Features

- User-Configurable Battery Charger for Lithium, Nickel and Lead Battery Packs
- Based on PIC16F785 with Integrated Shunt Regulator
- Firmware and Support Tools for easy design
- 10-bit ADC for Voltage, Current and Temperature Measurement:
  - Accurate voltage regulation (+/-1%)
  - Accurate current regulation (+/-5%)
- Advanced Charge Algorithms:
  - Chemistry dependent End-of-Charge determination
  - Charge qualification to detect shorted, damaged or heated cells
  - Precharge for deeply discharged cells
  - Configurable overtemperature and overvoltage charge suspension
  - Charge termination at user-specified minimum current or time-out
  - Configurable charge status display via two LEDs
- Maximum Integration for Optimal Size:
  - Integrated voltage regulator
  - Internal 8 MHz clock oscillator
  - High-Frequency Switch mode charging – configurable switching frequency up to 500 kHz

Applications

- Single-Cell and Multi-Cell Lithium, Nickel and Lead Battery Chargers
- Notebook Computers
- Personal Data Assistants
- Cellular Telephones
- Digital Still Cameras
- Camcorders
- Portable Audio Products
- Bluetooth® Devices

Pin Description

<table>
<thead>
<tr>
<th>PIC16HV785</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vdd</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LED2</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIN</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESET</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTRLIN</td>
<td>9</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHGOUT</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOOPIN</td>
<td>8</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOOPFBK</td>
<td>7</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHGFBK</td>
<td>6</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BATID</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IFBINA</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IFBINB</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HVOUT</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

20-Pin PDIP, SOIC, SSOP
USING THE PIC16HV785

Product Overview

The PIC16HV785 provides an unprecedented level of configurability for charging Lithium Ion/Lithium Polymer battery packs. Its precise, 10-bit Analog-to-Digital converter and high-frequency Pulse-Width Modulator enable the PIC16HV785 to provide optimum control of charging algorithms for lithium battery chemistries. Special features include an internal voltage regulator and an internal clock oscillator that reduce external component count. The PIC16HV785 can be configured as either a Switch mode or a linear charger. In Switch mode, it will support either primary or secondary side control. In Linear mode, it can be designed into applications requiring low-power supply noise.

MULTI-STEP CHARGING

To insure the proper treatment of lithium chemistries during extreme temperature and voltage conditions, multi-step charging is required. The PIC16HV785 starts the charging cycle upon sensing the presence of a battery pack and a valid charging supply. During charge qualification, the battery’s temperature and voltage are measured to determine the appropriate initial state. The initial states include Charge Suspend, Precharge and Current Regulation. Charge Suspend halts charging when the user-defined preset conditions for charging are not met. Precharge allows for the recovery of deeply discharged batteries by applying a low charge (or C) rate. Current Regulation provides constant current, voltage limited charge. Upon reaching the target voltage during Current Regulation, the Voltage Regulation state is entered. Charging continues at a constant voltage until the current decreases to the user-specified minimum current threshold. The user-specified minimum current threshold can be configured for various charging temperatures. At this threshold, charging is terminated and the End-of-Charge state is reached.

USER-CONFIGURABLE PARAMETERS

The PIC16HV785 supports user-configurable parameters that allow for customizing the charging profile without changing the charger’s hardware design. This feature allows for the maximum reuse of hardware, thus reducing time-to-market. These parameters include:

- Battery Temperature:
  - Minimum/maximum temperature for charge initiation
  - Maximum temperature allowed during charge
- Battery Voltage:
  - Minimum/maximum voltage for charge initiation
  - Target voltage during voltage regulation
  - Voltage at which the charger will restart charging after completion of a valid charge cycle
- Charge Current:
  - Target current during current regulation
  - Taper current threshold for End-of-Charge during voltage regulation
  - Target current during precharge
- Time:
  - Precharge time limit
  - Current regulation time limit
  - Voltage regulation time limit
- Status Display:
  - Duty cycle for the two LEDs denoting charge states can be modified

These parameters are configured through the PowerTool™ 200 Development Software for the PIC16HV785.

SPECIAL FEATURES

The PIC16HV785 includes a voltage regulator, a voltage reference, an internal clock oscillator and a high-frequency Pulse-Width Modulator.

- The internal voltage regulator has a maximum input voltage of 18V and eliminates the need for external references
- The precise, internal 8 MHz clock oscillator eliminates the need for external oscillator circuits
- The high-speed Pulse-Width Modulator is used for power regulation and can support frequencies up to 500 kHz
- In-circuit configurability utilizing on-board EEPROM
# TABLE 1: PINOUT DESCRIPTION

<table>
<thead>
<tr>
<th>Pin</th>
<th>Pin Name</th>
<th>Pin Type</th>
<th>Input Type</th>
<th>Output Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VDD</td>
<td>Supply</td>
<td>Power</td>
<td>—</td>
<td>Supply voltage</td>
</tr>
<tr>
<td>2</td>
<td>LED2</td>
<td>O</td>
<td>—</td>
<td>CMOS</td>
<td>Status indicator</td>
</tr>
<tr>
<td>3</td>
<td>VIN</td>
<td>I</td>
<td>Analog</td>
<td>—</td>
<td>Battery voltage input</td>
</tr>
<tr>
<td>4</td>
<td>RESET</td>
<td>I</td>
<td>ST</td>
<td>—</td>
<td>Reset</td>
</tr>
<tr>
<td>5</td>
<td>CTRLOUT</td>
<td>O</td>
<td>—</td>
<td>CMOS</td>
<td>PWM output for setting current level</td>
</tr>
<tr>
<td>6</td>
<td>CHGOUT</td>
<td>O</td>
<td>—</td>
<td>CMOS</td>
<td>PWM output to a buck converter for charge control</td>
</tr>
<tr>
<td>7</td>
<td>LOOPFBK</td>
<td>I</td>
<td>Analog</td>
<td>—</td>
<td>Current feedback loop</td>
</tr>
<tr>
<td>8</td>
<td>LOOPIN</td>
<td>I</td>
<td>Analog</td>
<td>—</td>
<td>Current feedback loop input</td>
</tr>
<tr>
<td>9</td>
<td>CTRLIN</td>
<td>I</td>
<td>Analog</td>
<td>—</td>
<td>Current level control</td>
</tr>
<tr>
<td>10</td>
<td>LED1</td>
<td>O</td>
<td>—</td>
<td>CMOS</td>
<td>Status indicator</td>
</tr>
<tr>
<td>11</td>
<td>HVOUT</td>
<td>O</td>
<td>—</td>
<td>HVOD</td>
<td>High-voltage, open-drain output pin (optional)</td>
</tr>
<tr>
<td>12</td>
<td>IFBINB</td>
<td>I</td>
<td>Analog</td>
<td>—</td>
<td>Current feedback input pin B used for current scaling</td>
</tr>
<tr>
<td>13</td>
<td>IFBINA</td>
<td>I</td>
<td>Analog</td>
<td>—</td>
<td>Current feedback input pin A used for current scaling</td>
</tr>
<tr>
<td>14</td>
<td>IFBOUT</td>
<td>O</td>
<td>—</td>
<td>Analog</td>
<td>Current feedback output</td>
</tr>
<tr>
<td>15</td>
<td>BATID</td>
<td>I</td>
<td>Analog</td>
<td>—</td>
<td>Battery ID select</td>
</tr>
<tr>
<td>16</td>
<td>CHGFBK</td>
<td>I</td>
<td>Analog</td>
<td>—</td>
<td>Charge control feedback</td>
</tr>
<tr>
<td>17</td>
<td>SHDN</td>
<td>O</td>
<td>—</td>
<td>Analog</td>
<td>Shutdown signal, active-low</td>
</tr>
<tr>
<td>18</td>
<td>VOVP</td>
<td>I</td>
<td>Analog</td>
<td>—</td>
<td>Overvoltage protection</td>
</tr>
<tr>
<td>19</td>
<td>TEMP</td>
<td>I</td>
<td>Analog</td>
<td>—</td>
<td>Battery temperature input</td>
</tr>
<tr>
<td>20</td>
<td>VSS</td>
<td>Supply</td>
<td>Power</td>
<td>—</td>
<td>Supply ground</td>
</tr>
</tbody>
</table>

**Legend:**  
*I* = Input,  
*O* = Output,  
*ST* = Schmitt Trigger Input Buffer,  
*HVOD* = High-Voltage Open-Drain
PIC16HV785 HARDWARE OVERVIEW

The PIC16HV785 is a configurable, Switch mode charger which is comprised of a PIC16F microcontroller core and precision analog circuitry. This section explores the hardware features in relation to generic Switch mode charging. The PIC16HV785 hardware is a PIC16F785 device with an integrated shunt regulator, to allow the device to be powered directly from a battery stack, or from charger voltage. It is available in a 20-pin PDIP, SOIC or SSOP package. See the PIC16F785 data sheet for more hardware description. Hardware features include:

- Oscillator
- Power-Saving Sleep mode
- Power-on Reset (POR)
- Brown-out Reset (BOR)
- High-Endurance Flash/EEPROM Cell:
  - 100,000 write Flash endurance
  - 1,000,000 write EEPROM endurance
  - Flash/Data EEPROM retention: > 40 years
- High-Speed Comparator module with:
  - Two independent analog comparators
- Operational Amplifier module with two independent op amps
- Two-Phase Asynchronous Feedback PWM
- Voltage Regulator
- 10-bit A/D Converter
- In-Circuit Serial Programming™ (ICSP™) via two pins

Hardware Features

The PIC16HV785 features are well-suited for Switch mode battery charging. The PIC16HV785 device’s block diagram (Figure 1) is to be used in conjunction with the Switch mode charger example (Figure 10, page 9).

- Current/Voltage Measurement Block – The Current/Voltage Measurement Block consists of a 10-bit Analog-to-Digital converter, operational amplifiers and a comparator. The output of this block is fed into the charge control module (refer to Figure 1). The inputs into this block are to be connected as described in Figure 10. The following signals are inputs into this block:
  - LOOPFBK: to comparator
  - LOOPIN: to op amp and ADC
  - CTRLIN: to op amp
  - IFBINA: to op amp
  - IFBINB: to op amp
  - BATID: to ADC
  - TEMP: to ADC
  - CHGFBK: to comparator
The following signals are outputs from this block:
  - IFBOUT: from op amp

- Charge Control Module – The charge control module generates a Pulse-Width Modulated signal called CHGOUT. Its frequency is configurable and can be set up to 1 MHz. This signal is connected to an external DC/DC buck converter.

- Voltage Regulator – The integrated voltage regulator is designed to work with unregulated DC supplies.

- The precision internal 8 MHz clock oscillator eliminates the need for external oscillator circuits.
- In-circuit configurability utilizing 256 bytes of on-board EEPROM.

- Power on Reset – The POR insures the proper start-up of the PIC16HV785 when voltage is applied to VDD.
- Brown-out Reset – The BOR is activated when the input voltage falls to 2.1V; the PIC16HV785 is reset.
FIGURE 1: PIC16HV785 BLOCK DIAGRAM

[Block diagram showing the PIC16HV785 block diagram with various components and connections, including Voltage Regulator, Voltage Reference, Internal Oscillator, Charge Control Module, and Current/Voltage Measurement Block.]

- Voltage Regulator
- Voltage Reference
- Internal Oscillator
- Charge Control Module
- Current/Voltage Measurement Block
- Components labeled: CTRLIN, LOOPIN, LOOPFBK, CHGFBK, IFBINB, IFBINA, I FBOUT, VIN, BATID, TEMP, VDD, VSS, VOVP, SHDN, CTRLOUT, CHGOUT, LED1, LED2, HVOUT.
REFERENCE SCHEMATIC

Theory of Operation

In this schematic, the PIC16HV785 is being used to control a step-down buck converter. A buck converter uses a square wave pulse train to turn on and off a switch that provides current into an inductor. The ratio of output voltage to input voltage is the duty cycle of the pulse. Current and voltage feedback are used to control the duty cycle to regulate the output voltage and current.

Buck Converter

The inductor L1, the capacitor COUT and diode D1 comprise the buck converter. The MOSFET Q1 is the switch that applies the charger voltage when turned on. It is driven by a pulse train applied by the PIC16HV785.

In the above diagram, when a constant voltage is applied to VIN and a pulse train of constant frequency and duty cycle is applied to the gate of the MOSFET, the result is a constant voltage at VOUT which is a fraction of VIN equal to the duty cycle of the pulse.

The voltage drop across the inductor is:

\[ V_L = L \frac{di}{dt} \]

With the voltage regulated at VOUT, the drop across the inductor is \( V_{IN} - V_{OUT} \), thus the current through the inductor is:

\[ i = \int (V_{IN} - V_{OUT}) \, dt \]

This integral taken over one pulse cycle can be broken down into pulse on and pulse off time. When the pulse is on, \( V_{IN} = V_{CHARGE} \), and when the pulse is low, \( V_{IN} = 0 \). Since the current is the same at the beginning of each cycle, the equation becomes:

\[ (V_{CHARGE} - V_{OUT}) \cdot T_{ON} - V_{OUT} \cdot T_{OFF} = 0 \]

or

\[ V_{CHARGE} \cdot T_{ON} = V_{OUT} \cdot T_{ON} + V_{OUT} \cdot T_{OFF} \]

\[ V_{OUT} = (T/T_{ON}) \cdot V_{CHARGE} \]

\[ V_{OUT} = V_{CHARGE} \cdot \text{Duty Cycle} \]

When the pulse goes high, the current through the inductor increases as a response. When the pulse goes low, the current decreases. The graph (Figure 3) shows the current through the inductor as a response to the input pulse, and the resulting voltage drop across the inductor.

When the current through the inductor is increasing, as a result of the pulse going high, the voltage drop across the inductor is positive (\( \frac{di}{dt} \) is positive). This drop is subtracted from the applied charge voltage to produce VOUT. When the current through the inductor is decreasing (\( \frac{di}{dt} \) is negative), the voltage drop across the inductor is negative, adding to the zero input voltage to produce VOUT.

Feedback Circuits

The circuit uses feedback for two purposes. One is to provide the ramp waveform that defines the PWM duty cycle. The other is the current sense that is compared to a reference voltage to determine if the current is being regulated at the correct level. This is also fed back into the PWM to modulate the duty cycle.
RAMP FEEDBACK

The CHGFBK pin (pin 16) receives the ramp sawtooth waveform that controls the duty cycle of the PWM signal. This sawtooth needs to be generated externally by an RC network connected to the PWM output. The RC network uses the frequency of the PWM to generate the sawtooth waveform. When the PWM is triggered high, the sawtooth starts to ramp up. When the sawtooth reaches a certain point (determined internally by reference voltage and current feedback), the PWM output is sent low, also driving the sawtooth low. The sawtooth starts up again when the internal oscillator sends the PWM high again.

The RC circuit can be placed on the output of the PWM signal. A clamping diode can be used to control the total voltage drop.

CURRENT FEEDBACK

The aforementioned reference voltage is determined by current feedback in order to regulate the current. A second PWM, which is under firmware control, is used to create a DC level to which to compare the sensed current. The voltage drop across a current sense resistor is applied to pin 13 (IFBINA) and is internally amplified by an op amp. The output of this op amp is available on pin 14 (IFBOUT). The output on pin 14 is then fed into pin 8 (LOOPIN) which is the input to another op amp. The other input of this op amp is a DC level that is created by the firmware controlled PWM. The firmware controlled PWM is output on pin 5 (CTRLOUT) and fed into an RC circuit whose time constant is high enough to create a rough DC level. This DC level will vary with the duty cycle of the firmware controlled PWM. This DC level is then applied to pin 9 (CTRLIN). This DC level is compared to the current feedback by op amp 1. The output of op amp 1 is fed to the main internal comparator where it is compared to the sawtooth waveform to determine the duty cycle of the main PWM, which regulates current through the buck converter.

The actual circuit implementation, including op amp feedback RC networks, is shown below.
Power Supply Shunt Regulator

The PIC16HV785 has a built-in shunt regulator to allow the device to be powered directly by the charging voltage. The integrated voltage regulator is designed to work with unregulated DC supplies. While there is, theoretically, no limit to the charging voltage, there are guidelines that should be followed. A series limiting resistor (RVDD) should be placed between the unregulated supply and the VDD pin. The value for this series resistor (RVDD) must be between RMIN and RMAX as shown in the following equation:

\[
R_{\text{MIN}} = \frac{V_s(\text{MIN}) - 5V}{0.95 \times 50 \text{ mA}}
\]

\[
R_{\text{MAX}} = \frac{V_s(\text{MAX}) - 5V}{1.05 \times (16 \text{ mA} + I(\text{led}))}
\]

Where:
- RMAX = maximum value of series resistor (ohms)
- RMIN = minimum value of series resistor (ohms)
- Vs(MIN) = minimum value of charger DC supply (VDC)
- Vs(MAX) = maximum value of charger DC supply (VDC)
- I(led) = total current drawn by all LEDs when illuminated simultaneously

Note: The 1.05 and .95 constants are included to compensate for the tolerance of 5% resistors. The 16 mA constant is the anticipated load presented by the PIC16HV785, including the loading, due to external components and a 4 mA minimum current for the shunt regulator itself. The 50 mA constant is the maximum acceptable current for the shunt regulator.

Overvoltage Protection

The PIC16HV785 has a comparator that is gated to the PWM which compares the reference voltage to an external divided voltage applied to pin 18 (VovP). When the voltage on pin 18 exceeds the reference voltage, the PWM is turned off. The external voltage divider should be chosen such that the preferred overvoltage safety point is used.

A/D Inputs

The internal A/D converter is used to measure the charging voltage on pin 3 (Vin), the current on pin 13 (IFBINA) and optionally, the temperature on pin 19 (TEMP) if there is a thermistor present. An external voltage divider is used on pin 3 to measure the charge voltage.
FIGURE 10: PIC16HV785 SWITCHING CHARGER SCHEMATIC

- **FIGURE 10:** PIC16HV785 SWITCHING CHARGER SCHEMATIC

- **POWER SUPPLY INPUT**
  - SUPPLY MUST BE FUSED OR CURRENT LIMITED
  - POWER SUPPLY INPUT

- **CHARGER OUTPUT TO BATTERY**
  - CHARGER OUTPUT TO BATTERY

- **THERMISTOR**
  - THERMISTOR

- **SUPPLY MUST BE FUSED OR CURRENT LIMITED**
  - SUPPLY MUST BE FUSED OR CURRENT LIMITED

- **OVP DIVIDER**
  - OVP DIVIDER

- **VIN DIVIDER**
  - VIN DIVIDER

- **D2: REVERSE CURRENT BLOCKING DIODE**
  - D2: REVERSE CURRENT BLOCKING DIODE

- **CONNECTIONS**
  - CONNECTIONS

- **FIRMWARE CURRENT CONTROL OUTPUT:**
  - FIRMWARE CURRENT CONTROL OUTPUT:

- **COMPONENTS THAT DO NOT HAVE A NUMERICAL VALUE ASSOCIATED WITH THEM ARE DEPENDENT UPON THE VOLTAGE/CURRENT SPECIFICATIONS OF THE PARTICULAR APPLICATION. CONSULT MICROCHIP FOR GUIDANCE IN DEFINING THESE COMPONENT VALUES.**

- **4.7 μF**
  - 4.7 μF
FUNCTIONAL DESCRIPTION:
LITHIUM CHEMISTRY

Lithium Charging
To ensure the proper treatment of lithium chemistries during extreme temperature and voltage conditions, multi-step charging is required. The PIC16HV785 measures key voltage, temperature and time parameters. It compares them to user-defined voltage, temperature and time limits.

CHARGE PENDING STATE – BEGINNING THE CHARGE CYCLE
The PIC16HV785 is initially set in the Charge Pending state. In this state, the presence of a battery pack must be sensed in order to begin the charging cycle. The PIC16HV785 comes up in the Charge Pending state after a Reset, independent of the previous state.

CHARGE QUALIFICATION STATE
During Charge Qualification, the battery’s temperature and voltage are measured to determine the next charging state. There are four possible next states:
1. If the battery’s temperature is outside of the limits for charge initiation (StartTMax, TempMin), then the next state is Charge Suspend.
2. If the battery’s voltage is less than the precharge threshold (PCVolt) and its temperature is within the limits for charge initiation (StartTMax, TempMin), then the next state is Precharge.
3. If the battery’s voltage is above the precharge threshold (PCVolt) and its temperature is within the limits for charge initiation (StartTMax, TempMin), then the next state is Current Regulation.
4. If the battery's voltage is above the voltage at which charging will restart (Vrchg), then the next state is Charge Complete.

PRECHARGE STATE
The Precharge state allows for the recovery of a deeply discharged battery pack by applying a low charge rate. In this state, a user-configured precharge current is applied to the battery, resulting in an increase in the battery's voltage. There are three possible next states:
1. If the battery’s voltage is above the precharge threshold (PCVolt) and its temperature is within the limits for charge initiation (StartTMax, TempMin), then the next state is Current Regulation.
2. If the Precharge state time limit is exceeded (PCTimeMax) and the battery’s temperature is less than the minimum temperature for charge initiation (StartTMin), then the next state is Charge Suspend.
3. If the battery pack is taken away, then the PIC16HV785 enters the Charge Pending state.

CHARGE SUSPEND STATE
In the Charge Suspend state, no current is applied to the battery pack. There are two possible next states:
1. If the battery’s temperature is within the limits for charge initiation (StartTMax, TempMin) and its voltage is less than the voltage at which charging would restart (VRVrech), then the next state is Precharge.
2. If the battery pack is taken away, then the PIC16HV785 enters the Charge Pending state.

CURRENT REGULATION STATE
The Current Regulation state can be entered from the Precharge state or Charge Qualification state. Battery charging is initiated. This state provides constant current, voltage limited charging. The charge current is referred to as CRCurrent or the regulation current. While the current is applied, the battery's voltage increases until it reaches a voltage limit referred to as CRVTarg or regulation voltage. Charging continues, during which battery voltage and temperature are monitored. There are three possible next states:
1. If the battery’s voltage reaches or exceeds the voltage limit (CRVTarg) and its temperature remains below the maximum allowable during current regulated charging (TempMax), then the next state is Voltage Regulation.
2. If the battery exhibits any one of the following conditions, then the next state is Charge Suspend:
   - Battery voltage exceeds upper voltage limit for charging (Vmax)
   - Battery temperature exceeds upper temperature limit for charging (TempMax)
   - Battery temperature is below the lower temperature limit for charging (TMIN)
   If the time in the Current Regulation state exceeds the time limit (CRTimeMax), then the next state is Charge Suspend.
3. If the battery pack is taken away, then the PIC16HV785 enters the Charge Pending state.
VOLTAGE REGULATION STATE
Voltage Regulation provides charging at a constant voltage while the charge current decreases (or tapers) to the user-specified minimum current threshold (VRIMin). There are three possible next states.
1. When the charge current reaches the taper current threshold for End-of-Charge (VRIMin) and the battery’s voltage remains at the regulated voltage value (CRVTarg), then the battery has reached the Charge Complete state.
2. If the battery exhibits any one of the following conditions, then the next state is Charge Suspend.
   - Battery voltage exceeds upper voltage limit for charging (Vmax)
   - Battery temperature exceeds upper temperature limit for charging (TempMax)
If the time in the Voltage Regulation state exceeds the time limit (VRTimeMax), then the next state is Charge Suspend.
3. If the battery pack is taken away, then the PIC16HV785 enters the Charge Pending state.

CHARGE CYCLE COMPLETE STATE
When the current is less than the taper current threshold (VRIMin) and the voltage is greater than the target voltage (CRVTarg), End-of-Charge is triggered. At this threshold, charging is terminated and the End-of-Charge state is reached. The PIC16HV785 can renew the charge cycle by entering the Charge Pending state when: 1) the battery is removed, or 2) if the battery’s voltage falls below the recharge threshold voltage (VRVrech).

FIGURE 11: PIC16HV785 LITHIUM CHARGING PROFILE
FUNCTIONAL DESCRIPTION:
NICKEL CHEMISTRY

Nickel Charging
To ensure the proper treatment of nickel chemistries during extreme temperature and voltage conditions, multi-step charging is required. The PIC16HV785 measures key voltage, temperature and time parameters. It compares them to user-defined voltage, temperature and time limits.

CHARGE PENDING STATE – BEGINNING THE CHARGE CYCLE
The PIC16HV785 is initially set in the Charge Pending state. In this state, the presence of a battery pack must be sensed in order to begin the charging cycle. The PIC16HV785 comes up in the Charge Pending state after a Reset, independent of the previous state.

CHARGE QUALIFICATION STATE
During charge qualification, the battery’s temperature and voltage are measured to determine the next charging state. There are four possible next states.
1. If the battery’s temperature is greater than the charge limit (MaxTmp), or the voltage is greater than the charge limit (MaxVolt), then the next state is Trickle Charge.
2. If the battery’s voltage is less than the precharge threshold (PCVolt), or its temperature is outside of the limits for fast charge (PCTempLo, PCTempHi), then the next state is Precharge.
3. If the battery’s voltage is above the precharge threshold (PCVolt) and its temperature is within the limits for fast charge (PCTempLo, PCTempHi), then the next state is Fast Charge.

PRECHARGE STATE
The Precharge state allows for the recovery of a deeply discharged battery pack by applying a low charge rate, or a slow recharge of a battery when the temperature is too high or too low to accept a fast charge. In this state, a user-configured precharge current is applied to the battery, resulting in an increase in the battery’s voltage. There are three possible next states.
1. If the battery’s voltage is above the precharge threshold (PCVolt) and the battery’s temperature is within the limits for charge initiation (PCTempLo, PCTempHi), then the next state is Fast Charge.
2. If the Precharge state time limit is exceeded (PCTimeMax) while still in Precharge state, then the next state is Trickle Charge.
3. If the battery pack is taken away, then the PIC16HV785 enters the Charge Pending state.

FAST CHARGE STATE
In Fast Charge state, the full charging current is applied to the battery. There are three possible next states.
1. The charger will enter Trickle Charge state under any of the following conditions:
   - The voltage exceeds MaxVolt.
   - The temperature exceeds MaxTmp.
   - The time exceeds FCTimeMax.
   - -dV (minus delta V) full trigger is detected and time is greater than FCTimeMin.
   - dT/dt full trigger is detected and time is greater than FCTimeMin and Top Off state is disabled.
2. The charger will enter Top Off state when dT/dt full trigger is detected and time is greater than FCTimeMin and Top Off state is enabled.
3. If the battery pack is taken away, then the PIC16HV785 enters the Charge Pending state.

TOP OFF STATE
In the Top Off state, the dT/dt trigger has been reached and a lower amount of current is delivered to top off the battery. There are two possible next states.
1. The battery will exit Top Off state and enter Trickle Charge under the following conditions:
   - The voltage exceeds MaxVolt
   - The temperature exceeds MaxTmp
   - The time exceeds TPTimeMax
2. If the battery pack is taken away, then the PIC16HV785 enters the Charge Pending state.

TRICKLE CHARGE STATE
Trickle Charge state provides a very small constant current to maintain a full charge on the battery. There are two possible next states.
1. The battery will enter Charge Complete state if the voltage exceeds MaxVolt or the time exceeds TPTimeMax.
2. If the battery pack is taken away, then the PIC16HV785 enters the Charge Pending state.

CHARGE CYCLE COMPLETE STATE
When the charge is complete, all charge current terminates. The battery will re-enter Trickle Charge mode when the voltage is less than Vrech and voltage recharge is enabled. Charge complete is determined by the dT/dt method or the -dV method.
1. dT/dt: When the rate of change of the temperature increases suddenly, the temperature increases by greater than dTDelta in time dTTTime and the charging can be declared complete.
2. -dV: When the voltage changes suddenly in the negative direction, drops by dVDetect suddenly after rising throughout the charge cycle, the charging can be declared complete.
CONFIGURABLE PARAMETERS

The PIC16HV785 device’s configurable parameters allow for flexible changes in designing battery chargers. The parameters are categorized as follows:

- Configuration
- Lithium Charging
- Nickel Charging
- LED Display Configuration
- Look-up Tables

Configuration Parameters

The configuration parameters provide an identity to the battery pack and provide its basic characteristics to the PIC16HV785.

