AN1384

Ni-MH Battery Charger Application Library

Author: *Mihnea Rosu-Hamzescu*
Microchip Technology Inc.

INTRODUCTION

The Battery Charger Application Library easily allows adding battery charging functionality to portable applications. Since it is very compact (uses less than 2k words of program space and less than 128 bytes of RAM) it fits on small, cost-effective parts like the 14-pin PIC16F616.

Hardware requirements (for the basic 2-slot charger):

- 2k Words Program Space
- 128 Bytes RAM
- 1 Enhanced PWM module with Auto-Shutdown (uses a 16-bit timer)
- 1 16-Bit Timer for the 125+ ms Tick
- 1 Comparator (for PWM pulse-by-pulse current limiting)
- 1 Band Gap Type Voltage Reference for Compensating Supply Variations (or external ADC reference)
- 1 CVREF module (or any other DAC type) for setting the Current Limit
- (10-bit Minimum) AD Converter:
  - 2 channels for battery voltage readings
  - 1 channel (shared with the comparator input) for reading average current value
  - 2 channels for temperature readings (optional); currently the fast charge termination algorithms are voltage-based (if temperature slope is used, they are mandatory)
- 2 Output Capable IO Ports to Drive Battery Switch Transistors
- 2 Output Capable IO Ports for LED Status Signaling (optional)
- 1 Output Capable IO Port for UART Debugging (optional)

The basic charger has a single power source in a buck converter configuration with current feedback. To charge both batteries, it switches between the batteries every second. When a battery is “active”, its voltage and current readings are updated, then the state machine uses the readings to decide the next step in the charging process. If debugging is enabled, the readings are sent to the PC for logging. The average current received by each battery is the current set value divided by the number of slots.

THE NICKEL-METAL HYDRIDE CELL

The nickel-metal hydride cell has become widespread in many high-end portable electronic products where battery performance is a major consideration. First adoption of the nickel-metal hydride cell had occurred in the cellular phone and portable computer markets. Currently, most portable electronics are powered by either nickel-metal hydride or Li-Ion batteries. As production volume increased, the metal hydride cells have replaced nickel-cadmium cells in most applications, with a few possible exceptions in specialty niches.

COMPARISON OF NI-MH AND NI-CD CELLS

The technology for metal hydride cells is essentially an extension of the sealed nickel-cadmium technology. The negative cadmium-based electrode is substituted by a hydrogen-absorbing alloy. This substitution increases the cell electrical capacity for the same weight and volume and eliminates heavy metal toxicity concerns, while the rest of the metal hydride cell is quite similar to the cadmium-based product. Exchanging the cell types in a design usually involves a few significant design changes.

© 2011 Microchip Technology Inc.

TABLE 1: COMPARISON BETWEEN NI-MH AND NI-CD CELLS

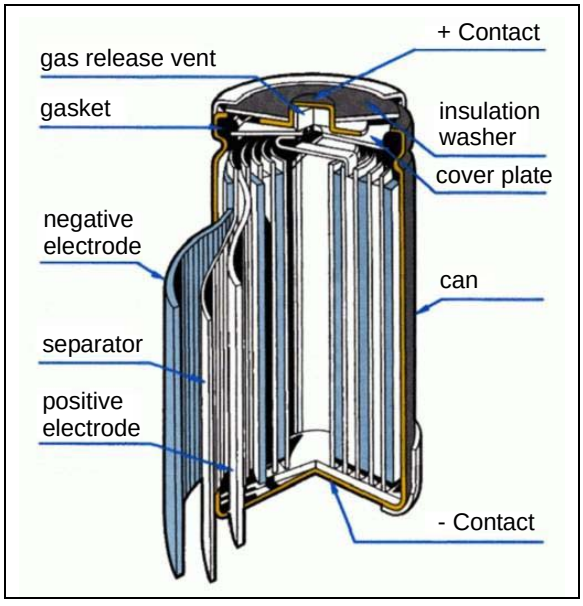
Feature	Ni-MH	Ni-Cd
Nominal voltage	Same, 1.25V	
Discharge capacity	Up to 40% higher	
Discharge profile and cut-off voltage	Equivalent, 0.9V	
High rate discharge capabilities	Essentially the same, higher for Ni-Cd	
Charging process	Similar; multi-step constant current with overcharge control	
Charge termination	Generally similar; Ni-MH transitions are more subtle and multiple termination protocols recommended	Generally similar; Ni-MH chargers should work for Ni-Cd cells
Self-discharge rate	Higher for Ni-MH	
Cycle life	Similar, but Ni-MH more application dependent	Similar but more resistant to abuse
Mechanical properties	Equivalent	
Environmental issues	Reduced with Ni-MH; Nickel is a mild toxin	Cadmium heavy metal toxicity concerns

PRINCIPLE AND STRUCTURE OF THE NI-MH BATTERY

The Hydrogen Absorbing Alloy

The negative electrode of a metal hydride cell is constructed from a special hydrogen-absorbing alloy. This alloy can absorb as much as 1000 times its own volume in hydrogen gas, which allows it to become a metal hydride. The alloy can also reversibly release the gas it has stored. Different manufacturers have developed many hydrogen-absorbing alloys used in battery manufacturing.

FIGURE 1: CYLINDRICAL CELL CONSTRUCTION



Electrochemistry of the Ni-MH battery

The nickel-metal hydride makes use of the reversibility of the hydrogen absorption/release in the special hydrogen-absorbing alloy. The positive electrode of the battery uses nickel oxide while the negative electrode uses a hydrogen-absorbing alloy. The electrolyte is an alkaline solution, containing a few substances like potassium hydroxide (KOH).

EQUATION 1: ELECTROCHEMISTRY

The Positive Electrode:

$$NiO(OH) + H_2O + e^- \leftrightarrow Ni(OH)_2 + OH^-$$

The Negative Electrode:

$$MH + OH^- \leftrightarrow M + H_2O + e^-$$

Overall Reaction:

$$NiO(OH) + MH \leftrightarrow Ni(OH)_2 + M$$

(MH: metal hydride, M: hydrogen-absorbing alloy) The right side of the reversible reaction shows the charged state while the left side shows battery discharge.

© 2011 Microchip Technology Inc.

A very important issue for sealed metal hydride cells is internal gas pressure build-up. Usually, the negative electrode (hydrogen-absorbing alloy) is made bigger than the positive electrode to allow sealing. The hydrogen formed during charging by electrolytic reaction of water is absorbed on the negative electrode. Also, the oxygen formed on the positive electrode diffuses through the separator and is consumed at the negative electrode by oxidizing hydrogen into water. This way all gasses formed in the charge/discharge process are normally reabsorbed. Of course, extreme conditions may lead to a high pressure build-up, in which case the gas is vented through an emergency valve to avoid explosion. This irreparably damages the battery in a way ranging from permanently reduced capacity to making it unusable.

Discharge Performance

The discharge behavior of nickel-metal hydride cells is well-suited for today's portable electronic products. Voltage is stable at 1.2V for 80% of the battery discharge time and high capacity ensures extended periods of operation.

Battery Capacity

The most important battery parameter to a product designer is usually the run time available for a certain equipment-use profile. While establishing actual run times of the product is vital before adopting the final design, all of the tests are performed using rated capacities. That is why designers should understand the conditions under which cell capacities are established and the impact of differences on the estimated performance.

The standard cell rating, abbreviated as C, is the capacity obtained from a new (but conditioned) cell subjected to a constant current discharge at room temperature, after it has been optimally charged. Battery capacity varies inversely with the discharge current. This means higher discharge currents will result in lower battery capacities. For metal hydride cells, the rated capacity is normally recorded for a discharge rate that completely depletes the cell in five hours.

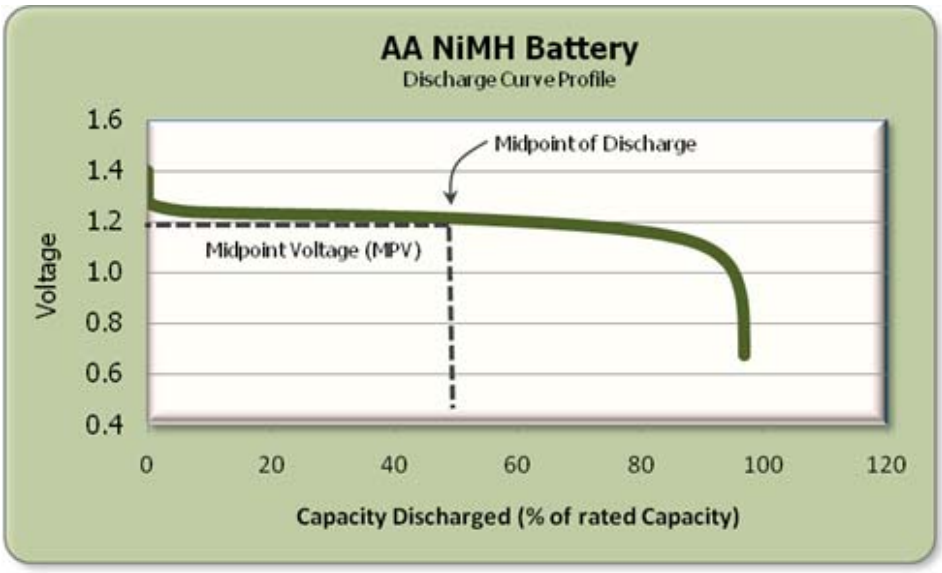
The marked value may reflect either an average or a minimum for all cells, depending on the manufacturer. Typically, nickel-cadmium cells are rated based on minimum values, while nickel-metal hydride cells are rated on average values. The difference between two cells may be significant (about 10%), depending on the manufacturing process. That is why it is more convenient to normalize charge and discharge parameters by the C rate. This is based on the fact that performance is often identical compared on the C basis, within a family of different cell sizes and capacities.

Voltage Discharge Curve Behavior

The discharge profile is mainly affected by temperature and discharge rate. Fortunately, under most conditions the voltage curve retains the flat plateau desirable for electronic applications.

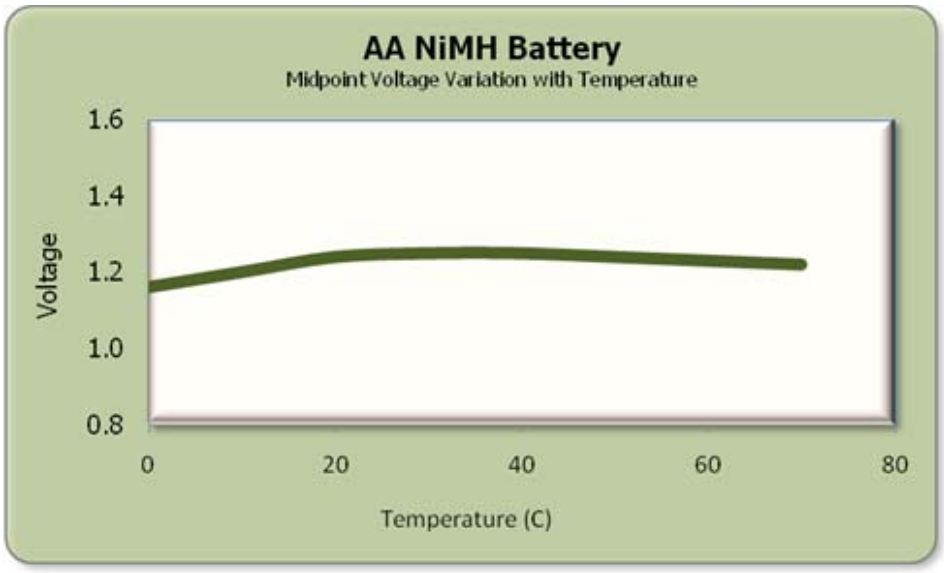
A typical discharge profile for a cell discharged at a 5 hour rate (C/5 rate) is shown in Figure 2. The initial drop from an open-circuit voltage of about 1.4V to the 1.2V plateau occurs very quickly. At near depletion the cell exhibits a sharp knee, where the voltage drops very quickly.

