AVR458: Charging Lithium-Ion Batteries with ATAVRBC100

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**
- **Serial Interface for Communication with External Master**
- **One-wire Interface for Communication with Battery EEPROM**
- **Analogue Inputs for Reading Battery ID and Temperature**
- **Internal Temperature Sensor for Enhanced Thermal Management**
- **On-chip EEPROM for Storage of Battery and Run-Time Parameters**

1 Introduction

This application note is based on the ATAVRBC100 Battery Charger reference design (BC100) and focuses on how to use the reference design to charge Lithium-Ion (Li-Ion) batteries. The firmware is written entirely in C language (using IAR® Systems Embedded Workbench) and is easy to port to other AVR® microcontrollers.

This application is based on the ATtiny861 microcontroller but it is possible to migrate the design to other AVR microcontrollers, such as pin-compatible devices ATtiny261 and ATtiny461. Low pin count devices such as ATtiny25/45/85 can also be used, but with reduced functionality.

8-bit AVR® Microcontrollers

Application Note

Rev. 8080A-AVR-09/07

2 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.

2.1 Li-Ion Battery Technology

Lithium-Ion batteries have the highest energy/weight and energy/space ratios of modern rechargeable batteries /1/ (See “References” section on page 29). 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 /2/. 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.

2.1.1 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 (positive 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.

2.2 Charging Li-Ion Batteries

There is only one way to charge lithium-based batteries /3/. 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, 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. The battery takes damage from overcharging and may explode if overcharged.

2.2.1 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.

2.2.2 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 10^{th} or the 100^{th} 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.

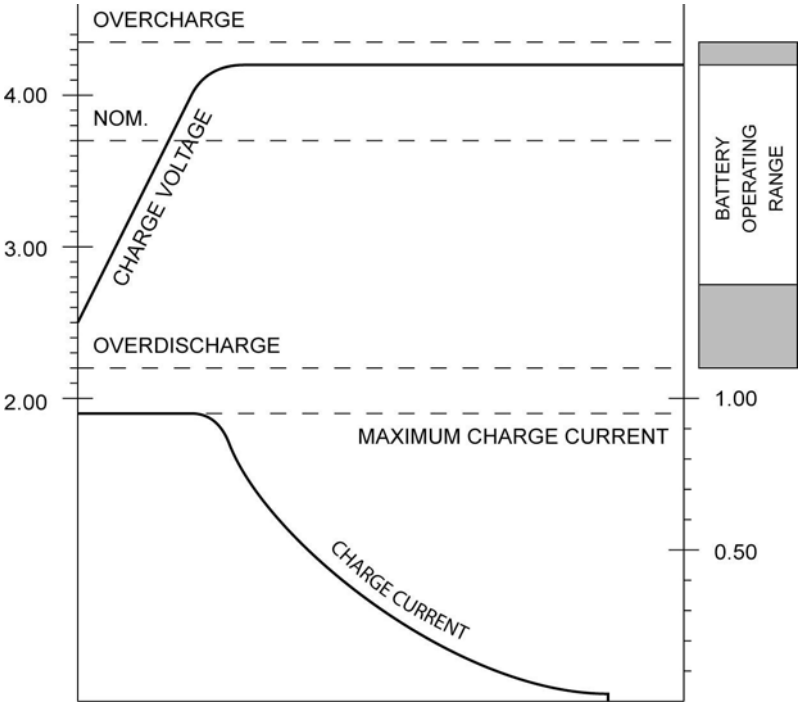
2.2.3 Charge Stages

There are two charge stages of a Lithium-Ion battery, as follows:

1. Constant current. Charging of a Li-Ion battery starts 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.
2. 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 1-1. Charge stages and limits of a Varta PoLiFlex® cell



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. Recommended battery operating voltage is typically a margin away from overcharge and discharge limits.

2.2.4 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 1-1. Typical Charge Characteristics

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

2.2.5 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 1-2. Manufacturer's data for Varta PoLiFlex range of lithium-ion batteries /4/

Parameter	PLF 443441	PLF 383562	PLF 503562	2P/PLF 503562	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 (resistor ID)	3.9	6.8	10	24	kΩ
NTC	10				kΩ
B-value	3435				K
Overcharge detection	4.35				V
Overdischarge detection	2.20				V

2.3 Battery Charger

This application note is based on the ATAVRBC100 Battery Charger reference design by Atmel®. The reference design is rather complex and has loads of features but this application focuses on the low end of the design, only. For more information on the BC100, please see AVR451 - BC100 Hardware User's Guide /5/.

2.3.1 Microcontroller

The BC100 hosts two microcontrollers; a master (ATmega644, by default) and a slave (an ATtiny25/45/85 or ATtiny261/461/861, by default). The master microcontroller is outside the scope of this application but it may be noted that the microcontrollers are capable of communicating with each other such that the master may request data from the slave at any time.

The slave microcontroller is fully capable of handling all tasks related to battery charging and it does not require a master microcontroller to be present. It constantly scans the connectors for batteries and, if found, charges them when required. The slave microcontroller also constantly monitors the hardware for any anomalies.

2.3.2 Power supply

This application note does not focus on the power supply. It may, however, be noted that the firmware constantly monitors the input voltage levels in order to make sure operation is reliable.

