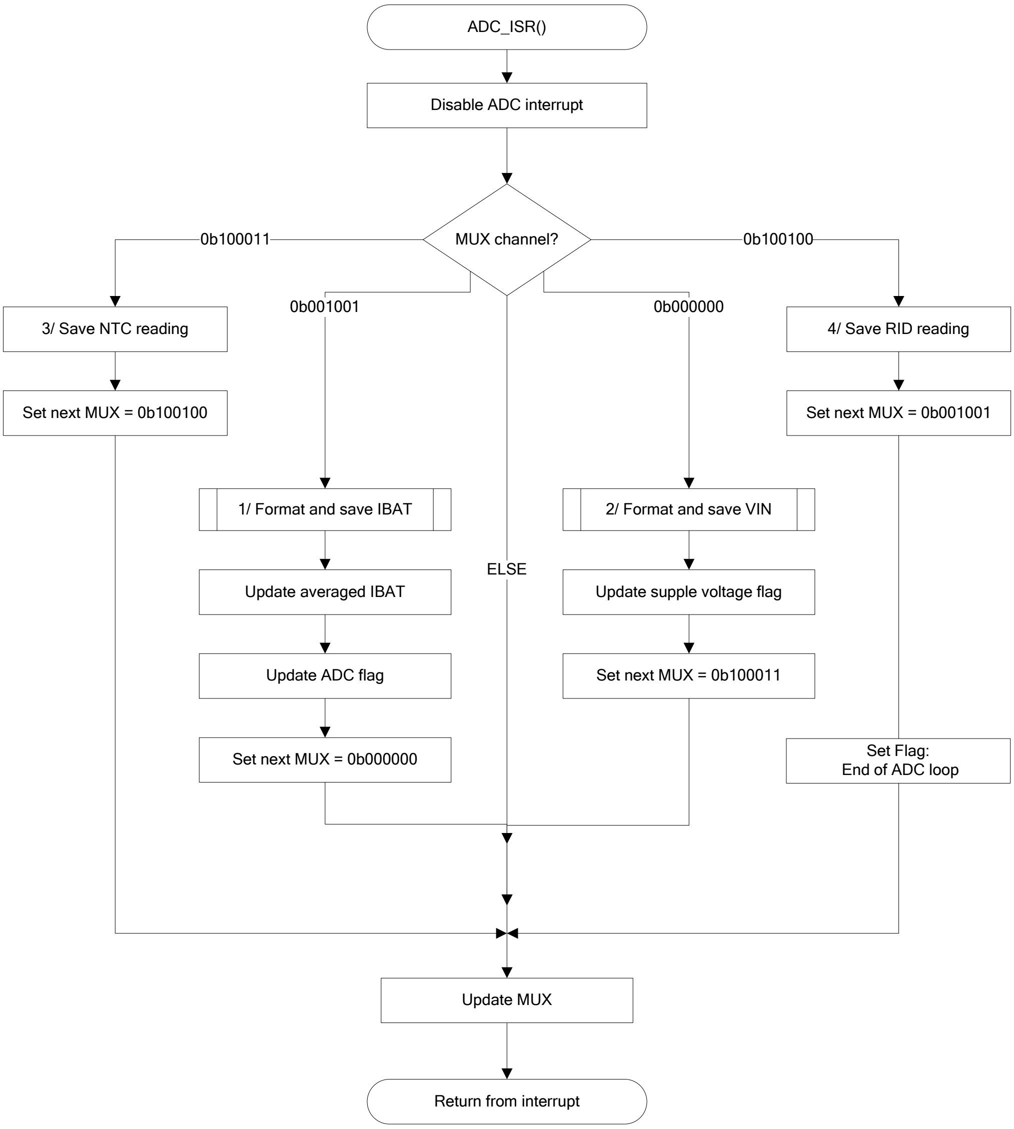


4.6 Other Functions

4.6.1 A/D Conversion

The A/D converter uses the multiplexer to read in data from several channels. At the end of a conversion the ADC Interrupt Service Routine (ISR) is called, as illustrated in the flow chart below. After the ISR is complete program execution will return to normal. For all MUX values, the ADC reference voltage is the 2.56V internal reference.

Figure 5-8. Flow chart of ADC interrupt service routine



4.7 Implementation

This section describes how to configure, create and download the software.

4.7.1 Configuration

The most important compile-time constants are discussed in the table below. See file battery.h for more program constants.