FIGURE 2: TYPICAL DISCHARGE VOLTAGE PROFILE FOR A NICKEL-METAL HYDRIDE CELL



The temperature has a significant influence over the location of the discharge curve, but small variations (±10° Celsius) from room temperature do not have an appreciable effect. Major variations, especially lower temperatures, will lower the nominal voltage.

FIGURE 3: MID-POINT VOLTAGE VARIATION WITH TEMPERATURE



The discharge rate will not have a significant effect on the shape of the discharge curves for rates under 1C. However, for rates over 1C the beginning and end transients will consume a larger portion of the duration.

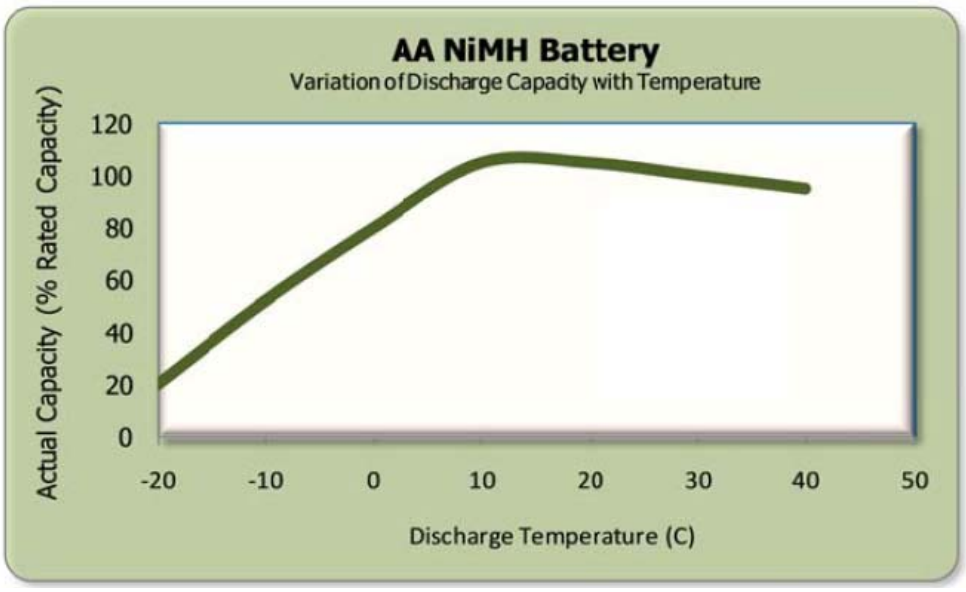
Discharge Capacity Behavior

Similar to the voltage profile, capacity during a discharge is dramatically affected by temperature and the rate of discharge. Capacity is also heavily influenced by the cell history of charge/discharge/storage. Obviously, if a cell has not been optimally charged or has been stored for extended periods, it will not be able to return the full capacity during discharge.

Temperature effects on discharge capacity are dramatic at lower temperatures. For example, a metal hydride cell discharged at 0°C will show about 70% capacity, 40% at -10°C and 20% at -20°C. Higher temperatures have a lesser effect, discharging at 50°C will reduce capacity to about 80%.

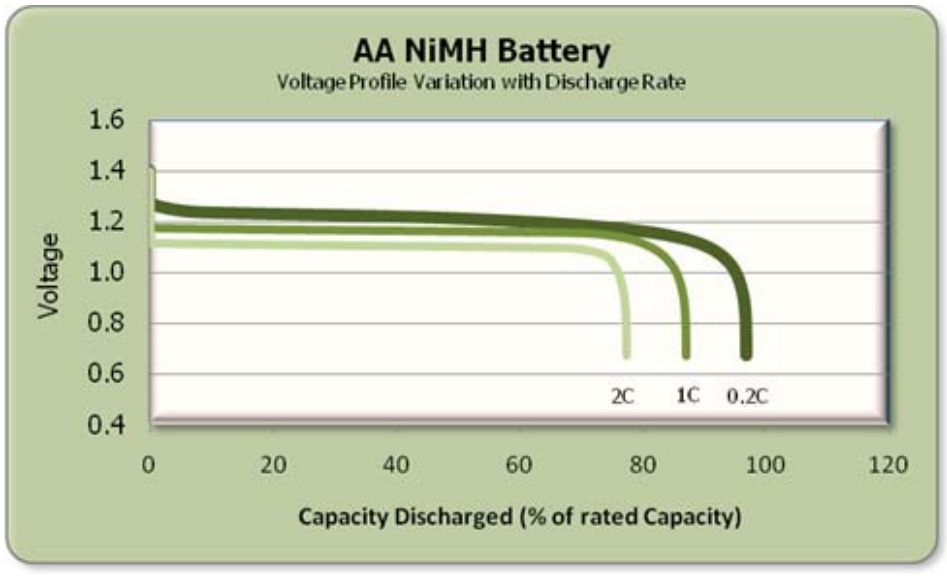
© 2011 Microchip Technology Inc.

FIGURE 4: VARIATION OF DISCHARGE CAPACITY WITH TEMPERATURE



The discharge rate will only start to affect capacity significantly at rates above 1C. As significant reductions in voltage delivery occur at high rates, it will result in capacity reduction depending on the choice of the discharge termination protocol.

FIGURE 5: VOLTAGE PROFILE VARIATION WITH DISCHARGE RATE



Measuring the state of charge for nickel-metal hydride cells is a real challenge because of the flatness of the voltage curve under normal discharge conditions. Voltage sensing will not be of any use for determining the charge state of the battery. This is vital for portable electronics (it is a real problem for computers), as they require some kind of fuel gauge to help determine when to recharge or to save the current work.

To date, the only form of state of charge measurement that can give reasonably accurate results is called “coulometry.” This method measures electrical flows during charge and discharge to compute remaining capacity. Many devices have integrated electronics that perform sophisticated tracking of current flows and self-discharge estimation. With initial calibration and compensation for environmental conditions, it is possible to compute remaining charge value with an accuracy of 5-10%.

Voltage depression or “memory” has always been a problem in applications using nickel-cadmium cells. When nickel-cadmium cells are routinely partially discharged, a depression of about 150 mV has been reported to appear in the discharge voltage profile. The severity of this problem is open to interpretation, but it is agreed that the cause of the problem is in the structure of the cadmium electrode. Since the nickel-metal hydride cells use hydrogen-absorbing alloy instead of cadmium, voltage depression is no longer a concern.

Discharge Termination

Incorrect discharge termination may lead to irreversible damage to the cell, caused by cell reversal. Removal of load from the cell(s) before total discharge is highly recommended. The typical voltage profile for a cell carried through total discharge has three plateaus. The first plateau is at about 1.2V during normal discharge. The second and the third plateaus are caused by the discharge of first the positive electrode, and then the residual capacity in the negative electrode (which is usually bigger than the positive one).

At the point where both electrodes are reversed, a substantial quantity of hydrogen is released. This leads to gas venting and irreparable damage to the electrode structure.

Voltage cutoff (load disconnect) is normally based on a cell voltage of 0.9V per cell (75% of the 1.2V midpoint voltage). This is an excellent practice for discharge rates below 1C.

However, high-drain usage (1-4C) somewhat shifts the midpoint to a lower value, and the “knee” to the near depletion curve is more rounded, meaning that voltage cutoff at 0.9V may be premature. A significant portion of the capacity is left unused. For this reason, it is a better choice to use a value equal to 75% of the application operating midpoint in high-drain applications.

Discharge termination in batteries is a more complicated matter. Normal manufacturing process produces a range of capacities for battery cells. As these cells are combined into batteries, the effect of cell capacity variations is amplified by the number of cells in the battery. Using termination voltage based on a simple 0.9V/cell multiple may lead to battery damage, as the weaker cells are driven into reverse significantly before the termination voltage is reached. Selection of proper discharge voltage, especially for large batteries, should be done in consultation with the cell manufacturer.

Charge Characteristics

Proper charging of the nickel-metal hydride cells is a key element in their performance. A successful charging scheme balances the need for quick, thorough charging with the need to minimize overcharging, a key factor in prolonging life.

The nickel-metal hydride cell is more sensitive to charging conditions than the nickel-cadmium cell and usually requires careful monitoring of the process.

There are a few key elements required for successfully charging a metal hydride cell:

- use a three-step charging strategy
- design to enable detection of more subtle indications of entry into overcharge
- use redundant fast-charge termination techniques
- provide fail-safe charge termination back up

Using these guidelines allow fast and reliable charging of nickel-metal hydride cells while maximizing cycle life.

Cell Behavior during Charge

Unlike discharge performance, where nickel-metal hydride and nickel-cadmium cells are very similar, there are some significant differences in behavior during charging. These differences are related to basic electrochemical composition of each type of cell. Nickel-cadmium cells are endothermic on charge (they absorb heat), while nickel-metal hydride are exothermic (produce heat). This difference manifests for each cell type in the relationship between pressure, voltage and temperature.

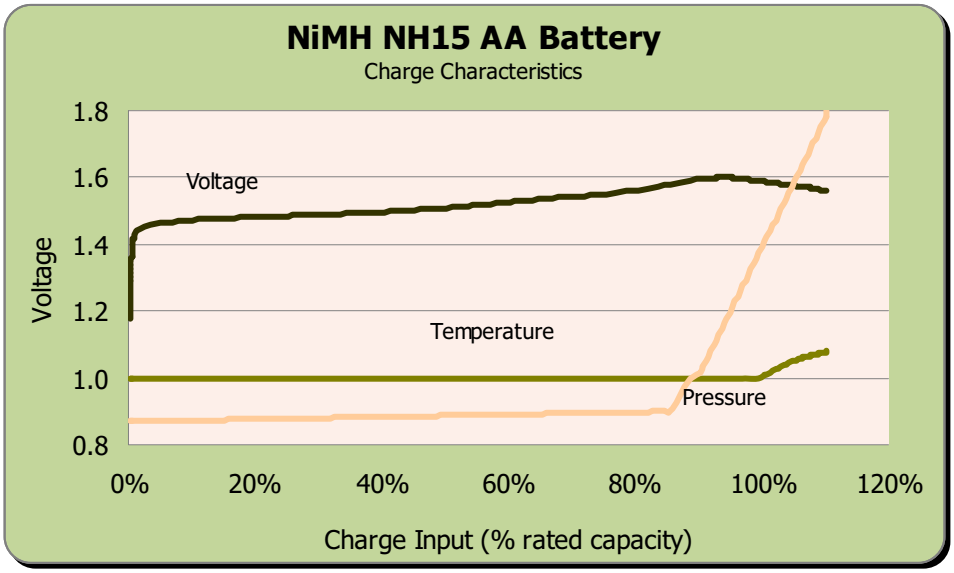
On charging a metal hydride cell, voltage spikes up initially, then slowly rises until full charge is achieved. As the cell reaches overcharge, the voltage peaks and then gradually trends down.