2.3.3 Buck switches

The firmware on the slave microcontroller controls any of the three buck switches on board the BC100. The default is to use a high-frequency PWM output of the microcontroller to adjust the voltage and current flow to the battery. The voltage (and current) of the buck switches are directly proportional to the duty cycle of the PWM signal.

3 Battery Charger Hardware

This application note is based on the ATAVRBC100 Battery Charger reference design. A detailed hardware description will not be provided in this document. Please see AVR451 - BC100 Hardware User's Guide for detailed information.

3.1 Configuration

The ATAVRBC100 Battery Charger reference design must be configured as detailed below.

3.1.1 Microcontroller

The hardware should be populated as follows:

- Make sure socket SC300 is empty
- Populate socket SC301 with an ATtiny861

It is possible to use other AVR microcontrollers but this application has been optimised for using ATtiny861. Pin compatible replacements such as ATtiny261 and ATtiny461 /6/ may be used if the compiled code size is decreased. This can be done by increasing the optimisation of the compiler and by removing unwanted features from the firmware.

Other microcontroller options include ATtiny25, ATtiny45 and ATtiny85 /7/. These (as well as other 8-pin AVR microcontrollers) use the SC300 socket on BC100. It should be noted that due to reduced pin count the 8-pin microcontrollers provide less features than the default 20-pin.

3.1.2 Programming Connector

The microcontroller can be programmed via 6-pin connector J301, using either SPI or debugWIRE.

Please note that in some hardware revisions of BC100 it may be necessary to remove R303 and disconnect pin 15 of U202. This procedure frees the /RESET line for use by external programmer or debugger but removes the possibility for the master microcontroller to reset the slave. Do not engineer the board unless required. Alternatively, the microcontroller can always be programmed off-board.

3.1.3 Jumpers

The jumpers should be configured as follows:

- J400, J401, J407 & J408: Set jumpers to use Buck Switch C (20V / 1A)
- J405 & J406: Set jumpers to 1/4 (max measurable voltage 10V)

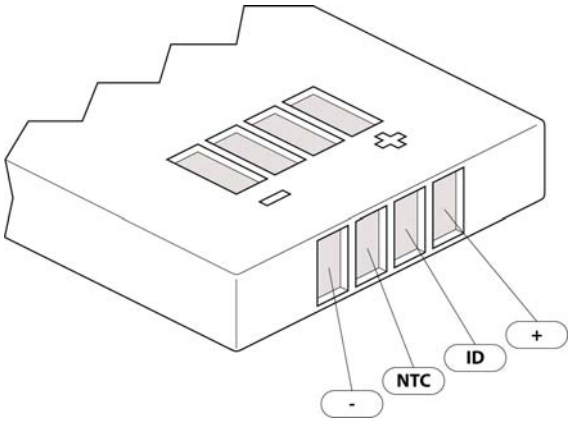
Other configurations are possible, but may require firmware changes. See variable VBAT_RANGE in file ADC.h.

3.1.4 Battery

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 done to firmware and hardware. See section 4.5.1 and file battery.h.

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

Figure 1-2. Connection pads of a Varta PoLiFlex cell.



The battery is connected to the battery charger as follows.

Table 1-3. 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+	

3.1.5 Data EPROM

Some batteries are equipped with an embedded EPROM for storing charge and manufacturing data. This application supports the use of EPROM via a one-wire interface. The default is a DS2502 EPROM connected as follows.

Table 1-4. Connecting external EPROM DS2502 to charger

EPROM Pin	Charger Connector
DATA	1-WIRE/SDA
GND	BATTERY-

If an EPROM is not connected to the battery charger the application will simply disregard its absence.

3.1.6 Supply Voltage

The higher the supply voltage, the higher the minimum current the buck switches can provide. For example, if supply voltage is about 9 V and buck charger C is used to charge a battery at 4.20 V then the minimum attainable current is about 80 mA. At this point the smallest decrease in PWM duty cycle (i.e. reducing the contents of OCR1B by 1) will effectively turn off the current to the battery.

It is recommended to use a supply voltage some three volts above battery charge voltage. In this application the battery is being charged at 4.20 V and the recommended supply voltage is therefore 7.5 V.

Another method to lower the minimum charge current the hardware can provide is to use a buck switch with a large inductor. In BC100 this means Buck Switch A.

4 Battery Charger Software

The firmware is written in C language using IAR Systems Embedded Workbench, version 4.20. 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 1-5. Project files (see IAR EW workspace file BC100_tiny.eww)

File	Type	Note
ADC.c	C source code	Functions related to A/D converter
ADC.h	Header file	
AVR458.c		Functions related to the different states and charging
AVR458.h		
battery.c	C source code	Battery-specific definitions and functions related to battery control & data acquisition
battery.h	Header file	
main.c	C source code	Main program / Program entry point
main.h	Header file	
menu.c	C source code	State machine definitions
menu.h	Header file	
OWI.c	C source code	Functions related to one-wire interface
OWI.h	Header file	
PWM.c	C source code	Functions related to generating pulse-width modulated output
PWM.h	Header file	
time.c	C source code	Functions related to timekeeping and measurement of time
time.h	Header file	
USI.c	C source code	Functions related to serial interface
USI.h	Header file	

4.1 Overview

The firmware integrates all functions required to charge two lithium-ion batteries. Batteries are connected to separate ports such that one may be charged while the other is idle. The firmware is fully automated and capable of stand-alone battery monitoring and charging but it may also be used together with a master microcontroller, such as the one implemented in BC100.