Table 5-1. Battery-related compile-time constants (see source file battery.h)

Label	Description
CELL_VOLTAGE_SAFETY	In case unmatched batteries are to be charged, this constant is subtracted from CELL_VOLTAGE_MAX for every extra cell in the battery, ie. BAT_CELL_NUMBER – 1.
CELL_VOLTAGE_MAX	The voltage at which a cell should be charged.
CELL_VOLTAGE_LOW	The lowest voltage at which a cell is considered charged. Charging will start when voltage drops below this level.
CELL_VOLTAGE_MIN	The lowest voltage at which charging may be initiated. Should generally be set to the voltage limit under which further discharge of batteries will cause damage.
CELL_VOLTAGE_PREQUAL	The voltage to which a cell should be charged to during prequalification.
BAT_TEMPERATURE_MAX	The highest battery temperature allowed. Charging will stop / not start if above this.
BAT_TEMPERATURE_MIN	The lowest battery temperature allowed. Charging will stop / not start if below this.
BAT_CURRENT_PREQUAL	Charge current during prequalification mode.
BAT_CURRENT_HYST	Charge current hysteresis.
BAT_VOLTAGE_HYST	Charge voltage hysteresis.
BAT_VOLTAGE_PREQUAL	Target voltage during prequalification stage. If this voltage is not achieved the battery will be marked as worn out.
BAT_TIME_PREQUAL	Maximum amount of time to spend in prequalification stage.
DEF_BAT_CAPACITY	Default battery capacity.
DEF_BAT_CURRENT_MAX	Default maximum charge current.
DEF_BAT_TIME_MAX	Default maximum charge time.
DEF_BAT_CURRENT_MIN	Default cut-off charge current.
ALLOW_NO_RID	If defined, batteries without RID (or not matching the lookup-table) will cause the charger to use the battery defaults. Otherwise, charge is halted.
RID[ ].Low and RID[ ].High	Assume RID resistance match if value within these limits.
RID[ ].Capacity	Battery capacity for given RID.
RID[ ].Icharge	Charge current for given RID.
RID[ ].tCutOff	Maximum charge time for given RID.
RID[ ].IcutOff	Charge termination current for given RID.
NTC[ ]	Temperature look-up table.

4.7.2 Compilation

Both IAR and GCC project are available. The GCC project can use an external makefile (see Makefile in \gcc\default) or use the options defined in AVR Studio project.

Table 5-2. Compiler configuration

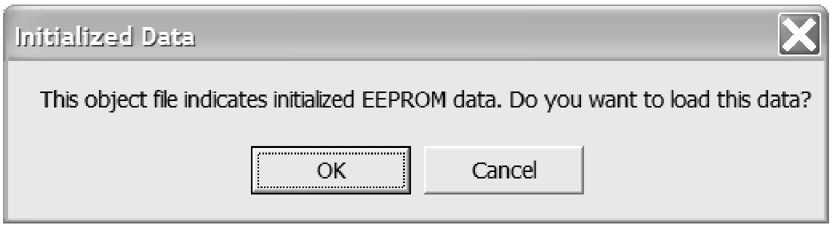
Section	Tab	Field	Value
General Options	Target	Processor configuration	ATmega16/32U4
		Memory model	Small
	System	Data stack	0x100
		Return address stack	32
		Enable bit definitions …	None
C/C++ Compiler	Language	Require prototypes	Selected
Linker	Output	Format	Other: ubrof8
	Extra Options	Command Line	-y(CODE) -Ointel-extended,(DATA)=EXE_DIR\$PROJ_FNAME$_data.hex -Ointel-extended,(XDATA)=EXE_DIR\$PROJ_FNAME$_eeprom.hex

4.7.3 Programming

The compiled code is conveniently downloaded to the target device using AVR Studio® and a debugger or programming tool of choice, such as the JTAGICE mkII.

Note that the compiled code is ready to contain EEPROM data if needed. This feature is only for further development. Answer OK when AVR Studio asks if EEPROM contents should be loaded. This is illustrated in the figure below.

Figure 5-9. Loading initialised data to EEPROM



Initialized Data

This object file indicates initialized EEPROM data. Do you want to load this data?

The program expects the use of the internal oscillator and that the clock signal is not prescaled. Some fuse bits must be programmed to ensure proper program execution. The fuse bit settings that deviate from the default factory configuration are listed in the table below.