As the charge process is exothermic for the metal hydride cell, heat is released throughout the charging process. When the cell reaches overcharge, where the bulk of incoming energy is converted to heat, cell temperature increases dramatically.

© 2011 Microchip Technology Inc.

Pressure also increases slowly during charging, but rises dramatically in overcharge as greater quantities of gas than the cell can recombine are generated. Without a safety vent, uncontrolled charging at C rate (or more) can result in physical damage to the cell.

FIGURE 6: NICKEL-METAL HYDRIDE CELL CHARGE CHARACTERISTICS



The effect of temperature on charging efficiency is another difference between metal hydride and cadmium cells. Charge acceptance in the metal hydride cell will decrease with higher temperature, starting below 20°C. The nickel-cadmium cell has a peak in charge acceptance at room temperature. With either cell type, the lower charge acceptance at higher temperatures is a serious concern. Product designers should exercise care when mounting the cells close to heat sources or in compartments with limited ventilation.

Charge acceptance of nickel-metal hydride also increases with charge rate, making fast chargers more efficient.

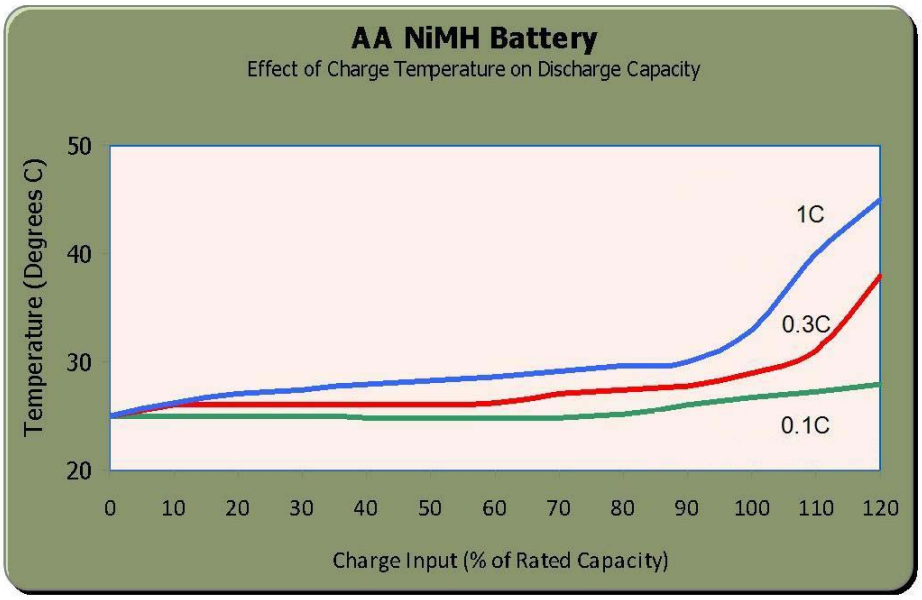
It is critical to detect entry into overcharge quickly and reliably, especially in schemes that use high charging rates. This is the key to maximizing cell life.

Primary charge control schemes usually depend on sensing either a quick rise in cell temperature or a peak in voltage.

Charge control based on temperature is the most reliable way to determine when to terminate fast charging of the nickel-metal hydride cell, and thus is recommended for the primary control mechanism over voltage sensing.

© 2011 Microchip Technology Inc.

FIGURE 7: TEMPERATURE PROFILES DURING CHARGE



Today's fast chargers require much higher rates than the 0.1 to 0.3C often used in older chargers because of the market's trend to shorter charge times. Also, these higher charge rates serve to accentuate the slope changes used to detect both temperature and voltage termination conditions. A charge rate of 1C is recommended for restoring full capacity to a discharged cell. For charging schemes that use a timed "topping" charge to ensure complete charge, a 0.1C rate balances charge input with minimum adverse effects from overcharge. After that, a maintenance charge rate of 0.025C (C/40) is adequate to counter cell self-discharge. Unfortunately, in standard type cells self discharge can be quite high in the first day (5-10%), but it levels to less than 1% per day afterwards.

Today's charging strategies are easily divided into two categories: two-stage chargers (or slow chargers), and three-stage chargers (or fast chargers).

Two-stage chargers only use a timer to switch from the initial charge rate to maintenance charge rate. Usually, for cost reasons, there is no voltage or temperature sensing and charge rates are kept below 0.1C to minimize overcharge impact on the cell's life and performance. Charge time is 16 to 24 hours to ensure full recharge on a depleted cell. Even if economical, this scheme does not take into consideration environmental conditions or the cell's degree of discharge, and its use is rarely recommended for typical nickel-metal hydride applications.

Three-stage chargers use a fast charge to restore about 90% of the capacity, an intermediate timed charge completes the charge, then a maintenance charge provides a trickle current to balance the cells and compensate for self-discharge. The fast charge (currents in the 1C range) is usually terminated by using a temperature-sensing technique that triggers at the onset of overcharge.

The intermediate charge uses a 0.1C rate for a timed duration selected based on cell capacity or battery pack configuration. This step replaces the need to fast-charge deeply into the overcharge region to ensure full charge.

For voltage-based termination, the intermediate charge step is usually skipped, since the battery goes much further into overcharge.

Three-step charging requires more complex circuitry that raises cost, but it reduces cell exposure to overcharge, thus extending cell life.

© 2011 Microchip Technology Inc.

THE BATTERY CHARGER LIBRARY

Multi-Step Charging

Proper treatment of Nickel chemistry batteries requires multiple charging steps, especially in fast chargers. Specific mechanisms are needed for detecting battery insertion, charge termination conditions and removal. Also, deeply depleted cells need to be trickle-charged up to a defined point before starting to fast charge. Depending on the protocols used for fast-charge termination, a top-off step may or may not be needed. If batteries are to be left in the charger for extended periods of time before being used, a very small maintenance charge current will counter battery self-discharge.

Configurable Parameters

The charger library supports many configurable parameters that allow easy customization of the charge profiles without making any changes in hardware. Of course, there are some limits. For example, if the inductor used in the buck converter has a maximum specified current of 1.5A, it is clear that setting a current threshold that corresponds to 2A will not work, or worse, result in permanent hardware damage.

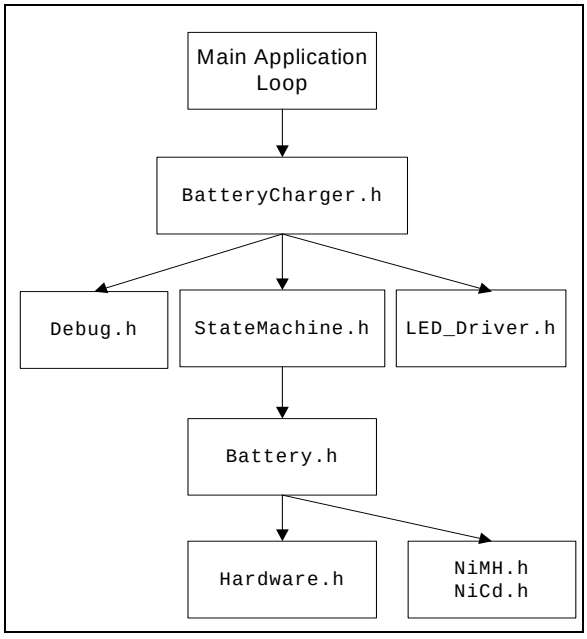
- Charger functionality:
  - Slow charger with optional maintenance charge (obsolete method)
  - Fast charger with pre-charging of depleted cells and optional top-off and maintenance charge
- Battery voltage:
  - Insertion/removal detection voltage
  - Minimum and maximum float voltage for charge initiation
  - Maximum voltage during charging
  - Maximum difference between charging and floating voltages for defective cell detection (impedance test)
- Battery temperature:
  - Minimum and maximum temperature for charge initiation
  - Maximum temperature during charging
- Time:
  - Slow charge time limit
  - Pre-charge time limit
  - Fast charge time limit
  - Top-off time limit
  - Maintenance time limit
- Charge current (depending on DAC resolution):
  - Slow charge current
  - Pre-charge current
  - Fast charge current
  - Top-off current
  - Maintenance current
- Charge status:
  - LED signaling

Library Structure

The library has a tree structure and the simplest way to use it would be to include the root file (`BatteryCharger.h`) and add the `DoCharger()` function into the application main loop. For different hardware platforms, changes are also needed in the `Hardware.h` file to define the correct values for the microcontroller peripheral Configuration registers and the AD channels to which the batteries, thermistors or current-sense amplifier are connected. The `NiMH.h` file contains charging parameters suitable for most commercial metal hydride single cell batteries, but the user should at least check them or consult with the battery manufacturer for the correct values. They are still a good starting point for testing purposes.

The only requirement for using the library is that the main charger function must be called every 125 ms or faster. The period is measured typically using a 16-bit timer with a prescaler. The PWM timer (Timer2 in this case) is also used for measuring smaller time intervals.

FIGURE 8: LIBRARY STRUCTURE



CHARGER LIBRARY FILES AND FUNCTIONS

BatteryCharger.h

This is the root file of the library and must be included in the user project. It contains a few high-level status functions that should be used in the user main application loop.

`void InitializeCharger(void)`

This function must be called before the main application loop to ensure that the hardware has been properly initialized. The peripheral Configuration register values are defined in the `Hardware.h` file and should be carefully inspected when using the library on a new platform. See `Hardware.h` file.