By default, the firmware fits into an ATtiny861 (build option: debug) or an ATtiny461 (build option: release). Memory requirements of the firmware are summarised in the table below.

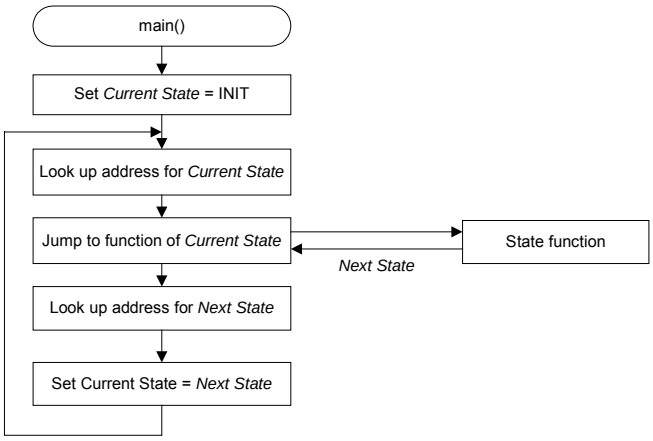
Table 1-6. Memory requirements of firmware

Build option	Memory	Approximate value
Debug	CODE (Flash)	5800 bytes
	DATA (SRAM)	270 bytes
	XDATA (EEPROM)	130 bytes
Release	CODE (Flash)	3900 bytes
	DATA (SRAM)	270 bytes
	XDATA (EEPROM)	130 bytes

4.2 State Machine

The state machine is rather simple and resides in the main() function. 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 1-3. 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 1-7. State machine codes (see source code, menu.h)

Label [1]	Related Function [2]	Description
INIT	Initialize()	Entry state
BATCON	BatteryControl()	Check hardware and batteries
PREQUAL	Charge()	Raise battery voltage, safety check
SLEEP	Sleep()	Low power consumption mode
CCURRENT	Charge()	Charge with constant current
CVOLTAGE	Charge()	Charge with constant voltage
ENDCHARGE	Charge()	End of successful charge
DISCHARGE	Discharge()	
ERROR	Error()	Resolve error, if possible

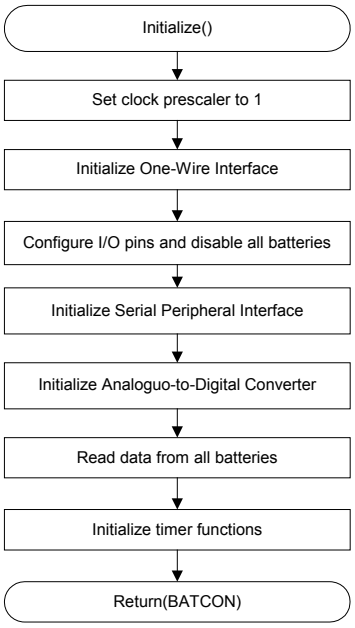
Notes: 1. Name of label, excluding leading “ST_”
2. Function name, as declared in source code

State functions are described in the following sections.

4.2.1 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 1-4. Flow chart of initialisation function

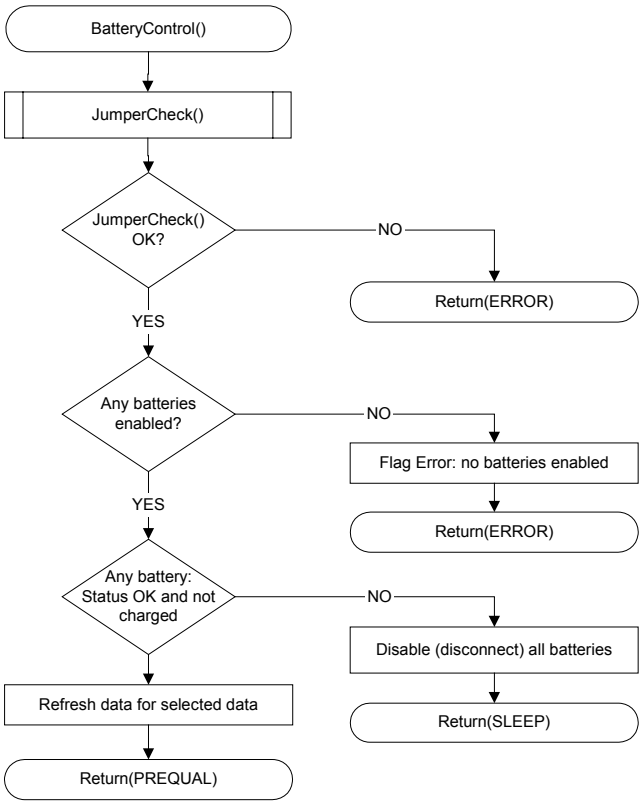


The initialisation function always exits with the same return code, pointing to the state function for battery control.

4.2.2 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 1-5. Flow chart of battery control function



4.2.3 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 sinks beneath 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 central variables in this function. The program flow of this state function is illustrated in the figure below.