Lithium Charging

The lithium parameters govern precharge conditions, current regulation conditions and voltage regulation conditions, as well as when the battery is full and when charging should be suspended.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th># Bytes</th>
<th>Typical Value</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BandgapCF</td>
<td>2</td>
<td>248</td>
<td>integer</td>
<td>Internal band gap calibration factor.</td>
</tr>
<tr>
<td>BattDetectVolt</td>
<td>2</td>
<td>50</td>
<td>mV</td>
<td>Voltage threshold for battery detection.</td>
</tr>
<tr>
<td>BattIDMax</td>
<td>1</td>
<td>255</td>
<td>A/D full scale divided by 255</td>
<td>BATID input pin value maximum. When using BATID pin battery detection, voltage on BATID pin must be between BattIDMax and BattIDMin for battery present.</td>
</tr>
<tr>
<td>BattIDMin</td>
<td>1</td>
<td>0</td>
<td>A/D full scale divided by 255</td>
<td>BATID input pin value minimum. When using BATID pin battery detection, voltage on BATID pin must be between BattIDMax and BattIDMin for battery present.</td>
</tr>
<tr>
<td>Capacity (mAh)</td>
<td>2</td>
<td>2000</td>
<td>mAh</td>
<td>Full-charge capacity of the battery pack. For reference only.</td>
</tr>
<tr>
<td>CurrentCF</td>
<td>2</td>
<td>2553</td>
<td>integer</td>
<td>Current calibration factor.</td>
</tr>
<tr>
<td>DevName</td>
<td>—</td>
<td>PIC16HV785</td>
<td>ASCII</td>
<td>Device name. For reference only.</td>
</tr>
<tr>
<td>MfgName</td>
<td>—</td>
<td>Microchip</td>
<td>ASCII</td>
<td>Manufacturer’s name. For reference only.</td>
</tr>
<tr>
<td>Mode</td>
<td>1</td>
<td>00000001b</td>
<td>binary</td>
<td>Configuration Register: bit 7: 1 = Always step through Precharge state bit 6: 1 = Enable GPIO cutoff logic bit 5-3: Unused bit 2: 1 = Battery present on BATID bit 1: 1 = Battery present on voltage sense bit 0: 1 = Battery present always</td>
</tr>
</tbody>
</table>

Nickel Charging

The nickel parameters govern when the cells should be trickle charged, fast charged, topped off, and when the cells are full based on negative voltage drop or temperature rate of change.

LED Display Configuration

The PIC16HV785 supports a 2-LED charging state display. These LEDs can be configured to identify the seven unique charger states.

Look-up Tables

The look-up tables are grids of data that perform thermistor measurement linearization and PWM adjustment based on feedback measurements.
### Configuration Parameters (Cont.)

<table>
<thead>
<tr>
<th>Parameter</th>
<th># Bytes</th>
<th>Typical Value</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode2</td>
<td>1</td>
<td>00100000b</td>
<td>binary</td>
<td>Configuration Register: bit 7: 1 = Disable auto-offset calibration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 6: 1 = Enable clock output on BATID pin after Reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 5: 1 = Use constant temperature from EEPROM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 4: 1 = NiMh algorithm 0 = Li Ion algorithm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 3: Unused</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 2: 1 = Disable voltage recharge trigger (VRVrech)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 1: 1 = Disable voltage cutoff in regulator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 0: 1 = Disable PWM auto-shutdown</td>
</tr>
<tr>
<td>OscTrim</td>
<td>1</td>
<td>0</td>
<td>integer</td>
<td>Oscillator trim calibration value.</td>
</tr>
<tr>
<td>PWMFreq</td>
<td>1</td>
<td>15</td>
<td>integer</td>
<td>LUT value which determines the PWM frequency.</td>
</tr>
<tr>
<td>PatternID</td>
<td>2</td>
<td>0x102</td>
<td>integer</td>
<td>ID for parameter set.</td>
</tr>
<tr>
<td>SHUNT</td>
<td>1</td>
<td>100</td>
<td>mOhms</td>
<td>Shunt resistor value.</td>
</tr>
<tr>
<td>SeriesCells</td>
<td>1</td>
<td>4</td>
<td>integer</td>
<td>Number of series connected cells in the battery pack.</td>
</tr>
<tr>
<td>Tdefault</td>
<td>1</td>
<td>112</td>
<td>code</td>
<td>Default temperature when using constant temperature in EEPROM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(°C * 10 + 200)/4.</td>
</tr>
<tr>
<td>TempCF</td>
<td>2</td>
<td>8192</td>
<td>integer</td>
<td>Temperature calibration value.</td>
</tr>
<tr>
<td>TimerEOCRecheck</td>
<td>1</td>
<td>20</td>
<td>.25 sec.</td>
<td>Recheck timer for End-of-Charge condition.</td>
</tr>
<tr>
<td>TimerStChng</td>
<td>1</td>
<td>20</td>
<td>.25 sec.</td>
<td>Recheck timer for state change.</td>
</tr>
<tr>
<td>VoltageCF</td>
<td>2</td>
<td>5121</td>
<td>integer</td>
<td>Voltage calibration value.</td>
</tr>
</tbody>
</table>

### Li Ion Charging Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th># Bytes</th>
<th>Typical Value</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRCurrent</td>
<td>2</td>
<td>800</td>
<td>mA</td>
<td>Charging current during current regulation.</td>
</tr>
<tr>
<td>CRTimeMax</td>
<td>1</td>
<td>0</td>
<td>4 min.</td>
<td>Current regulation time limit.</td>
</tr>
<tr>
<td>CRVTarg</td>
<td>2</td>
<td>4200</td>
<td>mV</td>
<td>Target cell voltage in current regulation. This is set to the fully charged</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>voltage of one cell, typically, as specified by the cell manufacturer.</td>
</tr>
<tr>
<td>PCCurrent</td>
<td>2</td>
<td>200</td>
<td>mA</td>
<td>Charging current during precharge.</td>
</tr>
<tr>
<td>PCTempThresh</td>
<td>1</td>
<td>175</td>
<td>code</td>
<td>Maximum temperature required to enable charging with precharge conditions</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(°C * 10 + 200)/4.</td>
</tr>
<tr>
<td>PCTimeMax</td>
<td>1</td>
<td>60</td>
<td>4 min.</td>
<td>Duration of precharge.</td>
</tr>
<tr>
<td>PCVolt</td>
<td>2</td>
<td>3000</td>
<td>mV</td>
<td>Cell voltage under which precharge occurs.</td>
</tr>
<tr>
<td>StartTMax</td>
<td>1</td>
<td>code</td>
<td></td>
<td>Maximum temperature under which charging may begin (°C * 10 + 200)/4.</td>
</tr>
<tr>
<td>TempMax</td>
<td>1</td>
<td>200</td>
<td>code</td>
<td>Maximum temperature under which charging is allowed to continue (°C * 10 +</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>200)/4.</td>
</tr>
<tr>
<td>TempMin</td>
<td>1</td>
<td>50</td>
<td>code</td>
<td>Minimum temperature over which charging is allowed to continue or start</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(°C * 10 + 200)/4.</td>
</tr>
<tr>
<td>Vmax</td>
<td>2</td>
<td>4300</td>
<td>mV</td>
<td>Voltage which triggers Charge Suspend mode when exceeded.</td>
</tr>
</tbody>
</table>
### TABLE 2: PIC16HV785 LITHIUM/NICKEL CONFIGURATION PARAMETERS (CONTINUED)

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th># Bytes</th>
<th>Typical Value</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VRIMin</td>
<td>2</td>
<td>50</td>
<td>mA</td>
<td>Voltage regulation fully charged current. This is the value of the taper current which will determine that the battery is fully charged.</td>
</tr>
<tr>
<td>VRTTimeMax</td>
<td>1</td>
<td>90</td>
<td>4 min.</td>
<td>Voltage regulation time limit.</td>
</tr>
<tr>
<td>VRVrech</td>
<td>2</td>
<td>4100</td>
<td>mV</td>
<td>Voltage regulation recharge cell voltage. Charger will begin recharging if cell voltage of pack falls below SeriesCells * VRVrech.</td>
</tr>
<tr>
<td>Vsafty</td>
<td>2</td>
<td>4350</td>
<td>mV</td>
<td>Charger will shut down immediately when voltage exceeds Vsafty.</td>
</tr>
</tbody>
</table>

#### Li Ion Charging Parameters (Cont.)

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th># Bytes</th>
<th>Typical Value</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChargeVolt</td>
<td>2</td>
<td>2000</td>
<td>mV</td>
<td>Charge voltage.</td>
</tr>
<tr>
<td>DischVolt</td>
<td>2</td>
<td>1000</td>
<td>mV</td>
<td>Discharge target voltage.</td>
</tr>
<tr>
<td>FCCurr</td>
<td>2</td>
<td>2000</td>
<td>mA</td>
<td>Fast charge current.</td>
</tr>
<tr>
<td>FCTimeMax</td>
<td>1</td>
<td>0</td>
<td>4 min.</td>
<td>Fast charge time limit.</td>
</tr>
<tr>
<td>FCTimeMin</td>
<td>1</td>
<td>0</td>
<td>4 min.</td>
<td>Fast charge time minimum.</td>
</tr>
<tr>
<td>MaxTmp</td>
<td>1</td>
<td>175</td>
<td>code</td>
<td>Maximum temperature for charging (°C * 10 + 200)/4.</td>
</tr>
<tr>
<td>MaxVolt</td>
<td>2</td>
<td>1800</td>
<td>mV</td>
<td>Maximum voltage for charging.</td>
</tr>
</tbody>
</table>

#### Nickel Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th># Bytes</th>
<th>Typical Value</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChargeVolt</td>
<td>2</td>
<td>2000</td>
<td>mV</td>
<td>Charge voltage.</td>
</tr>
<tr>
<td>DischVolt</td>
<td>2</td>
<td>1000</td>
<td>mV</td>
<td>Discharge target voltage.</td>
</tr>
<tr>
<td>FCCurr</td>
<td>2</td>
<td>2000</td>
<td>mA</td>
<td>Fast charge current.</td>
</tr>
<tr>
<td>FCTimeMax</td>
<td>1</td>
<td>0</td>
<td>4 min.</td>
<td>Fast charge time limit.</td>
</tr>
<tr>
<td>FCTimeMin</td>
<td>1</td>
<td>0</td>
<td>4 min.</td>
<td>Fast charge time minimum.</td>
</tr>
<tr>
<td>MaxTmp</td>
<td>1</td>
<td>175</td>
<td>code</td>
<td>Maximum temperature for charging (°C * 10 + 200)/4.</td>
</tr>
<tr>
<td>MaxVolt</td>
<td>2</td>
<td>1800</td>
<td>mV</td>
<td>Maximum voltage for charging.</td>
</tr>
</tbody>
</table>

#### Mode3

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th># Bytes</th>
<th>Typical Value</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode3</td>
<td>1</td>
<td>00111010b</td>
<td>binary</td>
<td>Configuration Register:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 7:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = Enable external discharge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 6:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = Enable external trickle circuit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 5:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = Bypass Trickle Charge state</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 4:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = Enable Top Off state</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 3:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = Enable Trickle Charge if voltage drops below recharge voltage</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 2:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unused</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 1:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = Enable dT/dt EOC method</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 0:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = Enable -dV EOC method</td>
</tr>
</tbody>
</table>

#### Other Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th># Bytes</th>
<th>Typical Value</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCCurrent</td>
<td>2</td>
<td>100</td>
<td>mA</td>
<td>Precharge current.</td>
</tr>
<tr>
<td>PCTempHi</td>
<td>1</td>
<td>160</td>
<td>code</td>
<td>High temperature limit over which the Precharge state is entered (°C * 10 + 200)/4.</td>
</tr>
<tr>
<td>PCTempLo</td>
<td>1</td>
<td>75</td>
<td>code</td>
<td>Low temperature limit under which the Precharge state is entered (°C * 10 + 200)/4.</td>
</tr>
<tr>
<td>PCTimeMax</td>
<td>1</td>
<td>0</td>
<td>4 min.</td>
<td>Precharge time limit.</td>
</tr>
<tr>
<td>PCTimeMin</td>
<td>1</td>
<td>0</td>
<td>4 min.</td>
<td>Precharge time minimum.</td>
</tr>
<tr>
<td>PCVolt</td>
<td>2</td>
<td>800</td>
<td>mV</td>
<td>Voltage threshold under which Precharge state occurs.</td>
</tr>
<tr>
<td>ShutdownVolt</td>
<td>2</td>
<td>1900</td>
<td>mV</td>
<td>Voltage which causes immediate shutdown when exceeded.</td>
</tr>
<tr>
<td>TPCurr</td>
<td>2</td>
<td>200</td>
<td>mA</td>
<td>Top Off state current.</td>
</tr>
<tr>
<td>TPTimeMax</td>
<td>1</td>
<td>11</td>
<td>4 min.</td>
<td>Top Off state time limit.</td>
</tr>
<tr>
<td>TRCurr</td>
<td>2</td>
<td>80</td>
<td>mA</td>
<td>Trickle state current.</td>
</tr>
<tr>
<td>TRTTimeMax</td>
<td>1</td>
<td>150</td>
<td>4 min.</td>
<td>Trickle state time limit.</td>
</tr>
<tr>
<td>VoltCal</td>
<td>2</td>
<td>2544</td>
<td>integer</td>
<td>Voltage calibration value.</td>
</tr>
<tr>
<td>Vrech</td>
<td>2</td>
<td>1300</td>
<td>mV</td>
<td>Voltage threshold under which Trickle Charge state is re-entered.</td>
</tr>
<tr>
<td>dTDelta</td>
<td>1</td>
<td>10</td>
<td>0.1°C</td>
<td>Change in temperature for dT/dt EOC condition.</td>
</tr>
<tr>
<td>dTTime</td>
<td>1</td>
<td>120</td>
<td>0.5 sec.</td>
<td>Change in time for dT/dt EOC condition.</td>
</tr>
<tr>
<td>dVDetect</td>
<td>2</td>
<td>10</td>
<td>mV</td>
<td>Negative change in voltage for -dV EOC condition.</td>
</tr>
<tr>
<td>Parameter Name</td>
<td># Bytes</td>
<td>Typical Value</td>
<td>Units</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
<td>---------------</td>
<td>-------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td><strong>LUT Parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWMAdjust1</td>
<td>1</td>
<td>12</td>
<td>integer</td>
<td>PWM adjustment for regulation control.</td>
</tr>
<tr>
<td>PWMAdjust2</td>
<td>1</td>
<td>10</td>
<td>integer</td>
<td>PWM adjustment for regulation control.</td>
</tr>
<tr>
<td>PWMAdjust3</td>
<td>1</td>
<td>5</td>
<td>integer</td>
<td>PWM adjustment for regulation control.</td>
</tr>
<tr>
<td>PWMAdjust4</td>
<td>1</td>
<td>1</td>
<td>integer</td>
<td>PWM adjustment for regulation control.</td>
</tr>
<tr>
<td>Vhh</td>
<td>1</td>
<td>19</td>
<td>mV</td>
<td>Voltage PWM adjustment zone limit.</td>
</tr>
<tr>
<td>Vh</td>
<td>1</td>
<td>6</td>
<td>mV</td>
<td>Voltage PWM adjustment zone limit.</td>
</tr>
<tr>
<td>VI</td>
<td>1</td>
<td>6</td>
<td>mV</td>
<td>Voltage PWM adjustment zone limit.</td>
</tr>
<tr>
<td>VII</td>
<td>1</td>
<td>44</td>
<td>mV</td>
<td>Voltage PWM adjustment zone limit.</td>
</tr>
<tr>
<td>ChI</td>
<td>1</td>
<td>5</td>
<td>mA</td>
<td>Current PWM adjustment zone limit.</td>
</tr>
<tr>
<td>T_LUT_N</td>
<td>1</td>
<td>8</td>
<td>integer</td>
<td>Number of temperature linearization LUT entries.</td>
</tr>
<tr>
<td>T_LUT_T_0</td>
<td>1</td>
<td>38</td>
<td>integer</td>
<td>Temperature A/D reading axis point.</td>
</tr>
<tr>
<td>T_LUT_T_1</td>
<td>1</td>
<td>48</td>
<td>integer</td>
<td>Temperature A/D reading axis point.</td>
</tr>
<tr>
<td>T_LUT_T_2</td>
<td>1</td>
<td>61</td>
<td>integer</td>
<td>Temperature A/D reading axis point.</td>
</tr>
<tr>
<td>T_LUT_T_3</td>
<td>1</td>
<td>79</td>
<td>integer</td>
<td>Temperature A/D reading axis point.</td>
</tr>
<tr>
<td>T_LUT_T_4</td>
<td>1</td>
<td>105</td>
<td>integer</td>
<td>Temperature A/D reading axis point.</td>
</tr>
<tr>
<td>T_LUT_T_5</td>
<td>1</td>
<td>183</td>
<td>integer</td>
<td>Temperature A/D reading axis point.</td>
</tr>
<tr>
<td>T_LUT_T_6</td>
<td>1</td>
<td>207</td>
<td>integer</td>
<td>Temperature A/D reading axis point.</td>
</tr>
<tr>
<td>T_LUT_M_0</td>
<td>2</td>
<td>-23362</td>
<td>integer</td>
<td>Temperature linearization slope LUT entry.</td>
</tr>
<tr>
<td>T_LUT_B_0</td>
<td>2</td>
<td>1418</td>
<td>integer</td>
<td>Temperature linearization Y-intercept LUT entry.</td>
</tr>
<tr>
<td>T_LUT_M_1</td>
<td>2</td>
<td>-19864</td>
<td>integer</td>
<td>Temperature linearization slope LUT entry.</td>
</tr>
<tr>
<td>T_LUT_B_1</td>
<td>2</td>
<td>1352</td>
<td>integer</td>
<td>Temperature linearization Y-intercept LUT entry.</td>
</tr>
<tr>
<td>T_LUT_M_2</td>
<td>2</td>
<td>-15709</td>
<td>integer</td>
<td>Temperature linearization slope LUT entry.</td>
</tr>
<tr>
<td>T_LUT_B_2</td>
<td>2</td>
<td>1255</td>
<td>integer</td>
<td>Temperature linearization Y-intercept LUT entry.</td>
</tr>
<tr>
<td>T_LUT_M_3</td>
<td>2</td>
<td>-12572</td>
<td>integer</td>
<td>Temperature linearization slope LUT entry.</td>
</tr>
<tr>
<td>T_LUT_B_3</td>
<td>2</td>
<td>1162</td>
<td>integer</td>
<td>Temperature linearization Y-intercept LUT entry.</td>
</tr>
<tr>
<td>T_LUT_M_4</td>
<td>2</td>
<td>-10154</td>
<td>integer</td>
<td>Temperature linearization slope LUT entry.</td>
</tr>
<tr>
<td>T_LUT_B_4</td>
<td>2</td>
<td>1071</td>
<td>integer</td>
<td>Temperature linearization Y-intercept LUT entry.</td>
</tr>
<tr>
<td>T_LUT_M_5</td>
<td>2</td>
<td>-8631</td>
<td>integer</td>
<td>Temperature linearization slope LUT entry.</td>
</tr>
<tr>
<td>T_LUT_B_5</td>
<td>2</td>
<td>990</td>
<td>integer</td>
<td>Temperature linearization Y-intercept LUT entry.</td>
</tr>
<tr>
<td>T_LUT_M_6</td>
<td>2</td>
<td>-10154</td>
<td>integer</td>
<td>Temperature linearization slope LUT entry.</td>
</tr>
<tr>
<td>T_LUT_B_6</td>
<td>2</td>
<td>1127</td>
<td>integer</td>
<td>Temperature linearization Y-intercept LUT entry.</td>
</tr>
<tr>
<td>T_LUT_M_7</td>
<td>2</td>
<td>-12875</td>
<td>integer</td>
<td>Temperature linearization slope LUT entry.</td>
</tr>
<tr>
<td>T_LUT_B_7</td>
<td>2</td>
<td>1402</td>
<td>integer</td>
<td>Temperature linearization Y-intercept LUT entry.</td>
</tr>
<tr>
<td><strong>LED Parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LED1State1</td>
<td>1</td>
<td>00000000b</td>
<td>binary</td>
<td>LED1 display during state 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Lithium: Charge Pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Nickel: Charge Pending</td>
</tr>
<tr>
<td>LED1State2</td>
<td>1</td>
<td>00000000b</td>
<td>binary</td>
<td>LED1 display during state 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Lithium: Charge Qualification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Nickel: Charge Qualification</td>
</tr>
<tr>
<td>LED1State3</td>
<td>1</td>
<td>00000000b</td>
<td>binary</td>
<td>LED1 display during state 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Lithium: Precharge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Nickel: Precharge</td>
</tr>
</tbody>
</table>
### LED Parameters (Cont.)

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th># Bytes</th>
<th>Typical Value</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
</table>
| LED1State4     | 1       | 00000000b     | binary| LED1 display during state 4  
Lithium: Current Regulation 
Nickel: Fast Charge |
| LED1State5     | 1       | 00000000b     | binary| LED1 display during state 5  
Lithium: Voltage Regulation 
Nickel: Top Off |
| LED1State6     | 1       | 00000000b     | binary| LED1 display during state 6  
Lithium: Charge Cycle Complete 
Nickel: Charge Cycle Complete |
| LED1State7     | 1       | 00000000b     | binary| LED1 display during state 7  
Lithium: Charge Suspended 
Nickel: Unused |
| LED1State8     | 1       | 00000000b     | binary| LED1 display during state 8  
Lithium: Unused 
Nickel: Trickle Charge |
| LED2State1     | 1       | 00000000b     | binary| LED1 display during state 1  
Lithium: Charge Pending 
Nickel: Charge Pending |
| LED2State2     | 1       | 00000000b     | binary| LED1 display during state 2  
Lithium: Charge Qualification 
Nickel: Charge Qualification |
| LED2State3     | 1       | 00000000b     | binary| LED1 display during state 3  
Lithium: Precharge 
Nickel: Precharge |
| LED2State4     | 1       | 00000000b     | binary| LED1 display during state 4  
Lithium: Current Regulation 
Nickel: Fast Charge |
| LED2State5     | 1       | 00000000b     | binary| LED1 display during state 5  
Lithium: Voltage Regulation 
Nickel: Top Off |
| LED2State6     | 1       | 00000000b     | binary| LED1 display during state 6  
Lithium: Charge Cycle Complete 
Nickel: Charge Cycle Complete |
| LED2State7     | 1       | 00000000b     | binary| LED1 display during state 7  
Lithium: Charge Suspend 
Nickel: Unused |
| LED2State8     | 1       | 00000000b     | binary| LED1 display during state 8  
Lithium: Unused 
Nickel: Trickle Charge |

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lithium</td>
<td>current Regulation</td>
</tr>
<tr>
<td>nickel</td>
<td>fast charge</td>
</tr>
<tr>
<td>lithium</td>
<td>voltage regulation</td>
</tr>
<tr>
<td>nickel</td>
<td>top off</td>
</tr>
<tr>
<td>lithium</td>
<td>charge cycle complete</td>
</tr>
<tr>
<td>nickel</td>
<td>charge cycle complete</td>
</tr>
<tr>
<td>lithium</td>
<td>charge suspended</td>
</tr>
<tr>
<td>nickel</td>
<td>unused</td>
</tr>
<tr>
<td>lithium</td>
<td>unused</td>
</tr>
<tr>
<td>nickel</td>
<td>trickle charge</td>
</tr>
</tbody>
</table>

**TABLE 2: PIC16HV785 LITHIUM/NICKEL CONFIGURATION PARAMETERS (CONTINUED)**
FIRMWARE SUMMARY

Initialization

During initialization, the firmware will define constants, allocate resources and configure registers. This includes mapping the GPIO, setting up the timers, setting the initial PWM frequency, outputting the optional BATID frequency check signal, configuring the LED pins and configuring the HVOUT pin.

Once the resources are configured, RAM is cleared and the main loop is entered.

Four of the initialization functions are described below:

1. Programming the initial PWM frequency.
2. Configuring the BATID pin as an analog input and output of the clock frequency.
3. Configuring the LED2 pin as LED or communication.
4. Configuring the HVOUT pin for one of its multiple functions.

The initial PWM frequency is configured by writing to PWMFreq, where the following table determines the PWM frequency as a function of the bits in the PWMP register.

```
<table>
<thead>
<tr>
<th>PWMP&lt;6:0&gt;</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>F: 8,000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>8000</td>
<td>4000</td>
<td>2000</td>
<td>1000</td>
</tr>
<tr>
<td>1</td>
<td>4000</td>
<td>2000</td>
<td>1000</td>
<td>500</td>
</tr>
<tr>
<td>2</td>
<td>2667</td>
<td>1333</td>
<td>667</td>
<td>333</td>
</tr>
<tr>
<td>3</td>
<td>2000</td>
<td>1000</td>
<td>500</td>
<td>250</td>
</tr>
<tr>
<td>4</td>
<td>1600</td>
<td>800</td>
<td>400</td>
<td>200</td>
</tr>
<tr>
<td>5</td>
<td>1333</td>
<td>667</td>
<td>333</td>
<td>167</td>
</tr>
<tr>
<td>6</td>
<td>1143</td>
<td>571</td>
<td>286</td>
<td>143</td>
</tr>
<tr>
<td>7</td>
<td>1000</td>
<td>500</td>
<td>250</td>
<td>125</td>
</tr>
<tr>
<td>8</td>
<td>889</td>
<td>444</td>
<td>222</td>
<td>111</td>
</tr>
<tr>
<td>9</td>
<td>800</td>
<td>400</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>727</td>
<td>364</td>
<td>182</td>
<td>91</td>
</tr>
<tr>
<td>11</td>
<td>667</td>
<td>333</td>
<td>167</td>
<td>83</td>
</tr>
<tr>
<td>12</td>
<td>615</td>
<td>308</td>
<td>154</td>
<td>77</td>
</tr>
<tr>
<td>13</td>
<td>571</td>
<td>286</td>
<td>143</td>
<td>71</td>
</tr>
<tr>
<td>14</td>
<td>533</td>
<td>267</td>
<td>133</td>
<td>67</td>
</tr>
<tr>
<td>15</td>
<td>500</td>
<td>250</td>
<td>125</td>
<td>63</td>
</tr>
<tr>
<td>16</td>
<td>471</td>
<td>235</td>
<td>118</td>
<td>59</td>
</tr>
<tr>
<td>17</td>
<td>444</td>
<td>222</td>
<td>111</td>
<td>56</td>
</tr>
<tr>
<td>18</td>
<td>421</td>
<td>211</td>
<td>105</td>
<td>53</td>
</tr>
<tr>
<td>19</td>
<td>400</td>
<td>200</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>20</td>
<td>381</td>
<td>190</td>
<td>95</td>
<td>48</td>
</tr>
<tr>
<td>21</td>
<td>364</td>
<td>182</td>
<td>91</td>
<td>45</td>
</tr>
<tr>
<td>22</td>
<td>348</td>
<td>174</td>
<td>87</td>
<td>43</td>
</tr>
<tr>
<td>23</td>
<td>333</td>
<td>167</td>
<td>83</td>
<td>42</td>
</tr>
<tr>
<td>24</td>
<td>320</td>
<td>160</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>25</td>
<td>308</td>
<td>154</td>
<td>77</td>
<td>38</td>
</tr>
<tr>
<td>26</td>
<td>296</td>
<td>148</td>
<td>74</td>
<td>37</td>
</tr>
<tr>
<td>27</td>
<td>286</td>
<td>143</td>
<td>71</td>
<td>36</td>
</tr>
<tr>
<td>28</td>
<td>276</td>
<td>138</td>
<td>69</td>
<td>34</td>
</tr>
<tr>
<td>29</td>
<td>267</td>
<td>133</td>
<td>67</td>
<td>33</td>
</tr>
<tr>
<td>30</td>
<td>258</td>
<td>129</td>
<td>65</td>
<td>32</td>
</tr>
<tr>
<td>31</td>
<td>250</td>
<td>125</td>
<td>63</td>
<td>31</td>
</tr>
</tbody>
</table>
```
The BATID pin is used to determine if a battery is present by measuring the voltage on the pin and comparing it to the proper EEPROM parameters. Alternatively, after a Reset and during initialization, this pin can be configured by the Mode2 parameter to output a single burst of 256 clocks in order to determine the frequency of the internal oscillator.

The LED2 pin is configured as either an LED driver or as the communication pin. See the “Communication” section for more information.

The HVOUT pin is a general purpose, open-drain output that can be configured as one of three mutually exclusive functions, as chosen by the Mode and Mode3 parameters.

Mode<6> = 1: Charge Current Switch
Used as an indication of charge current flowing.

HVOUT = 1: Charge current flowing
HVOUT = 0: No charge current flowing

Mode3<7> = 1: Enable External Discharge Circuit
An option in Nickel charging is to provide a load to discharge a battery immediately prior to charging. This can help in detection of End-of-Charge conditions.