`unsigned char Get_Charger_State(void)`

Returns the current charger state to the main application. The return value is one of the following:

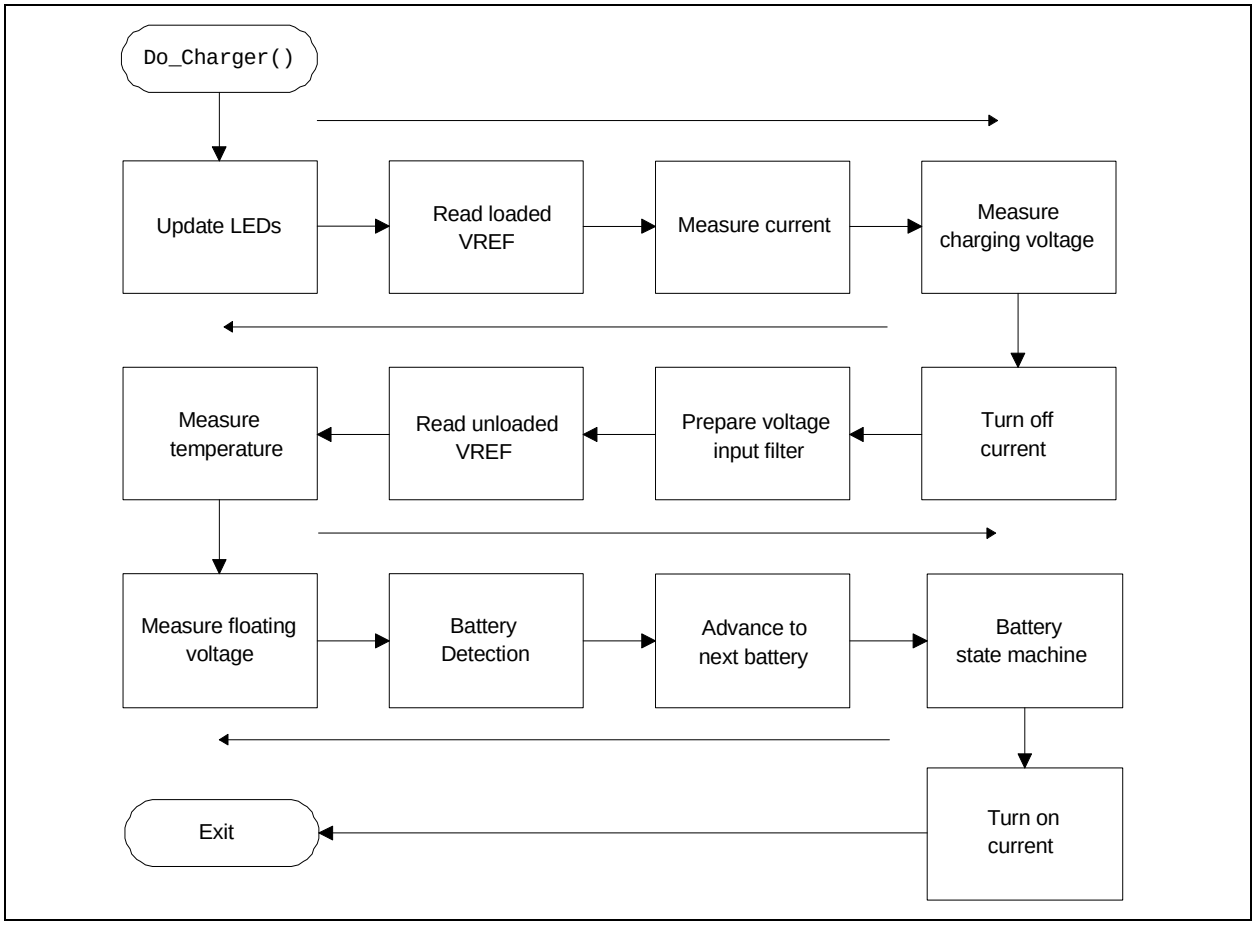
- `CHARGER_IDLE` – no batteries are inserted in the charger
- `CHARGER_WORKING` – at least one battery is inserted and is in a charging state
- `CHARGER_DONE` – all inserted batteries are charged and may be removed for use
- `CHARGER_FAULT` – at least one battery is in a Fault state and should be checked for safety

`void Do_Charger(void)`

This is the main charger function. It must be called at least every 125 ms from the main application loop. It handles all voltage, current and temperature measurement, presence detection, battery state machines, LED signaling and sending debug data.

The function checks for timer overflow upon entry and reloads. The default timer tick is 125 ms.

FIGURE 9: MAIN CHARGER FUNCTION



© 2011 Microchip Technology Inc.

Charge current readings and charge voltage readings are always taken while injecting current into the battery. A small downside exists when using low-side current sensing, because the small voltage drop on the current shunt appears in the readings. With proper filtering on both current and voltage, the shunt voltage drop can be easily eliminated in the firmware. Using high-side current sensing completely eliminates the problem, but is a bit more demanding in terms of hardware design.

Floating voltage readings are always taken with the current source and the battery switches off. Temperature readings are taken preferably with the current source off to avoid noise. This may become very important for temperature slope-based charge termination where 0.1 degree or better resolution is needed.

Battery presence is based on the floating voltage readings, so usually a value above a few hundred mV indicates a battery is inserted. All the voltage, current and temperature readings taken previously are used by the battery state machine and sent to the PC for debugging (if the option is enabled).

The next step is to advance the active battery and start injecting current.

EXAMPLE 1: BASIC USAGE EXAMPLE

```
#include “BatteryCharger.h”
.
.
.
void main(void)
{
    .
    // User initialization code
    .
    InitializeCharger();
    while(1)
    {
        Do_Charger();
        .
        // User application code
        .
    }
}
```

LED_Driver.h

The `LED_Driver.h` and `LED_Driver.c` only contain the functions used for signaling the battery or charger status.

`void LED_Blink(void)`

This function will turn the LEDs on and off according to a predefined scheme. The function is easily customizable, since it is known that it will be called every timer tick. In the default library configuration, each battery LED will have the following behavior:

- no battery detected – LED is off
- battery is charging – LED is blinking at 0.5 Hz
- battery is done charging or in Maintenance mode – LED is on
- battery is in a Fault state – LED is blinking at 2 Hz

Debug.h

Typically contains a function that sends battery data to the PC in a certain format for debugging purposes. The basic library uses the UART software to send data to the PC.

`void SendStatus(void)`

The function will send every second a data packet with the following structure:

TABLE 2: DATA PACKET STRUCTURE

Size (bytes)	Description
1	Charger ID
1	Battery count
1	Packet sequence number
2	Unloaded V_{DD} calculated value
1	Battery A charge state
2	Battery A floating voltage
2	Battery A charging voltage
2	Battery A peak voltage
2	Battery A current
2	Battery A temperature
2	Battery A state timer
1	Battery A Fault code
1	Battery B charge state
2	Battery B floating voltage
2	Battery B charging voltage
2	Battery B peak voltage
2	Battery B current
2	Battery B temperature
2	Battery B state timer
1	Battery B Fault code
2	Board thermistors B(eta) parameter
2	Board thermistors default resistance (usually at 25 Celsius)
2	0xFFFF Packet end sync field (only used for USART communication)

The associated C style structure can be easily used in a union with the 64-byte USB buffer for easy access.

EXAMPLE 2:

```
struct
    {
        unsigned char      ID;
        unsigned char      BatteryCount;
        unsigned char      sequence;
        unsigned int       Vdd;

        unsigned char      B1State;
        unsigned int       B1FloatV;
        unsigned int       B1ChargeV;
        unsigned int       B1PeakV;
        unsigned int       B1Current;
        unsigned int       B1TempC;
        unsigned int       B1StateTimer;
        unsigned char      B1FaultCode;

        unsigned char      B2State;
        unsigned int       B2FloatV;
        unsigned int       B2ChargeV;
        unsigned int       B2PeakV;
        unsigned int       B2Current;
        unsigned int       B2TempC;
        unsigned int       B2StateTimer;
        unsigned char      B2FaultCode;

        unsigned int       ThermistorB;
        unsigned int       ThermistorR0;

    } bat_values;
```

Uart.h

The UART software is written in assembly language for speed and size. The function will work well on PIC16 devices. The user needs to check and modify a few defines in `Uart.as` so that the correct bit rate is used, as well as the correct output pin.

EXAMPLE 3:

```
#define TX                _PORTA,1
#define UARTOscillator    8000000
#define UARTBaudRate      38400
```

In this example, the output pin is RA1, the PIC® device runs at 8 MHz (2 MIPS) and the transmission speed is 38,400 bps. Also, the user must make sure the selected pin is output capable (no OD/OC) and it is configured for output.

Since only a transmit pin exists and no two-way communication is possible with the PC, the 2-byte sync field was added to the end of the debug packet.

If a hardware UART is available, the user must add the initialization code and write the transmit function.

`void UartTx(unsigned char)`

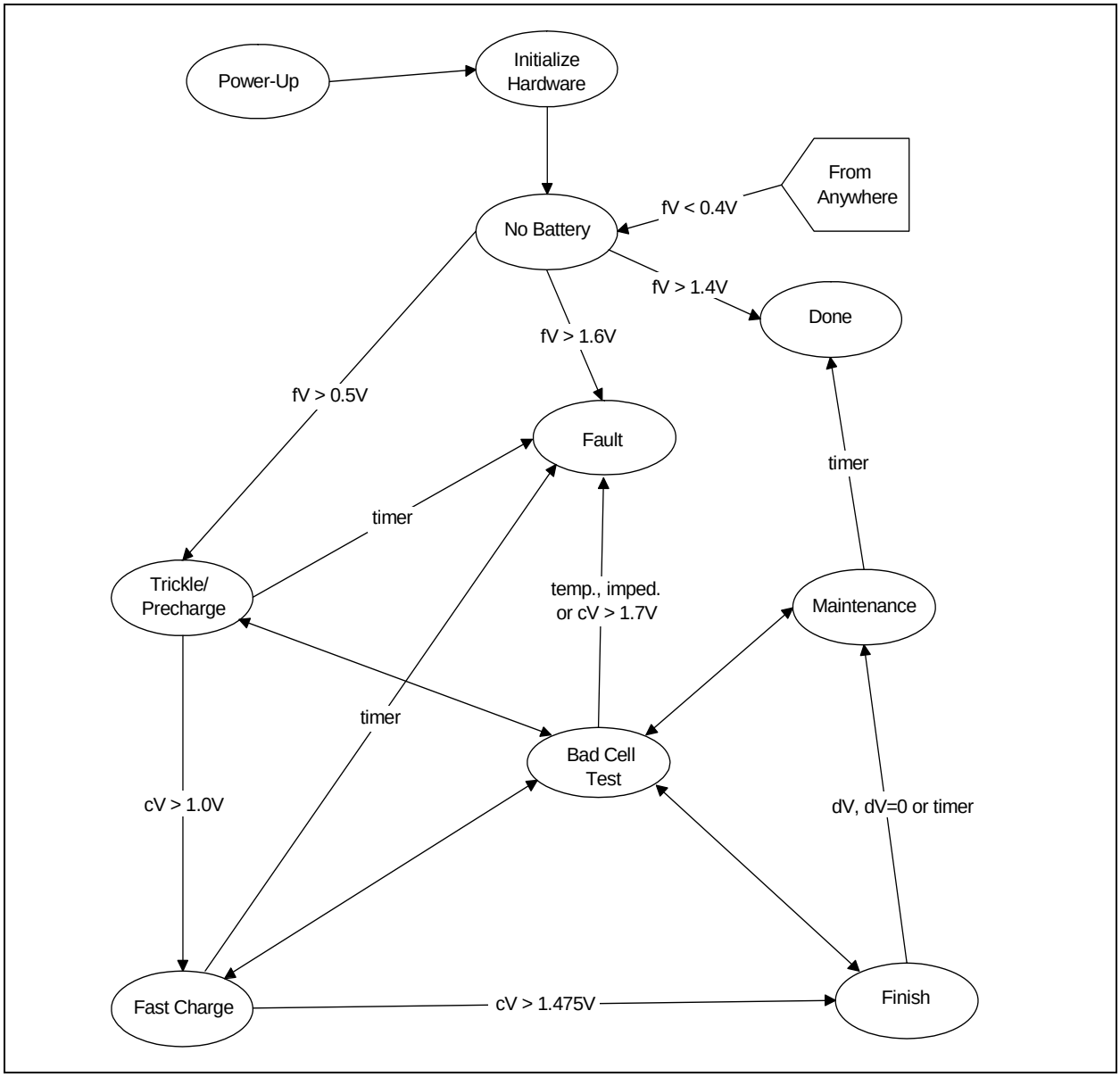
This function sends a single character serially using the predefined port speed.