Figure 1-6. Flow chart of the charge state function

4.2.4 Discharge()

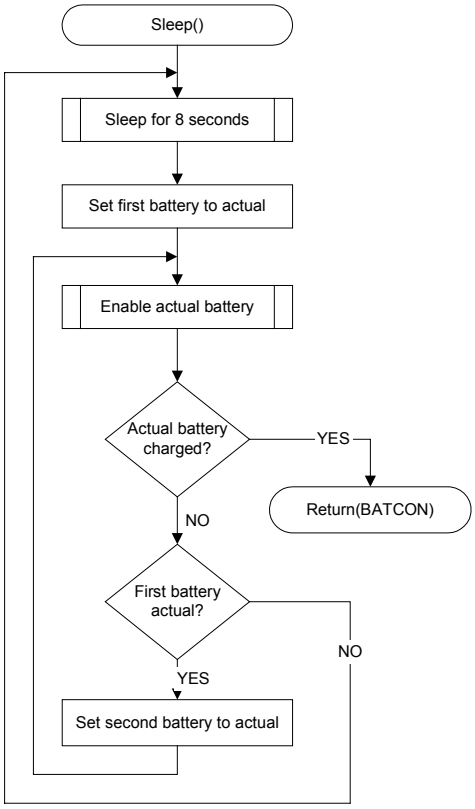
This function has not been implemented.

4.2.5 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 1-7. Flow chart of sleep function

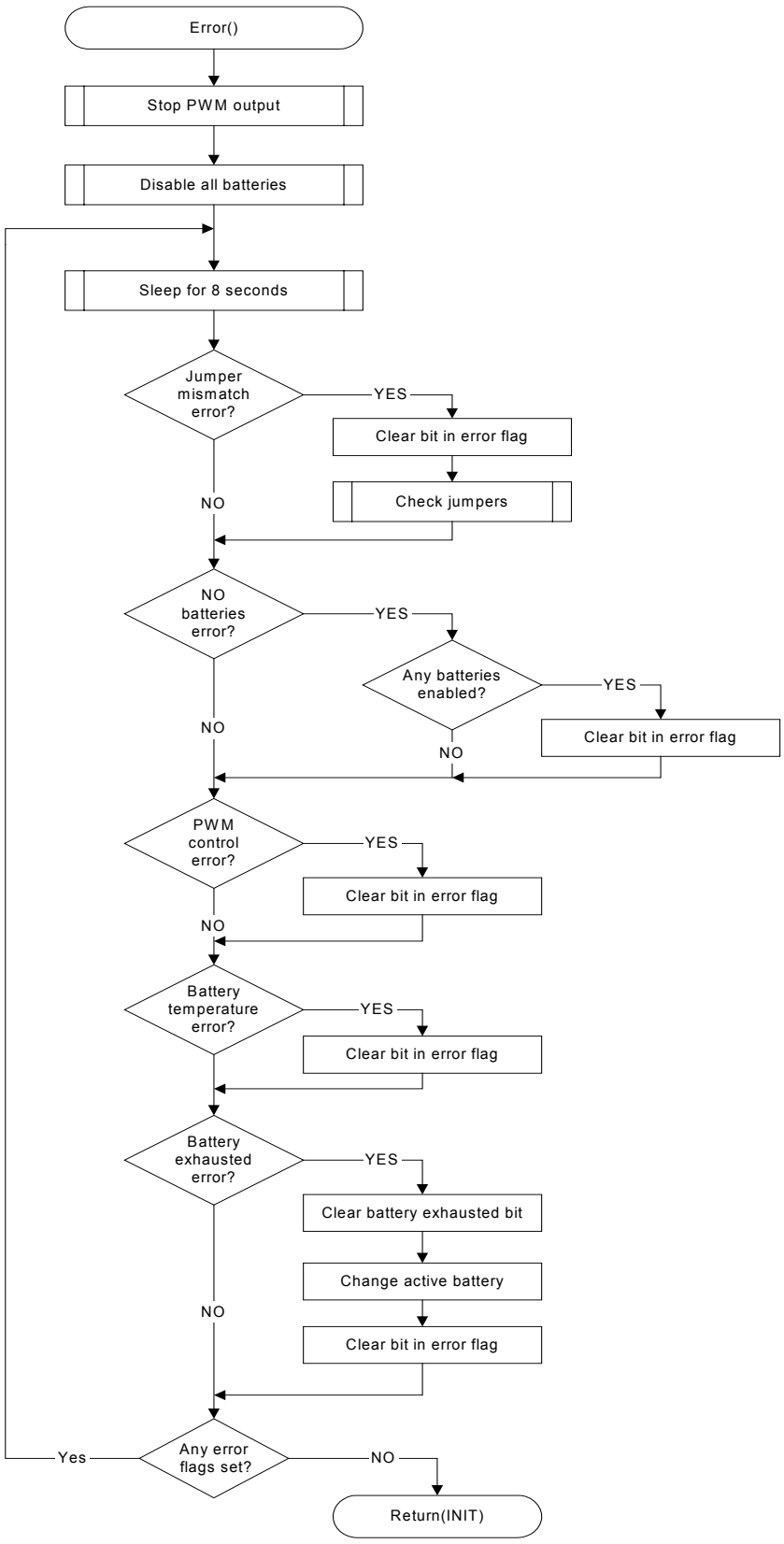


4.2.6 Error()

Program flow is diverted here when an error has occurred. The error handler contains some simple algorithms that try to resolve the most common problems. 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 1-8. Flow chart of error handler