HVOUT = 1: Default
HVOUT = 0: Discharge circuit enabled

Mode3<6> = 1: Enable External Trickle Charge Circuit
If the charger cannot satisfactorily regulate a low-level trickle charge current, an external trickle charge circuit may be used. The HVOUT pin, in this case, can control the trickle charge circuit when the Trickle Charge state is entered.

HVOUT = 1: Default
HVOUT = 0: Trickle charge circuit enabled

Main Loop

The main loop cycles through the following functions:

- Performs A/D measurements
- Checks measurements against triggers and determines the charge state
- Adjusts the PWM to regulate current
- Operates the LEDs
- Maintains the timers
- Performs EEPROM reads and writes
- Performs communication transactions

The actual subroutines are:

- adc_svc: Receive the finished A/D conversions, process the data with calibration constants, etc., and store in RAM
- adc_start: Start a new set of conversions to be completed for the next cycle
- check_triggers: Compare the A/D results with parameters to determine what state the charging should be in
- chg_state_svc: Put the charger into the proper state based on A/D results
- regulate: Adjust the PWM to regulate current based on charge state and feedback measurements
- led_svc: Operate two LEDs to display the charge state
- timer_svc: Maintain the firmware timers
- ee_write_buf: Background process to write the data block in the RAM buffer into EEPROM
- ccmd_svc: React to communication commands
- status_build: Build the status byte communication register

Triggers and Charge States

Once data is received from the A/D, it is compared to the parameters using charge state formulas to determine the proper charge states, as explained above in the functional descriptions.

Regulating the PWM

The PWM duty cycle is adjusted by the firmware in response to the charge state and the feedback measurements. It is increased or decreased to keep the voltage and current as close to the charge requirements as possible without exceeding those requirements. The feedback measurements of voltage and current are compared to the required voltage and current of the particular charge state the device is in. The PWM is either kept the same, increased or decreased a little, or increased or decreased a lot as a function of the difference between the feedback measurements and the requirements.

As Table 4 shows, if the voltage feedback is no greater than $V_h$ more than the requirement, and no less than $V_l$ lower than the requirement, the PWM is unchanged. If the feedback voltage exceeds the required voltage by more than $V_l$, the PWM is decreased by PWMAjust4, etc.
Table 4 shows the PWM adjustment factors as a function of current difference and voltage difference when comparing feedback to requirements:

<table>
<thead>
<tr>
<th>Voltage Zones</th>
<th>Current Zones</th>
<th>Current Zones</th>
<th>Current Zones</th>
<th>Current Zones</th>
<th>Current Zones</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; Vhh</td>
<td>-PWMAdjust1</td>
<td>-PWMAdjust1</td>
<td>-PWMAdjust1</td>
<td>-PWMAdjust1</td>
<td>-PWMAdjust1</td>
</tr>
<tr>
<td>&gt; Vh</td>
<td>-PWMAdjust4</td>
<td>-PWMAdjust4</td>
<td>-PWMAdjust4</td>
<td>-PWMAdjust4</td>
<td>-PWMAdjust2</td>
</tr>
<tr>
<td>Vh to -VI</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-PWMAdjust4</td>
<td>-PWMAdjust2</td>
</tr>
<tr>
<td>&lt; -VI</td>
<td>+PWMAdjust4</td>
<td>+PWMAdjust4</td>
<td>0</td>
<td>-PWMAdjust4</td>
<td>-PWMAdjust2</td>
</tr>
<tr>
<td>&lt; -VII</td>
<td>+PWMAdjust3</td>
<td>+PWMAdjust4</td>
<td>0</td>
<td>-PWMAdjust4</td>
<td>-PWMAdjust2</td>
</tr>
</tbody>
</table>

**LED Control**

Two LED Configuration registers (one for each LED) determine how the LEDs are displayed when controlling on/off, flashing, flash counts and on/off times.

**TABLE 5: LED CONFIGURATION Registers**

<table>
<thead>
<tr>
<th>Mode&lt;7,3&gt;</th>
<th>Mode Description</th>
<th>N&lt;6:4&gt;</th>
<th>F&lt;2:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>Off</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>0 1</td>
<td>Flash N + 1 Times, Pause, Repeat</td>
<td>Flash Count = N + 1</td>
<td>On Time = Off Time = F + 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pause Time = (F + 1) * 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Max = 3</td>
</tr>
<tr>
<td>1 0</td>
<td>On</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>1 1</td>
<td>Flash Continuously</td>
<td>On Time = N + 1</td>
<td>Off Time = F + 1</td>
</tr>
</tbody>
</table>

EEPROM parameters are used to define the settings above for each charge state. The **LED1State1-8** and **LED2State1-8** parameters are used to program the above configuration parameters based on what state the charger is in.

**A/D Starting and Processing**

The A/D operations consist of starting the A/D readings on up to 5 channels, retrieving the data and calibrating the data.

To start the readings, the firmware programs the A/D Control registers (see the “PIC16F785 Data Sheet” (DS41249)) to perform the required measurements. Up to five channels are used for the charger function. They include the following:

- Reference Voltage
- Current
- Voltage
- Temperature
- BATID

When conversions are complete, flags are set so the firmware can perform the calibration and processing. For filtering purposes, the average of 16 consecutive readings are used for valid data.

**REFERENCE VOLTAGE**

The band gap reference voltage (VR) is calibrated or translated from the raw A/D measurement (A/DRAW) as follows:

$$VR = A/D_{RAW} \times 16384/\text{BandgapCF}$$

**BandgapCF** is typically around 248 since:

$$VR/VDD = A/D_{RAW}(\text{FULLSCALE}) = 1212/5000 \times 1023 = 248$$

Since the reference voltage is fixed, this calibration factor is used to compensate for a variance in VDD. It is used to correct any readings that use VDD as a reference.
CURRENT
The current reading is calibrated or translated from the raw A/D measurement (A/DRaw) as follows:

**EQUATION 6:**

When referenced to VR:
Current = A/DRaw * CurrentCF/65536

When referenced to VDD:
Current = (A/DRaw * VR/16384) * CurrentCF/65535

The CurrentCF is determined by examining Equation 6 at full scale, for example:

**EQUATION 7:**

Current(full scale) = VREF/AMPgain/SHUNT = 5000/19.6/0.100 = 2551 mA
2551 = 1023 * CurrentCF
CurrentCF = 2.494

Representing the decimal fraction as a ratio using a power of 2:

**EQUATION 8:**

CurrentCF Base = 1024
CurrentCF = 2553

VOLTAGE
The voltage reading is calibrated or translated from the raw A/D measurement (A/DRaw) as follows:

**EQUATION 9:**

When referenced to VR:
Voltage = A/DRaw * VoltageCF/1024

When referenced to VDD:
Voltage = (A/DRaw * VR/16384) * VoltageCF/1024

Where VoltageCF is determined as follows:
Voltage = A/DRaw * VoltageCF
A/DRaw = (Voltage * Cells) * R/VREF * 1023
Where:
R = Resistor Divider Ratio
VREF = 5000 mV
This means:
Voltage = VoltageCF * Voltage * Cells * R/VREF * 1023
or
VoltageCF = VREF/(Cells * R * 1023)
and using integer arithmetic:
VoltageCF = VoltageCF' * 1024
So that:
Voltage = VoltageCF' * A/DRaw/1024

Table 6 shows the typical VoltageCF values for the PS2070 evaluation module with a different number of cells and different voltage dividers selected:

<table>
<thead>
<tr>
<th>Cells</th>
<th>R#</th>
<th>R1</th>
<th>R2</th>
<th>R Ratio</th>
<th>VoltageCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.232</td>
<td>10.0</td>
<td>0.9773</td>
<td>5121</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>10.500</td>
<td>10.0</td>
<td>0.4878</td>
<td>5130</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>20.500</td>
<td>10.0</td>
<td>0.3279</td>
<td>5088</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>30.900</td>
<td>10.0</td>
<td>0.2445</td>
<td>5117</td>
</tr>
</tbody>
</table>

BATID
The BATID pin is measured in raw A/D units, scaled to 0 to 255, and compared to EEPROM parameters that are in raw A/D units, scaled to 0 to 255, so no calibration is performed.

TEMPERATURE
The current reading is calibrated or translated from the raw A/D measurement (A/DRaw) as follows:

**EQUATION 10:**

Temperature = A/DRaw * TempCF/8192

Where temperature is in the internal units of:
(°C + 20) * 10

TempCF is typically 8192 and is set by comparing a known temperature to the measured temperature.

The temperature response of the thermistor is then subjected to linearization by a look-up table as described in the next section.

**Thermistor Linearization**

The thermistor reading is subjected to piecewise linear interpolation using a look-up table of line equations. Since the variance of voltage with temperature for the thermistor is not always along the same line (same slope and intercept), multiple line equations must be used for interpolation depending on where the measurement is. The look-up table was developed by rating raw A/D values; that is why TempCF can typically be set to 1.
The look-up table is a series of slopes and Y-intercepts corresponding to regions of temperature A/D readings. \( T_{\text{LUT}}\) represents the number of entries in the table, in this case eight entries.

### TABLE 7: THERMISTOR LINEARIZATION

<table>
<thead>
<tr>
<th>A/D Reading</th>
<th>Slope</th>
<th>Y-intercept</th>
</tr>
</thead>
<tbody>
<tr>
<td>( &lt; T_{\text{LUT}} T_0 )</td>
<td>( T_{\text{LUT}} M_0 )</td>
<td>( T_{\text{LUT}} B_0 )</td>
</tr>
<tr>
<td>( &lt; T_{\text{LUT}} T_1 )</td>
<td>( T_{\text{LUT}} M_1 )</td>
<td>( T_{\text{LUT}} B_1 )</td>
</tr>
<tr>
<td>( &lt; T_{\text{LUT}} T_2 )</td>
<td>( T_{\text{LUT}} M_2 )</td>
<td>( T_{\text{LUT}} B_2 )</td>
</tr>
<tr>
<td>( &lt; T_{\text{LUT}} T_3 )</td>
<td>( T_{\text{LUT}} M_3 )</td>
<td>( T_{\text{LUT}} B_3 )</td>
</tr>
<tr>
<td>( &lt; T_{\text{LUT}} T_4 )</td>
<td>( T_{\text{LUT}} M_4 )</td>
<td>( T_{\text{LUT}} B_4 )</td>
</tr>
<tr>
<td>( &lt; T_{\text{LUT}} T_5 )</td>
<td>( T_{\text{LUT}} M_5 )</td>
<td>( T_{\text{LUT}} B_5 )</td>
</tr>
<tr>
<td>( &lt; T_{\text{LUT}} T_6 )</td>
<td>( T_{\text{LUT}} M_6 )</td>
<td>( T_{\text{LUT}} B_6 )</td>
</tr>
<tr>
<td>( &gt; T_{\text{LUT}} T_6 )</td>
<td>( T_{\text{LUT}} M_7 )</td>
<td>( T_{\text{LUT}} B_7 )</td>
</tr>
</tbody>
</table>

The typical values are:

### TABLE 8: A/D TEMPERATURE READINGS

<table>
<thead>
<tr>
<th>A/D Reading</th>
<th>Slope</th>
<th>Y-intercept</th>
</tr>
</thead>
<tbody>
<tr>
<td>( &lt; 38 )</td>
<td>-23362</td>
<td>1418</td>
</tr>
<tr>
<td>( &lt; 48 )</td>
<td>-19864</td>
<td>1352</td>
</tr>
<tr>
<td>( &lt; 61 )</td>
<td>-15709</td>
<td>1255</td>
</tr>
<tr>
<td>( &lt; 79 )</td>
<td>-12572</td>
<td>1162</td>
</tr>
<tr>
<td>( &lt; 105 )</td>
<td>-10206</td>
<td>1071</td>
</tr>
<tr>
<td>( &lt; 183 )</td>
<td>-8631</td>
<td>990</td>
</tr>
<tr>
<td>( &lt; 207 )</td>
<td>-10154</td>
<td>1127</td>
</tr>
<tr>
<td>( &gt; 207 )</td>
<td>-12875</td>
<td>1402</td>
</tr>
</tbody>
</table>

Communication

Communication for memory reads and writes, typically used for changing parameters, is performed using the LED2 I/O pin (pin 2). Pin 2 is configured during Reset initialization to either be the communication pin or an LED driver. If pin 2 is driven low during initialization, pin 2 will become the LED driver. If pin 2 is driven high during initialization, communication will be enabled and pin 2 will be the communication pin.

The communication protocol is the Single Pin Serial (SPS) protocol. SPS communication is an asynchronous return-to-one protocol. The signal requires an external pull-up resistor. The timing of the driven low pulses defines the communication. A Break cycle starts a command from the host to the PIC16HV785. The command is eight bits long. After this, eight data bits are either written to the PIC16HV785, or read from the PIC16HV785.

A Break cycle is defined by a low period of time equal to or greater than time \( t_{bs} \), then returned high for a time greater than or equal to \( t_{sr} \).

The data bits consist of three sections each:

1. Start: low for at least time \( t_{str} \).
2. Data: data high or low valid by time \( t_{dsuh/v} \) and held until time \( t_{dh/v} \).
3. Stop: high by time \( t_{ssuh/v} \) and held until time \( t_{cyc} \).

All transactions either read or write an 8-bit register. Each register has a 7-bit address, plus a read/write bit, for a total of 8 bits. Bit 7 is the read/write bit. When bit 7 is ‘1’, the register is written. When bit 7 is ‘0’, the register is read. Of the possible 128 addressable registers, only 10 are implemented.

A read transaction will receive a single byte of data. A write transaction can write multiple 8-bit data values to a register:

**READ:** BREAK, REG_ADDR, DATA

**WRITE:** BREAK, REG_ADDR, DATA, DATA, ... DATA

The communication protocol is the Single Pin Serial (SPS) protocol. SPS communication is an asynchronous return-to-one protocol. The signal requires an external pull-up resistor. The timing of the driven low pulses defines the communication. A Break cycle starts a command from the host to the PIC16HV785. The command is eight bits long. After this, eight data bits are either written to the PIC16HV785, or read from the PIC16HV785.

A Break cycle is defined by a low period of time equal to or greater than time \( t_{bs} \), then returned high for a time greater than or equal to \( t_{sr} \).

The data bits consist of three sections each:

1. Start: low for at least time \( t_{str} \).
2. Data: data high or low valid by time \( t_{dsuh/v} \) and held until time \( t_{dh/v} \).
3. Stop: high by time \( t_{ssuh/v} \) and held until time \( t_{cyc} \).

All transactions either read or write an 8-bit register. Each register has a 7-bit address, plus a read/write bit, for a total of 8 bits. Bit 7 is the read/write bit. When bit 7 is ‘1’, the register is written. When bit 7 is ‘0’, the register is read. Of the possible 128 addressable registers, only 10 are implemented.

A read transaction will receive a single byte of data. A write transaction can write multiple 8-bit data values to a register:

**READ:** BREAK, REG_ADDR, DATA

**WRITE:** BREAK, REG_ADDR, DATA, DATA, ... DATA
FIGURE 12: SINGLE PIN SERIAL TIMING

Break Timing
Break bit
Break Reset

Host to PIC16HV785
Start bit Data bit Stop bit

PIC16HV785 to Host
Start bit Data bit Stop bit

CMD and Data Protocol
Break LSB CMD Address MSB LSB Data to or from PIC16HV785

Communication Example
CMD Addr = 04 hex Data = 25 hex
Break 0 1 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0

Break bit Data bit Stop bit
Break bit Break Reset

Break bit Break Reset
### TABLE 9: REGISTER SUMMARY

<table>
<thead>
<tr>
<th>Name</th>
<th>ADDR</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM_ADDR</td>
<td>0x00</td>
<td>R/W</td>
<td>Indirect Memory Address</td>
</tr>
<tr>
<td>STATUS</td>
<td>0x01</td>
<td>R</td>
<td>Status</td>
</tr>
<tr>
<td>CONFIG</td>
<td>0x02</td>
<td>R/W</td>
<td>Configuration</td>
</tr>
<tr>
<td>CMND</td>
<td>0x03</td>
<td>R/W</td>
<td>Command</td>
</tr>
<tr>
<td>DATA_LO</td>
<td>0x04</td>
<td>R/W</td>
<td>Data</td>
</tr>
<tr>
<td>DATA_HI</td>
<td>0x05</td>
<td>R/W</td>
<td>Data</td>
</tr>
<tr>
<td>N/A</td>
<td>0x06</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>UNLOCK</td>
<td>0x07</td>
<td>W</td>
<td>Unlock Key = 0x96</td>
</tr>
<tr>
<td>MEM_ACCESS</td>
<td>0x08</td>
<td>R/W</td>
<td>Accesses Memory Indirectly through MEM_ADDR</td>
</tr>
<tr>
<td>MEM_ACCESS_IA</td>
<td>0x0C</td>
<td>R/W</td>
<td>Accesses Memory Indirectly through MEM_ADDR and Post-Increments Memory Address</td>
</tr>
</tbody>
</table>

### REGISTER DESCRIPTIONS

#### REGISTER 0: MEM_ADDR

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:0</td>
<td>MEM_ADDR</td>
<td>Indirect memory address used for reading and writing data</td>
</tr>
</tbody>
</table>

#### REGISTER 1: STATUS

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>EE_Busy</td>
<td>( \dfrac{}{\cap} ) = EEPROM write is in progress; busy</td>
</tr>
<tr>
<td>6</td>
<td>EE_Err</td>
<td>( \dfrac{1}{\cap} ) = Error encountered during last EEPROM write</td>
</tr>
<tr>
<td>5</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>REG_ACTIVE</td>
<td>( \dfrac{1}{\cap} ) = Regulation active</td>
</tr>
<tr>
<td>3</td>
<td>CHGCON</td>
<td>( \dfrac{1}{\cap} ) = Charge controller active</td>
</tr>
<tr>
<td>2</td>
<td>SIM_ACTIVE</td>
<td>( \dfrac{1}{\cap} ) = Data simulation active</td>
</tr>
<tr>
<td>1</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Unused</td>
<td></td>
</tr>
</tbody>
</table>

#### REGISTER 2: CONFIG

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:6</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>SUSPEND</td>
<td>( \dfrac{1}{\cap} ) = Suspend/skip all processing (used when writing EEPROM)</td>
</tr>
<tr>
<td>4</td>
<td>CHGCON_OFF</td>
<td>( \dfrac{1}{\cap} ) = Suspend charge controller</td>
</tr>
<tr>
<td>3:2</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MEMBANK_EE</td>
<td>( \dfrac{1}{\cap} ) = Indirect memory addressing refers to EEPROM</td>
</tr>
<tr>
<td>0</td>
<td>MEMBANK_23</td>
<td>( \dfrac{1}{\cap} ) = Indirect memory addressing refers to 2nd bank of RAM</td>
</tr>
</tbody>
</table>
## REGISTER 3: CMND

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>VERSION</td>
<td>1 = Load Data registers (Register 4 and Register 5) with firmware version number</td>
</tr>
<tr>
<td>6</td>
<td>PWM_SET</td>
<td>1 = Load control PWM with contents of Data registers</td>
</tr>
<tr>
<td>5</td>
<td>REG_ON</td>
<td>1 = Enable regulation module</td>
</tr>
<tr>
<td>4</td>
<td>EE_RQ</td>
<td>1 = Request EEPROM write of data block in RAM</td>
</tr>
<tr>
<td>3</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>RESET</td>
<td>1 = Reset firmware (branch to Reset vector from Idle loop)</td>
</tr>
<tr>
<td>1</td>
<td>FORCE_CHGSTATE</td>
<td>1 = Force branch to Charge Controller state</td>
</tr>
<tr>
<td>0</td>
<td>SIM_RQ</td>
<td>1 = Load simulation data previously written to RAM</td>
</tr>
</tbody>
</table>

## REGISTER 4: DATA_LO

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:0</td>
<td>DATA_LO</td>
<td>Generic data used in memory reads and writes (LSB)</td>
</tr>
</tbody>
</table>

## REGISTER 5: DATA_HI

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:0</td>
<td>DATA_HI</td>
<td>Generic data used in memory reads and writes (MSB)</td>
</tr>
</tbody>
</table>

## REGISTER 6: UNUSED

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:0</td>
<td>Unused</td>
<td></td>
</tr>
</tbody>
</table>

## REGISTER 7: UNLOCK

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:0</td>
<td>UNLOCK</td>
<td>Unlock code is written here</td>
</tr>
</tbody>
</table>

## REGISTER 8: MEM_ACCESS

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:0</td>
<td>MEM_ACCESS</td>
<td>Data written to Register 8 is actually sent to the memory address contained in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Register 0 and the bank indicated by Register 2 (bits&lt;1:0&gt;)</td>
</tr>
</tbody>
</table>

## REGISTER C: MEM_ACCESS_IA

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:0</td>
<td>MEM_ACCESS_IA</td>
<td>Data written to Register 8 is actually sent to the memory address contained in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Register 0 and the bank indicated by Register 2 (bits&lt;1:0&gt;); Register 0 will be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>post-incremented</td>
</tr>
</tbody>
</table>
Host Driven Operations

Host driven operations refer to a host communicating with the PIC16HV785 in order to read or write memory locations. This is typically done during programming, parameter changing, or troubleshooting. The four basic functions are EEPROM read, EEPROM write, RAM read and RAM write. The host will employ the Single Pin Serial protocol and the registers described in the “Register Descriptions” section to accomplish the functions.

RAM READ

There are three steps to the RAM read:
1. Select the bank: Set Communication Register 2 (bit 0 = 0); select bank 0/1 since bank 2/3 is not implemented.
2. Select the address: Set Communication Register 0 to the starting RAM address.
3. Read the data: Read the contents of the Memory Access register (Register 8 or Register C). When using Register C, the address will auto-increment, so step 3 can be repeated to receive more data.

RAM WRITE

There are three steps to the RAM write:
1. Select the bank: Set Communication Register 2 (bit 0 = 0); select bank 0/1 since bank 2/3 is not implemented.
2. Select the address: Set Communication Register 0 to the starting RAM address.
3. Write the data: Write the data to the Memory Access register (Register 8 or Register C). When using Register C, the address will auto-increment, so step 3 can be repeated to write more data.

EEPROM READ

There are three steps to the EEPROM read:
1. Select the bank: Set Communication Register 2 (bits<1:0> = 10); select bank = EEPROM.
2. Select the address: Set Communication Register 0 to the starting EEPROM address.
3. Read the data: Read the contents of the Memory Access register (Register 8 or Register C). When using Register C, the address will auto-increment, so step 3 can be repeated to receive more data.

EEPROM WRITE

The EEPROM write follows a more secure protocol in which a “control packet” of data is written to a RAM buffer first. The RAM buffer begins at address 0xA0. A control bit is then set to trigger the writing of the data in the control packet to EEPROM. The control packet takes the following form:

<table>
<thead>
<tr>
<th>Byte</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ADDR</td>
<td>Starting EEPROM Address to be Written</td>
</tr>
<tr>
<td>1</td>
<td>COUNT</td>
<td>Byte Count (N), Maximum = 29</td>
</tr>
<tr>
<td>2</td>
<td>DATA</td>
<td>Data[0]</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N + 1</td>
<td>DATA</td>
<td>Data[N – 1]</td>
</tr>
<tr>
<td>N + 2</td>
<td>CHKSUM</td>
<td>Checksum = Sum (byte[0]:byte[N + 1])</td>
</tr>
</tbody>
</table>

The total procedure is a five step process:
1. Suspend normal operation: Set Communication Register 2 = 0x20 (bit 5 = 1).
2. Check if the EEPROM is busy: Does Communication Register 1 (bit 7 = 1)?
3. If not busy, write the control block data to RAM, beginning at address 0xA0, using RAM write procedure.
4. When all data is written, trigger EEPROM write; set Communication Register 3 (bit 4 = 1).
5. Issue a firmware Reset: Set Communication Register 3 (bit 2 = 1).
Software License Agreement

The software supplied herewith by Microchip Technology Incorporated (the “Company”) is intended and supplied to you, the Company’s customer, for use solely and exclusively with products manufactured by the Company.

The software is owned by the Company and/or its supplier, and is protected under applicable copyright laws. All rights are reserved. Any use in violation of the foregoing restrictions may subject the user to criminal sanctions under applicable laws, as well as to civil liability for the breach of the terms and conditions of this license.

THIS SOFTWARE IS PROVIDED IN AN “AS IS” CONDITION. NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.

FIRMWARE SOURCE CODE

Define Constants, Registers and EEPROM Locations

The following section defines variables used by the firmware to control the charging regime. The EEPROM parameters described in the functional description are assigned addresses and variable names. Note that the internal firmware variable names for these parameters may not match the names used in the functional descriptions above. The names in the functional descriptions match the names in the PowerTool™ 200 software. The software and data sheet names have been given names that are more user-friendly.

The mode bits are defined which become user-selectable functions and charge features as described in the functional descriptions. Variable names are defined for hardware interface registers, such as A/D control and data, timers, PWM configuration and GPIO.