StateMachine.h

Contains a collection of functions related to battery charging states and the rules of transitioning between states.

© 2011 Microchip Technology Inc.

FIGURE 10: BATTERY CHARGING-RELATED FUNCTIONS



`void Battery_Detection(void)`

This function takes care of the battery presence detection. Floating voltage readings on a certain battery slot over 0.5V usually mean that a battery has been inserted and it starts charging. If the floating voltage is above 1.6V, this is certainly an alkaline battery or some other wrong type of rechargeable and the slot goes into Fault. If the voltage is only above 1.4V, then it goes directly to done, since it is either a properly charged nickel type cell or partially charged alkaline. Temperatures below 0°C and above 45°C will put the battery slot into Fault.

If a battery is already inserted and, at any time, a floating voltage below 0.4V is detected, it means the battery has been removed from that slot and it goes into No Battery mode.

For the voltage and temperature thresholds used in battery detection, see `Hardware.h`.

`void Battery_Fault_Check(void)`

Checks if the battery inserted in the active slot is defective or the wrong kind. Overheating is also considered a fault and charging is stopped. Even if nickel chemistry batteries are not as dangerous as lithium-ion types, which may explode if charged improperly, they may rupture and leak toxic/corrosive substances.

There are three kinds of safety tests performed in this function:

- absolute voltage fault – will stop charging if detecting voltages over 1.7V
- absolute temperature fault – will stop charging for temperatures over 50°C
- impedance fault – will stop charging if the difference between charging and floating voltage exceeds a threshold calculated using the charging current and the maximum allowed battery impedance

The temperature threshold is defined in `Hardware.h` as it applies to all battery types (overheating is generally bad).

EXAMPLE 4:

```
#define MAX_CHARGE_TEMP
```

The voltage thresholds and impedance limit are defined in the `NiMH.h` file.

EXAMPLE 5:

```
#define Z_TEST_LIMIT
#define INSTANT_CHARGE_CURRENT
#define ALKALINE_DETECTION_VOLTAGE
```

`void Battery_Slow_Charge(void)`

This function is used only if the charger is configured as a slow charger. There are no charge termination protocols active in this mode and charging will stop after the timer expires. Fault detection is always active. The timer value is located in the `NiMH.h` file:

EXAMPLE 6:

```
#define CHARGER_OVERNIGHT
#define OVERNIGHT_TIMER
#define CHARGE_VOLTAGE_LIMIT
```

`void Battery_Precharge(void)`

Trickle mode is the first state in a fast charger because of the specific high-impedance of deeply depleted batteries. Instead of using full current, a battery is charged with a low rate (about C/10) until the charging voltage goes above 1.0V. This prevents overheating or faulting the battery until it is able to accept a greater charge rate. If the state timer expires before the battery voltage reaches 1.0V, a Fault state follows.

Related defined values for the trickle-charge state are defined in `NiMH.h`:

EXAMPLE 7:

```
#define TRICKLE_CHARGE_ENABLED
#define TRICKLE_CHARGE_CURRENT
#define TRICKLE_CHARGE_TIME_LIMIT
```

`void Battery_Fast_Charge(void)`

In this state the battery is charged with a high current until charging voltage exceeds a certain threshold or the timer expires and the battery switches to Fault mode. The average charging current depends on the number of battery slots (`BATTERY_COUNT`) and the maximum current setting. Because the battery slot changes every second, each battery (if inserted) receives the set current for one second every `BATTERY_COUNT` seconds. Obviously, the duty cycle is 50% for two slots and 25% for four slots.

There are no charge termination protocols active in the fast-charge state, but Fault detection is active. If the timer expires and the battery charging voltage has not reached the finish charge threshold (for the basic library version it is set to 1.475V), it switches to Fault mode. Otherwise, the battery goes into Finish mode.

`void Battery_Finish_Charge(void)`

Finish charge state has two very important functions. It has to finalize the charging, and it has to detect entry into overcharge quickly. Temperature and voltage may be used to detect a battery entering the overcharge region. This is the only charging state to use charging termination protocols.

Temperature slope is the best method as it is triggered very early, helping extend battery life. Neither overheating, nor pressure build-up occurs if this is done properly. The usual values for the temperature slope trigger are between 1.0 and 2.0°C/min. The downside is that very good thermal coupling between the thermal sensor and the battery case is needed for accurate readings. Each battery slot needs its own sensor and sometimes an additional sensor for the environment. If the sensors are not linear, then the interpolation code takes a lot of program space and memory. After this charge termination triggers, a short top-off charge at low charge rates (C/10) is recommended to finalize the charge.

Voltage-based termination protocols are easier to implement in hardware, but may prove tricky in software because of system noise and ADC resolution. Over-sampling and LP filtering are highly recommended. Negative delta V is the first sign to look for. After receiving full charge, nickel chemistry batteries exhibit a voltage peak and a small drop, followed by a flat portion. Ni-Cd batteries make it easy, since the drop is well into the 100 mV range. Ni-MH batteries have a much smaller drop, only 5-10 mV, making it difficult to detect on 10-bit converters.

Detecting a flat charging voltage is simple enough. The battery charging voltage should go up steadily all through the charging process, with a steeper slope at the end. If there is no voltage rise for a defined period, flat delta V is triggered. This usually happens if the negative delta V was missed, or the battery is old enough not to show a detectable drop.

© 2011 Microchip Technology Inc.

Any of these two triggers will switch the battery into Maintenance mode or end the charge, depending on defined options.

Related defines are found in the NiMH.h file:

EXAMPLE 8:

```
#define FINISH_CHARGE_CURRENT
#define FINISH_CHARGE_TIME_LIMIT
#define FINISH_CHARGE_TIME_RESULT
#define FINISH_CHARGE_FULL_CURRENT

#define NDV_VOLTAGE_DROP
#define NDV_SAMPLES
#define FDV_SAMPLES
#define PEAK_UPDATE_SAMPLES
```

`void Battery_Topoff(void)`

The battery switches to this state only after a successful temperature slope trigger in Finish mode. Because temperature slope is an early overcharge indicator, it is recommended to top off the battery charge with a low-rate charge. If battery life is more important than capacity, this step can be skipped directly to maintenance or charge end.

Related defines are found in the NiMH.h file:

EXAMPLE 9:

```
#define TOPOFF_CHARGE_ENABLED
#define TOPOFF_CHARGE_TIME_LIMIT
```

`void Battery_Maintenance(void)`

Battery maintenance injects a very small current, just to counter battery self-discharge until the user removes the battery to install it into the intended device. This is necessary because the rate of self-discharge in standard type cells can be 5-10% in the first day after a charge cycle. Even if the current is very small, it is not recommended to keep batteries into Maintenance mode extended periods of time. After the state timer expires, the battery charge is ended. If maintenance charge is not enabled, the state machine will skip to end of charge.

Related defines are found in the NiMH.h file:

EXAMPLE 10:

```
#define MAINTENANCE_CHARGE_ENABLED
#define MAINTENANCE_CHARGE_TIME_LIMIT
```

`void Battery_State(void)`

This function checks for the active battery state and calls the appropriate function.

`void CleanUp_Vars(void)`

This is a special function that clears the battery timer, voltage and current values when it enters any of the non-charging states: NO_BATTERY, Fault or Done. This is necessary because, when a battery is inserted, the state machine expects to find clear values.

Battery.h

All battery related functions, structures and variables are defined at this library level. Each battery has its own data structures BAT_STRUCT and BAT_STATE_VARS.

BAT_STRUCT contains battery measured and calculated values:

- floating voltage – voltage reading on the disconnected battery
- charging voltage – voltage reading on the battery while current is being injected
- peak voltage – biggest charging voltage reading recorded to the moment
- current – voltage reading on the current shunt (I-V conversion)
- temperature – voltage reading on the thermistor/temperature sensor
- PWM duty – switching power supply duty cycle
- power – calculated power input for the battery

BAT_STATE_VARS contains the battery state, timers and debouncing counters:

- total charge time – total time in seconds the battery has been receiving charge
- state timer – time in seconds remaining for current charging state
- peak update counter – counts remaining for new charging voltage peak
- flat voltage counter – counts remaining to flat voltage trigger
- negative delta V trigger counter – counts remaining to negative delta V trigger

Battery charge states for each slot are defined as following:

- NO_BATTERY (0x00) – no battery inserted
- Fault (0x01) – defective battery or wrong type detected
- Done (0x02) – battery is charged and resting
- Maintenance (0x04) – battery is in Maintenance mode; a very low current is injected to counter self-discharge
- Trickle (0x08) – battery is trickle-charged to reach 1.0V before going to Fast mode

© 2011 Microchip Technology Inc.

- Fast (0x10) – battery is charged with maximum current every BATTERY_COUNT seconds; no charge termination active
- Finish (0x20) – battery is charged with maximum current; peak voltage is tracked and charge termination protocols are active
- Topoff (0x40) – battery is trickle-charged for a specified time to complete charge after certain finish charge termination triggers like temperature slope

`unsigned char Get_Battery_State(unsigned char n)`

Returns battery state of the selected battery (n) to the user application.

`unsigned long Get_Battery_Time(unsigned char n)`

Returns the total charging time (seconds) of the selected battery (n) to the user application.

`unsigned int Get_Battery_Volts(unsigned char n)`

Returns the floating voltage (mV) of the selected battery (n) to the user application.

`void Measure_Current(void)`

This function measures the current shunt voltage present at the positive comparator input.

The shunt value and amplification are known values so it is very easy to derive current by division.

`void Measure_Charge(void)`

Measures charging voltage on the active battery. This is only possible while the battery is active and current is being injected. If the current shunt is a low side type, the voltage drop must be subtracted from the battery voltage to obtain the correct value.

`void Measure_Float(void)`

Measures floating voltage on the active battery. For this operation, the current is turned off and the battery top transistor is switched off. The RC filter that connects the battery to the AD Converter input must be discharged and then charged from the battery to obtain the correct value. Charging time depends on the RC filter time constant and should be at least 5 times the time constant (98.5% accuracy).

`void Measure_Temp(void)`

Measures the voltage on the temperature sensor attached to the active battery. For obvious reasons it is better to measure with the current source off to avoid getting noisy readings.

`void Switch_Battery(void)`

This function is very important because it switches to the next battery slot and checks the battery state to decide if the current source should be turned on or not. It also increments total battery time for the current battery and decrements the state timer.

A special define is present in `NiMH.h` and will prevent switching the battery once it reaches the FINISH state:

EXAMPLE 11:

```
#define FINISH_CHARGE_FULL_CURRENT
```

This practically increases the average current put into that battery to about 90% of the set limit. Termination protocols like negative delta V or temperature slope need charging rates of at least C/2 to become reliable so using all available current on a single battery until one of these conditions triggers, helps a lot with detection.

`Hardware.h`

This is a very important file because all initialization values are defined here. When using a different hardware platform, the user must carefully modify these values to match the hardware.

EXAMPLE 12:

```
#define    NiMH
#define    NiCd
#define    Li_ION
#define    NiZn
```

Only the correct type of battery must be defined (in this case, NiMH), so the other ones should be commented out. This define selects the special header file suited for the battery chemistry.