4.3 Charging Functions

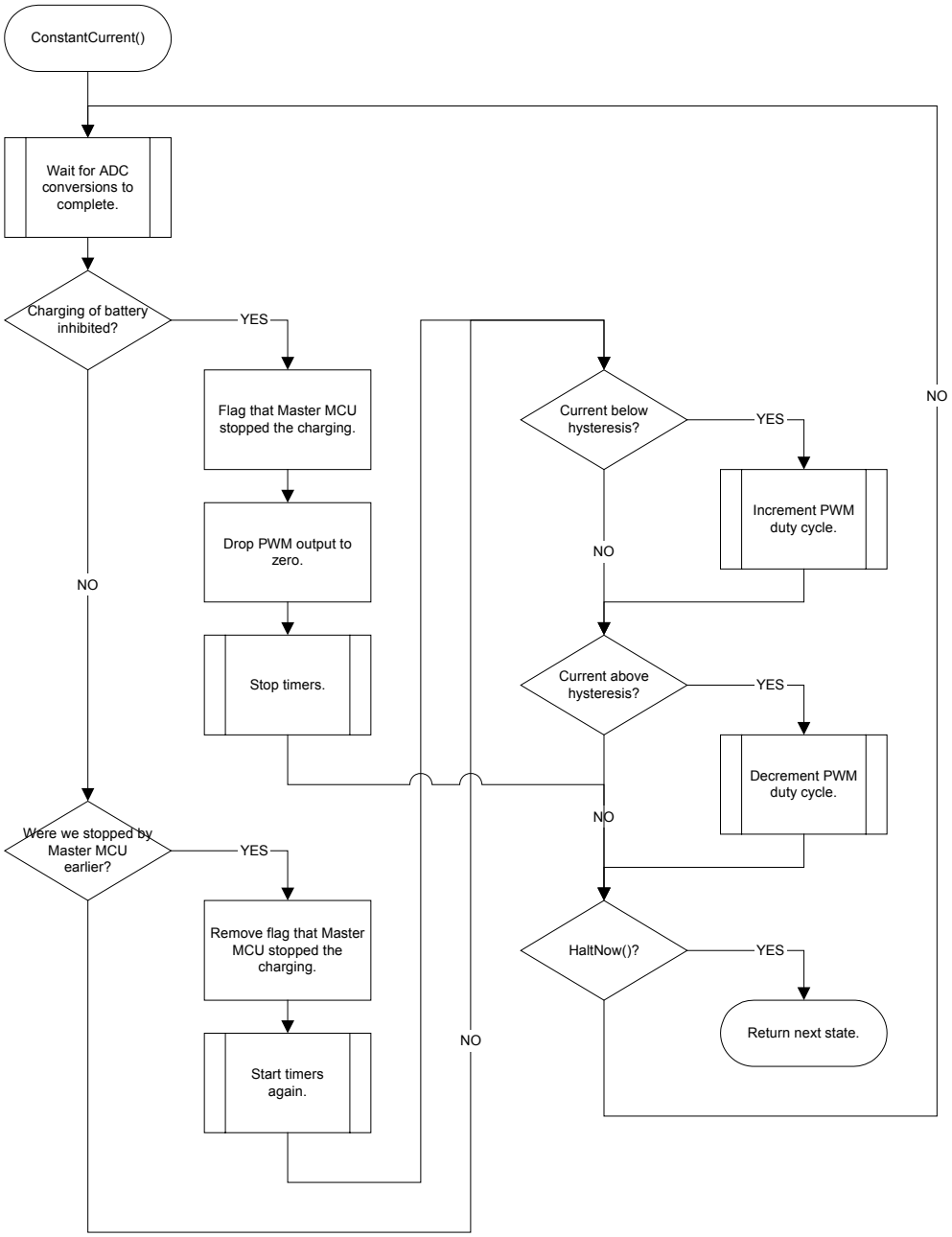
These functions are called by Charge() after all parameters have been set.

4.3.1 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.

If a Master microcontroller is present, it may temporarily stop the charging by flagging a charge inhibit. This is to prevent battery damage during prolonged serial transfers.

Figure 1-9. Flow chart for ConstantCurrent()



4.3.2 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 f.ex. 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.