Table 5-3. Non-default fuse bit settings

Fuse Bit	Setting	Description
CKDIV8	1 (unprogrammed)	Do not divide clock by eight
CKSEL3…0	0010	Use internal oscillator

On the EVK527 Rev1.0.0, the JTAG pins and Joystick buttons share the same IOs. It is the reason of the CDC key pressed application removal.

The HWB button is used to start the sending of data to the HyperTerminal.

After the download of software with AVR Studio, the ATmega16/32U4 bootloader is erased. If a download is needed by using FLIP (ATMEL ISP), a download of bootloader software (with AVR Studio) is needed.

5 EVK527 Rev1.0.0 Schematics

Figure 6-1. Page 1/5 (Schematics Rev3.0.0 corresponds to Board Rev1.0.0)

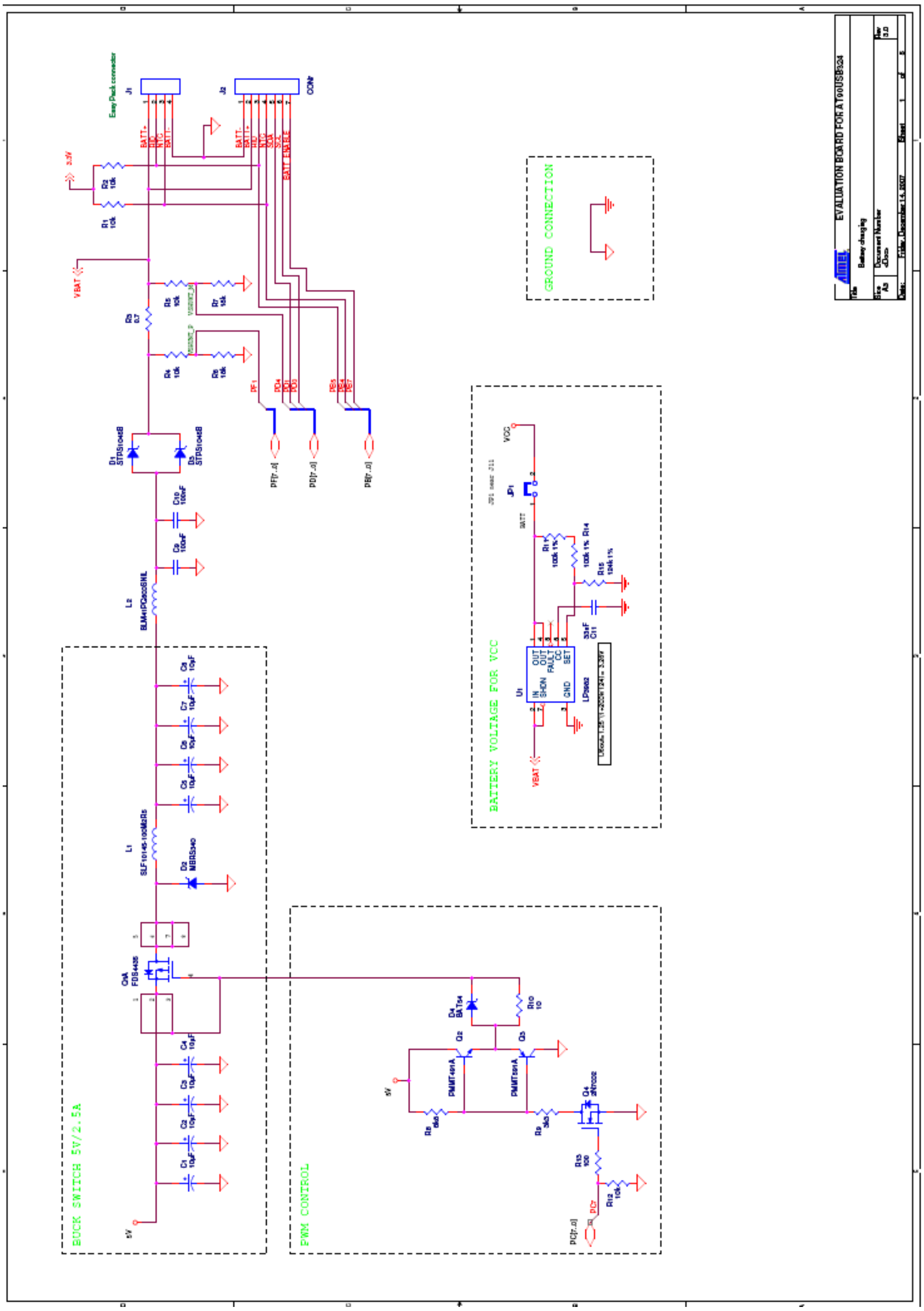


Figure 6-2. Page 2/5

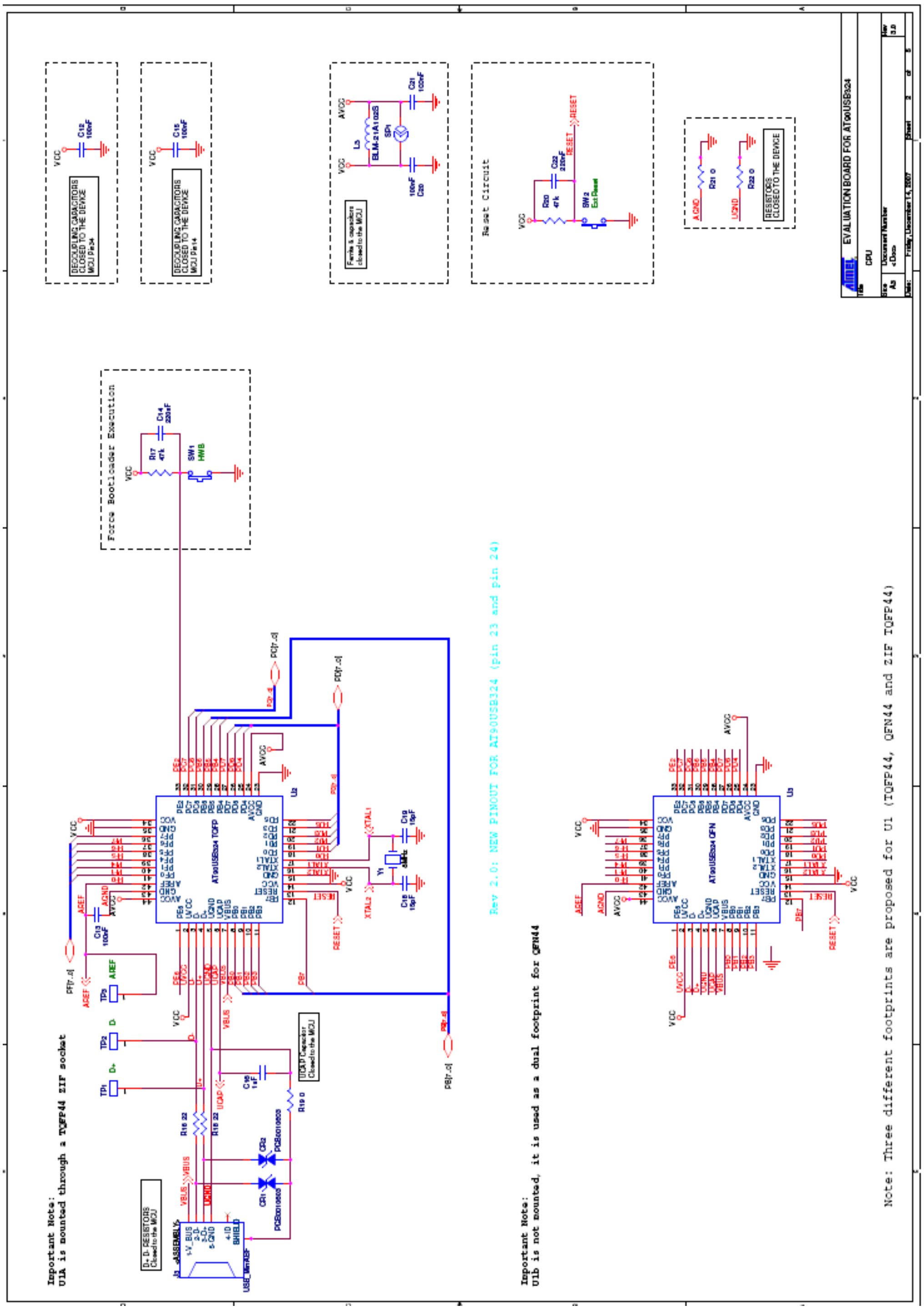


Figure 6-3. Page 3/5

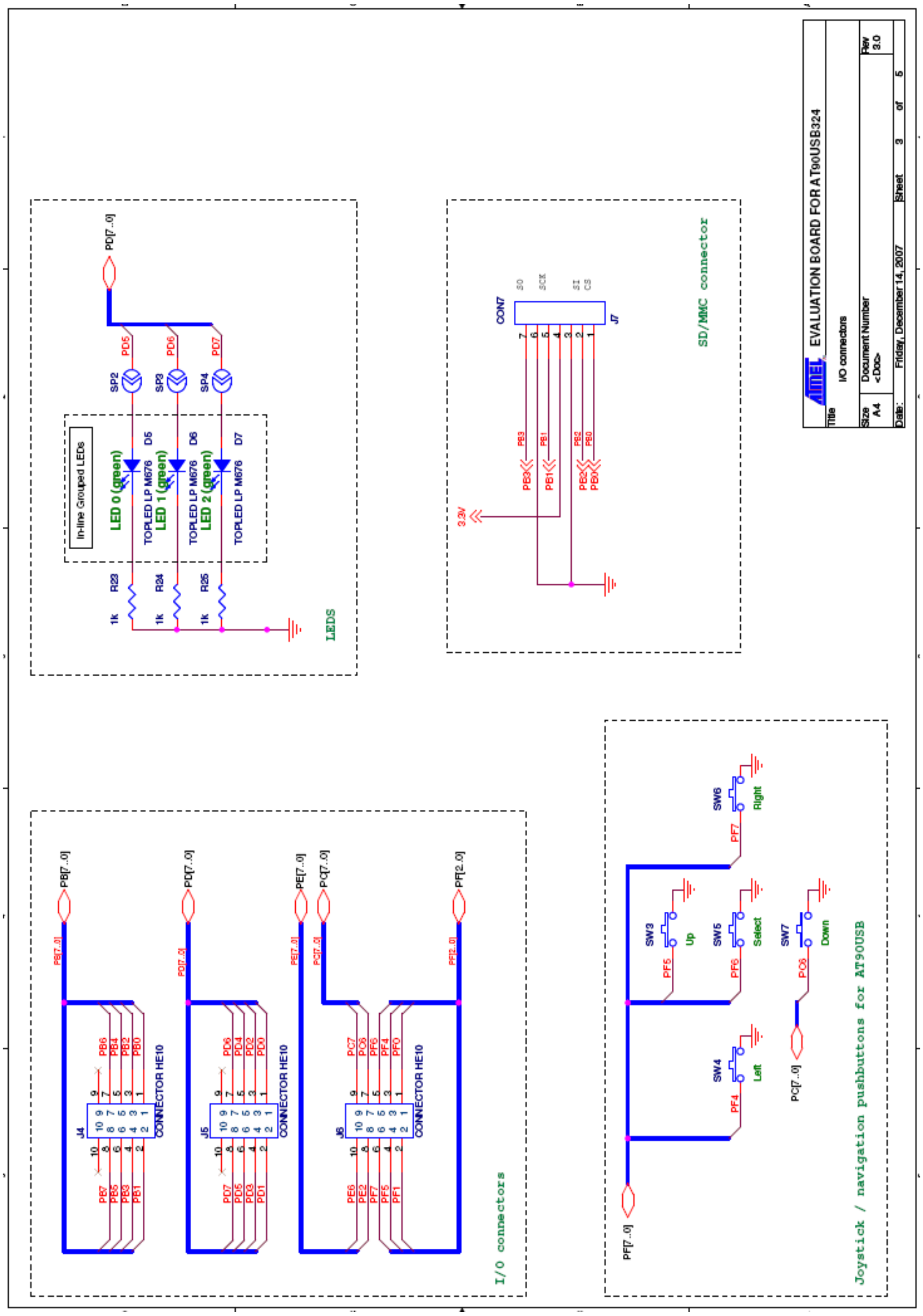