EXAMPLE 13:

```
#define XTAL_FREQ             8000000
#define WORKING_FREQ   XTAL_FREQ / 4
```

Crystal frequency must be defined to allow correct timing calculation. Instruction cycle is 4 oscillator periods for PIC16/18 families.

© 2011 Microchip Technology Inc.

EXAMPLE 14:

```
#define T2_INIT                  0x7C
#define PR2_INIT                 24

#define T2_POSTSCALER            16
#define T2_FLAG_TICK WORKING_FREQ / (PR2_INIT + 1) / T2_POSTSCALER
```

Timer2 is used to generate the PWM signal for the switching power supply. Also, using the prescaler it is very simple to generate shorter timings, in this case, the PWM ramp timing and the charge/discharge timings for the battery voltage reading.

EXAMPLE 15:

```
#define    PWM_PRECHARGE  (PR2_INIT+1)*4/4
#define    PWM_FAST       (PR2_INIT+1)*4/2
#define    PWM_FINISH     (PR2_INIT+1)*4/2
#define    PWM_TOPOFF     (PR2_INIT+1)*4/4
#define    PWM_MAINT      (PR2_INIT+1)*4/20
#define    PWM_TRICKLE    (PR2_INIT+1)*4/4
#define    PWM_OVERNIGHT  (PR2_INIT+1)*4/4
#define    PWM_OFF         0

#define    PWM_RAMP_STEP   1
```

Duty cycle should not be more than 50% when using a buck converter configuration. Fast and finish charge have the maximum allowed duty cycle. Trickle/pre-charge have only ¼ of the maximum duty cycle since the current threshold is lower. Maintenance mode has a very low duty cycle and operates in Open-Loop Discontinuous mode.

Because most of the power supplies do not respond well to load steps, the duty cycle is ramped up or down with a certain step every time the battery current goes on or off.

EXAMPLE 16:

```
#define    FTIMER_1SEC    (unsigned long)    T2_FLAG_TICK * 1000 / 1000
#define    FTIMER_1MSEC   (unsigned long)    T2_FLAG_TICK * 1 / 1000
#define    FTIMER_5MSEC   (unsigned long)    T2_FLAG_TICK * 5 / 1000
#define    FTIMER_10MSEC  (unsigned long)    T2_FLAG_TICK * 10 / 1000
#define    FTIMER_50MSEC  (unsigned long)    T2_FLAG_TICK * 50 / 1000
#define    FTIMER_60MSEC  (unsigned long)    T2_FLAG_TICK * 60 / 1000
#define    FTIMER_100MSEC (unsigned long)    T2_FLAG_TICK * 100 / 1000
```

Postscaled PWM clock is used to measure smaller time intervals.

EXAMPLE 17:

```
#define            T1_INIT               0x31
#define            TIMER_OVERFLOW        TMR1IF

#define            ONE_SECOND            4
#define            LED_TIME              (ONE_SECOND / 4)
#define            TIMER_250_MSEC        WORKING_FREQ / 8 / ONE_SECOND
```

Timer1 is used to generate the 250 ms timer tick for the main application. This is the longest interval that can be generated at 2 MIPS with a prescaler of 8 (62500 * 8 * 4 = 2000000). Non-integer division should be avoided for obvious accuracy reasons.

© 2011 Microchip Technology Inc.

EXAMPLE 18:

```
#define    TRIS_INPUT     1
#define    TRIS_OUTPUT0

#define    PORTA_INIT     0b11101111
#define    PORTC_INIT     0b11110111
#define    TRISA_INIT     0b11011100
#define    TRISC_INIT     0b11001110

#define    ANSEL_INIT     0xEC

#define    PIE1_INIT      0x00
#define    INTCON_INIT    0x00

#define    CCP1CON_INIT   0x0F
#define    PWM1CON_INIT   0x80
#define    ECCPAS_INIT    0x15

#define    OPTION_INIT    0xF9

#define    VRCON_INIT     0xE0
#define    VRCON_TRICKLE  0xE1
#define    VRCON_FAST     0xE5
#define    VRCON_FINISH   0xE5

#define    CM1CON0_INIT   0x96
#define    CM2CON0_INIT   0x00
#define    CM2CON1_INIT   0x00

#define    ADCON0_MASK    0x81
#define    ADCON1_INIT    0x50
```

These register initialization are used in the `InitializeHardware()` function. They should be modified to match the user application hardware.

EXAMPLE 19:

```
#define LED_A          RC4
#define LED_B          RA5

#define BAT_A_SW       RA0
#define BAT_B_SW       RC0

#define BAT_ON         0
#define BAT_OFF        1

#define LED_OFF        0
#define LED_ON         1

#define TRUE           1
#define FALSE          0

#define BAT_A_TRIS     TRISA4
#define BAT_B_TRIS     TRISC3

#define BAT_A_VOLT     RA4
#define BAT_B_VOLT     RC3
```

Port allocation and direction related defines.

EXAMPLE 20:

```
#define BAT_A_CHANNEL      0x03
#define BAT_B_CHANNEL      0x07
#define TEMP_A_CHANNEL     0x02
#define TEMP_B_CHANNEL     0x05
#define CURRENT_CHANNEL    0x06

#define CVREF              0x0C
#define VREF_06            0x0D
#define VREF_12            0x0E
```

For each battery slot a voltage reading and a temperature reading ADC channel is needed. For the current reading, an additional channel (multiplexed with a comparator input) is needed. Internal references have separate ADC channels.

EXAMPLE 21:

```
#define ADC_CHARGE_DELAY               20

#define VREF1200                       1185
#define VREF600                        600

#define ADC_SAMPLES                    64
#define ADC_RESULT_SHIFT               6
#define ADC_OVERSAMPLE_MAX             65535
#define ADC_MAX                        1023
#define FILTER_BITS                    6
#define FILTER_MAX                     1023
#define FILTER_SAMPLES                 1

#define CHARGE_VOLTAGE_OVERSAMPLING    64
#define CHARGE_VOLTAGE_SAMPLES         1
#define CHARGE_VOLTAGE_FILTER_BITS     6

#define FLOAT_VOLTAGE_SAMPLES          64
#define FLOAT_VOLTAGE_FILTER_BITS      0

#define CURRENT_OVERSAMPLING           64
#define CURRENT_FILTER_BITS            6
```

Since the switching power supply makes measurements quite noisy, longer acquisition times, over-sampling and filtering is needed. On a 10-bit ADC, LP filters up to 64 samples may be used to obtain 16-bit values. This is highly recommended for the charging voltage when using negative delta V termination.

EXAMPLE 22:

```
#define SHUNT_AMPLIFICATION    10
#define SHUNT_RESISTANCE       50 //in mohms
#define HIGH_SIDE_SENSING
```

The current shunt value and voltage amplification determines the current limit value. If the type of current sensing is not “high side”, then the shunt voltage drop must be subtracted from the charging voltage to get the correct value.

© 2011 Microchip Technology Inc.

EXAMPLE 23:

```
#define THERMISTOR_NTC

#define NTC_0C                (unsigned int)0xC353
#define NTC_25C               (unsigned int)0x7FFF
#define NTC_45C               (unsigned int)0x4E3B
#define NTC_50C               (unsigned int)0x43BC

#define TEMP_0C               ADC_OVERSAMPLE_MAX - NTC_0C
#define TEMP_25C              ADC_OVERSAMPLE_MAX - NTC_25C
#define TEMP_45C              ADC_OVERSAMPLE_MAX - NTC_45C
#define TEMP_50C              ADC_OVERSAMPLE_MAX - NTC_50C

#define MAX_CHARGE_TEMP       TEMP_50C
#define MAX_START_CHARGE_TEMP TEMP_50C

#define MIN_CHARGE_TEMP       TEMP_0C
#define MIN_START_CHARGE_TEMP TEMP_0C
```

The example board has NTC type thermistors and the temperature points are 16-bit voltage readings calculated using the part data sheet. To avoid confusion from having statements like “if (temperature < NTC_50C)” which actually check if the temperature is greater than a given value, not the other way around, all the values have been subtracted from `0xFFFF`. Placing the NTC thermistor on the high side of the voltage divider also works.

`unsigned int Measure_ADC(unsigned char channel, unsigned char samples)`

The function acquires the given number of samples on the selected ADC channel and returns the sum. For a 10-bit ADC, the maximum number of samples is 64.

`void InitializeHardware(void)`

Initializes the peripherals using the values defined in the *Hardware.h* file.

`void Set_Current(unsigned int current)`

This function sets the current limit for the switching power supply. On systems using a second PWM or a high resolution DAC to generate analog voltages, this function may be able to calculate the voltage needed for a certain mA input value. The example board uses the internal CVREF which has 200 mV steps. In this case, the function accepts PWM duty cycle values and sets the voltage reference accordingly.

`void Bat_Switches_Off(void)`

Switches off all battery-top switches. This is needed before measuring correctly the floating voltage on the batteries.

`NiMH.h`

This header file contains all charge settings related to NiMH chemistry. Everything that has to do with charge times, termination protocols and specific Fault conditions is defined here.

```
#define CHARGER_OVERNIGHT
#define CHARGER_FAST
```

The charger library may operate in one of these modes. Trickle (overnight) charger uses a small current for 12-16 hours to charge the batteries. No charge termination protocols are active. Fast charging uses three charging stages and uses a much higher current (usually between C/2 and C). This is the recommended charging method for most batteries.

EXAMPLE 24:

```
#define CHG_TERM_NEG_DELTA_V_ENABLE
#define CHG_TERM_DELTA_TEMP_ENABLE
#define CHG_TERM_FLAT_DELTA_V_ENABLE

#define CHG_TERM_MIN_I_ENABLE
#define CHG_TERM_FLAT_DELTA_I_ENABLE
```

Charge termination protocols may be enabled here. Negative delta V and flat delta V should go together. Delta temperature is usually a very reliable termination protocol, but in case it does not trigger, it is recommended to enable the voltage-based protocols, too. At least flat delta V should be active at the same time. Normally, the first to trigger is temperature, followed by negative delta V and the last is flat delta V.

Current-based termination protocols are not used for Ni-MH and Ni-Cd batteries. They are specific to CC/CV types like Li-Ion, Ni-Zn or lead acid batteries.

© 2011 Microchip Technology Inc.

EXAMPLE 25:

```
#define NDV_SAMPLES           15         //1 sample = BATTERY_COUNT seconds
#define MIN_I_SAMPLES         15
#define FDV_SAMPLES           15*60
#define FDI_SAMPLES           15*60
#define PEAK_UPDATE_SAMPLE S  5
#define DT_TEMP_RISE_RATE     10         //temperature rise per minute in 1/10 C
#define NDV_VOLTAGE_DROP      5          //drop needed for NDV in mV
```

Negative delta V needs charging voltage peak tracking. If the current voltage is higher than the last recorded, a straight number of samples (defined above), then the peak value is updated. In a similar way, if the charging voltage is lower than the peak voltage by 5 mV for a set number of samples, negative delta V triggers. Flat delta V triggers if there is no new peak for the set number of samples.

Delta temperature triggers if the rise rate per minute is at least 1.0°C degrees.

Since every battery has its state refreshed every BATTERY_COUNT seconds, it is easy to calculate how long it takes for a condition to trigger.