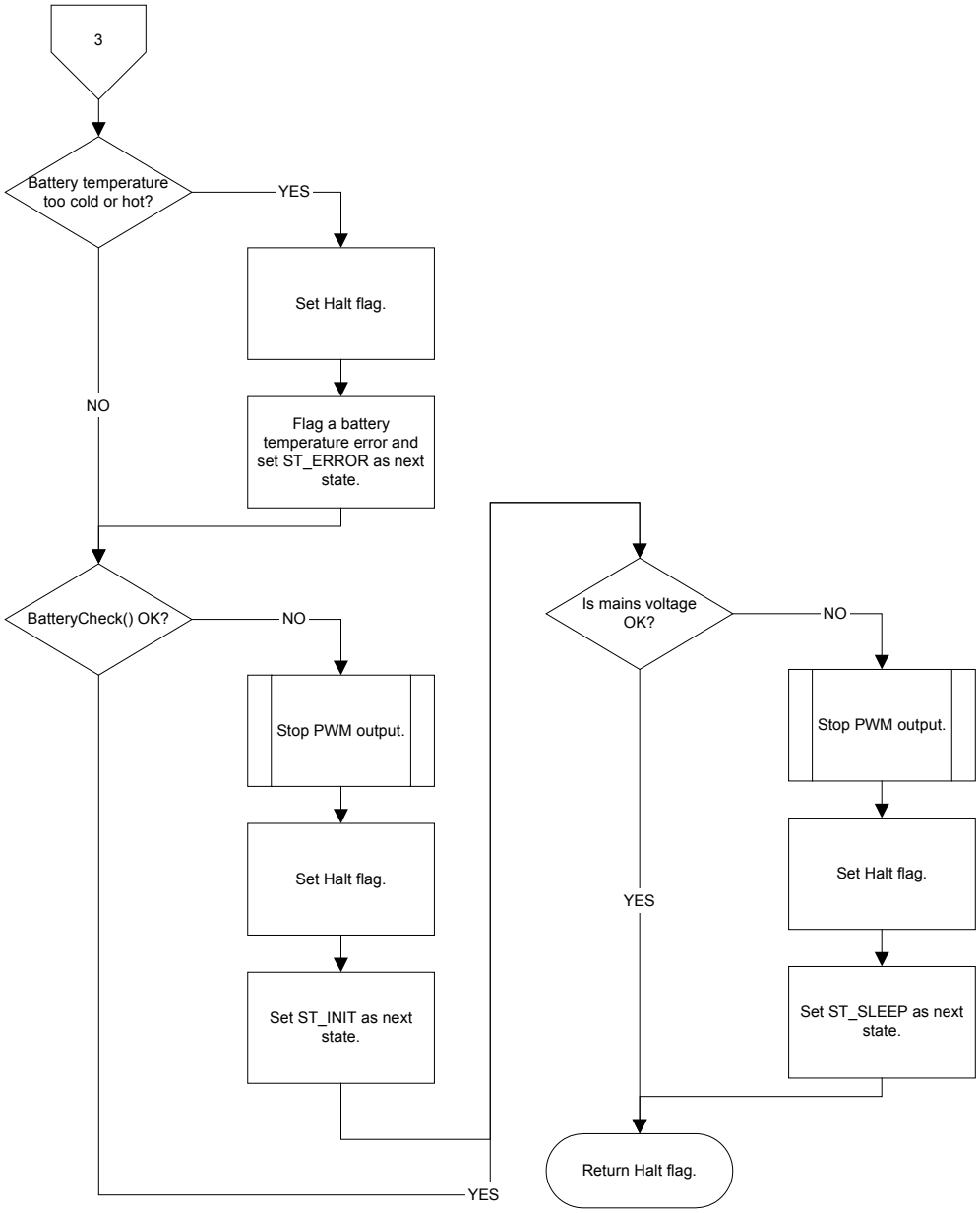
Lastly, the function checks if temperature is within limits, if the battery is OK and if mains voltage is above minimum. Should any of these tests fail, the next state is set to an appropriate error handler (ST_ERROR, ST_INIT or ST_SLEEP) and charging is aborted.

Figure 1-10. Flow chart for HaltNow() part 1.

Figure 1-11. Flow chart for HaltNow() part 2

Figure 1-12. Flow chart for HaltNow() part 3

Figure 1-13. Flow chart for HaltNow() part 4



4.4 Other Functions

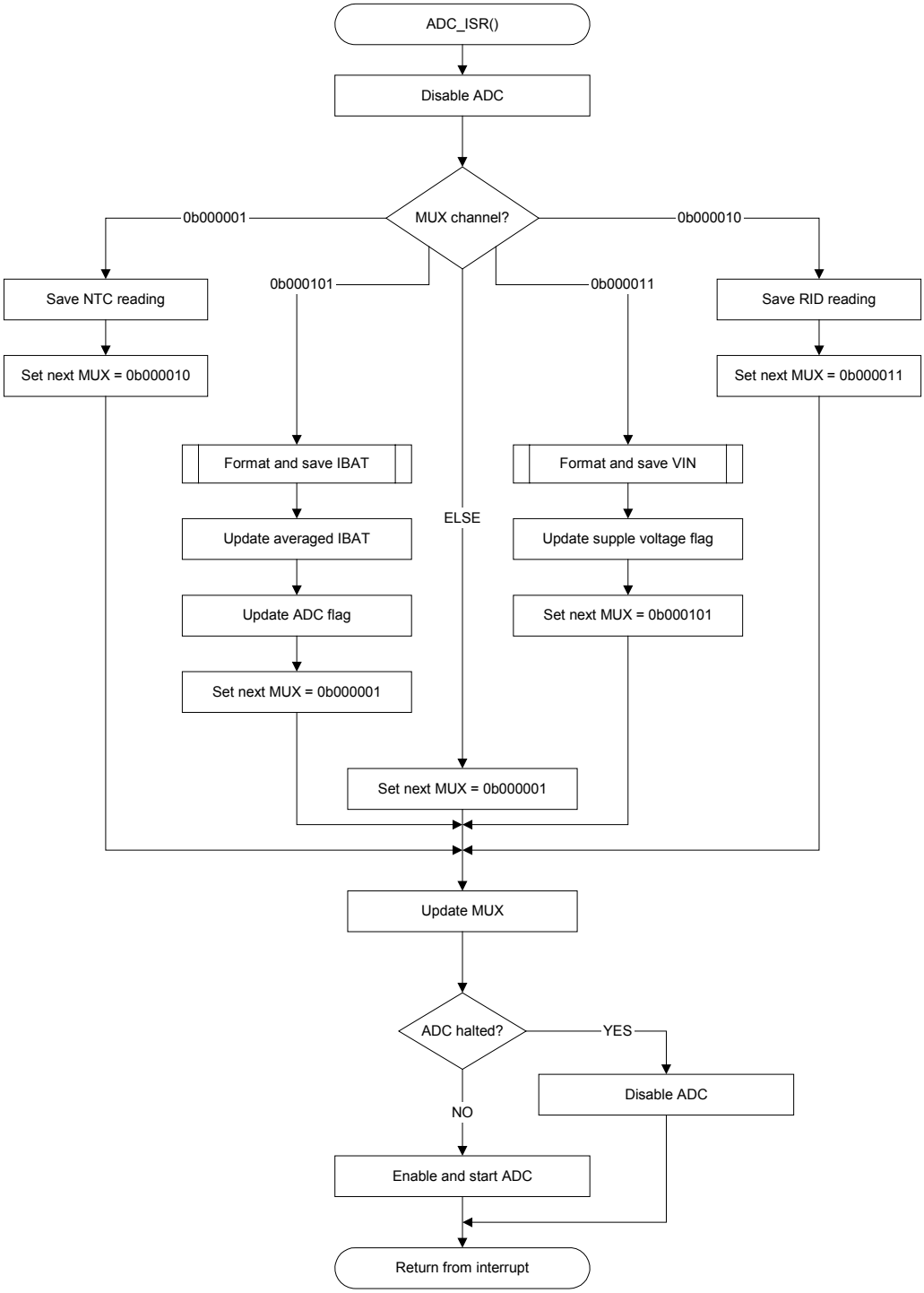
The program flow is mainly state-based, but some processing takes place in the background. This includes A/D conversion, time keeping and serial interface handling. All of these functions are interrupt-driven.