;=====================================================================
;=====================================================================
;text defines
;#define CLOCK_4MHZ
#define CLOCK_8MHZ
;#define ENABLE_COMM_LOCK
;#define DEBUG_ENABLE_TOGGLE

;--- firmware version
#define FW_VERSION_LO 0x01
#define FW_VERSION_HI 0x02
#include "pl16f785.inc"

;--- configuration
__CONFIG _CP_OFF & _CPD_OFF & _BOD_OFF & _BOR_OFF & _MCLRE_ON & _PWRTE_ON & _WDT_OFF & _INTRC_OSC_NOCLKOUT

;-----------------------------------------------------------
;--- registers: special function
;-----------------------------------------------------------
r_indf  equ INDF
r_tmr0  equ TMRO
r_pcl   equ PCL
r_status equ STATUS
r_fsr   equ FSR
r_port_a equ PORTA
r_port_b equ PORTB
r_port_c equ PORTC
r_pclath equ PCLATH
r_intcon equ INTCON
r_pir1  equ PIR1
r_tmr1l  equ TMR1L
r_tmr1h  equ TMR1H
r_t1con  equ T1CON
r_tmr2  equ TMR2
r_t2con  equ T2CON
r_ccpr1l equ CCP1L
r_ccpr1h equ CCP1H
r_ccp1con equ CCP1CON
r_wdtcon equ WDTCON
r_adresh equ ADRESH
r_adcon0 equ ADCON0
r_option_reg equ OPTION_REG
r_tris_a equ TRISA
r_tris_b equ TRISB
r_tris_c equ TRISC
r_pcon equ PCON
r_oscccon equ OSCCON
r_osctune equ OSCTUNE
r_ansel0 equ ANSEL0
r_pr2 equ PR2
r_ansel1 equ ANSEL1
r_wpua equ WPUA
r_ioca equ ICA
r_refcon equ REFCON
r_vrcon equ VRCON
r_eedata equ EDATA
r_eeadr equ EADDR
r_econ1 equ ECON1
r_econ2 equ ECON2
r_adres1 equ ADRESL
r_adcon1 equ ADCON1
r_pwmcon1 equ PWMCON1
r_pwmcon0 equ PWMCON0
r_pwmclk equ PWMCLK
r_pwmph1 equ PWMPH1
r_pwmph2 equ PWMPH2
r_cm1con0 equ CM1CON0
r_cm2con0 equ CM2CON0
r_cm2con1 equ CM2CON1
r_opa1con equ OPA1CON
r_opa2con equ OPA2CON

;*** register bank limits
#define ram0_start 0x20
#define ram0_end 0x7f
#define ram0_length ram0_end - ram0_start + 1
#define raml_start 0xa0
#define raml_end 0xef
#define raml_length raml_end - raml_start + 1

;-----------------------------------------------------------
;--- registers: user
;-----------------------------------------------------------
org 0x20 ; *** bank 0
r_mode res 1 ; operational mode register
r_chg_state res 1 ; charge controller *state*
r_adc_0 equ $ ;
r_adc_0_L res 1 ; adc result - channel 0
r_adc_0_H res 1 ;
r_adc_1 equ $ ;
r_adc_1_L res 1 ; adc result - channel 1
r_adc_1_H res 1 ;
r_adc_2 equ $ ;
r_adc_2_L res 1 ; adc result - channel 2
r_adc_2_H res 1 ;
r_adc_3 equ $ ;
r_adc_3_L res 1 ; adc result - channel 3
r_adc_3_H res 1 ;

###
r_adc_4        equ $ ; adc result - channel 4
r_adc_4_L      res 1
r_adc_4_H      res 1
r_pwm_L        res 1 ; pwm setting
r_pwm_H        res 1 ; pwm setting
r_reg_c        res 2 ; regulation target: current (mA)
r_reg_v        res 2 ; regulation target: voltage (mV)
r_comm_reg     equ $ ; comm "registers"
r_comm_reg_0   res 1 ; indirect address register
r_comm_reg_1   res 1 ; status
r_comm_reg_2   res 1 ; config flags
r_comm_reg_3   res 1 ; command flags
r_comm_reg_4   res 1 ; data lo
r_comm_reg_5   res 1 ; data hi
r_comm_reg_6   res 1 ;
r_comm_reg_7   res 1 ;
r_sim          res 1 ;
r_chg_timer_a  res 1 ; hysteresis timer
r_chg_timer_b  res 1 ; hysteresis timer
r_chg_timer_c  res 1 ; hysteresis timer
r_chg_timer_d  res 1 ; hysteresis timer
r_temp_1       res 1 ; location sensitive (init ram clear)
r_temp_2       res 1 ;
r_temp_3       res 1 ;
r_temp_4       res 1 ;
r_tempi_1      res 1 ; temporary reg for isr
r_timer_a1     res 1 ;
r_timer_b      res 1 ;
r_timer_b1     res 1 ;
r_timer_c      res 1 ;
r_timer_d      res 1 ;
r_timer_d1     res 1 ;
r_led_config_1 res 1 ;
r_led_ctrl1_1  res 1 ;
r_led_config2  res 1 ;
r_led_ctrl2_1  res 1 ;
r_adc_control  res 1 ; adc control
r_adc_raw_L    res 1 ;
r_adc_raw_H    res 1 ;
r_count_1      res 1 ;
r_accD_L       res 1 ; math - accumulator - D
r_accD_H       res 1 ;
r_accC_L       res 1 ; math - accumulator - C
r_accC_H       res 1 ;
r_accB_L       res 1 ; math - accumulator - B
r_accB_H       res 1 ;
r_accA_L       res 1 ; math - accumulator - A
r_accA_H       res 1 ;
r_comm_count   res 1 ;
r_comm_data    res 1 ;
r_comm_flags   res 1 ;
r_comm_data_cmd res 1 ;
r_mode2        res 1 ;

org 0x60
r_adc_accum    res 2 ;
r_adc_accum_count res 1 ;
r_adc_avg      res 2 ;
r_adc_avg_shadow res 2 ;
r_adc_1_ofs    res 1 ;
r_adc_2_save   res 2 ;
r_adc_3_save_a res 2 ;
r_adc_3_save_b  res 2  

r_mode3  res 1  

r_timer_d2  res 1  

;-----------------------------------------------------------

;--- registers: bank0,1,2,3 (common)  
;-----------------------------------------------------------

; org 0x70  

r_shadow_1  res 1  

r_shadow_2  res 1  

r_shadow_3  res 1  

; r_shadow_4  res 1  

r_flags_1  res 1  

; assorted bit flags  

r_flags_2  res 1  

; assorted bit flags  

r_flags_3  res 1  

; assorted bit flags  

r_flags_4  res 1  

; assorted bit flags  

r_flags_5  res 1  

; assorted bit flags  

r_flags_6  res 1  

; assorted bit flags  

r_isr_w  res 1  

; interrupt context  

r_isr_status  res 1  

; interrupt context  

r_isr_pclath  res 1  

; interrupt context  

r_isr_fsr  res 1  

; interrupt context  

r_isr_pclath  res 1  

; interrupt context  

r_isr_fsr  res 1  

; interrupt context  

r_ee_data  res 1  

; eeprom data  

r_ee_addr  res 1  

; eeprom address  

r_tempc_1  res 1  

;-----------------------------------------------------------

#define flag0_mode_pchg_always r_mode, 7  

; always start with pchg  

#define flag0_mode_gpio_cutoff r_mode, 6  

; enable gpio cutoff logic  

#define flag0_mode_bpres_battid r_mode, 2  

; use battid for batt present  

#define flag0_mode_bpres_v r_mode, 1  

; battery present on voltage sense  

#define flag0_mode_bpres_always r_mode, 0  

; battery present - always  

#define flag0_mode_coefs_dis r_mode2, 7  

; current offset - disable  

#define flag0_mode_oscout r_mode2, 6  

; enable oscillator out on battid  

#define flag0_mode_temp_k r_mode2, 5  

; use constant temperature 25degC  

#define flag0_mode_nm r_mode2, 4  

; nickel metal hydride algorithm  

#define flag0_mode_vrcchg_dis r_mode2, 2  

; voltage recharge - disable  

#define flag0_mode_vregc_dis r_mode2, 1  

; regulation voltage cutoff - disable  

#define flag0_mode_pwnmas_dis r_mode2, 0  

; pwm auto shutdown - disable  

#define flag0_mode_nm_extdcchg_en r_mode3, 7  

; enable discharge  

#define flag0_mode_nm_exttrik_en r_mode3, 6  

; enable discharge  

#define flag0_mode_nm_trik_byp r_mode3, 5  

; disable trickle stage  

#define flag0_mode_nm_topoff_en r_mode3, 4  

; enable topoff stage  

#define flag0_mode_nm_vrcchg_en r_mode3, 3  

; enable trickle re-charge  

#define flag0_mode_nm_slochrg r_mode3, 2  

; enable slow charge mode  

#define flag0_mode_nm_eoc_dtdt r_mode3, 1  

; enable eoc method - dtdt  

#define flag0_mode_nm_eoc_dv r_mode3, 0  

; enable eoc method - minus dv  

BN_CREG_EE_BUSY  equ .7  

; ee write busy  

BN_CREG_EE_ERR  equ .6  

; error on last ee write  

;BN_CREG_UNLOCKED equ .5  

; comm unlocked  

BN_CREG_REG  equ .4  

; regulation active  

BN_CREG_CHGCON  equ .3  

; charge controller enabled  

BN_CREG_SIM  equ .2  

; simulation active (>=1 channel)  

#define flag0_creg_st_ee_busy r_comm_reg_1, BN_CREG_EE_BUSY  

#define flag0_creg_st_ee_err r_comm_reg_1, BN_CREG_EE_ERR  

;#define flag0_creg_st_unlocked r_comm_reg_1, BN_CREG_UNLOCKED  

#define flag0_creg_st_reg r_comm_reg_1, BN_CREG_REG  

#define flag0_creg_st_chgcon r_comm_reg_1, BN_CREG_CHGCON  

#define flag0_creg_st_sim r_comm_reg_1, BN_CREG_SIM  

#define flag0_creg_suspend r_comm_reg_2, 5
#define flag0_creg_chgcon_off r_comm_reg_2, 4
#define flag0_creg_membank_ee r_comm_reg_2, 1
#define flag0_creg_membank_23 r_comm_reg_2, 0
#define flag0_creg_version r_comm_reg_3, 7
#define flag0_creg_pwm_set r_comm_reg_3, 6
#define flag0_creg_reg_on r_comm_reg_3, 5
#define flag0_creg_ee_rq r_comm_reg_3, 4
#define flag0_creg_test r_comm_reg_3, 3
#define flag0_creg_reset r_comm_reg_3, 2
#define flag0_creg_fchgstate r_comm_reg_3, 1
#define flag0_creg_sim_rq r_comm_reg_3, 0
#define flag0_creg_ee_rq r_comm_reg_2, 1
#define flag0_creg_membank_ee r_comm_reg_2, 0
#define flag0_creg_version r_comm_reg_3, 7
#define flag0_creg_pwm_set r_comm_reg_3, 6
#define flag0_creg_reg_on r_comm_reg_3, 5
#define flag0_creg_ee_rq r_comm_reg_3, 4
#define flag0_creg_test r_comm_reg_3, 3
#define flag0_creg_reset r_comm_reg_3, 2
#define flag0_creg_fchgstate r_comm_reg_3, 1
#define flag0_creg_sim_rq r_comm_reg_3, 0
#define flag_ee_busy r_flags_1, 7
#define flag_ee_rq r_flags_1, 6
#define flag_ee_err r_flags_1, 5
#define flag_chg_state_timer r_flags_1, 4
#define flag_chg_state_timer r_flags_1, 3
#define flag_math_temp r_flags_1, 2
#define flag_timer_0 r_flags_1, 1
#define flag_led_timer r_flags_1, 0
#define flag_v_le_vmin r_flags_2, 7
#define flag_v_le_vmax r_flags_2, 6
#define flag_v_le_vreg r_flags_2, 5
#define flag_v_le_vpchg r_flags_2, 4
#define flag_t_le_tmin r_flags_2, 3
#define flag_v_le_vpchg r_flags_2, 2
#define flag_v_le_vreg r_flags_2, 1
#define flag_v_le_vpchg r_flags_2, 0
#define flag_unlocked r_flags_3, 7
#define flag_temp_1 r_flags_3, 6
#define flag_temp_2 r_flags_3, 5
#define flag_neg r_flags_3, 4
#define flag_chg_timer r_flags_3, 3
#define flag_adcset_2_rq r_flags_3, 2
#define flag_adcset_1_rq r_flags_3, 1
#define flag_adcset_0_rq r_flags_3, 0
#define flag_reg_timer r_flags_5, 7
#define flag_battpres1 r_flags_5, 6
#define flag_battpres r_flags_5, 5
#define flag_adcset_2_rdy r_flags_4, 7
#define flag_adcset_2_rqq r_flags_4, 6
#define flag_adcset_1_rdy r_flags_4, 5
#define flag_adcset_1_rqq r_flags_4, 4
#define flag_adcset_0_rdy r_flags_4, 3
#define flag_adcset_0_rqq r_flags_4, 2
#define flag_adcset_0_rdy r_flags_4, 1
#define flag_adcset_0_rqq r_flags_4, 0
#define flag_reg_timer r_flags_5, 7
#define flag_battpres1 r_flags_5, 6
#define flag_battpres r_flags_5, 5
#define flag_comm_active r_flags_5, 4
#define flag_reg_on r_flags_5, 3
#define flag_vreg r_flags_5, 2
#define flag_vreg_2 r_flags_5, 1
#define flag_vreg_1 r_flags_5, 0

#define BN_CHGN_TSEL 0
#define flag_chg_ti1_done r_flags_6, 7
#define flag_chg_ti2_done r_flags_6, 6
#define flag_chgn_tsel r_flags_6, BN_CHGN_TSEL
#define MASK_CHGN_TSEL 1<<BN_CHGN_TSEL
#define REG_CHGN_TSEL r_flags_6
#define ADCH_4 0x10
#define ADCH_3 0x08
#define ADCH_2 0x04
#define ADCH_1 0x02
#define ADCH_0 0x01
#define ADCSET_0  ADCH_4 | ADCH_3 | ADCH_2 | ADCH_1 | ADCH_0
#define ADCSET_1  ADCH_2 | ADCH_1
#define ADCSET_2  ADCH_4 | ADCH_3 | ADCH_2 | ADCH_1
#define COMM_UNLOCK_KEY 0x96
#define flag_comm_pin r_comm_flags, 5
#define flag_comm_timeout r_comm_flags, 4
#define flag_comm_cmnd r_comm_flags, 3
#define flag_comm_bit r_comm_flags, 2
#define flag_comm_H2L r_comm_flags, 1
#define flag_comm_xmit r_comm_flags, 0

;-----------------------------------------------------------
;--- registers: bank1
;-----------------------------------------------------------
org 0xa0 ; *** bank 1
r_buf1 equ $ ; sim data, ee write buf data
r_ee_buf equ $ ;
r_ee_buf_adr res .1 ;
r_ee_buf_cnt res .1 ;
r_ee_buf_dta res .29 ;
r_ee_buf_ptr res .1 ;
org 0xb0
r_buf2 res .16 ; overlaps 2nd half of r_buf1
; scratchpad for LUT

;-----------------------------------------------------------
;--- registers: bank2
;-----------------------------------------------------------
org 0x110 ; *** bank 2

;-----------------------------------------------------------
;--- registers: bank3
;-----------------------------------------------------------
org 0x190 ; *** bank 3

;-----------------------------------------------------------
;--- constants: timing
;-----------------------------------------------------------

;=== option reg
#ifdef CLOCK_8MHZ
#define clk_p equ .8000000 ; (mhz) clock frequency
#define OSCCON_DEFAULT equ 0x70 ;
;--- option
option_default equ l<<NOT_RAPU | 0x02

tmr1_default equ 0x10 ; 2:1 scale, lusec tic
TIME_COMM_USEC_T equ l
#endif

clk_i equ clk_p / .4 ; (mhz) instruction clock timer resolution (class b)

TIMER_A_USEC equ .1024 ; (usec) timer resolution (class a)
TIMER_B_MSEC equ .250 ; (msec) timer resolution (class b)
TIMER_C_MSEC equ .1000 ; (msec) timer resolution (class c)
TIMER_D_SEC equ .240 ; (sec) timer resolution (class d)
TIMER_A1_MSEC equ .20 ; (msec) regulation timer

TIMER_A1_TA equ ((TIMER_A1_MSEC * .1000) + TIMER_A_USEC / 2) / TIMER_A_USEC
TIMER_B_TA equ (TIMER_B_MSEC * .1000) / TIMER_A_USEC
TIMER_C_TB equ (TIMER_C_MSEC) / TIMER_B_MSEC
TIMER_D_TC equ (TIMER_D_SEC * .1000) / TIMER_C_MSEC

TIME_COMM_REPLY_USEC equ .250
TIME_COMM_B1_LO_USEC equ .20
TIME_COMM_B1_HI_USEC equ .230
TIME_COMM_B0_LO_USEC equ .170
TIME_COMM_B0_HI_USEC equ .080
TIME_COMM_0_MAX_USEC equ .175
TIME_COMM_1_MAX_USEC equ .70
TIME_COMM_BREAK_USEC equ .200

TIME_COMM_REPLY_T equ TIME_COMM_REPLY_USEC / TIME_COMM_USEC_T
TIME_COMM_B1_LO_T equ TIME_COMM_B1_LO_USEC / TIME_COMM_USEC_T
TIME_COMM_B0_LO_T equ TIME_COMM_B0_LO_USEC / TIME_COMM_USEC_T
TIME_COMM_B1_HI_T equ TIME_COMM_B1_HI_USEC / TIME_COMM_USEC_T
TIME_COMM_B0_HI_T equ TIME_COMM_B0_HI_USEC / TIME_COMM_USEC_T
TIME_COMM_0_MAX_T equ TIME_COMM_0_MAX_USEC / TIME_COMM_USEC_T
TIME_COMM_1_MAX_T equ TIME_COMM_1_MAX_USEC / TIME_COMM_USEC_T
TIME_COMM_BREAK_T equ TIME_COMM_BREAK_USEC / TIME_COMM_USEC_T

;-----------------------------------------------------------
;--- constants: i/o configuration
;-----------------------------------------------------------
#define TRIS_A_COMM b'11111011'
#define TRIS_B_DEFAULT b'00111111'
#define TRIS_C_BIOUT b'11001101'
#define TRIS_C_DEFAULT b'11001111'
#define p_led_1 r_port_b, 7
#define p_gpio r_port_b, 6
#define p_led_2 r_port_a, 5
#define p_comm r_port_a, 5
#define p_batid r_port_c, 1

;--- IOCA
#define IOCA_DEFAULT l<<IOCA5

;--- WPJA
#define WPJUA_DEFAULT l<<WPJUA5

;--- OPA1CON
;debug OVP
#define OPA1CON_DEFAULT l<<OPA1CON
#define OPA1CON_DEFAULT 0<<OPA1CON
```c
//--- OPA2CON
#define OPA2CON_DEFAULT  1<<OPAON

//--- CM1: INPUTS: RA1/C1Ref  SPEED: NORM, OUTPUT: INT
#if define CM1CON0_DEFAULT  1<<C1R | 1<<C1SP | 1<<C1ON
debug OVP
#define CM1CON0_DEFAULT  1<<C1R | 1<<C1SP | 1<<C1ON | 1<<C1OE

#define CM2CON0_DEFAULT  1<<C2ON | 0<<C2POL | 0<<C2SP | 0<<C2R | 0<<C2CH0
#define CM2CON1_DEFAULT  0

//--- VRCON: default 1.2V
#define VRCON_DEFAULT  0

//--- REFCON: ENABLED
#define REFCON_DEFAULT  1<<VREN | 0<<VROE

//--- ANSEL0
#define ANSEL0_DEFAULT  1<<ANS0 | 1<<ANS1 | 0<<ANS2 | 1<<ANS3 | 1<<ANS4 | 0<<ANS5

//--- ANSEL1
#define ANSEL1_DEFAULT  1<<ANS8 | 1<<ANS9 | 1<<ANS10 | 1<<ANS11

//--- PWMCON0
#define PWMCON0_DEFAULT  0<<BLANK2 | 1<<PH2EN
#define PWMCON0_DEFAULT  0<<BLANK2 | 1<<PH2EN | 1<<PASEN

//--- PWMCLK
#define PWMCLK_DEFAULT  0<<PWMP0 | 0<<PER0

//--- PWMH2
#define PWMH2_DEFAULT  0<<POL | 1<<C2EN | 0<<C1EN | 1<<PH0

//--- ADCON
#define ADC_ADCON0_DEFAULT  1<<ADFM | 1<<ADON
#define ADC_ADCON0_0 ADC_ADCON0_DEFAULT | .13<<CHS0 | 0<<VCFG
#define ADC_ADCON0_1 ADC_ADCON0_DEFAULT | .06<<CHS0 | 0<<VCFG
#define ADC_ADCON0_2 ADC_ADCON0_DEFAULT | .03<<CHS0 | 0<<VCFG
#define ADC_ADCON0_3 ADC_ADCON0_DEFAULT | .00<<CHS0 | 0<<VCFG
#define ADC_ADCON0_4 ADC_ADCON0_DEFAULT | .05<<CHS0 | 0<<VCFG
#define ADC_CHANNEL_MASK 0x1F
#define ADC_ADCON1_DEFAULT 0x05<<ADCS0

//--- ADC_TAQ (# of 3-instruction loops ... 8mhz => 1.5us/loop)
; .83 => 125usec  .21 => 32usec
#define ADC_TAQ  .21

//--- CCP1CON
#define CCP1CON_DEFAULT  0x0c

//--- constants: interrupts

//--- constants:
#define PWM_DEFAULT  equ .000

//--- EE MAP
#define EE_PATTERN  equ .0
```
`EE_NCELLS` equ .2
`EE_CAPACITY` equ .19
`EE_PWM_FREQ` equ .21
`EE_MODE` equ .22
`EE_MODE2` equ .23
`EE_OSC_TRIM` equ .24

`EE_LED1_CFG` equ .32
`EE_LED2_CFG` equ .40

`EE_REG_P1` equ .52
`EE_REG_P2` equ .53
`EE_REG_P3` equ .54
`EE_REG_P4` equ .55
`EE_REG_VHH_VH` equ .56
`EE_REG_VH` equ .57
`EE_REG_VL` equ .58
`EE_REG_VLL_VL` equ .59
`EE_REG_CNULL` equ .60
`EE_REG_VSAFETY` equ .61

`EE_CHG_V_MIN` equ .67
`EE_CHG_V_MAX` equ .69
`EE_CHG_V_RCHG` equ .71
`EE_CHG_V_REG` equ .73
`EE_CHG_V_PCHG` equ .75
`EE_CHG_V_MIN_BP` equ .77
`EE_CHG_T_MIN` equ .79
`EE_CHG_T_MAXCHGI` equ .80
`EE_CHG_T_MAXCHG` equ .81
`EE_CHG_T_PCHG` equ .82
`EE_CHG_C_PCHG` equ .83
`EE_CHG_C_CREG` equ .85
`EE_CHG_C_MIN` equ .87
`EE_CHG_TI_PCHG` equ .89
`EE_CHG_TI_CREG` equ .90
`EE_CHG_TI_VREG` equ .91
`EE_CHG_TIME_0` equ .92
`EE_CHG_TIME_1` equ .93
`EE_CHG_TIME_2` equ .94
`EE_CHG_TIME_3` equ .95
`EE_CHG_TIME_4` equ .96
`EE_CHG_TIME_5` equ .97
`EE_BATTID_MIN` equ .98
`EE_BATTID_MAX` equ .99

`EE_CAL_ADC` equ .108
`EE_CAL_ADC_0` equ .108
`EE_CAL_ADC_1` equ .110
`EE_CAL_ADC_2` equ .112
`EE_CAL_ADC_3` equ .114
`EE_CAL_ADC_4` equ .116
`EE_SHUNT` equ .118
`EE_T_DEFAULT` equ .119

`EE_T_LUT_N` equ .124
`EE_T_LUT_T` equ .125
`EE_T_LUT_MB` equ .132

`EE_MODE3` equ .170
`EE_CAL_ADC_2_NM` equ .171
`EE_CHGN_V` equ .173
`EE_CHGN_V_DCHG` equ .175
`EE_CHGN_T_MAX` equ .177
`EE_CHGN_V_MAX` equ .178
EE_CHGN_C_PCHG        equ .180
EE_CHGN_T_PCHG_LO     equ .182
EE_CHGN_T_PCHG_HI     equ .183
EE_CHGN_V_PCHG        equ .184
EE_CHGN_TI_PCHG_MN    equ .186
EE_CHGN_TI_PCHG_MX    equ .187
EE_CHGN_C_FCHG        equ .188
EE_CHGN_TI_FCHG_MN    equ .190
EE_CHGN_TI_FCHG_MX    equ .191
EE_CHGN_TI_DTDT       equ .192
EE_CHGN_T_DTDT        equ .193
EE_CHGN_V_DV          equ .194
EE_CHGN_C_TOPOFF      equ .196
EE_CHGN_TI_TOPOFF     equ .198
EE_CHGN_V_RCHG        equ .199
EE_CHGN_C_TRIK        equ .201
EE_CHGN_TI_TRIK       equ .203
EE_REG_VSAFETY_NM     equ .204

;=====================================================================
Interrupt Service

This routine sets up the Reset vector, then the Interrupt Status register for “PORTA” GPIO, and the interrupt and communication timers.

```assembly
;=====================================================================
or 0x00
vector_reset: 
  goto start 

org 0x0004
isr:
  vector_isr: 
    movwf r_isr_w ; save context
    swapf r_status, w ;
    clrf r_status ;
    movwf r_isr_status ;
    movf r_pclath, w ;
    movwf r_isr_pclath ;
    movf r_fsr, w ;
    movwf r_isr_fsr ;

isr_rac: ; isr: PORTA CHANGE
  btfss r_intcon, RAIF ;
  goto isr_rbc_x ;
  movf r_port_a, w ;
  bcf flag_comm_pin ;
  btfsc r_port_a, 5 ;
  bsf flag_comm_pin ;
  bcf r_intcon, RAIF ;
  ; incf r_temp_3, f ;
  call comm_isr ;
  call blink_3 ;

isr_rbc_x: ;

isr_t1: ; isr: TMR1
  btfss r_intcon, TMR1IF ;
  goto isr_t1_x ;
  bcf r_intcon, TMR1IF ;
  bcf r_t1con, TMR1ON ;
  bsf flag_comm_timeout ;
  call comm_isr ;

isr_t1_x: ;

isr_t0: ; isr: TMR0
  btfss r_intcon, T0IF ;
  goto isr_t0_x ;
  bcf r_intcon, T0IF ;
  bsf flag_timer_0 ;

isr_t0_x: ;

isr_x: 
  movf r_isr_fsr, w ;
  movwf r_fsr ;
  movf r_isr_pclath, w ;
  movwf r_pclath ;
  swapf r_isr_status, w ;
  movwf r_status ;
  swapf r_isr_w, f ;
  swapf r_isr_w, w ;
  retfie ;
;=====================================================================
```

© 2006 Microchip Technology Inc.
#ifdef DEBUG_ENABLE_TOGGLE
#define p_toggle r_port_c, 1

blink_4:
    bsf r_port_c, 1 ;
    bcf r_port_c, 1 ;

blink_3:
    bsf r_port_c, 1 ;
    bcf r_port_c, 1 ;

blink_2:
    bsf r_port_c, 1 ;
    bcf r_port_c, 1 ;

blink_1:
    bsf r_port_c, 1 ;
    bcf r_port_c, 1 ;
    return ;

toggle:
    btfss flag_temp_1 ;
    goto toggle_1 ;
    bcf flag_temp_1 ;
    bcf r_port_c, 1 ;
    return ;

toggle_1:
    bsf flag_temp_1 ;
    bsf r_port_c, 1 ;
    return ;

#endif
Start-up Initialization

This routine runs whenever the part is first powered up or reset. This includes initial hardware configurations, such as oscillator, GPIO ports and voltage reference configurations. It sets the initial PWM frequency, checks for communication and outputs the clock on the BAT1D pin if requested.