EXAMPLE 26:

```
#define Z_TEST_LIMIT              200 //mohms
#define INSTANT_CHARGE_CURRENT    800 //mA
#define CHARGE_MINUS_FLOAT_LIMIT  (unsigned long) INSTANT_CHARGE_CURRENT * Z_TEST_LIMIT / 1000
```

Impedance test is very important for fast detection of damaged NiMH batteries or alkaline types inserted in the charger. The impedance limit value is in milliohms and the instantaneous maximum charge limit in milliamperes. Since voltage is much easier to read, we deduce the maximum allowed difference between charging voltage and floating voltage.

EXAMPLE 27:

```
#define MAX_BATTERY_CAPACITY_AAA      1000  //mAh
#define MAX_BATTERY_CAPACITY_AA       3000  //mAh

#define CHARGE_EFFICIENCY             125   // 125% power needed to restore battery to 100%
                                                charge level

#define FINISH_CHARGE_TIME_FRACTION   33    // FINISH charge max time is 33% of FAST charge

#define FAST_CHARGE_TIME_NOLOSS       (unsigned long) MAX_BATTERY_CAPACITY_AAA * 60 * 60 /
                                      INSTANT_CHARGE_CURRENT
```

Charging time should be based on battery charge acceptance at a certain charge rate and maximum battery capacity. Charge acceptance is better at higher currents. A slow charger needs around 14-16 hours at C/10 rate to fully charge a battery. On the other hand, a fast charger needs only 65-70 minutes at C rate to fully charge a battery.

Maximum capacity, charge efficiency and current limit are used to calculate the maximum time needed to charge fully. This helps minimize the time spent in overcharge, if all charge termination protocols fail. Finish charge is defined as a fraction of the total charge time.

© 2011 Microchip Technology Inc.

EXAMPLE 28:

```
#define TOPOFF_CHARGE_ENABLED
#define TOPOFF_CHARGE_TIME_LIMIT         60*60   // 60 minutes
#define MAINTENANCE_CHARGE_ENABLED
#define MAINTENANCE_CHARGE_TIME_LIMIT    360*60  //6 hours

#define TRICKLE_CHARGE_ENABLED
#define TRICKLE_CHARGE_CURRENT           100
#define TRICKLE_CHARGE_TIME_LIMIT        6*60    //6 minutes pre-charge time

#define FAST_CHARGE_CURRENT              INSTANT_CHARGE_CURRENT / BATTERY_COUNT
#define FAST_CHARGE_TIME_LIMIT           (unsigned long) FAST_CHARGE_TIME_NOLOSS *
                                         CHARGE_EFFICIENCY / 100

#define FINISH_CHARGE_CURRENT            INSTANT_CHARGE_CURRENT
#define FINISH_CHARGE_TIME_LIMIT         (unsigned long) FAST_CHARGE_TIME_LIMIT *
                                         FINISH_CHARGE_TIME_FRACTION / 100

#define FINISH_CHARGE_TIME_RESULT        1       // 1 is GOOD, 0 is FAULT
#define FINISH_CHARGE_FULL_CURRENT

#define TOPOFF_CHARGE_ENABLED
#define TOPOFF_CHARGE_CURRENT            100
#define TOPOFF_CHARGE_TIME               60*60   // 60 minutes

#define OVERNIGHT_CHARGE_ENABLED
#define OVERNIGHT_CHARGE_CURRENT         100
#define OVERNIGHT_CHARGE_TIME            900*60  // 15 hours
```

For each of the enabled charging states, the countdown timer must be specified or calculated. Finish charge also has a special option that switches the battery to Fault mode, if the timer expires without any charger termination protocol triggering.

EXAMPLE 29:

```
#define BATTERY_REMOVAL_VOLTAGE       400
#define BATTERY_INSERTION_VOLTAGE     500
#define PRECHARGE_TERMINATION_VOLTAGE 1000
#define ALKALINE_DETECTION_VOLTAGE    1600
#define CHARGE_VOLTAGE_LIMIT          1700
#define FINISH_CHARGE_START_VOLTAGE   1470
#define FULL_BATTERY_VOLTAGE          1400
```

All voltage thresholds are defined in mV, but they are internally converted to 16-bit voltage readings (with a 5V nominal reference).

`BATTERY_REMOVAL_VOLTAGE` and `BATTERY_INSERTION_VOLTAGE` are the battery presence thresholds.

`PRECHARGE_TERMINATION_VOLTAGE` is the threshold for stopping pre-charge and switching to fast charge.

`ALKALINE_DETECTION_VOLTAGE` is the floating voltage threshold for newly detected batteries that sends them directly to Fault mode.

`CHARGE_VOLTAGE_LIMIT` is the maximum charging voltage allowed before going into fault.

`FINISH_CHARGE_START_VOLTAGE` is the charging voltage threshold for switching to finish charge.

`FULL_BATTERY_VOLTAGE` is the floating voltage for newly inserted batteries that goes directly to Done without charging (battery already full).

CONFIGURING THE LIBRARY

The first thing when configuring the library for a given hardware platform is to check the requirements. The requirements are listed in detail in the introduction of this paper. The selected part needs an ECCP peripheral with auto-shutdown, a comparator, a 10-bit ADC with several channels and general purpose IO pins. An internal or external voltage reference is recommended.

The next step is to set the Configuration registers initialization values in `Hardware.h` to the correct value. All the values are defines and are used in the `InitializeHardware()` function.

The first thing that has to be defined is the battery type the charger is going to be used for. This is by default Ni-MH. Currently, there is no support for CC-CV charging (Li-Ion, lead-acid or Ni-Zn).

EXAMPLE 30:

```
#define NiMH
```

Internal running frequency and instruction speed are needed for automatic calculation of certain timing parameters like the PWM frequency or the timer tick pre-load value.

EXAMPLE 31:

```
#define XTAL_FREQ   12000000
#define WORKING_FREQ XTAL_FREQ / 4

#define T2_INIT     0x7C
#define PWM_FREQ    120000
#define PR2_INIT ((WORKING_FREQ / PWM_FREQ)-1)
```

The PWM duty cycles for each charging state are calculated based on the period value. The buck converter duty cycle should not go above 50%.

By enabling the postscaler for the PWM timer makes it easy enough to use for timing in the charger application.

EXAMPLE 32:

```
#define T2_POSTSCALER 16
#define T2_FLAG_TICK WORKING_FREQ / (PR2_INIT + 1) / T2_POSTSCALER
```

To avoid sudden variations in supply voltage, there is a ramp on the PWM when the current is turned on or off. The defined value is added to or subtracted from (on or off) the current PWM value every PWM timer overflow. Do not forget to set the correct Configuration register value for the timer and the postscaler value for other calculations.

EXAMPLE 33:

```
#define PWM_RAMP_STEP1
```

The main 16-bit timer is set by default to 100 ms and all the other main charger loop values are derived from it. The Timer1 pre-load value is also calculated based on this. The LED signaling offers a simple two-type blinking scheme: a slow 0.5 Hz blink and a configurable faster blink.

EXAMPLE 34:

```
#define ONE_SECOND      10
#define LED_TIME        (ONE_SECOND / 2)
#define T1_PRESCALER    8
```

Depending on the values of the components in the RC filter at the battery voltage ADC inputs, charge and discharge times are set using the PWM timer. With 100ko/100nF the RC constant is 10 ms.

```
#define BAT_FILT_DISCHARGE_TIME FTIMER_1MSEC
#define BAT_FILT_CHARGE_TIME FTIMER_50MSEC
```

Setting the Configuration registers is quite straightforward. There are still a few delicate points.

First, the PWM must be configured to shut down automatically every pulse when the maximum current is reached, and restart on the next period. The shutdown signal is the output of a comparator which is connected to the current shunt and a reference voltage. In this case, the reference voltage is the CVREF.

Never start the PWM on maximum duty cycle, or the CVREF on maximum value, before the current loop is confirmed to be working properly.

© 2011 Microchip Technology Inc.

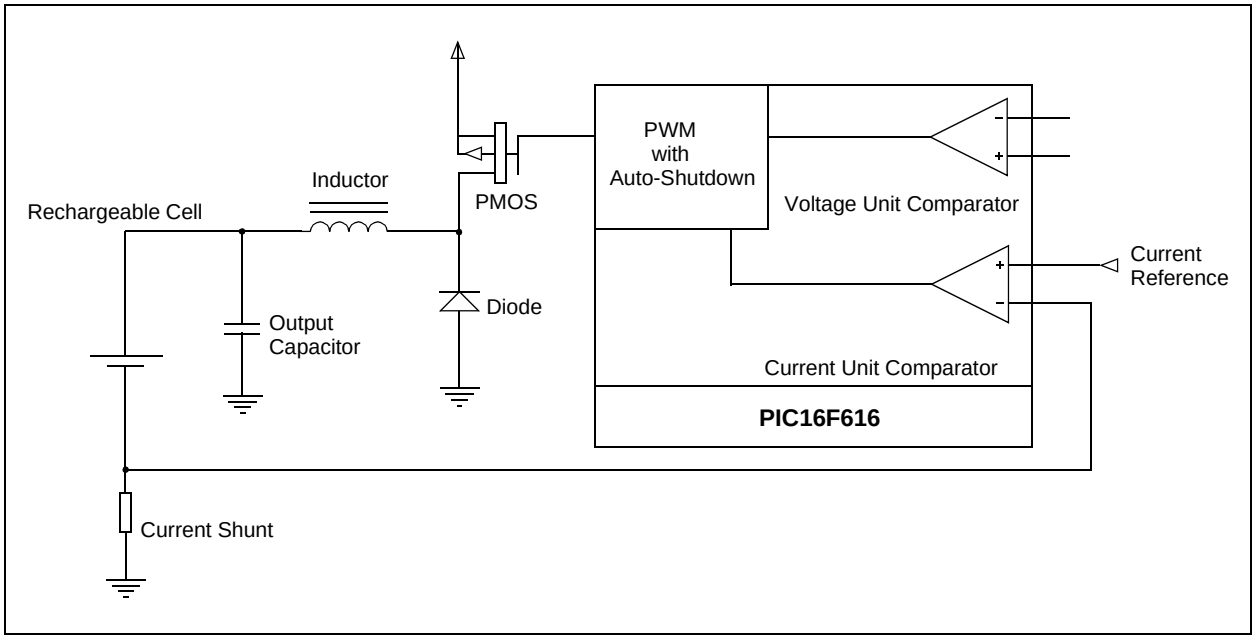
```
#define CCP1CON_INIT      0x0F
#define PWM1CON_INIT      0x80
#define ECCPAS_INIT       0x15
#define PSTRCON_INIT      0x02

#define CM1CON0_INIT      0x9D
#define CM2CON0_INIT      0x00
#define CM2CON1_INIT      0x00

#define REFCON0_INIT      0x90
#define REFCON1_INIT      0xE8
#define REFCON2_INIT      0x00

#define REFCON_TRICKLE    0x04
#define REFCON_FAST       0x11
#define REFCON_FINISH     0x11
```

FIGURE 11: SCHEMATIC



The second issue is related to the ADC reference. VDD is usually not accurate or stable enough and some kind of voltage reference is needed. If an external reference is used, then there is no need for further compensation. Still, if the exact VDD value is needed, the reference must be read on a regular channel while VDD is the reference and deduce the correct supply value.