4.4.1 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.

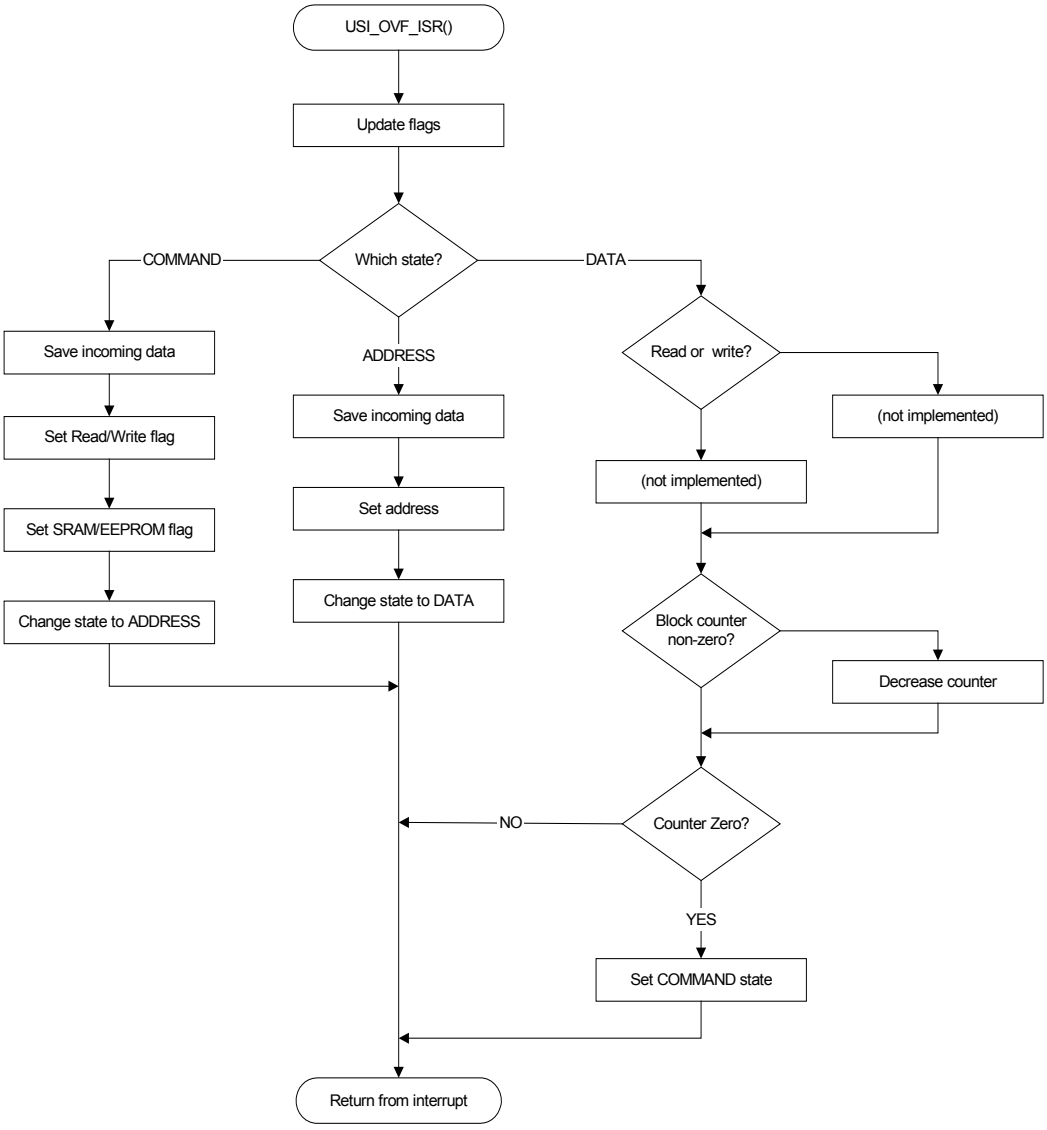
Figure 1-14. Flow chart of ADC interrupt service routine