;    org 0x100
;start:

    clrf      r_port_a
    clrf      r_port_b
    clrf      r_port_c

;;;--- default gpio
    bsf       p_gpio                  
    bsf       r_status, RP0          ; *** bank=1

;;;--- configure oscillator
    movlw     EE_OSC_TRIM            
    call      ee_read_waddr
    bsf       r_status, RP0          ; *** bank=1
    movwf     r_osctune
    movlw     OSCCON_DEFAULT         
    movwf     r_osccon

;;;--- configure ports
    movlw     TRIS_A_COMM            
    movwf     r_tris_a
    movlw     IOCA_DEFAULT           
    movwf     r_ioca
    movlw     WPUA_DEFAULT           
    movwf     r_wpua
    movlw     TRIS_B_DEFAULT         
    movwf     r_tris_b
    movlw     TRIS_C_DEFAULT         
    movwf     r_tris_c

;;;--- option
    movlw     option_default         
    movwf     r_option_reg

;;;--- vrcon
    movlw     VRCON_DEFAULT          
    movwf     r_vrcon

;;;--- refcon
    movlw     REFCON_DEFAULT         
    movwf     r_refcon

;;;--- ansiel
    movlw     ANSEL0_DEFAULT         
    movwf     r_anse10
    movlw     ANSEL1_DEFAULT         
    movwf     r_anse11
    bcf       r_status, RP0          ; *** bank=0
    bcf       r_status, RP1          ; *** bank=2

;;;--- opamps
    movlw     OPA1CON_DEFAULT        
    movwf     r_opa1con
    movlw     OPA2CON_DEFAULT        
    movwf     r_opa2con

;;;--- comparators
movlw CM1CON0_DEFAULT; 
movwf r_cm1con0; 
movlw CM2CON0_DEFAULT; 
movwf r_cm2con0; 
movlw CM2CON1_DEFAULT; 
movwf r_cm2con1; 
bcf r_status, RP1; *** bank=0 
bsf r_tlcon, TMR1ON; enable timer 1  

;--- clear ram  
bcf r_status, IRF; 
movlw 0x20; 
movwf r_fsr; 
movlw .96; 
call ram_clear; 
movlw 0xA0; 
movwf r_fsr; 
movlw .32; 
call ram_clear;  

;--- setup timer(s)  
movlw tmr1_default; 
movwf r_t1con; 

;--- move MODE params from ee  
movlw EE_MODE; 
call ee_read_waddr_ia; 
movwf r_mode; 
call ee_read; 
movwf r_mode2; 
movlw EE_MODE3; 
call ee_read_waddr; 
movwf r_mode3;  

;--- pwm  
movlw EE_PWM_FREQ; 
call ee_read_waddr; 
andlw 0x7f; 
bsf r_status, RP1; *** bank=2 
movwf r_pwmclk1; 
movlw PWMPH2_DEFAULT; 
movwf r_pwmph2; 
bcf r_status, RP1; *** bank=0 
movlw PWMCN0_AS_DIS; 
btfsc flag0_mode_pwm_dis; 
movlw PWMCN0_AS_EN; 
btfsc flag0_mode_nm; nimh mode? - disable pwmas 
movlw PWMCN0_AS_DIS; 
bsf r_status, RP1; *** bank=2 
movwf r_pwmcon0; 
bcf r_status, RP1; *** bank=0 

;--- config pwm  
call pwm_config; 

init: ; 
---------------------------------------<<<<<<<<<<<<<<<<<<<<<<<<<************** 
main_init: ; 

;--- init chg controller timer  
movlw 0xff; 
movwf r_chg_timer_a;
;--- comm?
comm_chk:
    call comm_pin_input_
    movlw .255
    movwf r_temp_1
comm_chk_loop:
    btfsc p_comm
    goto comm_chk_on
    decfsz r_temp_1, f
    goto comm_chk_loop
comm_chk_off:
    call comm_off
    goto comm_chk_x
comm_chk_on:
    call comm_on
comm_chk_x:

;--- pwm default
    movlw low PWM_DEFAULT
    movwf r_pwm_L
    movlw high PWM_DEFAULT
    movwf r_pwm_H
    call pwm_config

;--- init chg controller
    call chg_state_0_init

;--- option: output clock on batid pin
osc_out:
    btfs flag0_mode_oscout
    goto osc_out_x
    bsf r_status, RP0 ; *** bank=1
    movlw TRIS_C_BIOUT
    movwf r_tris_c
    bcf r_status, RP0 ; *** bank=0
    movlw .0
    movwf r_temp_1
osc_out_loop:
    bsf p_batid
    bcf p_batid
    decfsz r_temp_1, f
    goto osc_out_loop
    bsf r_status, RP0 ; *** bank=1
    movlw TRIS_C_DEFAULT
    movwf r_tris_c
    bcf r_status, RP0 ; *** bank=0
osc_out_x:

;--- interrupts
    bsf r_intcon, T0IE
    bsf r_intcon, RAIE
    bsf r_intcon, GIE

#ifdef DEBUG_ENABLE_TOGGLE
    bsf r_status, RP0 ; *** bank=1
    movlw TRIS_C_BIOUT
    movwf r_tris_c
    bcf r_status, RP0 ; *** bank=0
#endif

;-----------------------------------------------
Main Loop

The main loop of this firmware cycles through the subroutines that call the primary functions:

- **adc_svc**: Receive the finished A/D conversions, process the data with calibration constants, etc., and store in RAM
- **adc_start**: Start a new set of conversions to be completed for the next cycle
- **check_triggers**: Compare the A/D results with parameters to determine what state the charging should be in
- **chg_state_svc**: Put the charger into the proper state based on A/D results
- **regulate**: Adjust the PWM to regulate current based on charge state and feedback measurements
- **led_svc**: Operate two LEDs to display the charge state
- **timer_svc**: Maintain the firmware timers
- **ee_write_buf**: Background process to write the data block in the RAM buffer into EEPROM
- **ccmd_svc**: React to communication commands
- **status_build**: Build the Status Byte Communication register

```plaintext
main:

;--- reset?
    btfsc flag0_creg_reset ; "reset" command flag ?
    goto vector_reset ; --- yes, goto reset vector

    btfsc flag0_creg_suspend ; "suspend" command flag ?
    goto main_suspended ; --- yes, skip processing steps ...

    call adc_svc ; service: ADC
    call adc_start ; service: ADC scheduler
    call check_triggers ; compare adc results to triggers
    call chg_state_svc ; service: charge state controller
    call regulate ; service: regulation

main_suspended:

    call led_svc ; service: LEDs
    call timer_svc ; service: timers
    call ee_write_buf ; service: background EE write
    call ccmd_svc ;
    call status_build ;

main_x:
    goto main ;
```

;-----------------------------------------------------------
FIGURE 13: MAIN LOOP FLOWCHART

- $r_{comm\_reg\_3<2>} = 1$? (Firmware Reset)
- $r_{comm\_reg\_2<5>} = 1$? ("Suspend" mode)

Call functions:
- `call adc_svc` (ADC Service)
- `call adc_start` (ADC Start)
- `call check_triggers` (Compare ADC Results to Threshold)
- `call chg\_state_svc` (Service Charge Control)
- `call regulate` (Dynamic Voltage/Current Control)
- `call led_svc` (LED Service)
- `call timer_svc` (Service Timer(s))
- `call ee_write_buf` (Execute EEPROM Control Packet)
- `call ccmd_svc` (Service Comm Control Commands)
- `call status_build` (Build Status Byte, i.e., "comm\_reg\_1")

Yes Yes

GOTO VECTOR_RESET
Communication Command Service

This routine is run in response to communication activities that use the control bits in the Communication registers to modify behavior. All of the communication functions, as described in the functional descriptions on communication, are implemented below, including all of the functionality in each bit of the Communication registers. Functions include reading the firmware version, setting the PWM, turning on regulation, forcing the branch to the charge controller state machine and running simulation.

;---------------------------------------
;--- comm command service
;---------------------------------------
ccmd_svc:                               
ccmd_ee_rq:                             
btfss flag0_creg_ee_rq                
goto ccmd_ee_rq_x                      
bcf flag0_creg_ee_rq                  
bsf flag_ee_rq                         
ccmd_ee_rq_x:                           
ccmd_version:                           
btfss flag0_creg_version              
goto ccmd_version_x                   
bcf flag0_creg_version                
movlw FW_VERSION_LO                   
movwf r_comm_reg_4                    
movlw FW_VERSION_HI                   
movwf r_comm_reg_5                    
ccmd_version_x:                         
ccmd_pwm_set:                           
btfss flag0_creg_pwm_set              
goto ccmd_pwm_set_x                   
bcf flag0_creg_pwm_set                
movf r_comm_reg_4, w                 
movwf r_pwm_L                         
movf r_comm_reg_5, w                 
movwf r_pwm_H                         
call pwm_set                          
ccmd_pwm_set_x:                         
ccmd_reg_on:                            
btfss flag0_creg_reg_on               
goto ccmd_reg_on_x                    
bcf flag0_creg_reg_on                 
call reg_on                           
ccmd_reg_on_x:                          
ccmd_chg_state_force:                   
btfss flag0_creg_fchgstate            
goto ccmd_chg_state_force_x           
bcf flag0_creg_fchgstate              
movf r_comm_reg_4, w                 
movwf r_chg_state                     
call chg_state_svc_jumptable          
ccmd_chg_state_force_x:                 
ccmd_sim_rq:                            
btfss flag0_creg_sim_rq               
goto ccmd_sim_rq_x                    
bcf flag0_creg_sim_rq                 
call sim_rq_proc                      

ccmd_sim_rq_x: ;

    return ;

;-----------------------------
;--- sim_rq_proc()
;-----------------------------
sim_rq_proc: ;
    movlw    r_buf1 ;
    movwf    r_fsr ;
    bcf      r_status, IRP ;
    movf     r_indf, w ;
    movwf    r_adc_3_L ;
    incf     r_fsr, f ;
    movf     r_indf, w ;
    movwf    r_adc_3_H ;
    bsf      flag_adc_3_sim ;
sim_rq_proc_x: ;
    return ;

;---------------------------------------
Status Register Build

This routine builds the Communication Status register – Communication Register 1. This includes bit-maps for signifying that EEPROM write is in progress, EEPROM write resulted in an error, regulation is active, charge controller is active and data simulation is active.

```assembly
;---------------------------------------
;--- status_build()
;---------------------------------------
status_build:                           
  clrf      r_temp_1                 ;
  btfsc     flag_ee_busy             ;
  bsf       r_temp_1, BN_CREG_EE_BUSY ;
  btfsc     flag_ee_err              ;
  bsf       r_temp_1, BN_CREG_EE_ERR ;
  btfsc     flag_reg_on              ;
  bsf       r_temp_1, BN_CREG_REG    ;
  btfss     flag0_creg_chgcon_off    ;
  bsf       r_temp_1, BN_CREG_CHGCON ;
  movlw     0x1f                     ;
  andwf     r_sim, w                 ;
  btfss     r_status, Z              ;
  bsf       r_temp_1, BN_CREG_SIM    ;
  movf      r_temp_1, w              ;
  movwf     r_comm_reg_1             ;
  return                             ;
```

;---------------------------------------
PWM Configuration (Subroutine of Start-up Initialization)

This routine sets the initial PWM value during start-up initialization.

```
pwm_config:
    ; *** bank=1
    movlw 0xff ;
    movwf r_pr2 ;
    ; *** bank=0
    movlw 0x80 ;
    movwf r_ccpr1l ;
    movlw 0x04 ; enable timer2 (TMR2ON)
    movwf r_t2con ;
    movlw CCP1CON_DEFAULT ; set "pwm" mode CCP1M3,2,1,0
    movwf r_ccp1con ;
    return ;
```
PWM Set (Subroutine of Regulate)

During the "regulate" phase of the main loop, this routine is used to load the PWM in response to changing PWM values. The values are typically changed because the charge state changed or the feedback measurements are not close enough to requirements. This routine loads the PWM with the new value.

```assembly
;---------------------------------------
;--- pwn_set()
;---------------------------------------
pwm_set:                                
       rrf       r_pwm_H, w               
       movwf     r_accA_H                 
       rrf       r_pwm_L, w               
       movwf     r_accA_L                 
       rrf       r_accA_H, f              
       rrf       r_accA_L, w              
       movwf r_ccpr1l; load bits: 9:2    
       swapf r_pwm_L, w               
       andlw 0x30;                       
       iorlw CCP1CON_DEFAULT;            
       movwf r_ccplcon;                  
       movf r_pwm_L, w               
       iorwf r_pwm_H, w               
       btfsc r_status, Z              
       goto pwm_disable;               
       pwm_enable:                             
       bsf       r_status, RP1            
       bsf r_pwmcon0, PH2EN             
       goto pwm_set_x;                  
pwm_disable:                            
       bsf       r_status, RP1            
       bcf       r_pwmcon0, PH2EN         
pwm_set_x:                              
       bcf       r_status, RP1            
       return                             
;---------------------------------------
```
PWM Adjust (Subroutine of Regulate)

This routine determines how much the PWM needs to change as a result of feedback measurements. The table discussed in the functional descriptions is followed to determine the PWM change value as a function of feedback measurements vs. requirements of voltage and current.

```assembly
;---------------------------------------
;--- pwm_adj()
;---------------------------------------
pwm_adj:                                
    movwf r_accB_L                    ; accB = pwm delta
    clrf r_accB_H                    ;
    btfsc flag_neg                    ; delta negative ?
    call math_neg_B                  ; --- yes, invert ...
    movlw r_pwm_L                    ; accA = pwm
    call math_add_16_load_A          ; accB = accA + accB = pwm + -(pwm delta)
    btfss r_accB_H, 7                ; result negative ?
    goto pwm_adj_pos                 ; --- no, skip ...
    clrf r_accB_L                    ; set result = 0
    clrf r_accB_H                    ;
    goto pwm_adj_x                   ;
pwm_adj_pos:                            
    movlw 0xfc                       ; result exceeds range of pwm setting ?
    andwf r_accB_H, w                ;
    btfsc r_status, Z                ;
    goto pwm_adj_x                   ; --- no, skip ...
    movlw 0x03                       ; --- yes, set result = 0x03ff
    movwf r_accB_H                    ;
    movlw 0x0ff                      ;
    movwf r_accB_L                    ;
pwm_adj_x:                              
    movlw r_pwm_L                    ; pwm = result
    call math_move_B                  ;
    return                           ;
;---------------------------------------
```

© 2006 Microchip Technology Inc. DS01012B-page 49
Ram Clear (Subroutine of Start-up Initialization)

The following routine clears all the RAM. This is typically performed on power-up or Reset.

;---------------------------------------
;--- util: ram clear
;
; call:
;     w       count
;     fsr      pointer
;     status[IRP] ram bank 0/1 or 2/3
;---------------------------------------
ram_clear:
    clrf      r_indf                   ; clear location
    incf      r_fsr, f                 ; bump pointer
    addlw     0xff                     ; decrement count
    btfss     r_status, Z              ;
    goto      ram_clear                ;
    return                             ;
;---------------------------------------
EEPROM Buffer Write

This routine writes the RAM EEPROM buffer to the EEPROM. As described in "Host Driven Operations", an EEPROM write is performed by writing a control block to a RAM buffer. A Communication register bit is then set to execute this routine to copy the RAM buffer to the EEPROM.

```assembly
;---------------------------------------
;--- ee buffer write
;---------------------------------------

ee_write_buf:                           
    bsf       r_status, RP0            ; *** bank=1
    btfsc     flag_ee_busy             ; busy ?
    goto      ee_write_buf_busy        ; --- yes, skip ...
    btfss     flag_ee_rq               ; request pending ?
    goto      ee_write_buf_x           ; --- no, exit ...

ee_write_buf_prep:                      
    movlw     r_ee_buf                 ; prep for checksum check
    movwf     r_fsr                    ;
    bcf       r_status, IRP            ;
    movf      r_ee_buf_cnt, w          ;
    addlw     .2                       ;
    call      chksum                   ; calc checksum
    xorwf     r_indf, w                ; compare to checksum in buffer
    btfss     r_status, Z              ; checksum ok ?
    goto      ee_write_buf_err         ; --- no, process error ...

    movlw     r_ee_buf_dta             ;
    movwf     r_ee_buf_ptr             ;
    bsf       flag_ee_busy             ; --- yes, set "busy" flag
    bcf       flag_ee_err              ;

ee_write_buf_busy:                      
    bsf       r_status, RP1            ; *** bank=3
    btfsc     r_eecon1, WR             ; ee write in progress ?
    goto      ee_write_buf_x           ; --- yes, exit ...
    bcf       r_status, RP1            ; *** bank=1
    movf      r_ee_buf_cnt, f          ;
    btfsc     r_status, Z              ;
    goto      ee_write_buf_done        ;

ee_write_buf_next:                      
    movf      r_ee_buf_ptr, w          ;
    movwf     r_fsr                    ;
    movf      r_indf, w                ; w = data
    movwf     r_ee_data                ;
    movf      r_ee_buf_adr, w          ;
    call      ee_write_waddr           ; (sets bank=0)
    bsf       r_status, RP0            ; *** bank=1
    decf      r_ee_buf_cnt, f          ; decrement count
    incf      r_ee_buf_adr, f          ; increment ee pointer
    incf      r_ee_buf_ptr, f          ; increment buffer pointer
    goto      ee_write_buf_x           ;

ee_write_buf_err:                       
    bsf       flag_ee_err              ;

ee_write_buf_done:                      
    bcf       flag_ee_busy             ;
    bcf       flag_ee_rq               ;

ee_write_buf_x:                         
    bcf       r_status, RP1            ;
    bcf       r_status, RP0            ;
    return                             ;
```

;---------------------------------------
;--- checksum
;
call:
    fsr  pointer to buffer
    (r_status[IRF] should be set appropriately)
    w    count
uses:
    r_temp_1
return:
    fsr  points to buffer[count]
    w    checksum
chksum:
    movwf r_tempc_1    ;
    movlw .0          ;
chksum_loop:                    ;
    addwf r_indf, w    ; add data byte
    incf r_fsr, f     ; increment pointer
    decfsz r_tempc_1, f ; loop ...
    goto chksum_loop   ;
    return              ;
chksum_loop:
Thermistor Temperature Processing (Subroutine of ADC_Service)

As described in the functional descriptions, the temperature measurement of the A/D converter uses a linearization scheme composed of a look-up table of line equations. This routine uses the look-up tables to piecewise linearly interpolate the temperature reading.

```assembly
;---------------------------------------
;--- thermistor temperature index
;  call:          
;  w         key  
;  exit:         
;  w         ram location  
;  r_temp_3  vector length (limited)  
;  r_temp_4  key
;---------------------------------------
therm_index:                            
  movwf     r_temp_4                 ; save key

;--- read vector length
  movlw     EE_T_LUT_N               
  call      ee_read_waddr            ; read LUT size (N)
  addlw     0xff                     ; temperature axis length = N-1
  movwf     r_temp_3                 ; vector length
  andlw     0x07                     ; limit vector length

;--- setup read ee to buffer
  movwf     r_temp_1                 
  movlw     EE_T_LUT_T               
  call      ee_read_buf              ; read temperature vector into ram

;--- index into temperature vector
  goto      lut_index_buf            

;-------------------------------------------------
;--- read ee data into scratch buffer "buf"
;  call:          
;  w         ee address  
;  r_temp_1  length (byte count)  
;-------------------------------------------------
ee_read_buf:                            
  movwf     r_ee_addr                ; save ee address
  ee_read_buf_:                           
  movlw     r_buf2                   
  movwf     r_fsr                    ; load RAM pointer
  goto      move_ee_ram_             ; read ee into ram buffer

;---------------------------------------
;--- lut index
;  call:          
;  w         ram location  
;  r_temp_3  vector length  
;  r_temp_4  search key  
;  return:  
;  w         vector index
;---------------------------------------
lut_index_buf:                          
  movlw     r_buf2                   
```
lut_index:
    ;
    movwf    r_far
    ;
    movf     r_temp_3, w
    ;
    movwf    r_temp_2
    ;
lut_index_loop:
    ;
    movf     r_temp_4, w
    ;
    subwf    r_indf, w
    ;
    btfsc    r_status, C
    ;
    goto     lut_index_x
    ;
    incf     r_far, f
    ;
    decfsz   r_temp_2, f
    ;
    goto     lut_index_loop
    ;
lut_index_x:
    ;
    movf     r_temp_2, w
    ;
    subwf    r_temp_3, w
    ;
    return
    ;
;---------------------------------------
Communication

This set of subroutines implements all the communication functions: enabling or disabling communication depending on start-up state of the communication GPIO, and receiving and transmitting the data.

```assembly
;-----------------------------------------------------------
;--- comm_on() - enable communication
;-----------------------------------------------------------
comm_on:
    bsf flag_comm_active

comm_reset:
    bcf r_intcon, PEIE
    bcf r_intcon, RAIE
    clrf r_comm_flags
    goto comm_rcv_byte_prep

;-----------------------------------------------------------
;--- comm_off() - disable communication
;-----------------------------------------------------------
comm_off:
    bcf r_intcon, RAIE
    bcf flag_comm_active
    call comm_pin_output_lo
    return

;-----------------------------------------------------------
;--- comm: receive byte prep
;-----------------------------------------------------------
comm_rcv_byte_prep:
    call comm_pin_input
    movlw .8
    movwf r_comm_count
    bcf flag_comm_xmit
    bsf r_intcon, RAIE
    return

;-----------------------------------------------------------
;--- comm: timer interrupt service
;-----------------------------------------------------------
comm_isr:
    btfsb flag_comm_xmit
    goto comm_isr_xmit

comm_isr_rcv:
    btfsb flag_comm_timeout
    goto comm_reset ; break!
    btfsb flag_comm_pin
    goto comm_isr_rcv_1
    bcf flag_comm_bit
    movlw TIME_COMM_BREAK_T
    call comm_timer_load
    goto comm_isr_x

comm_isr_rcv_1:
    btfsb flag_comm_bit
    goto comm_isr_x
    call comm_timer_off
    bcf r_tmr1l, w
    sublw TIME_COMM_BREAK_T
    movwf r_tempi_1
    sublw TIME_COMM_1_MAX_T
    btfsc r_status, C
```
goto      comm_isr_rcv_bit         ;
movlw     TIME_COMM_0_MAX_T        ;
subwf     r_tempi_1, w             ;
btfsc     r_status, C              ;
goto      comm_reset               ; break!

comm_isr_rcv_bit:                       ;
    rrf       r_comm_data, f           ;
decfsz    r_comm_count, f          ;
goto      comm_isr_x             ;

movf      r_comm_data, w           ;
btfsc     flag_comm_cmnd           ;
goto      comm_isr_rcv_data        ;

comm_isr_rcv_cmnd:                      ;
    bsf       flag_comm_cmnd           ;
movwf     r_comm_data_cmnd         ;
btfsc     r_comm_data, 7           ;
goto      comm_rcv_byte_prep       ;

comm_isr_reg_write:                     ;
    movlw     0x07                     ;
exorwf     r_comm_data_cmnd, w      ;
andlw     0x7f                      ;
btfss     r_status, Z              ;
goto      comm_isr_reg_write_      ;

comm_isr_reg_write_:                    ;
movf      r_comm_data, w           ;
movwf     r_indf                   ;
ibtfsc     r_comm_data_cmnd, 2      ;
incf      r_comm_reg_0, f          ;
goto      comm_rcv_byte_prep       ;
comm_isr_reg_read_ia:
    btfsc flag0_creg_membank_ee
    goto comm_isr_reg_read_ee
    movf r_indf, w
    movwf r_comm_data

comm_isr_reg_read_ia:
    btfsc flag0_creg_membank_ee
    goto comm_isr_reg_read_ee
    movf r_fsr, w
    call ee_read_waddr
    goto comm_isr_reg_read_ia_

comm_isr_xmit:
    btfss flag_comm_H2L
    goto comm_isr_xmit_hi

    ;--- comm xmit bit - set pin lo
    comm_isr_xmit_lo:
        call comm_pin_lo
        movlw TIME_COMM_B1_LO_T
        btfss r_comm_data, 0
        movlw TIME_COMM_B0_LO_T
        call comm_timer_load
        bcf flag_comm_H2L
        goto comm_isr_x

    ;--- comm xmit bit - set pin hi
    comm_isr_xmit_hi:
        call comm_pin_hover
        decfsz r_comm_count, f
        goto $+2
        goto comm_isr_x_

    return

;---------------------------------------
;--- comm timer load
;---------------------------------------

comm_xmit_byte_prep:
    movwf r_comm_data
    movlw .8
    movwf r_comm_count
    bsf flag_comm_xmit
    bsf flag_comm_H2L
    call comm_timer_load
    movlw TIME_COMM_REPLY_T
    call comm_timer_load
    return

;---------------------------------------
;--- comm: transmit byte prep
;---------------------------------------
comm_timer_load:                        ;
    bcf       r_t1con, TMR1ON          ; timer off
    movwf     r_tmr1l                  ; load timer
    comf      r_tmr1l, f               ;
    movlw     0xff                     ;
    movwf     r_tmr1h                  ;
    bsf       r_t1con, TMR1ON          ; timer on
    bcf       r_pir1, TMR1IF           ; clear timer interrupt flag
    bsf       r_status, RP0            ; *** bank=1
    bsf       r_pie1, TMR1IE           ; enable timer interrupt
    bcf       r_status, RP0            ; *** bank=0
    bsf       r_intcon, PEIE           ; enable peripheral interrupt(s)
    return                             ;

comm_timer_off:                        ;
    bcf       r_t1con, TMR1ON          ;
    return                             ;

;---------------------------------------
;--- set comm pin to output
;---------------------------------------

comm_pin_output_lo:                     ;
    bcf       p_comm                   ;
    goto      comm_pin_output_         ;

comm_pin_output:                        ;
    btfss     flag_comm_active         ;
    return                             ;
    bsf       r_port_a, 5              ;

comm_pin_output_:                       ;
    bsf       r_status, RP0            ; *** bank=1
    bcf       r_tris_a, 5              ;
    bcf       r_status, RP0            ; *** bank=0
    return                             ;

;---------------------------------------
;--- set comm pin to input
;---------------------------------------

comm_pin_input:                         ;
    btfss     flag_comm_active         ;
    return                             ;

comm_pin_input_:                        ;
    bsf       r_status, RP0            ; *** bank=1
    bcf       r_tris_a, 5              ;
    bcf       r_status, RP0            ; *** bank=0
    return                             ;

;---------------------------------------
;--- set comm pin lo
;---------------------------------------

comm_pin_lo:                            ;
    bcf       p_comm                   ;
    return                             ;

;---------------------------------------
;--- set comm pin hi
;---------------------------------------

comm_pin_hi:                            ;
    bsf       p_comm                   ;
    return                             ;

;---------------------------------------
LED Service

Two GPIOs can be used to perform charge state feedback with an LED display. Each LED can be programmed to be on, off or flashing, and the on/off/flash times can be programmed using the parameters described in the functional description. This routine uses the charge state and the LED configuration parameters to drive the GPIO to control the LED properly.

```assembly
;-------------------------------------------------
;--- led_init_1() - configureinitialize LED1
;-------------------------------------------------
led_init_1:                             ;
    movwf     r_led_config_1           ;
    clrf      r_led_contrl_1           ;
    bsf       r_led_contrl_1, 7        ; start "off" (comment to start "on")
    return                             ;

;-------------------------------------------------
;--- led_init_2() - configureinitialize LED2
;-------------------------------------------------
led_init_2:                             ;
    movwf     r_led_config_2           ;
    clrf      r_led_contrl_2           ;
    bsf       r_led_contrl_2, 7        ; start "off" (comment to start "on")
    return                             ;

;-----------------------------------------------------------
;--- led service
; configuration byte
; [7,3] - mode
; [6:4] - on time, count
; [2:0] - off time
; operations byte
; [7:7] - led on
; [6:4] - count
; [3:0] - timer
;-----------------------------------------------------------
led_svc:                                ;
    btfss     flag_led_timer           ;
    goto      led_svc_x                ;
    bcf       flag_led_timer           ;

;--- led 1
led_svc_1:                              ;
    movlw     r_led_config_1           ;
    movwf     r_fsr                    ;
    call      led_svc_modex            ;
    btfss     r_indf, 7                ;
    bcf       p_led_1                  ;
    btfsc     r_indf, 7                ;
    bsf       p_led_1                  ;

;--- led 2
led_svc_2:                              ;
    btfsc     flag_comm_active         ;
    goto      led_svc_2_x              ;
    movlw     r_led_config_2           ;
    movwf     r_fsr                    ;
    call      led_svc_modex            ;
    btfss     r_indf, 7                ;
    bcf       p_led_2                  ;
    btfsc     r_indf, 7                ;
    bsf       p_led_2                  ;
```
led_svc_2_x:                           

   return                             

;---------------------------------------
;--- led service: mode x
;---------------------------------------

led_svc_modex:                          

   btfss   r_indf, 7                
   goto   led_svc_modex_            
   btfss   r_indf, 3                
   goto   led_svc_mode2            
   goto   led_svc_mode3            

led_svc_modex_:                          

   btfsc   r_indf, 3                
   goto   led_svc_mode1            
   incf   r_fsr, f                 

;---------------------------------------
;--- led service: mode 0
;---------------------------------------

led_svc_mode0:                          

   incf   r_fsr, f                 
   bcf    r_indf, 7                
   goto   led_svc_x                

;---------------------------------------
;--- led service: mode 1
;---------------------------------------

led_svc_model1:                          

   incf   r_far,f                  ; timer==0 ?
   movlw   0x0f                     
   andwf   r_indf, w                
   btffs   r_status, Z              
   goto   led_svc_dec_x            ; --- no, decrement & exit ...
   btffs   r_indf, 7                ; led currently on ?
   goto   led_svc_mode1_off        ; --- no, jump ...

led_svc_model1_on:                       

   movf   r_indf, w                
   andlw   0x70                     
   btffs   r_status, Z              
   goto   led_svc_mode3_on         ; blink count<>0, go load timer ...

   clrf   r_indf                   ; load long *off* time
   decf   r_far, f                 
   movf   r_indf, w                
   andlw   0x07                     
   incf   r_far, f                 
   movwf   r_indf                   
   incf   r_indf, f                 
   movwf   r_indf                   
   bcf    r_status, C              
   rlff   r_indf, f                 
   bcf    r_status, C              
   rlff   r_indf, f                 
   addwf   r_indf, f                
   movlw   0xf0                     
   andwf   r_indf, w                
   movlw   0x00                     
   btffs   r_status, Z              
   movw    r_indf                   
   goto   led_svc_x                ;
led_svc_mode1_off:
  bsf   r_indf, 7
  movf  r_indf, w
  andlw 0x70
  btfss r_status, Z
  goto   led_svc_mode1_off_
  decf  r_fsr, f
  movf  r_indf, w
  andlw 0x70
  incf  r_fsr, f
  iorwf r_indf, f
  goto   led_svc_mode3_on_
led_svc_mode1_off_
  movlw 0x10
  subwf  r_indf, f
  goto   led_svc_mode3_on_

;---------------------------------------
;--- led service: mode 2
;---------------------------------------
led_svc_mode2:
  incf  r_fsr, f
  bsf   r_indf, 7
  goto   led_svc_x

;---------------------------------------
;--- led svc mode: 3
;---------------------------------------
led_svc_mode3:
  incf  r_fsr,f
  movlw 0X0f
  andwf r_indf, w
  btfss r_status, Z
  goto   led_svc_dec_x
  btfss r_indf, 7
  goto   led_svc_mode3_off
  led_svc_mode3_on:
  bcf   r_indf, 7
  decf  r_fsr, f
  movf  r_indf, w
  goto   led_svc_mode3_
led_svc_mode3_on_
  decf  r_fsr, f
  swaf  r_indf, w
  andlw 0x07
  led_svc_mode3_
  incf  r_fsr, f
  iorwf r_indf, f
  goto   led_svc_x
led_svc_mode3_off:
  bsf   r_indf, 7
  decf  r_fsr, f
  point to config register
  led_svc_mode3:
  movf  r_indf, w
  goto   led_svc_mode3_
led_svc_mode3_
  decf  r_fsr, f
  movf  r_indf, w
  goto   led_svc_x
led_svc_dec_x:
  decf  r_indf, f

;---
led_svc_x:
  return

;-----------------------------------------------------------
Timer Service

This routine maintains the timers that are used for various firmware purposes, including charge control limits.