Figure 6-4. Page 4/5

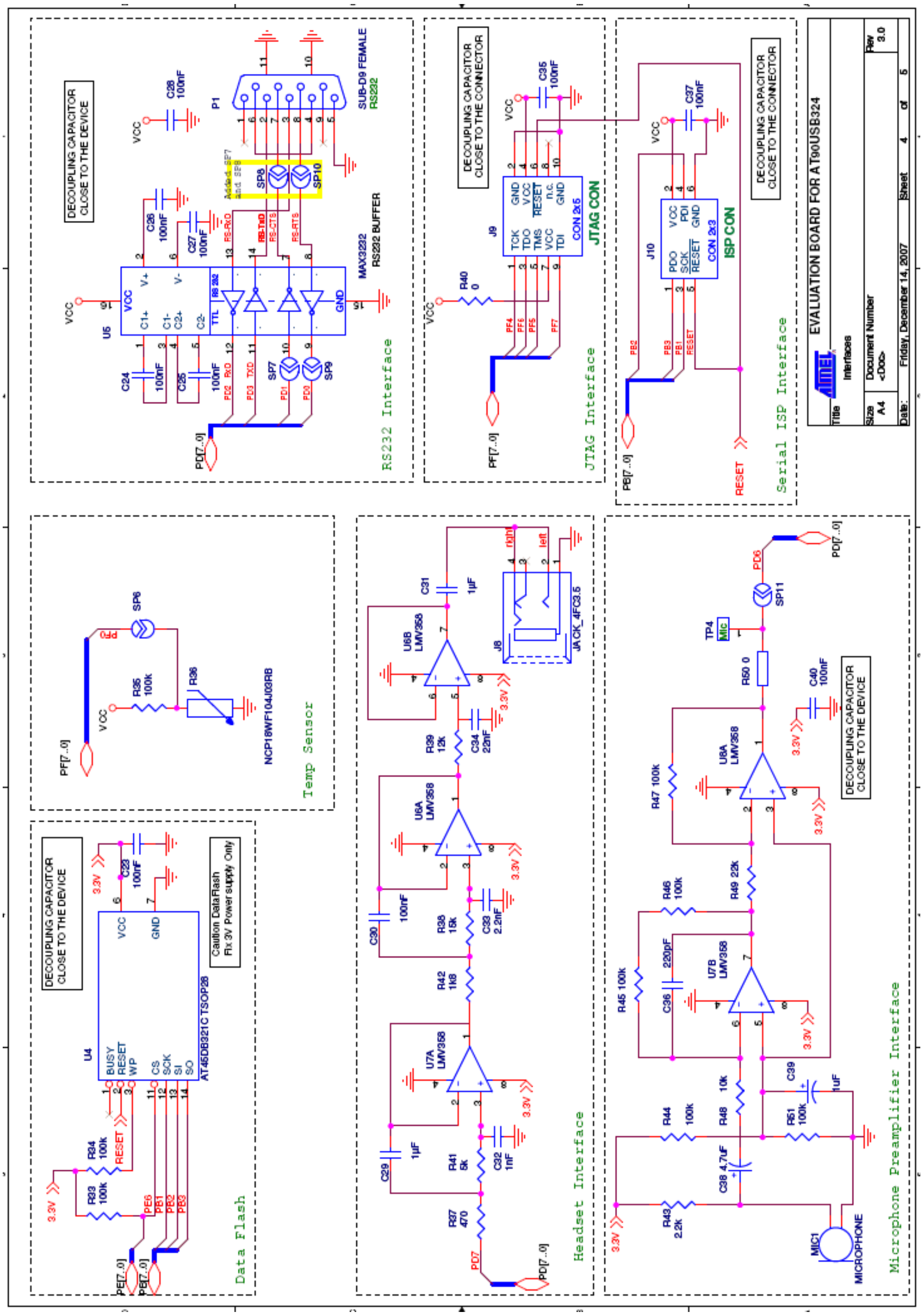
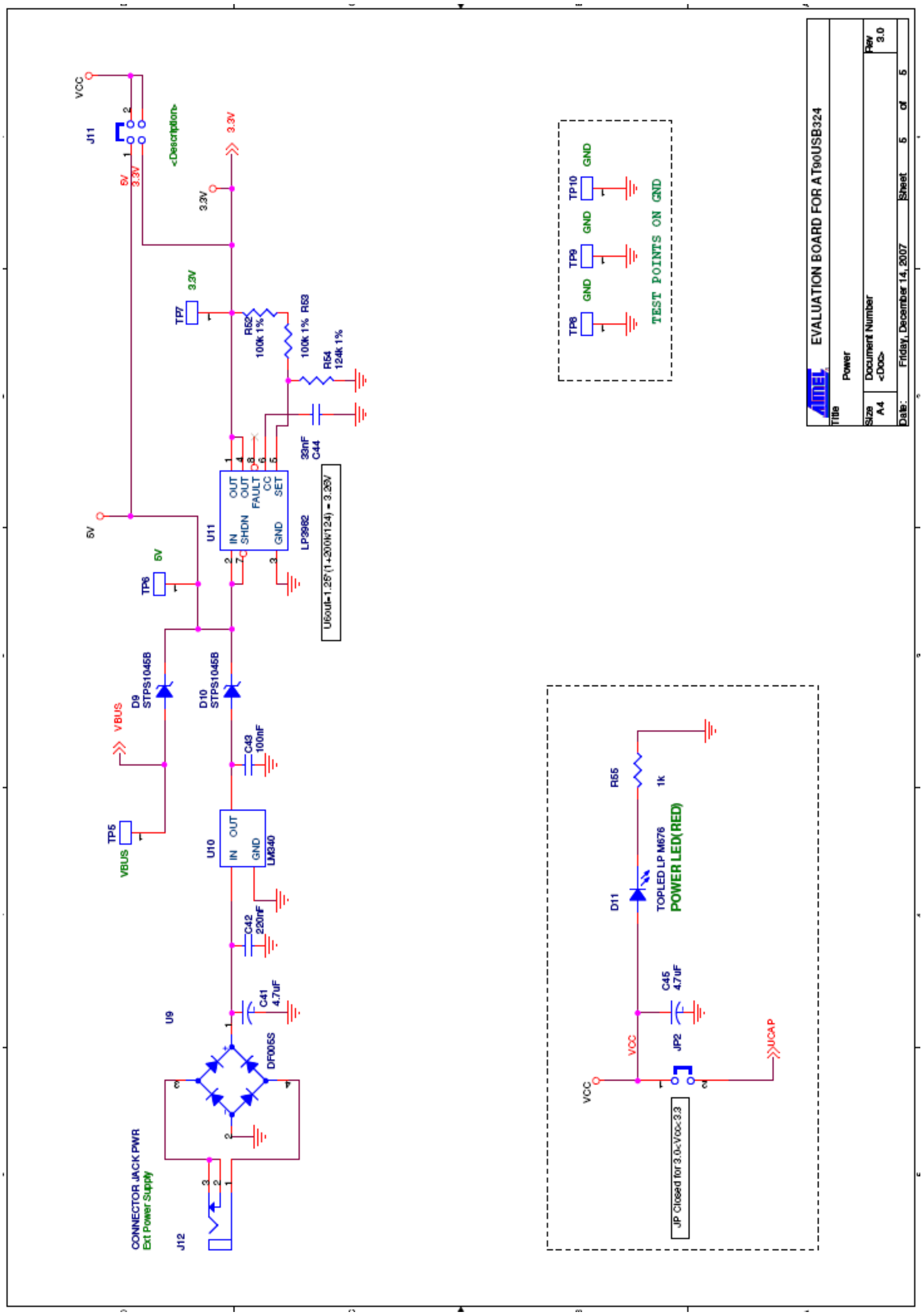


Figure 6-5. Page 5/5



6 References

1. "What's the best battery?". Retrieved April 3, 2007, from Battery University: http://www.batteryuniversity.com/partone-3.htm

2. "Lithium-ion safety concerns". Retrieved April 3, 2007, from Battery University: http://www.batteryuniversity.com/partone-5B.htm

3. "Charging lithium-ion batteries". Retrieved April 3, 2007, from Battery University: http://www.batteryuniversity.com/partone-12.htm

4. "VARTA™ EasyPack" datasheet:

   550mAh

   http://www2.varta-microbattery.com/en/mb_data/documents/promotion_varta_easypack/688819.pdf

   750mAh

   http://www2.varta-microbattery.com/en/mb_data/documents/promotion_varta_easypack/688820.pdf

   1000mAh

   http://www2.varta-microbattery.com/en/mb_data/documents/promotion_varta_easypack/688821.pdf

   2000mAh

   http://www2.varta-microbattery.com/en/mb_data/documents/promotion_varta_easypack/688822.pdf

5. "ATmega32U4". Available from Atmel web site: http://www.atmel.com/products/avr/

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

©2008 Atmel Corporation. **All rights reserved.** Atmel®, logo and combinations thereof, AVR®, AVR Studio® and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.