4.4.2 Master-Slave Communication

This application is designed to work as stand-alone but it also supports co-operation with other microcontrollers. The Universal Serial Interface (USI) can be used for communication between microcontrollers. The basic protocol for this interface has been developed but some functions need to be finalised.

Figure 1-15. Flow chart of USI overflow interrupt service routine



4.5 Implementation

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

4.5.1 Configuration

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

Table 1-8. Battery-related compile-time constants (see source file battery.h)

Label	Description
BAT_CELL_NUMBER	The number of cells in the battery. Each of the defined cell voltages gets multiplied by this, to define BAT_VOLTAGE_MAX, _LOW, _MIN and _PREQUAL.
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 above this.
BAT_CURRENT_PREQUAL	Charge current during prequalification mode.
BAT_CURRENT_HYST	Charge current hysteresis. Current will not be adjusted when within plus or minus this value from target.
BAT_VOLTAGE_HYST	Charge voltage hysteresis. Current will not be adjusted when within plus or minus this value from target.
BAT_VOLTAGE_PREQUAL	Target voltage during prequalification stage. If this voltage is not achieved the battery will be marked as exhausted.
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.

4.5.2 Compilation

Before compiling the code the following configurations should be made.

Table 1-9. Compiler configuration

Section	Tab	Field	Value
General Options	Target	Processor configuration	ATtiny861 (1)
		Memory model	Small
	System	Data stack	0x40
		Return address stack	24
		Enable bit definitions …	Selected
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

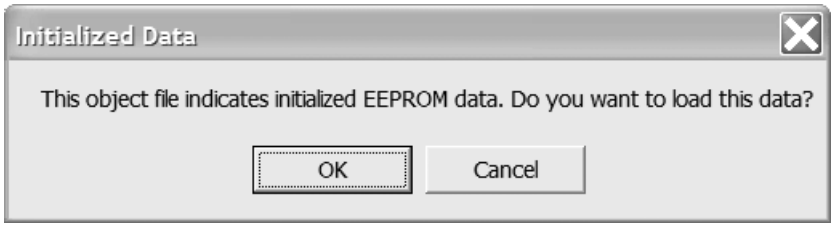
Notes: 1. Other options possible. See section 3.1.1 on page 7 for more information.

4.5.3 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 contains EEPROM data that must be loaded to the target for the software to work. Answer OK when AVR Studio asks if EEPROM contents should be loaded. This is illustrated in the figure below.

Figure 1-16. Loading initialised data to EEPROM



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 are listed in the table below.

Table 1-10. Non-default fuse bit settings

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

5 Known Limitations

Here are listed known limitations of the design.

5.1 Battery Current Measurement

Battery current is sensed using a shunt resistor with very low resistance. This means noise is easily picked up in the measured signal and that even noise with very low amplitude may disturb the measurements. As a remedy, the battery current measured is averaged over four samples.

Yet, it is not uncommon to find fluctuations in the order of 1 or 2 LSB. By default (see section 3.1.3) this means a measurement error of 7 or 14 mA (see function ScaleI() in file ADC.c). In practice, this may result in premature end of charge cycle.

The suggested solution is to optimise the size of the shunt resistor (R410: the larger, the better) and the resistor divider (R400…R410, R427, R428, R446 and R447).

5.2 RID Sensing

Battery identification resistor is sensed via pin PA2 (ADC2). The default pull-up resistor on this line (R305 in ATAVRBC100 Battery Charger reference design) is 4.7 kohm. This limits the size of the sense resistor to TBD ohm.

When using Varta PoLiFlex batteries this means the largest battery size that can be reliably sensed is 1000 mAh. For larger sense resistors / battery sizes the pull-up resistor on BC100 must be changed. In addition, the software must be updated to reflect the new pull-up resistor value.

5.3 Buck chargers

The choice of buck charger (and supply voltage) sets a limit on how low the minimum charge current may be. The higher the supply voltage and the smaller the buck switch inductor, the higher will the minimum charge current be. This means some configurations may result in premature end of charge cycle.

The remedy is to use a low supply voltage and a buck switch with a large inductor.

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 PoLiFlex Sales Program and Technical Handbook”. Retrieved May 10, 2007, from VARTA Microbattery: http://www.varta-microbattery.com/en/oempages/index.htm

5. “AVR451 - BC100 Hardware User's Guide”. Available from Atmel web site: http://www.atmel.com/products/avr/

6. “ATtiny261/461/861 Data Sheet”. Available from Atmel web site: http://www.atmel.com/products/avr/

7. “ATtiny25/45/85 Data Sheet”. 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

8080A-AVR-09/07