;-----------------------------------------------------------
;--- timer service
;-----------------------------------------------------------
timer_svc:
    btfss flag_timer_0
    goto timer_svc_x
    bcf flag_timer_0

;--- class "a" timer
timer_svc_a1:
    incf r_timer_a1, f
    movlw TIMER_A1_TA
    subwf r_timer_a1, w
    btfss r_status, C
    goto timer_svc_a1_x
    clrf r_timer_a1
    bsf flag_reg_timer

    timer_svc_a1_x:

;--- class "b" timer
timer_svc_b:
    incf r_timer_b, f
    movlw TIMER_B_TA
    subwf r_timer_b, w
    btfss r_status, C
    goto timer_svc_x
    clrf r_timer_b
    bsf flag_led_timer
    bsf flag_chg_state_timer

    timer_svc_b_x:

;--- class "c" timer
timer_svc_c:
    incf r_timer_c, f
    movlw TIMER_C_TB
    subwf r_timer_c, w
    btfss r_status, C
    goto timer_svc_x
    clrf r_timer_c

    timer_svc_c_x:

;--- class "d" timer
timer_svc_d1:
    incf r_timer_d, f
    movlw TIMER_D_TC
    subwf r_timer_d, w
    btfss r_status, C
    goto timer_svc_x
    clrf r_timer_d

    timer_svc_d1_x:
timer_svc_d2:                           
  movf  r_timer_d2, f                  
  btfsc r_status, Z                   
  goto  timer_svc_d2_x                 
  decf  r_timer_d2, f                 
  btfsc r_status, Z                   
  bsf   flag_chg_ti2_done             
timer_svc_d2_x:                         

timer_svc_x:                            
  return                                
;-----------------------------------------------------------
EEPROM Read and Write

These are general purpose routines to move data between EEPROM and RAM.

;-----------------------------------------------------------
;--- move_ee_ram() - move data from EEPROM to RAM
;    call:
;      r_fsr          ram address
;      r_eeprom_addr  EEPROM address
;      w              byte count
;    uses:
;      r_temp_1
;    return:
;      n/a
;-----------------------------------------------------------
move_ee_ram:                            
movwf     r_temp_1                 
move_ee_ram_:                           
move_ee_ram_loop:                       
call      ee_read_ia               
movwf     r_indf                   
incf      r_fsr, f                 
decfsz    r_temp_1, f              
goto      move_ee_ram_loop         
return                             ;-----------------------------------------------------------
;--- EEPROM Read
;-----------------------------------------------------------
ee_read_waddr_ia:                       
movwf     r_eeprom_addr            ; save address
ee_read_ia:                              
call      ee_read                  
incf      r_eeprom_addr, f          
return                             

ee_read:                                
bcf       r_status, RP1            ; *** bank=1
bsf       r_status, RP0            
movf      r_eeprom_addr, w         ; load address
movwf     r_eeadr                  
bsf       r_eecon1, RD             ;
movf      r_eedata, w              ; read data into WREG
bcf       r_status, RP0            ; *** bank=0
movwf     r_ee_data                
return                             ;-----------------------------------------------------------
;--- EEPROM Write
;-----------------------------------------------------------
ee_write_wdata_ia:                      
movwf     r_ee_data                ; save data
call      ee_write                 
incf      r_eeprom_addr, f          
return                             
ee_write_waddr:                         
movwf     r_eeprom_addr            ; save address
ee_write:                                
    bcf    r_status, RP1                ; *** bank=1
    bsf    r_status, RP0
    btfsc  r_eecon1, WR                ; wait for write in progress
    goto   §-1
    movf   r_ee_addr, w
    movwf   r_eeadr
    movf   r_ee_data, w
    movwf   r_eedata
    bsf    r_eecon1, WREN
    bcf    r_intcon, GIE
    movlw   0x55
    movwf   r_eecon2
    movlw   0xaa
    movwf   r_eecon2
    bsf    r_eecon1, WR
    bcf    r_intcon, GIE
    bcf    r_eecon1, WREN
    bcf    r_status, RP0                ; *** bank=0
    return

;-----------------------------------------------------------
ADC Service

This routine receives the raw A/D data for voltage, current, temperature and BATID and calibrates and converts it to a form usable by the algorithm, as described in the calibration section in the firmware descriptions.

```assembly
;-----------------------------------------------------------
;--- adc_svc() - service adc conversion
;-----------------------------------------------------------
adc_svc:                                
  btfss     r_adc_control, 7         ; conversion started ?
  goto      adc_svc_x                ; --- no,  exit ...
  btfsc     r_adcon0, GO             ; conversion complete ?
  goto      adc_svc_x                ; --- no,  exit ...
  bcf       r_adc_control, 7         ; clear "conversion started" flag

;---------------------------------------
;--- fetch "raw" result
;---------------------------------------
  bsf       r_status, RP0            ; *** bank=1
  movf      r_adresl, w              
  bcf       r_status, RP0            ; *** bank=0
  movwf     r_adc_raw_L              
  movf      r_adresh, w              
  movwf     r_adc_raw_H              ; load raw data
  movlw     r_adc_raw_L              ;
  call      math_load_B              ; accB = raw
  movlw     r_adc_accum              ; accA = running accum
  call      math_add_16_load_A       ; accB = raw + accum
  movlw     r_adc_accum              ;
  call      math_move_B              ; move result to register: accum
  incf      r_adc_accum_count, f     ;
  btfss     r_adc_accum_count, 4     ; accum complete ?
  goto      adc_svc_x                ; --- no,  exit ...
  movlw     .4                       ;
  call      math_shift_BC            ; accB = accum / count = avg
  movlw     r_adc_avg                ;
  call      math_move_D              ; save result

;---------------------------------------
;--- process raw data
;---------------------------------------
adc_svc_0:                                
  btfss     r_adc_control, 0         ;
  goto      adc_svc_0_x              ;
  bcf       r_adc_control, 0         ;

;debug
;  movlw     r_adc_avg_shadow         ;
;  call      math_move_D              ; save result
```
move  low .16384         ;
movf  r_accA_L         ;
movlw  high .16384     ;
movf  r_accA_H         ;
call  math_mul_16_prep_ ; accBC = result

movlw  r_accB_L         ;
call  math_load_A       ;
movlw  r_accC_L         ;
call  math_load_B       ;
movlw  EE_CAL_ADC_0     ;
call  ee_read_waddr_ia  ;
movf  r_accC_L         ;
call  ee_read_ia        ;
movf  r_accC_H         ;
call  math_div_32       ;
movf  r_accB_L, w       ;
movf  r_accB_H, w       ;
movf  r_accB_H         ;

adc_svc_0_x:                       
goto  adc_svc_x          

goto  adc_svc_0_x

;---------------------
;--- adc_1: current
;---------------------
adc_svc_1:

  btfss  r_adc_control, 1     
  goto  adc_svc_1_x          
  bcf   r_adc_control, 1     
  call  adc_refcal           

;debug
;  movlw  r_adc_avg_shadow 
;  call  math_move_D         ; save result

movlw  EE_CAL_ADC_1          
call  load_A_ee             ; accA = cal factor
call  math_mul_16_prep_      ; accBC = result
movlw  .2                   
call  math_shift_BC         ; accB_L,accC_H = result/1024

movlw  r_accC_H             
call  math_load_A           ; accA = result

;--- remove offset (option)
adc_svc_1_cofs:             
  clrf   r_accB_L            
  clrf   r_accB_H            
  btfsc  flag0_mode_cofs_dis 
  goto  adc_svc_1_cofs_x    
  movlw  r_adc_1_ofs        
  call  math_load_B         ; accB = -offset
  clrf   r_accB_H            
  call  math_neg_B           
  call  math_add_16         ; accA + accB = result - offset
  btfss  r_accB_H, 7         ; negative ?
  goto  adc_svc_1_cofs_x    ; --- no, skip ...
  clrf   r_accB_L            ; --- yes, make zero
  clrf   r_accB_H            

adc_svc_1_l_cofs_x:
    movlw r_adc_1_L
    call math_move_B
    goto adc_svc_x


adc_svc_1_x:

;------------------
;--- adc_2: voltage
;------------------
adc_svc_2:
    btfss r_adc_control, 2
    goto adc_svc_2_x
    bcf r_adc_control, 2

;debug
;    movlw r_adc_avg_shadow
;    call math_move_D

; debug
    movlw ER_CAL_ADC_2
    btfsc flag0_mode_nm
    movlw ER_CAL_ADC_2_NM
    call load_A_ee
    call math_mul_16_prep
    movlw .2
    call math_shift_BC
    movf r_accC_H, w
    movwf r_adc_2_L
    movf r_accB_L, w
    movwf r_adc_2_H
    goto adc_svc_x

adc_svc_2_x:

;------------------
;--- adc_3: temperature
;------------------
adc_svc_3:
    btfss r_adc_control, 3
    goto adc_svc_3_x
    bcf r_adc_control, 3

adc_svc_3_sim:
    btfsc flag_adc_3_sim
    goto adc_svc_3_x

adc_svc_3_sim_x:

adc_svc_3_k:
    btfss flag0_mode_temp_k
    goto adc_svc_3_k_x
    constant temperature option
    movlw ER_T_DEFAULT-1
    call load_B_ee
    clrf r_accB_L
    accB_L = temp param
    movlw .6
    call math_shift_BC
    accB /= 64 (i.e. param * 4)
    goto adc_svc_3_B

adc_svc_3_k_x:

;debug
;    movlw r_adc_avg_shadow
;    call math_move_D

; debug
    movlw ER_CAL_ADC_3
    call load_A_ee
    call math_mul_16_prep
    accB,C = TSCALE = RAW * CF
    movlw .5

call    math_shift_BC            ;
movlw   r_accC_H                 ;
call    math_load_D              ; accD = TSCALE' = TSCALE/8192
movlw   .2                        ;
call    math_shift_BC            ;
movf    r_accC_H, w              ; W = TSCALE' / 4
movf    r_accC_H, w
movf    r_accC_H, w
bcf     r_status, C              ;
rlf     r_ee_addr, f             ;
rlf     r_ee_addr, f             ;
movlw   EE_T_LUT_MB              ;
addwf   r_ee_addr, f             ;
call    ee_read_ia               ;
movwf   r_accB_L                 ;
call    ee_read_ia               ;
movwf   r_accB_H                 ;
bcf     flag_math_temp           ;
btfss   r_accB_H, 7              ;
goto    $+3                      ;
call    math_neg_B               ;
bsf     flag_math_temp           ;
call    math_mul_16_prep_        ; accB,C = TSCALE' * M
movlw   .5                       ;
call    math_shift_BC            ;
movf    r_accB_L, w              ;
movf    r_accB_L, w
movf    r_accB_H, w
movf    r_accB_H, w
movf    r_accB_H
movlw   EE_T_LUT_MB              ;
addwf   r_ee_addr, f             ;
call    ee_read_ia               ;
movwf   r_accA_L                 ;
call    ee_read_ia               ;
movwf   r_accA_H                 ;
call    math_add_16              ; accB = accB + yint = temperature
adc_svc_3_B:

adc_svc_3_under:
    btfss   r_accB_H, 7              ; check limit: tcode < 0
goto    adc_svc_3_under_x        ;
clrwf    r_accB_H                 ;
clrwf    r_accB_L
    goto    adc_svc_3_x_            ;
adc_svc_3_under_x:

adc_svc_3_over:
    movlw   0xFC                     ; check limit: tcode >= 1024
    andwf   r_accB_H, w              ;
btfssc   r_status, Z              ;
goto    adc_svc_3_over_x         ;
    movlw   0x03
    movwf   r_accB_H
    movlw   0x0F
    movwf   r_accB_L
    movlw   r_accB_L
adc_svc_3_over_x:
ADC Service Routine

```assembly
adc_svc_3_x:                           
    movlw r_adc_3_L                
    call math_move_B              ; save results 
    goto adc_svc_x                
adc_svc_3_x:                           

;-----------------------------
;--- adc_4: battid
;-----------------------------
adc_svc_4:                              
    btfss r_adc_control, 4         
    goto adc_svc_4_x              
    bcf r_adc_control, 4         
    ;debug
    movlw r_adc_avg_shadow         
    call math_move_D              ; save result 
    movlw .2                       
    call math_shift_BC            ; accB = result/4 (i.e. 0->255) 
    movlw r_adc_4_L                
    call math_move_B              
    bcf flag_battpres1           
    movlw EE_BATTID_MIN            
    call ee_read_waddr_ia         ; w = lower limit 
    subwf r_adc_4_L, w             
    btfss r_status, C              
    goto adc_svc_4_x_             
    call ee_read                  ; w = upper limit 
    subwf r_adc_4_L, w             
    btfss r_status, Z              
    goto adc_svc_4_ok             
    btfss r_status, Z              
    goto adc_svc_4_x_             
adc_svc_4_ok:                           
    bsf flag_battpres1           
adc_svc_4_x_:                           
    goto adc_svc_x                
adc_svc_4_x:                            

adc_svc_x:                              
    return                             

;----------------------------------------
;--- adc_refcal()
;    call:
;        accD = adc-raw
;----------------------------------------
adc_refcal:                             
    movlw r_adc_0_L                
    call math_mul16_prep         ; accBC = result 
    movlw .6                       
    call math_shift_BC            
    movf r_accB_L, w              
    movwf r_accD_H                
    movf r_accC_H, w              
    movwf r_accD_L                ; accD = adc raw corrected for vref 
    return                             
```

--- Microchip Technology Inc. 2006 ---
ADC Start

This routine monitors the A/D status to see when a new reading should be performed, then programs the A/D registers to perform the correct measurements using the correct channels and resolutions.

;------------------------------------------
;--- adc_start_reset() - reset adc scheduler
;------------------------------------------
adc_start_reset:                        
    clrf r_adc_control            
    return                         
;
;------------------------------------------
;--- adc_start() - start/initiate new conversion
;------------------------------------------
adc_start:                              
    btfsc r_adc_control, 7         ; conversion active ?
    goto adc_start_x              ; --- yes, exit ...

;------------------------------------------
;--- start conversion                     
;------------------------------------------
    movlw ADC_CHANNEL_MASK         ;
    andwf r_adc_control, w         ;
    btfsc r_status, Z              ;
    goto adc_start_done           ;
    bsf r_adc_control, 6         ;
    btfsc r_adc_control, 4         ;
    movlw ADC_ADCON0_4             ;
    btfsc r_adc_control, 3         ;
    movlw ADC_ADCON0_3             ;
    btfsc r_adc_control, 2         ;
    movlw ADC_ADCON0_2             ;
    btfsc r_adc_control, 1         ;
    movlw ADC_ADCON0_1             ;
    btfsc r_adc_control, 0         ;
    movlw ADC_ADCON0_0             ;

;--- finish initialization
adc_start_:                             
    movwf r_adcon0                 ;
    movlw ADC_TAQ                  ;
    movwf r_temp_1                 ;
    adc_start_loop:               ;
    decfsz r_temp_1, f            ;
    goto adc_start_loop           ;
    bsf r_adcon0, GO             ;
    bsf r_adc_control, 7         ;
    goto adc_start_x              ;

;--- done
adc_start_done:                         

;--- set "data ready" flag(s)
adc_start_done_0:                       
    btfss flag_adcset_0_rq        ;
    goto adc_start_done_0_x      ;
    bcf flag_adcset_0_rq         ;
    bcf flag_adcset_0_rqq        ;
    bsf flag_adcset_0_rdy        ;
ADC start done 0_x:

ADC start done 1:
  btfss flag_adcset_1_rq
  goto adc_start_done_1_x
  bcf flag_adcset_1_rq
  bcf flag_adcset_1_rqq
  bsf flag_adcset_1_rdy
adc_start_done_1_x:

ADC start done 2:
  btfss flag_adcset_2_rq
  goto adc_start_done_2_x
  bcf flag_adcset_2_rq
  bcf flag_adcset_2_rqq
  ; bsf flag_adcset_2_rdy
  bsf flag_adcset_0_rdy
adc_start_done_2_x:

ADC start new:
  movlw .0

ADC start new 0:
  btfss flag_adcset_0_rq
  goto adc_start_new_0_x
  bsf flag_adcset_0_rqq
  iorlw ADCSET_0
adc_start_new_0_x:

ADC start new 1:
  btfss flag_adcset_1_rq
  goto adc_start_new_1_x
  bsf flag_adcset_1_rqq
  iorlw ADCSET_1
adc_start_new_1_x:

ADC start new 2:
  btfss flag_adcset_2_rq
  goto adc_start_new_2_x
  bsf flag_adcset_2_rqq
  iorlw ADCSET_2
adc_start_new_2_x:

movwf r_adc_control

ADC start_x:
  return

;-----------------------------------------------------------
Charge State Service

As a result of the trigger checks, this routine enters the charger into the correct charge state, as described in the functional descriptions. Charge Suspend, Fast Charge, Trickle Charge, etc., are entered when appropriate. See Lithium and Nickel charge state descriptions in the functional description sections.

;-----------------------------------------------------------
;--- charge state service
;-----------------------------------------------------------
chg_state_svc:                        
  btfss  flag_chg_state_timer      ; service timer expired ?
  goto  chg_state_svc_x            ; --- no, exit ...
  bcf   flag_chg_state_timer      ;
  btfss  flag_adcset_0_rdy        ; adc data ready ?
  goto  chg_state_svc_x            ; --- no, exit ...
  bcf   flag_adcset_0_rdy        ;

;--- charge state service enabled ?
chg_state_on:                        
  btfsc flag0_creg_chgcon_off    ;
  goto  chg_state_svc_x            ;
chg_state_on_x:                        

;-----------------------------
;--- battery present
;-----------------------------
  bcf   flag_battpres            ;

;--- battery present - voltage min
chg_state_svc_bpv:                    
  btfss  flag0_mode_bpres_v       ;
  goto  chg_state_svc_bpv_x      ;
  movlw  EE_CHG_V_MIN_BP          ;
  call  check_voltage            ;
  btfss  r_status, C              ;
  bsf   flag_battpres            ;
chg_state_svc_bpv_x:                   

;--- battery present - force
  btfsc flag0_mode_bpres_always  ;
  bsf   flag_battpres            ;

;--- battery present - battid
chg_state_svc_bpbi:                   
  btfss  flag0_mode_bpres_battid ;
  goto  chg_state_svc_bpbi_x     ;
  btfsc flag_battpres            ;
  bsf   flag_battpres            ;
chg_state_svc_bpbi_x:                   

call  checkchg_timer_a    ;
  btfss  r_status, Z            ;
  goto  chg_state_svc_x        ;
  movf  r_chg_state, f         ;
  btfss  r_status, Z            ;
  goto  chg_state_svc_x        ;

chg_state_svc_bp:                      
  btfsc flag_battpres            ; battery present ?
  goto  chg_state_svc_bp_x      ; --- yes, skip ...
  movf  r_chg_state, f         ;
  btfss  r_status, Z            ;
  goto  chg_state_0_init        ; --- no, go initialize state 0
chg_state_svc_bp_x:                      

call  checkchg_timer_a    ;
  btfss  r_status, Z            ;
  goto  chg_state_svc_x        ;
  goto  chg_state_svc_jumptable ;
;---------------------------
;--- charge state: 0 - reset
;---------------------------
chg_state_0:
    ;
call      reg_off
    ;
btfsc     flag_battpres
    ;
goto      chg_state_1_init
    ;
chg_state_0_x:
    ;
goto      chg_state_x
    ;

;---------------------------
;--- charge state: 1 - charge qualification
;---------------------------

call      reg_off
chg_state_1:
    ;
btfss     flag0_mode_nm
    ;
goto      chg_state_1_lion
    ;
chg_state_1_nimh:
    ;

;--- discharge enabled ?

chg_state_1_nimh_dchg:
    ;
btfss     flag0_mode_nm_extdchg_en
    ;
goto      chg_state_1_nimh_dchg
    ;
btfsc     flag_v_le_dchg_nm
    ;
goto      chg_state_1_nimh_dchg_x
    ;
bcf       p_gpio
    ;
goto      chg_state_x
    ;
chg_state_1_nimh_dchg_x:
    ;
bsf       p_gpio
    ;
chg_state_1_nimh_dchg_x:
    ;

;--- max exceeded?

btfsc     flag0_mode_nm_slochg
    ;
goto      chg_state_7_init
    ;
btfss     flag_t_le_tmaxchg_nm
    ;
goto      chg_state_57_init
    ;
btfss     flag_v_le_vmaxchg_nm
    ;
goto      chg_state_57_init
    ;

;--- pchg?

btfsc     flag0_mode_pchg_always
    ;
goto      chg_state_2_init
    ;
btfsc     flag_v_le_vpchg_nm
    ;
goto      chg_state_2_init
    ;
btfsc     flag_t_le_tpchg_lo_nm
    ;
goto      chg_state_2_init
    ;
btfss     flag_t_le_tpchg_hi_nm
    ;
goto      chg_state_2_init
    ;
chg_state_2_init:
    ;
chg_state_1_lion:
    ;
;--- suspend?

btfss     flag_v_le_vmax
    ;
goto      chg_state_6_init
    ;
btfss     flag_t_le_tmin
    ;
goto      chg_state_6_init
    ;
chg_state_6_init:
    ;
btss     flag_t_le_tmaxchg
    ;
goto      chg_state_6_init
    ;
chg_state_6_init:
    ;
btss     flag_t_le_tmaxchg
;--- charge state: 2 - precharge

chg_state_2:
  btfs s flag0_mode_nm
  goto chg_state_2_lion

chg_state_2_nimh:
  btfs s flag_reg_on
  goto chg_state_0_init

;--- max exceeded?
  btfs s flag_t_le_tmaxchg_nm
  goto chg_state_57_init
  btfs s flag_v_le_vmaxchg_nm
  goto chg_state_57_init
  btfs s flag_chg_til_done
  goto chg_state_57_init

;--- min/holdoff timer expired ?
  btfs s flag_chg_ti2_done
  goto chg_state_x

;--- check pchg criteria
  btfs s flag_v_le_vpchg_nm
  goto chg_state_x
  btfs s flag_t_le_tpchg_lo_nm
  goto chg_state_x
  btfs s flag_t_le_tpchg_hi_nm
  goto chg_state_x
  goto chg_state_3_init

chg_state_2_lion:
  btfs s flag_reg_on
  goto chg_state_6_init
  btfs s flag_v_le_vmax
  goto chg_state_6_init
  btfs s flag_t_le_tmin
  goto chg_state_6_init
  btfs s flag_t_le_tmaxchg
  goto chg_state_6_init
  btfs s flag_chg_til_done
  goto chg_state_6_init

  btfs s flag_t_le_tpchg
  goto chg_state_x
  btfs s flag_v_le_vpchg
  goto chg_state_x

  goto chg_state_3_init
chg_state_3:
    btfss flag0_mode_nm
    goto chg_state_3_lion

chg_state_3_nimh:
    btfss flag_reg_on
    goto chg_state_0_init
    ;--- max exceeded?
    btfss flag_t_le_tmaxchg_nm
    goto chg_state_57_init
    btfss flag_v_le_vmaxchg_nm
    goto chg_state_57_init
    btfsb flag_chg_ti1_done
    goto chg_state_57_init
    ;--- minus delta V?
    chg_state_3_nimh_dv:
        btfsb flag0_mode_nm_eoc_dv
        goto chg_state_3_nimh_dv_x
        btfss flag_chg_ti2_done
        goto chg_state_x
        btfss r_adc_2_save+1, 7
        goto chg_state_3_nimh_dv_ok
        movf r_adc_2, w
        movsf r_adc_2_save
        movf r_adc_2+1, w
        movsf r_adc_2_save+1
        chg_state_3_nimh_dv_ok:
        call check_nm_dv
        btfsb flag_temp_1
        goto chg_state_7_init
        chg_state_3_nimh_dv_x:

    ;--- dtdt?
    chg_state_3_nimh_dtdt:
        btfsb flag0_mode_nm_eoc_dtdt
        goto chg_state_3_nimh_dtdt_x
        call check_nm_dtdt
        btfsb flag_chg_ti2_done
        goto chg_state_x
        btfsb flag_temp_1
        goto chg_state_3_nimh_dtdt_x
        btfss flag0_mode_nm_topoff_en
        goto chg_state_57_init
        goto chg_state_4_init
        chg_state_3_nimh_dtdt_x:
        goto chg_state_3_x

chg_state_3_lion:
    btfss flag_reg_on
    goto chg_state_6_init
    ;--- yes, exit to state 6 ...
    btfsb flag_v_le_vpchg
    goto chg_state_6_init
    btfss flag_v_le_vmax
    goto chg_state_6_init
    btfsb flag_t_le_tpchg
    goto chg_state_6_init
    btfsb flag_t_le_tmaxchg
    goto chg_state_6_init
    btfsb flag_chg_til_done
    goto chg_state_6_init
    btfsb flag_vreg
    
goto  chg_state_4_init;

chg_state_3_x:
  goto  chg_state_x;

;-------------------------------------------------
;--- check_nm_dtdt() - check dT/dt criteria
;    flag_temp_1=1 dtdt > threshold
;    flag_temp_1=0 dtdt negative or small positive
;-------------------------------------------------
check_nm_dtdt:
  bcf    flag_temp_1; clear flag
  call   check_chg_timer_b;
  btfss  r_status, Z;
  goto   check_nm_dtdt_x;
  movlw  r_adc_3_save_a;
  btfss  flag_chgn_tsel;
  movlw  r_adc_3_save_b;
  call   math_load_B;
  call   math_neg_B;
  movlw  r_adc_3;
  call   math_add_16_load_A; accB = deltaT
  btfsc  r_accB_H, 7; deltaT positive? (T > T_SAVE) ?
  goto   check_nm_dtdt_x;
  -- no, exit ...

  call   math_neg_B;
  movlw  EE_CHGN_T_DTDT;
  call   math_load_B;
  movlw  r_adc_3_L;
  call   math_load_B;
  movlw  r_adc_3_save_a;
  btfss  flag_chgn_tsel;
  movlw  r_adc_3_save_b;
  call   math_move_B; archive temp in ping-pong buffer
  movlw  MASK_CHGN_TSEL;
  xorwf  REG_CHGN_TSEL, f; toggle buffer selector
  movlw  EE_CHGN_TI_DTDT;
  call   ee_read_waddr;
  movwf  r_chg_timer_b; re-load timer

check_nm_dtdt_x::
  return;

;-------------------------------------------------
;--- check_nm_dv(): check minus delta V
;    flag_temp_1=1 deltaV > threshold
;    flag_temp_1=0 deltaV positive or < threshold
;-------------------------------------------------
check_nm_dv:
  bcf    flag_temp_1; clear flag
  movlw  r_adc_2_save;
  call   math_load_B;
  call   math_neg_B;
  movlw  r_adc_2;
  call   math_add_16_load_A; accB = -(adc_2_save) [i.e. -VMAX]
  call   math_neg_B;
  movlw  r_adc_2;
  btfss  flag_chgn_tsel;
  movlw  r_adc_2_save_a;
  call   math_move_B; delta V negative
  goto   check_nm_dv_newmax;

check_nm_dv_newmax::
  -- no, skip ...