If the reference is internal or can not serve as VREF for the ADC for any reason, it must be read every time the current is turned on or off. Since the reference should be stable over a large VDD range, it is easy to deduce the correct VDD value and compensate the other ADC readings based on the reference value.

EXAMPLE 35:

```
#define ADCON0_MASK    0x01
#define ADCON1_INIT    0x04
#define ADCON1_VDD     0x00
#define ADCON2_INIT    0x85
```

ADC channels for the battery voltage, temperature and current VREF must be correctly defined. The external VREF is usually the channel marked VREF+. The rest of the defines depend on what type of reference is used.

EXAMPLE 36:

```
#define BAT_A_CHANNEL      0x07
#define BAT_B_CHANNEL      0x08
#define TEMP_A_CHANNEL     0x09
#define TEMP_B_CHANNEL     0x09
#define CURRENT_CHANNEL    0x05

#define FVREF_CHANNEL      0x0E
#define DAC_CHANNEL        0x0F
#define VREF_EXT_CHANNEL   0x04
```

The define #COMPENSATE_VDD should be uncommented when the ADC uses VDD as a reference and all readings will be normalized to a nominal VDD of 5.000V. The accuracy of this scaling depends on the accuracy of the voltage reference.

The rest of the registers are mainly port, tris and analog function settings.

EXAMPLE 37:

```
#define PORTA_INIT     0b00000000
#define PORTC_INIT     0b10110111
#define TRISA_INIT     0b11111111
#define TRISB_INIT     0b00001111
#define TRISC_INIT     0b11001011

#define ANSEL_INIT     0b10110000
#define ANSELH_INIT    0b00000011

#define PIE1_INIT      0x00
#define INTCON_INIT    0x00

#define TXSTA_INIT     0b10100100
#define RCSTA_INIT     0b10000000
#define BAUDCON_INIT   0b00001000
#define SPBRGH_INIT    0
#define SPBRG_INIT     77

#define WDTCON_INIT    0x00
```

The current shunt for the current limiting loop is usually small, in the 50-100mOhm range. This value must be defined to serve in other calculations related to instantaneous current and battery impedance. An OA amplifies the voltage on the shunt before it is fed to the comparator.

EXAMPLE 38:

```
#define    SHUNT_AMPLIFICATION    10
#define    SHUNT_RESISTANCE       56
```

If the shunt is a high-side type, uncomment this line. Otherwise leave it commented because the voltage drop on the shunt is subtracted from the battery charging voltage.

EXAMPLE 39:

```
#define HIGH_SIDE_SENSING
```

If the board is equipped with thermistor(s) for temperature shutdown, this is the type definition. Uncomment for NTC and leave commented for PTC. NTC thermistors have lower impedance for higher temperatures and the code may look atypical in some places because of this.

For example, "`if(temperature < TEMP_40C)`" triggers for temperatures higher than 40°C, not the other way around, if NTC readings are used directly.

All the NTC thermistor readings are subtracted from the maximum ADC value to make it look like the impedance rises with temperature.

EXAMPLE 40:

```
#define THERMISTOR_NTC
```

Normally, the thermistor(s) are only used for detecting environmental conditions. The charger will not start charging at all (and go into Fault) if the temperature is below 0°C or it is above 40°C. Also, if the temperature goes above 50°C while charging, it will generate a Fault condition.

Using a tight thermal coupling between the thermistors and the batteries (one themistor per battery needed) allows to detect the sharp slope present in the last part of the charging cycle and use it to terminate charging before the onset of overcharge. It is a very good charge termination technique, but the cost is a more complex mechanical setup needed to have tight thermal contact between the sensor and the battery. A temperature calculation function that offers 0.1C resolution needs to be implemented.

EXAMPLE 41:

```
#define MIN_START_CHARGE_TEMP     TEMP_0C
#define MIN_CHARGE_TEMP           TEMP_0C
#define MAX_START_CHARGE_TEMP     TEMP_40C
#define MAX_CHARGE_TEMP           TEMP_50C
```

Battery related defines are stored in the `Ni-MH.h` file. It is important to check and modify these values, because they determine the calculated values of the charge timers in each battery state. Threshold values specific to battery chemistry and the charge current used are also here and must be checked thoroughly.

The first thing that needs to be configured is the charger behavior. Fast charging or slow/overnight charging may be selected. If both are defined, fast charging will take precedence.

EXAMPLE 42:

```
#define CHARGER_OVERNIGHT
#define CHARGER_FAST
```


© 2011 Microchip Technology Inc.

Besides the obvious timer, voltage and temperature fail-safe mechanisms, there are two main voltage-based triggers for fast charge termination. It is highly recommended to keep negative delta voltage and flat voltage termination active when fast charging.

EXAMPLE 43:

```
#define CHG_TERM_NEG_DELTA_V_ENABLE
#define CHG_TERM_FLAT_DELTA_V_ENABLE
```

To avoid problems related to noise, charge termination protocols have special counters that trigger when they reach 0. For example, negative delta voltage triggers if the last 15 samples the charging voltage has been at least 5 mV lower than the last recorded peak. Also, the peak is only updated if the last 5 samples of the charging voltage are higher than the last recorded peak.

Flat delta voltage has a longer trigger time, set to 15 minutes. If the decrement condition is not true at any time, the counter value is returned to the initial value.

It is important to remember that the batteries are charged one second at a time and, unless the finish charge full current option is enabled, these values are actually multiplied by the number of battery slots (time-wise).

EXAMPLE 44:

```
#define NDV_SAMPLES           15
#define FDV_SAMPLES           15*60
#define PEAK_UPDATE_SAMPLES   5
#define NDV_VOLTAGE_DROP      5
```

Impedance testing is done in a very simple way. The maximum allowed impedance in mOhms is multiplied by the instantaneous fast charge current to obtain the maximum allowed difference between charging and floating voltage. A more accurate way to do this would be to multiply the current reading by the maximum impedance every time impedance Fault conditions are checked. Of course, this long integer multiplication eats up some extra memory.

EXAMPLE 45:

```
#define Z_TEST_LIMIT             150
#define INSTANT_CHARGE_CURRENT   850
#define CHARGE_MINUS_FLOAT_LIMIT  (unsigned long) INSTANT_CHARGE_CURRENT * Z_TEST_LIMIT / 1000
```

Fast charge and finish charge timers are calculated automatically using the maximum cell capacity allowed in the charger. If higher capacities are used, the charge will not be complete. A typical 80% efficiency (125% power) is used to calculate the actual timer values. Also, the finish charge is defined to be no more than 33% of the calculated fast charge time.

EXAMPLE 46:

```
#define MAX_BATTERY_CAPACITY_AAA     1000
#define MAX_BATTERY_CAPACITY_AA      3000

#define CHARGE_EFFICIENCY_INV        125
#define FINISH_CHARGE_TIME_FRACTION  33

#define FAST_CHARGE_TIME_NOLOSS      (unsigned long) MAX_BATTERY_CAPACITY_AAA * 60 * 60 /
                                     INSTANT_CHARGE_CURRENT

#define FAST_CHARGE_TIME_LIMIT       (unsigned long) FAST_CHARGE_TIME_NOLOSS *
                                     CHARGE_EFFICIENCY_INV / 100

#define FINISH_CHARGE_TIME_LIMIT     (unsigned long) FAST_CHARGE_TIME_LIMIT *
                                     FINISH_CHARGE_TIME_FRACTION /        100
```

The overnight and top-off (if applicable) charge times are declared manually.

EXAMPLE 47:

```
#define TOPOFF_CHARGE_TIME    60*60
#define OVERNIGHT_CHARGE_TIME 900*60
```

© 2011 Microchip Technology Inc.

One very important #define selects if finish charge will put all available current into one battery until one of the charge termination protocols triggers. This is very helpful in getting a reliable negative delta voltage waveform but may also cause additional heating. First, check if the average charging current (instantaneous current/no. of battery slots) is higher than C/2 current of the battery. In this example, the instantaneous current is 850 mA and there are two battery slots. Charging a 750 mAh battery without full current in finish charge should allow reliable –dV detection. Remember though, smaller charge currents delay the –dV waveform further into the overcharge region (even if the battery does not heat up).

EXAMPLE 48:

```
#define FINISH_CHARGE_FULL_CURRENT
```

A set of voltage thresholds typical to the Ni-MH chemistry (and the smaller AAA battery format) are used to transition charge states, to detect wrong battery types and other faults. They may be modified to suit the application's needs but when in doubt, obtain or discuss them with the cell manufacturer. All values are in mV.

Battery removal and detection is straightforward enough.

EXAMPLE 49:

```
#define BATTERY_REMOVAL_VOLTAGE   400
#define BATTERY_INSERTION_VOLTAGE 500
```

Deeply discharged cells are not fast-charged because the high internal impedance causes heating. They are instead trickle-charged until the voltage reaches 1.0V.

EXAMPLE 50:

```
#define PRECHARGE_TERMINATION_VOLTAGE 1000
```

Usually, a very high floating voltage upon insertion of a cell means it is the wrong type. The charger puts the slot in Fault mode.

EXAMPLE 51:

```
#define ALKALINE_DETECTION_VOLTAGE 1600
```

The maximum charge voltage limit is 1.7V. This may change if higher charging currents are used.

EXAMPLE 52:

```
#define CHARGE_VOLTAGE_LIMIT 1700
```

The transition voltage from fast charge to finish charge is set to 1470 mV. This may also change with charging current but do not set it too high or the battery might never change state. This is not good, since fast charge does not have any charge termination protocols.

EXAMPLE 53:

```
#define FINISH_CHARGE_START_VOLTAGE 1470
```

A high floating voltage upon insertion (but not high enough to cause a Fault) usually means the rechargeable cell is full or a partially depleted alkaline is present. The charger puts the slot automatically in the Done state. No charging is done.

EXAMPLE 54:

```
#define FULL_BATTERY_VOLTAGE 1400
```

Discharged batteries have a higher internal impedance, so it is sometimes recommended to check for impedance faults after the cell has reached a minimum charging voltage.

EXAMPLE 55:

```
#define IMPEDANCE_CHECK_VOLTAGE   1200
```


© 2011 Microchip Technology Inc.

CONCLUSIONS

The Microchip Battery Charger Library allows users to quickly add charging functionality to their applications. The hardware is very simple and the fact that it can charge more than one battery with one current source makes it cost-efficient. While there are not any magic recipes for charging batteries and each manufacturer has its own set of recommendations, the library is highly customizable and can be easily reconfigured to match them.

Whether it is a stand-alone charger or a more complex device, the battery charger library may prove to be a valuable tool in portable applications.

NOTES:

© 2011 Microchip Technology Inc.

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE**.** Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, PIC^{32} logo, rfPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Omniscient Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICtail, REAL ICE, rfLAB, Select Mode, Total Endurance, TSHARC, UniWinDriver, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2011, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.

ISBN: 978-1-61341-376-0

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2009

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario, Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hangzhou
Tel: 86-571-2819-3180
Fax: 86-571-2819-3189

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-6578-300
Fax: 886-3-6578-370

Taiwan - Kaohsiung
Tel: 886-7-213-7830
Fax: 886-7-330-9305

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

05/02/11

© 2011 Microchip Technology Inc.