---------------------------------------------------------------------
check_nm_dv_negative:                   
  movlw   EE_CHANGED_V_DV             
  call    load_A_ee                   
  call    math_add_16                 
  btfss   r_accB_H, 7                
  goto    check_nm_dv_x              
  bsf     flag_temp_1               
  goto    check_nm_dv_x              

check_nm_dv_newmax:                     
  movf    r_adc_2, w                 
  movwf   r_adc_2_save               
  movf    r_adc_2+1, w               
  movwf   r_adc_2_save+1             

check_nm_dv_x:                          
  return                             

;-----------------------------------------------------------
;--- charge state: 4
;-----------------------------------------------------------
chg_state_4:                             
  btfss   flag0_mode_nm              
  goto    chg_state_4_lion           
chg_state_4_nimh:                       
  btfss   flag_reg_on                
  goto    chg_state_0_init           
  btfsc   flag_t_le_tmaxchg_nm       
  goto    chg_state_57_init          
  btfsc   flag_v_le_vmaxchg_nm       
  goto    chg_state_57_init          
  btfsc   flag_chg_ti1_done          
  goto    chg_state_57_init          
  goto    chg_state_x                
chg_state_4_lion:                       
  btfss   flag_reg_on                
  goto    chg_state_6_init           
  btfsc   flag_v_le_vprechg          
  goto    chg_state_6_init           
  btfsc   flag_v_le_vmax             
  goto    chg_state_6_init           
  btfsc   flag_t_le_tpchg            
  goto    chg_state_6_init           
  btfsc   flag_chg_ti1_done          
  goto    chg_state_6_init           
chg_state_4_a:                           
  movlw   EE_CHANGED_C_MIN           
  call    check_current              
  btfss   r_status, C                
  goto    chg_state_4_a_1            
  call    check_chg_timer_b          
  btfsc   r_status, Z                
  goto    chg_state_6_init           
  goto    chg_state_4_a_x            

;-----------------------------------------------------------
;--- charge state: 4
;-----------------------------------------------------------
chg_state_4_lion:                       
  btfss   flag_reg_on                
  goto    chg_state_6_init           
  btfsc   flag_v_le_vprechg          
  goto    chg_state_6_init           
  btfsc   flag_v_le_vmax             
  goto    chg_state_6_init           
  btfsc   flag_t_le_tpchg            
  goto    chg_state_6_init           
  btfsc   flag_chg_ti1_done          
  goto    chg_state_6_init           
chg_state_4_a:                           
  movlw   EE_CHANGED_C_MIN           
  call    check_current              
  btfss   r_status, C                
  goto    chg_state_4_a_1            
  call    check_chg_timer_b          
  btfsc   r_status, Z                
  goto    chg_state_6_init           
  goto    chg_state_4_a_x            

;-----------------------------------------------------------
;--- charge state: 4
;-----------------------------------------------------------
chg_state_4_lion:                       
  btfss   flag_reg_on                
  goto    chg_state_6_init           
  btfsc   flag_v_le_vprechg          
  goto    chg_state_6_init           
  btfsc   flag_v_le_vmax             
  goto    chg_state_6_init           
  btfsc   flag_t_le_tpchg            
  goto    chg_state_6_init           
  btfsc   flag_chg_ti1_done          
  goto    chg_state_6_init           
chg_state_4_a:                           
  movlw   EE_CHANGED_C_MIN           
  call    check_current              
  btfss   r_status, C                
  goto    chg_state_4_a_1            
  call    check_chg_timer_b          
  btfsc   r_status, Z                
  goto    chg_state_6_init           
  goto    chg_state_4_a_x            

;-----------------------------------------------------------
;--- charge state: 4
;-----------------------------------------------------------
chg_state_4_lion:                       
  btfss   flag_reg_on                
  goto    chg_state_6_init           
  btfsc   flag_v_le_vprechg          
  goto    chg_state_6_init           
  btfsc   flag_v_le_vmax             
  goto    chg_state_6_init           
  btfsc   flag_t_le_tpchg            
  goto    chg_state_6_init           
  btfsc   flag_chg_ti1_done          
  goto    chg_state_6_init           
chg_state_4_a:                           
  movlw   EE_CHANGED_C_MIN           
  call    check_current              
  btfss   r_status, C                
  goto    chg_state_4_a_1            
  call    check_chg_timer_b          
  btfsc   r_status, Z                
  goto    chg_state_6_init           
  goto    chg_state_4_a_x            

chg_state_4_a_1:                        ;
  movlw  EE_CHG_TIME_1            ;
call   ee_read_waddr            ;
movwf  r_chg_timer_b            ;
chg_state_4_a_x:                        ;
goto   chg_state_x              ;

;-----------------------------------------------------------
;--- charge state: 5
;-----------------------------------------------------------
chg_state_5:                            ;
call   reg_off                  ;
btfss  flag0_mode_nm            ;
goto   chg_state_5_lion         ;
chg_state_5_nimh:                       ;
btfss  flag_t_le_tmaxchg_nm      ;
goto   chg_state_5_x            ;
btfss  flag_v_le_vmaxchg_nm      ;
goto   chg_state_5_x            ;
btfss  flag0_mode_nm_vrchg_en    ;
goto   chg_state_5_x            ;
btfsc  flag_v_le_rchg_nm        ;
goto   chg_state_7_init         ;
goto   chg_state_5_x            ;
chg_state_5_lion:                       ;
movlw  EE_CHG_V_RCHG            ;
call   check_voltage            ;
btfsc  r_status, C              ;
goto   chg_state_0_init         ;
chg_state_5_x:                          ;
goto   chg_state_x              ;

;-----------------------------------------------------------
;--- charge state: 6
;-----------------------------------------------------------
chg_state_6:                            ;
call   reg_off                  ;
btfss  flag0_mode_nm            ;
goto   chg_state_6_lion         ;
chg_state_6_nimh:                       ;
btfss  flag_t_le_tmaxchg_nm      ;
goto   chg_state_6_x            ;
goto   chg_state_6_x_           ;
chg_state_6_lion:                       ;
btfss  flag_v_le_vmax           ;
goto   chg_state_6_x            ;
btfsc  flag_t_le_tmin           ;
goto   chg_state_6_x            ;
btfss  flag_t_le_tmaxchg1       ;
goto   chg_state_6_x            ;
chg_state_6_x_:                         ;
goto   chg_state_0_init         ;
chg_state_6_x:                          ;
goto   chg_state_x              ;
;-----------------------------------------------------------
;--- charge state: 7
;-----------------------------------------------------------
chg_state_7:
    btfss flag_reg_on ; regulation module shutdown ?
    goto chg_state_0_init ; --- yes, exit to state 0 ...

;--- max exceeded?
    btfss flag_t_le_tmaxchg_nm
    goto chg_state_7_x2
    btfss flag_v_le_vmaxchg_nm
    goto chg_state_7_x2
    btfsc flag_chg_til1_done
    goto chg_state_7_x2
chg_state_7_x1:
    goto chg_state_x
chg_state_7_x2:
    btfsc flag0_mode_nm_exttrik_en
    bsf p_gpio
    goto chg_state_5_init

;-----------------------------------------------------------
;--- charge state exit(s) ...
;-----------------------------------------------------------
chg_state_x:
    btfss flag_reg_on ; regulation module shutdown ?
    goto chg_state_x_
    bsf r_status, RP1 ; auto shutdown ?
    movlw 1<<PWMASE
    andwf r_pwmclk, w ;
    bcf r_status, RP1
    btfss r_status, Z ; auto-shutdown trip ?
    goto chg_state_6_init ; --- yes, exit to state 6 ...
chg_state_x_

;--- exit -
chg_state_svc_x:
    bcf flag_adcset_0_rdy
    bcf flag_adcset_2_rdy
    btfss flag_reg_on
    bsf flag_adcset_0_rq
    btfsc flag_reg_on
    bsf flag_adcset_2_rq
chg_state_svc_x_
    return
;-----------------------------------------------------------
FIGURE 14: Li Ion CHARGE STATE FLOWCHART

STATE-0
CHARGE PENDING

STATE-1
CHARGE QUAL

STATE-2
PRECHARGE

STATE-3
CURRENT REG

STATE-4
VOLTAGE REG

STATE-5
EOC

STATE-6
CHARGE SUSPEND

[x0] [01] [12] [13] [16] [23] [26] [34] [36] [45] [46] [50] [60]
EXAMPLE 1: Li Ion CHARGE STATE FLOWCHART EQUATIONS

State Transition Criteria:

[12] => Transition from state 1 to state 2
[x1] => Transition from any state to state 1

[x0] = BP* or Reset
Highest priority and true for all states – for clarity, not included in all equations below.

[01] = [x0]*

[16] = (V > VMAX) or (T < TMIN) or (T > TMAXCHG)
[12] = [16]* and ((V < VPCHG) or (T < TPCHG))
[13] = [16]* and [12]*

[26] = (V > VMAX) or (T < TMIN) or (T > TMAXCHG) or (V > VSAFETY) or (TI > TIPCHG)
[23] = [26]* and (T > TPCHG) and (V > VPCHG)

[36] = (V < VPCHG) or (V > VMAX) or (T < TPCHG) or (T > TMAXCHG) or (V > VSAFETY) or (TI > TICREG)
[34] = [36]* and (V > VREG)

[46] = (V < VPCHG) or (V > VMAX) or (T < TPCHG) or (T > TMAXCHG) or (V > VSAFETY) or (TI > TIVREG)
[45] = [46]* and (C < CMIN)

[50] = [x0] or (V < VRCHG)

[60] = [x0] or ((V < VMAX) and (T > TMIN) and (T < TMAXCHG))
**Mode bit, TRICKLE_BYP, will cause STATE-7 to be bypassed, entering STATE-5 directly.
EXAMPLE 2: NICKEL CHARGE STATE SEQUENCE FLOWCHART EQUATIONS

State Transition Criteria:

[12] => Transition from state 1 to state 2
[x1] => Transition from any state to state 1

[x0] = BP* or Reset
Highest priority and true for all states – for clarity, not included in all equations below.

STATE-0
[01] = [x0]*

STATE-1
AA = (MODE_EXTDCHG_EN and (V > VdCHG))
[17] = AA* and ((V > VMAX) or (T > TMAX) or MODE_SLOCHG)
[12] = AA* and [17]* and ((T < TpCHG_LO) or (T > TpCHG_HI) or (V < VPCHG))
[13] = AA* and [17]* and [12]*

STATE-2
[27] = ((V > VMAX) or (T > TMAX) or (T1_1 > TIPCHG_MAX))
[23] = [27]* and (T2_2 > TIPCHG_MIN) and (T > TPCHG_LO) and (T < TPCHG_HI) and (V > VPCHG)

STATE-3
AA = (V > VMAX) or (T > TMAX) or (T1_1 > TIPCHG_MAX)
[37] = AA or ((T2_2 > TIPCHG_MIN) and (DTDT_DETECT and MODE_TOPOFF_EN))
[34] = AA* and (T2_2 > TIPCHG_MIN) and (DTDT_DETECT and MODE_TOPOFF_EN)

STATE-4
[47] = (V > VMAX) or (T > TMAX) or (T1_1 > TITOPOFF_MAX)

STATE-5
[57] = MODE_VRCHG_EN and (V < VMAX) and (T < TMAX) and (V < VRCHG)

STATE-7
[75] = (V > VMAX) or (T1_1 > TTTRICKLE_MAX)
FIGURE 16: NICKEL END-OF-CHARGE dT/dt METHOD FLOWCHART

DTDT: PROC

TIMER_DTDT

YES

FLAG_CHGN_SEL

YES

TEMP_DELTA = TEMP_SAVE_A - TEMP

NO

TEMP_DELTA > T_DTDT

NO

TEMP_DELTA = TEMP_SAVE_B - TEMP

YES

TIMER_D2 = TI_FCHG_MIN

NO

TIMER_D2 = TI_FCHG_MIN

TIMER_DTDT = TI_DTDT

TOGGLE FLAG_CHGN_SEL

EOC !
FIGURE 17: NICKEL END-OF-CHARGE MINUS dV METHOD FLOWCHART

DTDT: MDV

V_DELTA = V – V_SAVE

TEMP_DELTA > 0

YES

[V_DELTA] > V_MDV

YES

TIMER_D2

YES

EOC !

NO

NO

V_SAVE = V

NO

V_SAVE = V

NO

YES

TIMER_D2 = TI_FCHG_MIN

VT DTDT: MDV

V_SAVE = V
Charge State Initialization

This routine performs the initialization of variables required by each individual charge state. Each charge state will have a different set of triggers and variables that are required for exiting.

```assembly
;-------------------------------------------------
;--- charge state initialization
;-------------------------------------------------
chg_state_init:                         

;-----------------------------
;--- chg_state_0_init()
;-----------------------------
chg_state_0_init:                       
call      reg_off                  
clrf      r_adc_1_ofs              
movlw     .0                       
goto      chg_state_init_x_gpio_hi ;

;-----------------------------
;--- chg_state_1_init()
;-----------------------------
chg_state_1_init:                       
movf      r_adc_1_L, w             
movwf     r_adc_1_ofs              
movlw     .1                       
goto      chg_state_init_x_gpio_hi ;

;-----------------------------
;--- chg_state_2_init()
;-----------------------------
chg_state_2_init:                       
btfss     flag0_mode_nm            
goto      chg_state_2_init_lion    
chg_state_2_init_nimh:                  
movlw     EE_CHGN_C_PCHG           
call      reg_load_nimh            
movlw     EE_CHGN_TI_PCHG_MX       
call      load_timer_d1_ee         
movlw     EE_CHGN_TI_PCHG_MN       
call      load_timer_d2_ee         
goto      chg_state_2_init_x       ;

chg_state_2_init_lion:                  
movlw     EE_CHG_C_PCHG            
call      reg_load_lion            
movlw     EE_CHG_TI_PCHG           
call      load_timer_d1_ee         

chg_state_2_init_x:                     
movlw     .2                       
goto      chg_state_init_x_gpio_lo ;

;-----------------------------
;--- chg_state_3_init()
;-----------------------------
chg_state_3_init:                       
btfss     flag0_mode_nm            
goto      chg_state_3_init_lion    
```

© 2006 Microchip Technology Inc.
chg_state_3_init_nimh:                  ;
    bsf       r_adc_2_save + 1, 7      ; init voltage save register
    movlw     r_adc_3                  ;
    call      math_load_B              ;
    movlw     r_adc_3_save_a           ; init temperature save register a
    call      math_move_B              ;
    movlw     r_adc_3_save_b           ; init temperature save register b
    call      math_move_B              ;
    bcf       flag_chgn_tsel           ; init temperature save reg select
    movlw     EE_CHGN_TI_FCHG_MX       ; load max/timeout timer
    call      load_timer_d1_ee         ;
    movlw     EE_CHGN_TI_FCHG_MN       ; load min/holdoff timer
    call      load_timer_d2_ee         ;
    movlw     EE_CHGN_C_FCHG           ;
    call      reg_load_nimh            ;
    goto      chg_state_3_init_        ;

chg_state_3_init_lion:                  ;
    call      reg_load_lion            ;
    movlw     EE_CHG_TI_CREG           ;
    call      load_timer_d1_ee         ;

chg_state_3_init_:                      ;
    movlw     .3                       ;
    goto      chg_state_init_x_gpio_lo ;

;-----------------------------
;--- chg_state_4_init()
;-----------------------------

chg_state_4_init:                       ;
    btfss     flag0_mode_nm            ;
    goto      chg_state_4_init_lion    ;

chg_state_4_init_nimh:                  ;
    movlw     EE_CHGN_C_TOPOFF         ;
    call      reg_load_nimh            ;
    movlw     EE_CHGN_TI_TOPOFF        ;
    goto      chg_state_4_init_x       ;

chg_state_4_init_lion:                  ;
    call      reg_load_lion            ;
    movlw     EE_CHG_TI_VREG           ;
    call      load_timer_d1_ee         ;

chg_state_4_init_x:                     ;
    call      load_timer_d1_ee         ;
    movlw     .4                       ;
    goto      chg_state_init_x_gpio_lo ;

;-----------------------------
;--- chg_state_57_init()
; i.e. 5 or 7
;-----------------------------

chg_state_57_init:                       ;
    btfss     flag0_mode_nm_trik_byp   ;
    goto      chg_state_7_init         ;

;-----------------------------
;--- chg_state_5_init()
;-----------------------------

chg_state_5_init:                       ;
    call      reg_off                  ;
    movlw     .5                       ;
    goto      chg_state_init_x_GPIO_hi ;
Check Triggers
This routine checks all the triggers that are required to exit the current charge state and enter into a different one. The triggers, variables, and equations for each state are described in the functional description.

load_timer_d1() - load timer and reset pre-scaler

load_timer_d1: ;
call eereadwaddr ;
clr r_timer_d ;
bcf flag_chg_til_done ;
return ;

load_timer_d1_ee: ;
call eereadwaddr ;
load_timer_d1: ;
movf t_timer_d ;
clr r_timer_d ;
bcf flag_chg_til_done ;
return ;

;-----------------------
;--- chg_state_6_init()
;-----------------------
chg_state_6_init: ;
call reg_off ;
movlw .6 ;
goto chg_state_init_x_gpio_hi ;

;-----------------------
;--- chg_state_7_init()
;-----------------------
chg_state_7_init: ;
movlw EE_CHGN_C_TRIK ;
call reg_load_nimh ;
movlw EE_CHGN_TI_TRIK ;
call load_timer_d1_ee ;
btfsc flag0_mode_nm_exttrik_en ;
bcf p_gpio ;
movlw .7 ;
goto chg_state_init_x_gpio_lo ;

;-----------------------
chc_state_init_x_gpio_hi: ;
btfsc flag0_mode_gpio_cutoff ;
bsf p_gpio ;
goto chg_state_init_x ;
chc_state_init_x_gpio_lo: ;
btfsc flag0_mode_gpio_cutoff ;
bcf p_gpio ;
chc_state_init_x_: ;
movwf r_chg_state ; ; save state
movf r_chg_state, w ; ; configure led1
addlw EE_LED1_CFG ;
call eereadwaddr ;
call led_init_1 ;
movf r_chg_state, w ; ; configure led2
addlw EE_LED2_CFG ;
call eereadwaddr ;
call led_init_2 ;
movlw EE_CHG_TIME_0 ;
call eereadwaddr ; w = theshold from ee
movlw r_chg_timer_a ;

return ;

;-----------------------------------------------------------
Check Triggers
This routine checks all the triggers that are required to exit the current charge state and enter into a different one. The triggers, variables, and equations for each state are described in the functional description.

;-----------------------
;--- load_timer_d1() - load timer and reset pre-scaler
;-----------------------
load_timer_d1_ee: ;
call eereadwaddr ;
load_timer_d1: ;
movf t_timer_d1 ;
clr r_timer_d ;
bcf flag_chg_til_done ;
return ;
--- load_timer_d2() - load timer and reset pre-scaler
; used as a "holdoff" or "minimum" timeout; therefore,
; if setpoint==0, "done" flag set immediately
; if setpoint>0, "done" flag set when timer expires
;-----------------------------------------------------------
load_timer_d2_ee:                       ;
call      ee_read_waddr            ;
load_timer_d2:                          ;
movwf     r_timer_d2               ;
clf       r_timer_d                ;
bcf       flag_chg_ti2_done        ;
movf      r_timer_d2, f            ;
btfsc     r_status, Z              ;
bsf       flag_chg_ti2_done        ; set "done" if ==0
return                             ;
;---------------------------------------
;--- check timer a
;--- check timer b
;--- check timer c
;--- check timer d
; status[Z]=1 if timer=0
;---------------------------------------
check_chg_timer_a:                      ;
movlw     r_chg_timer_a            ;
goto      check_chg_timer_         ;
check_chg_timer_b:                      ;
movlw     r_chg_timer_b            ;
goto      check_chg_timer_         ;
check_chg_timer_c:                      ;
movlw     r_chg_timer_c            ;
goto      check_chg_timer_         ;
check_chg_timer_d:                      ;
movlw     r_chg_timer_d            ;
goto      check_chg_timer_         ;
check_chg_timer_ :                       ;
movwf     r_fsr                    ;
movf      r_indf, f                ;
btfss     r_status, Z              ;
decfsz    r_indf, f                ;
nop                                ;
return                             ;
;-----------------------------------------------------------
;--- check_current() - compare current to threshold
;
; call:
;     w    ee address of threshold
; uses:
;     r_accB_L
;     r_accB_H
;     r_fsr
; return:
;     r_status[C]=1  current <= threshold
;     r_status[C]=0  current > threshold
;-----------------------------------------------------------
check_current:                          ;
call      load_B_ee                ;
check_current_:                          ;
movlw     r_adc_1_H                ;
goto      math_cmp_16               ;
;--- check_voltage() - compare voltage to threshold
;
;    call:
;      w ee address of theshold
;    uses:
;      r_accB_L
;      r_accB_H
;      r_fsr
;    return:
;      r_status[C]=1  voltage <= threshold
;      r_status[C]=0  voltage > threshold
;-----------------------------------------------------------
check_voltage:                          
call      load_B_ee                
movlw     r_adc_2_H                
goto      math_cmp_16              

;-----------------------------------------------------------
;--- check_temperature() - compare temperature to threshold
;
;    call:
;      w ee address of theshold
;    uses:
;      r_accB_L
;      r_accB_H
;      r_fsr
;    return:
;      r_status[C]=1  temperature <= threshold
;      r_status[C]=0  temperature > threshold
;-----------------------------------------------------------
check_temperature:                      
  addlw     -.1                      ; accB_H = temp param
  call      load_B_ee                
  clrf      r_accB_L                 
  movlw     .6                       ;
  call      math_shift_BC            ; accB = temp param * 4
  movlw     r_adc_3_H                
goto      math_cmp_16              

;-----------------------------------------------------------
;--- load_A_ee() - load accA from EE
;-----------------------------------------------------------
load_A_ee:                              
call      ee_read_waddr_ia         
movwf     r_accA_L                 
call      ee_read_ia               
movwf     r_accA_H                 
return                             

;-----------------------------------------------------------
;--- load_B_ee() - load accB from EE
;-----------------------------------------------------------
load_B_ee:                              
call      ee_read_waddr_ia         
movwf     r_accB_L                 
call      ee_read_ia               
movwf     r_accB_H                 
return                             

;-----------------------------------------------------------
;--- check_triggers() - compare a/d values to trigger levels
;-----------------------------------------------------------
check_triggers:
    clerf r_flags_2 ; pre-clear flags
    btfss flag0_mode_nm ; nickel algorithm ?
    goto check_triggers_lion ; --- no, skip ...

check_triggers_nimh:
    movlw EE_CHGN_V_PCHG ;
    call check_voltage ;
    rlff r_flags_2, f ;
    movlw EE_CHGN_T_PCHG_LO ;
    call check_temperature ;
    rlff r_flags_2, f ;
    movlw EE_CHGN_T_PCHG_HI ;
    call check_temperature ;
    rlff r_flags_2, f ;
    movlw EE_CHGN_T_MAX ;
    call check_temperature ;
    rlff r_flags_2, f ;
    movlw EE_CHGN_V_MAX ;
    call check_voltage ;
    rlff r_flags_2, f ;
    movlw EE_CHGN_V_PCHG ;
    call check_voltage ;
    rlff r_flags_2, f ;
    movlw EE_CHGN_T_MAXCHGI ;
    call check_temperature ;
    rlff r_flags_2, f ;
    movlw EE_CHGN_T_MAXCHG ;
    call check_temperature ;
    rlff r_flags_2, f ;
    movlw EE_CHGN_T_PCHG ;
    call check_temperature ;
    rlff r_flags_2, f ;
    goto check_triggers_x ;

check_triggers_lion:
    movlw EE_CHG_V_MIN ;
    call check_voltage ;
    rlff r_flags_2, f ;
    movlw EE_CHG_V_MAX ;
    call check_voltage ;
    rlff r_flags_2, f ;
    movlw EE_CHG_V_REG ;
    call check_voltage ;
    rlff r_flags_2, f ;
    movlw EE_CHG_V_PCHG ;
    call check_voltage ;
    rlff r_flags_2, f ;
    movlw EE_CHG_T_MIN ;
    call check_temperature ;
    rlff r_flags_2, f ;
    movlw EE_CHG_T_MAXCHG ;
    call check_temperature ;
    rlff r_flags_2, f ;
    movlw EE_CHG_T_MAXCHGI ;
    call check_temperature ;
    rlff r_flags_2, f ;
    movlw EE_CHG_T_PCHG ;
    call check_temperature ;
    rlff r_flags_2, f ;

check_triggers_x:
    return ;
;-----------------------------------------------------------
Check Triggers

This routine checks all the triggers that are required to exit the current charge state and enter into a different one. The triggers, variables and equations for each state are described in the functional descriptions.

```assembly
;-----------------------------------------------------------
;--- load_timer_d1() - load timer and reset pre-scaler
;-----------------------------------------------------------
load_timer_d1_ee:
    call     ee_read_waddr
load_timer_d1:
    movwf    r_timer_d1
    clrf     r_timer_d
    bcf      flag_chg_ti1_done
    return

;-----------------------------------------------------------
;--- load_timer_d2() - load timer and reset pre-scaler
;    used as a "holdoff" or "minimum" timeout; therefore,
;    if setpoint==0, "done" flag set immediately
;    if setpoint<>0, "done" flag set when timer expires
;-----------------------------------------------------------
load_timer_d2_ee:
    call     ee_read_waddr
load_timer_d2:
    movwf    r_timer_d2
    clrf     r_timer_d
    bcf      flag_chg_ti2_done
    movf     r_timer_d2, f
    btfsc    r_status, Z
    bsf      flag_chg_ti2_done
    return

;---------------------------------------
;--- check timer a
;--- check timer b
;--- check timer c
;--- check timer d
;    status[Z]=1 if timer=0
;---------------------------------------
check_chg_timer_a:
    movlw    r_chg_timer_a
    goto     check_chg_timer_
check_chg_timer_b:
    movlw    r_chg_timer_b
    goto     check_chg_timer_
check_chg_timer_c:
    movlw    r_chg_timer_c
    goto     check_chg_timer_
check_chg_timer_d:
    movlw    r_chg_timer_d
    goto     check_chg_timer_
check_chg_timer_
    movwf    r_fsr
    movf     r_indf, f
    btfss    r_status, Z
    decfsz   r_indf, f
    nop
    return
```
--- check_current() - compare current to threshold

; call:
;   w ee address of threshold
; uses:
;   r_accB_L
;   r_accB_H
;   r_fsr
; return:
;   r_status[C]=1  current <= threshold
;   r_status[C]=0  current > threshold

check_current:
  call load_B_ee
check_current_
  movlw r_adc_1_H
  goto math_cmp_16

--- check_voltage() - compare voltage to threshold

; call:
;   w ee address of threshold
; uses:
;   r_accB_L
;   r_accB_H
;   r_fsr
; return:
;   r_status[C]=1  voltage <= threshold
;   r_status[C]=0  voltage > threshold

check_voltage:
  call load_B_ee
  movlw r_adc_2_H
  goto math_cmp_16

--- check_temperature() - compare temperature to threshold

; call:
;   w ee address of threshold
; uses:
;   r_accB_L
;   r_accB_H
;   r_fsr
; return:
;   r_status[C]=1  temperature <= threshold
;   r_status[C]=0  temperature > threshold

check_temperature:
  addlw -.1
  call load_B_ee
  clrf r_accB_L
  movlw .6
  call math_shift_BC
  movlw r_adc_3_H
  goto math_cmp_16

--- load_A_ee() - load accA from EE

;-----------------------------------------------------------
load_A_ee:  
call  ee_read_waddr_ia  
movwf  r_accA_L  
call  ee_read_ia  
movwf  r_accA_H  
return

load_B_ee:  
call  ee_read_waddr_ia  
movwf  r_accB_L  
call  ee_read_ia  
movwf  r_accB_H  
return

check_triggers:  
clrf  r_flags_2  ; pre-clear flags
btfss  flag0_mode_nm  ; nickel algorithm ?
goto  check_triggers_lion  ; --- no, skip ...

check_triggers_nimh:  
movlw  EE_CHGN_V_PCHG  
call  check_voltage  
rlf  r_flags_2, f  
movlw  EE_CHGN_T_PCHG_LO  
call  check_temperature  
rlf  r_flags_2, f  
movlw  EE_CHGN_T_PCHG_HI  
call  check_temperature  
rlf  r_flags_2, f  
movlw  EE_CHGN_T_MAX  
call  check_temperature  
rlf  r_flags_2, f  
movlw  EE_CHGN_V_MAX  
call  check_voltage  
rlf  r_flags_2, f  
movlw  EE_CHGN_V_PCHG  
call  check_voltage  
rlf  r_flags_2, f  
movlw  EE_CHGN_V_REG  
call  check_voltage  
rlf  r_flags_2, f  
rlf  r_flags_2, f  
goto  check_triggers_x

check_triggers_lion:  
movlw  EE_CHGN_V_MIN  
call  check_voltage  
rlf  r_flags_2, f  
movlw  EE_CHGN_V_MAX  
call  check_voltage  
rlf  r_flags_2, f  
movlw  EE_CHGN_V_REG  
call  check_voltage  
rlf  r_flags_2, f  
movlw  EE_CHGN_V_PCHG  
call  check_voltage  
rlf  r_flags_2, f  
movlw  EE_CHGN_V_PCHG  
call  check_voltage  
rlf  r_flags_2, f  

movlw EE_CHG_T_MIN           ;
call check_temperature       ;
rlf r_flags_2, f             ;
movlw EE_CHG_T_MAXCHGI       ;
call check_temperature       ;
rlf r_flags_2, f             ;
movlw EE_CHG_T_MAXCHG        ;
call check_temperature       ;
rlf r_flags_2, f             ;
movlw EE_CHG_T_PCHG          ;
call check_temperature       ;
rlf r_flags_2, f             ;

check_triggers_x:             ;
    return                   ;
;--------------------------------------------------------------------------
Regulate

The regulate routine controls the PWM so that the required voltage and current are always delivered to the battery. The charge state and the feedback measurements are used to determine if the PWM needs to be adjusted. If it does, then PWM_adjust is called to calculate the correct new value, and PWM_set is called to write that value to the PWM Control register.

;-----------------------------------------------------------
;--- regulate()
;-----------------------------------------------------------
#define   REG_V_H2   .25                ; voltage - upper limit 2
#define   REG_V_H1   .6                 ; voltage - upper limit 1
#define   REG_V_L1   .6                 ; voltage - lower limit 1
#define   REG_V_L2   .50                ; voltage - upper limit 2
#define   REG_C_NULL     .5             ; current - null limit
#define   REG_ADJ_P1     .12            ; pwm adjust
#define   REG_ADJ_P2     .10            ; pwm adjust
#define   REG_ADJ_P3     .5             ; pwm adjust
#define   REG_ADJ_P4     .1             ; pwm adjust

regulate:                               
   btfss     flag_reg_on              ; regulation enabled ?
   goto      regulate_x_              ; --- no,  exit ...

   btfss     flag_reg_timer           ; timer elapsed ?
   goto      regulate_x_              ; --- no,  exit ...
   bcf       flag_reg_timer           ;

   btfss     flag_adcset_1_rdy        ; adc conversions done ?
   goto      regulate_x               ; --- no,  chk adc & exit ...

regulate_co:                            
   btfsc     flag0_mode_vregco_dis    ;
   goto      regulate_co_x            ;
   movlw     EE_REG_VSAFETY           ;
   btfsc     flag0_mode_nm            ;
   movlw     EE_REG_VSAFETY_NM        ;
   call      check_voltage            ;
   btfss     r_status, C              ;
   goto      reg_off                  ;

regulate_co_x:                          

;---------------------------------------
;--- voltage
;---------------------------------------
regulate_v:                             
   bcf       flag_vreg_1              ;
   bcf       flag_vreg_2              ;
   bcf       flag_vreg                ;
   bcf       flag_neg                 ;

   movlw     r_adc_2_L                ;
   call      math_load_B              ; accB = voltage
   call      math_neg_B               ;
   movlw     r_reg_v                  ;
   call      math_add_16_load_A       ; accB = delta = reg_v - voltage
   btfss     r_accB_H, 7              ; delta negative ?
   goto      regulate_v_lo            ; --- no, voltage <= vreg, skip ...

;--- V > VREG

regulate_v_hi:                          
   bsf       flag_vreg                ;
   bsf       flag_neg                 ;

;-----------------------------------------------------------
;--- regulate()
movlw   EE_REG_VH                ;
call    ee_read_waddr            ;
call    math_add_8b              ;
btfsr   r_accB_H, 7              ; delta <= upper_limit ?
goto   regulate_v_x             ; --- yes, go check current ... 
bsf     flag_vreg_1              ;

movlw   EE_REG_VHH_VH            ;
call    ee_read_waddr            ;
call    math_add_8b              ;
btfsr   r_accB_H, 7              ; delta <= upper_limit ?
goto   regulate_v_x             ; --- yes, go check current ... 
bsf     flag_vreg_1              ;

movlw   EE_REG_P1                ;
call    ee_read_waddr            ;
go to   regulate_adj             ; --- no, make big adj (down) 

;===(zones: A1,A2,A3,A4,A5)
;--- V <= VREG

regulate_v_lo:                          
call    math_neg_B               ;

movlw   EE_REG_VL                ;
call    ee_read_waddr            ;
call    math_add_8b              ;
btfsr   r_accB_H, 7              ; delta <= upper_limit ?
goto   regulate_v_x             ; --- yes, go check current ... 
bsf     flag_vreg_1              ;

movlw   EE_REG_VLL_VL            ;
call    ee_read_waddr            ;
call    math_add_8b              ;
btfsr   r_accB_H, 7              ; delta <= upper_limit ?
goto   regulate_v_x             ; --- yes, go check current ... 
bsf     flag_vreg_2              ; V > LEVEL 2 !!

regulate_v_x:                           
btfsc   flag_vreg_1              ; set flag_vreg if null zone 
goto   regulate_v_x_             
btfss   flag_vreg_2              
bsf     flag_vreg                ;

regulate_v_x_:                          

;---------------------------------------
;--- current 
;---------------------------------------

regulate_c:                             
movlw   r_adc_1_L                ;
call    math_load_B              ;
call    math_neg_B               ;

movlw   r_reg_c                  ;
call    math_add_16_load_A       ; accB = delta = reg_c - current 
bcf     flag_neg                 ;
btfsr   r_accB_H, 7              ;
goto   $+3                      
bsf     flag_neg                 ; current > reg_c level? (set flag) 
call    math_neg_B               ; delta = -delta (make it positive)

;--- delta_c >= 256

regulate_c_1:                           
movf    r_accB_H, f              ; delta >= 256 mA 
btfsc   r_status, Z              
goto   regulate_c_1_x            
btfsc   flag_neg                 ;
goto regulate_adj_dn_1 ;===(zones: B5,C5,D5,E5)
regulate_c_1_x:

;--- delta_c < 256 - voltage too high, non-null
regulate_c_2:
  btfss flag_vreg
  goto regulate_2_x
  btfsc flag_vreg_1
  goto regulate_adj_dn_0 ;===(zones: B1,B2,B3,B4)
regulate_2_x:

regulate_c_3:
  call math_neg_B ; delta_c = -delta_c (make it negative)
  movlw EE_REG_CNULL
  call ee_read_waddr
  call math_add_8b
  btfss r_accB_H, 7
  goto regulate_x ;===(zones: C3,D3,E3) delta_c in null zone
  btfsc flag_neg
  goto regulate_adj_dn_0 ;===(zones: C4,D4,E4) current too high
  call math_neg_B ; delta = -delta (make it positive)
regulate_c_3_x:

regulate_c_4:
  btfsc flag_vreg_2
  goto regulate_c_4_x
  btfss flag_vreg_1
  goto regulate_x ;===(zones: C1,C2)
regulate_c_4_x:

regulate_c_5:
  movf r_accB_H, f ; delta <= -256 mA
  btfsc r_status, Z
  goto regulate_adj_up_0 ;===(zones: E2)
  movlw EE_REG_P3
  call ee_read_waddr
  goto regulate_adj ;===(zones: E1)
regulate_c_5_x:

regulate_adj_dn_1:
  movlw EE_REG_P2
  call ee_read_waddr
  goto regulate_adj_dn
regulate_adj_dn_0:
  movlw EE_REG_P4
  call ee_read_waddr
regulate_adj:
  bsf flag_neg
  goto regulate_adj
regulate_adj_up_1:
  movlw EE_REG_P2
  call ee_read_waddr
  goto regulate_adj_up
regulate_adj_up_0:
  movlw EE_REG_P4
  call ee_read_waddr
regulate_adj_up:
  bcf flag_neg
regulate_adj:
  call pwm_adj
  call pwm_set
regulate_x:
    bcf flag_adcset_1_rdy
    bsf flag_adcset_1_rq

regulate_x:
    return

;---------------------------------------
;--- reg_load()
;---------------------------------------
reg_load_c:
    call load_B_ee
    movlw r_reg_c
    call math_move_B
    return

reg_load_v:
    call load_B_ee
    movlw r_reg_v
    call math_move_B
    return

reg_load_nimh:
    call reg_load_c
    movlw EE_CHGN_V
    call reg_load_v
    goto reg_on

reg_load_lion:
    movlw EE_CHG_C_CREG
    reg_load_lion:
        call reg_load_c
        movlw EE_CHG_V_REG
        call reg_load_v
        goto reg_on

;---------------------------------------
;--- reg_on() - regulation on
;---------------------------------------
reg_on:
    bsf flag_reg_on
    bsf r_status, RP1
    bcf r_pwmclk, PWMASE
    bcf r_status, RP1
    return

;---------------------------------------
;--- reg_off() - regulation off
;---------------------------------------
reg_off:
    bcf flag_reg_on
    clrf r_reg_c
    clrf r_reg_c+1
    clrf r_reg_v
    clrf r_reg_v+1
    clrf r_pwm_L
    clrf r_pwm_H
    call pwm_set
    return
Math Functions

These are random math routines used by the firmware.

;-----------------------------------------------------
;--- math_add_8b() - add 8-bit positive value to accB
;-----------------------------------------------------
math_add_8b:                            
    movwf     r_accA_L                 
    clrf      r_accA_H                 
    goto      math_add_16              

;-----------------------------------------------------------
;--- math_mul_16_prep()
;
;    function:
;        *WREG - 16-bit operand
;-----------------------------------------------------------
math_mul_16_prep:                       
    call      math_load_A              
    math_mul_16_prep_:                      
    movlw     .16                      

;-----------------------------------------------------------
;--- math_mul_16
;
;    function:
;        16x16 multiplication
;
;    call:
;        WREG - count/shift ops
;        op1: accA
;        op2: accD
;
;    result:
;        accB,accC = accA * accD
;-----------------------------------------------------------
math_mul_16:                            
    movwf     r_count_1                
    clrf      r_accB_H                 ; clear result accumulator
    clrf      r_accB_L                 
    clrf      r_accC_H                 
    clrf      r_accC_L                 
    math_mul_16_loop:                       
    rrf       r_accD_H, F              ; shift operand2 lsb into C
    rrf       r_accD_L, F              
    btfss     r_status, C              ; C = 1?
    goto      math_mul_16_shift        ; --- no, go shift ...
    math_mul_16_add:                        
    call      math_add_16              
    math_mul_16_shift:                      
    rrf       r_accB_H, F              ; SHIFT result accumulator
    rrf       r_accB_L, F              
    rrf       r_accC_H, F              
    rrf       r_accC_L, F              
    decfsz    r_count_1, F             
    goto      math_mul_16_loop         ; loop
    retlw     0                        ;
--- div32

operands:
- dividend - accA, accB
- divisor - accC

result:
- quotient - accB
- remainder - accA
- overflow - WREG=0 else WREG=1?

---

math_div_32:

    movf r_accC_L, W
    subwf r_accA_L, W
    movf r_accC_H, W
    btfss r_status, C
    incf r_accC_H, W
    subwf r_accA_H, W
    btfsc r_status, C
    retlw 0 ; overflow or division by zero

    movlw .16
    movwf r_count_1

math_div_32_loop:

    bcf r_status, C
    rlf r_accB_L, F ; shift dividend (accA, accB << 1)
    rlf r_accB_H, F
    rlf r_accA_L, F
    rlf r_accA_H, F
    btfsc r_status, C ; if carry, go subtract
    goto math_div_32_sub

    movf r_accC_L, W
    subwf r_accA_L, W
    movf r_accC_H, W
    btfss r_status, C
    incf r_accC_H, W
    subwf r_accA_H, W
    btfss r_status, C ; if smaller than divisor, skip to next
    goto math_div_32_next

math_div_32_sub:

    movf r_accC_L, W ; subtract divisor from high
    subwf r_accA_L, F
    movf r_accC_H, W
    btfss r_status, C
    incf r_accC_H, W
    subwf r_accA_H, F
    bsf r_accB_L, 0

math_div_32_next:

    decfsz r_count_1, F
    goto math_div_32_loop

    retlw 1 ; no more overflow possible

---

--- math_add_16()

function:
add 16-bit operands

call:
o1: accA
o2: accB
;  result:
;  accB = accA + accB
;-----------------------------------------------------------
math_add_16_load_A: ;
call math_load_A ;
math_add_16: ;
movf r_accA_L, w ;
addwf r_accB_L, f ;
btfsC r_status, C ;
incf r_accB_H, f ;
movf r_accA_H, w ;
addwf r_accB_H, f ;
retlw 0 ;
;-----------------------------------------------------------
;--- math_neg_B()
; function:
; accB = -accB
;-----------------------------------------------------------
math_neg_B: ;
comf r_accB_L, f ;
incf r_accB_L, f ;
btfsC r_status, Z ;
decf r_accB_H, f ;
comf r_accB_H, f ;
retlw 0 ;
;-----------------------------------------------------------
;--- math_cmp_16()
; function:
; compare 2 16-bit values
; 1: accB  2: *WREG
; call:
; *WREG:  operand A (msb)
; accB:   operand B
; result:
; status[C]=1:  accB >= *WREG
; status[C]=0:  accB <  *WREG
;-----------------------------------------------------------
math_cmp_16: ;
movwf r_fsr ;
movf r_indf, w ;
subwf r_accB_H, w ;
btfsC r_status, Z ;
return ;
decf r_fsr, f ;
movf r_indf, w ;
subwf r_accB_L, w ;
return ;
;-----------------------------------------------------------
;--- math_load_utilities
; *WREG = lsb of source data
;-----------------------------------------------------------

;--- math_load_D()
math_load_D: ;
movwf r_fsr ;
math_load_D:                           
        movf    r_indf, w                
        movwf   r_accD_L                 
        incf    r_fsr, f                 
        movf    r_indf, w                
        movwf   r_accD_H                 
        goto    math_load_x              

;--- math_load_C()
math_load_C:                            
        movwf   r_fsr                    
math_load_C_:                           
        movf    r_indf, w                
        movwf   r_accC_L                 
        incf    r_fsr, f                 
        movf    r_indf, w                
        movwf   r_accC_H                 
        goto    math_load_x              

;--- math_load_B()
math_load_B:                            
        movwf   r_fsr                    
math_load_B_:                           
        movf    r_indf, w                
        movwf   r_accB_L                 
        incf    r_fsr, f                 
        movf    r_indf, w                
        movwf   r_accB_H                 
        goto    math_load_x              

;--- math_load_A()
math_load_A:                            
        movwf   r_fsr                    
math_load_A_:                           
        movf    r_indf, w                
        movwf   r_accA_L                 
        incf    r_fsr, f                 
        movf    r_indf, w                
        movwf   r_accA_H                 
        goto    math_load_x              
        incf    r_fsr, f                 
        return                           

;--- math_move_B:                           
        movwf   r_fsr                    
math_move_B_:                           
        movf    r_accB_L, w              
        movwf   r_indf                   
        incf    r_fsr, f                 
        movf    r_accB_H, w              
        goto    math_move_x              

;--- math_move_D:                           
        movwf   r_fsr                    
math_move_D_:                           
        movf    r_accD_L, w              
        movwf   r_indf                   
        incf    r_fsr, f                 
        movf    r_accD_H, w              
        goto    math_move_x              

math_move_x:                            
    movwf r_indf                      
    incf r_fsr, f                     
    return                            ;---

math_shift_BC:                          
    movwf r_count_1                   
math_shift_BC_loop:                     
    bcf r_status, C                   
    rrf r_accB_H, f                   
    rrf r_accB_L, f                   
    rrf r_accC_H, f                   
    rrf r_accC_L, f                   
    decfsz r_count_1, f               
goto math_shift_BC_loop             
return                                

org 0x7d0
chg_state_svc_jumptable:               
    movlw high $                     
    movwf r_pclath                   
    movf r_chg_state, w             
    andlw 0x0f                       
    addwf r_pcl, f                   
    goto chg_state_0                 
    goto chg_state_1                 
    goto chg_state_2                 
    goto chg_state_3                 
    goto chg_state_4                 
    goto chg_state_5                 
    goto chg_state_6                 
    goto chg_state_7                 
    goto chg_state_0_init           
    goto chg_state_1_init           
    goto chg_state_2_init           
    goto chg_state_3_init           
    goto chg_state_4_init           
    goto chg_state_5_init           
    goto chg_state_6_init           
    goto chg_state_7_init           
#if chg_state_svc_jumptable >> 8 != $>>8
    error "jump table page violation: chg_state_svc_jumptable"
#endif
;-----------------------------------------------------------
Default EEPROM Values

This sets the default values for the EEPROM parameters. Note that the internal names of EEPROM parameters may vary from the data sheet and PowerTool 200 names. The PowerTool 200 names are used in the functional description sections.

`;---------------------------------------
;--- EEPROM DEFAULT
;---------------------------------------
org 0x2100
DE   0x01, 0x00                    ; pattern
DE   .1                            ; ncells
DE   "microchp"                    ; manuf name
DE   "16HV785  "                    ; device name
DE   low .800, high .800           ; capacity
DE   .19                           ; pwm
DE   0x01                           ; mode
DE   0x20                           ; mode2
DE   .00                            ; oscillator trim

org 0x2120
DE   0x08, 0x18, 0x28, 0x38        ; LED1 CONFIG
DE   0x48, 0x58, 0x68, 0x78         ; LED1 CONFIG (cont)
DE   0x08, 0x18, 0x28, 0x38        ; LED2 CONFIG
DE   0x48, 0x58, 0x68, 0x78         ; LED2 CONFIG (cont)

org 0x2134
DE   .12, .10, .05, .01            ; regulation: pwm adj values
DE   .19, .06, .06, .44            ; regulation: voltage zone thresholds
DE   .5                             ; regulation: current zone thresholds
DE   low .4350, high .4350         ; regulation: v_safety (lion)

org 0x2143
DE   low .2000, high .2000         ; chg_v_min
DE   low .4250, high .4250         ; chg_v_max
DE   low .4000, high .4000         ; chg_v_rchg
DE   low .4200, high .4200         ; chg_v_reg
DE   low .3000, high .3000         ; chg_v_pchg
DE   low .0050, high .0050         ; chg_v_min_bp
DE   .50                            ; chg_t_min
DE   .175                           ; chg_t_chgi
DE   .200                           ; chg_t_chg
DE   .75                            ; chg_t_pchg
DE   low .100, high .100            ; chg_c_pchg
DE   low .800, high .800            ; chg_c_reg
DE   low .50, high .50             ; chg_c_min
DE   .0                              ; chg_t_pchg
DE   .0                              ; chg_t_creg
DE   .0                              ; chg_t_vreg
DE   .20, .20, .00, .00, .00, .00   ; chg_time
DE   .128, .255                      ; battid_min, _max

org 0x216c
DE   low .0248, high .0248         ; adc_cal_0 (reference)
DE   low .2553, high .2553         ; adc_cal_1 (current)
DE   low .5121, high .5121         ; adc_cal_2 (voltage)
DE   low .8192, high .8192         ; adc_cal_3 (temperature)
DE   low .6407, high .6407         ; adc_cal_4 (battid)
DE   .100                            ; shunt
DE   .112                            ; temperature default
org 0x217C
DE .8 ; TLUT - length
DE .38, .48, .61, .79 ; TLUT - temp axis
DE .105, .183, .207 ; TLUT - temp axis (cont)
DE low -.23362, high -.23362 ; TLUT - slope - 0
DE low .1418, high .1418 ; TLUT - yint - 0
DE low -.19864, high -.19864 ; TLUT - slope - 1
DE low .1352, high .1352 ; TLUT - yint - 1
DE low -.15709, high -.15709 ; TLUT - slope - 2
DE low .1255, high .1255 ; TLUT - yint - 2
DE low -.12572, high -.12572 ; TLUT - slope - 3
DE low .1162, high .1162 ; TLUT - yint - 3
DE low -.10206, high -.10206 ; TLUT - slope - 4
DE low .1071, high .1071 ; TLUT - yint - 4
DE low -.8631, high -.8631 ; TLUT - slope - 5
DE low .990, high .990 ; TLUT - yint - 5
DE low -.10154, high -.10154 ; TLUT - slope - 6
DE low .1127, high .1127 ; TLUT - yint - 6
DE low -.12875, high -.12875 ; TLUT - slope - 7
DE low .1402, high .1402 ; TLUT - yint - 7

org 0x21AA
; debug
DE 0x1A ; mode_3
DE 0x3A ; mode_3
DE low .2544, high .2544 ; adc_cal_2 (voltage) nimh
DE low .2000, high .2000 ; chgn_v
DE low .1000, high .1000 ; chgn_v_dchg
DE .175 ; chgn_t_max (175=50degC, 200=60degC)
DE low .1800, high .1800 ; chgn_v_max
DE low .1000, high .1000 ; chgn_c_pchg
DE .75 ; chgn_t_pchg_lo (75=10degC)
DE .162 ; chgn_t_pchg_hi (162=44.8degC)
DE low .0800, high .0800 ; chgn_v_pchg
DE .0 ; chgn_ti_pchg_mn
DE .0 ; chgn_ti_pchg_mx
DE low .2000, high .2000 ; chgn_c_fchg
DE .0 ; chgn_ti_fchg_mn
DE .0 ; chgn_ti_fchg_mx
DE .120 ; chgn_ti_dtdt (temp rise time criteria)
DE .10 ; chgn_t_dtdt (temp rise criteria)
DE low .10, high .10 ; chgn_v_dv (-delta voltage)
DE low .200, high .200 ; chgn_c_topoff
DE .11 ; chgn_ti_topoff (11=45min, i.e. 66% eff, 5% cap, c/10)
DE low .1300, high .1300 ; chgn_v_rchg (recharge voltage)
DE low .1900, high .1900 ; chgn_v_trik
DE .150 ; chgn_ti_trik (timeout for trickle charge)
DE low .1900, high .1900 ; chgn_v_safety

end
## Memory Map

### EEPROM

#### TABLE 11: EEPROM MEMORY MAP

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Len</th>
<th>Default</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dec</td>
<td>Hex</td>
<td>Dec</td>
<td>Hex</td>
<td></td>
</tr>
<tr>
<td>PATTERN</td>
<td>1</td>
<td>0001</td>
<td>2</td>
<td>1</td>
<td>1 coded</td>
</tr>
<tr>
<td>NCELLS</td>
<td>1</td>
<td>01</td>
<td>1</td>
<td>1</td>
<td>1 cells</td>
</tr>
<tr>
<td>MANUF_NAME</td>
<td>3</td>
<td>03</td>
<td>8</td>
<td>—</td>
<td>— ASCII</td>
</tr>
<tr>
<td>DEV_NAME</td>
<td>11</td>
<td>0B</td>
<td>8</td>
<td>—</td>
<td>— ASCII</td>
</tr>
<tr>
<td>CAPACITY</td>
<td>19</td>
<td>13</td>
<td>2</td>
<td>800</td>
<td>320 mAh</td>
</tr>
<tr>
<td>PWM_FREQ</td>
<td>21</td>
<td>15</td>
<td>1</td>
<td>19</td>
<td>13 coded</td>
</tr>
<tr>
<td>MODE</td>
<td>22</td>
<td>16</td>
<td>1</td>
<td>1</td>
<td>01 coded</td>
</tr>
<tr>
<td>MODE2</td>
<td>23</td>
<td>17</td>
<td>1</td>
<td>32</td>
<td>20 coded</td>
</tr>
<tr>
<td>OSC_TRIM</td>
<td>24</td>
<td>18</td>
<td>1</td>
<td>0</td>
<td>0 coded</td>
</tr>
<tr>
<td>Reserved</td>
<td>25</td>
<td>19</td>
<td>7</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td></td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LED1_CFG_0</td>
<td>32</td>
<td>20</td>
<td>1</td>
<td>8</td>
<td>08 coded</td>
</tr>
<tr>
<td>LED1_CFG_1</td>
<td>33</td>
<td>21</td>
<td>1</td>
<td>24</td>
<td>18 coded</td>
</tr>
<tr>
<td>LED1_CFG_2</td>
<td>34</td>
<td>22</td>
<td>1</td>
<td>40</td>
<td>28 coded</td>
</tr>
<tr>
<td>LED1_CFG_3</td>
<td>35</td>
<td>23</td>
<td>1</td>
<td>56</td>
<td>38 coded</td>
</tr>
<tr>
<td>LED1_CFG_4</td>
<td>36</td>
<td>24</td>
<td>1</td>
<td>72</td>
<td>48 coded</td>
</tr>
<tr>
<td>LED1_CFG_5</td>
<td>37</td>
<td>25</td>
<td>1</td>
<td>88</td>
<td>58 coded</td>
</tr>
<tr>
<td>LED1_CFG_6</td>
<td>38</td>
<td>26</td>
<td>1</td>
<td>104</td>
<td>68 coded</td>
</tr>
<tr>
<td>LED1_CFG_7</td>
<td>39</td>
<td>27</td>
<td>1</td>
<td>120</td>
<td>78 coded</td>
</tr>
<tr>
<td>LED2_CFG_0</td>
<td>40</td>
<td>28</td>
<td>1</td>
<td>8</td>
<td>08 coded</td>
</tr>
<tr>
<td>LED2_CFG_1</td>
<td>41</td>
<td>29</td>
<td>1</td>
<td>24</td>
<td>18 coded</td>
</tr>
<tr>
<td>LED2_CFG_2</td>
<td>42</td>
<td>2A</td>
<td>1</td>
<td>40</td>
<td>28 coded</td>
</tr>
<tr>
<td>LED2_CFG_3</td>
<td>43</td>
<td>2B</td>
<td>1</td>
<td>56</td>
<td>38 coded</td>
</tr>
<tr>
<td>LED2_CFG_4</td>
<td>44</td>
<td>2C</td>
<td>1</td>
<td>72</td>
<td>48 coded</td>
</tr>
<tr>
<td>LED2_CFG_5</td>
<td>45</td>
<td>2D</td>
<td>1</td>
<td>88</td>
<td>58 coded</td>
</tr>
<tr>
<td>LED2_CFG_6</td>
<td>46</td>
<td>2E</td>
<td>1</td>
<td>104</td>
<td>68 coded</td>
</tr>
<tr>
<td>LED2_CFG_7</td>
<td>47</td>
<td>2F</td>
<td>1</td>
<td>120</td>
<td>78 coded</td>
</tr>
<tr>
<td>Reserved</td>
<td>48</td>
<td>30</td>
<td>4</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
**TABLE 11: EEPROM MEMORY MAP (CONTINUED)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Len</th>
<th>Default</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dec</td>
<td>Hex</td>
<td>Dec</td>
<td>Hex</td>
<td></td>
</tr>
<tr>
<td>SUB-LED</td>
<td>52</td>
<td>34</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REG_P1</td>
<td>52</td>
<td>34</td>
<td>1</td>
<td>12</td>
<td>0C count</td>
</tr>
<tr>
<td>REG_P2</td>
<td>53</td>
<td>35</td>
<td>1</td>
<td>10</td>
<td>0A count</td>
</tr>
<tr>
<td>REG_P3</td>
<td>54</td>
<td>36</td>
<td>1</td>
<td>5</td>
<td>05 count</td>
</tr>
<tr>
<td>REG_P4</td>
<td>55</td>
<td>37</td>
<td>1</td>
<td>1</td>
<td>01 count</td>
</tr>
<tr>
<td>REG_VHH_VH</td>
<td>56</td>
<td>38</td>
<td>1</td>
<td>19</td>
<td>13 mV</td>
</tr>
<tr>
<td>REG_VH</td>
<td>57</td>
<td>39</td>
<td>1</td>
<td>6</td>
<td>06 mV</td>
</tr>
<tr>
<td>REG_VL</td>
<td>58</td>
<td>3A</td>
<td>1</td>
<td>6</td>
<td>06 mV</td>
</tr>
<tr>
<td>REG_VLL_VL</td>
<td>59</td>
<td>3B</td>
<td>1</td>
<td>44</td>
<td>2C mV</td>
</tr>
<tr>
<td>REG_CNULL</td>
<td>60</td>
<td>3C</td>
<td>1</td>
<td>5</td>
<td>05 mA</td>
</tr>
<tr>
<td>REG_VSAFETY</td>
<td>61</td>
<td>3D</td>
<td>2</td>
<td>4350</td>
<td>1F0E mV</td>
</tr>
<tr>
<td>Reserved</td>
<td>63</td>
<td>3F</td>
<td>4</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SUB-REG</td>
<td>67</td>
<td>43</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHG_V_MIN</td>
<td>67</td>
<td>43</td>
<td>2</td>
<td>2000</td>
<td>07D0 mV</td>
</tr>
<tr>
<td>CHG_V_MAX</td>
<td>69</td>
<td>45</td>
<td>2</td>
<td>4250</td>
<td>109A mV</td>
</tr>
<tr>
<td>CHG_V_RCHG</td>
<td>71</td>
<td>47</td>
<td>2</td>
<td>4000</td>
<td>FA0 mV</td>
</tr>
<tr>
<td>CHG_V_VREG</td>
<td>73</td>
<td>49</td>
<td>2</td>
<td>4200</td>
<td>1068 mV</td>
</tr>
<tr>
<td>CHG_V_PCHG</td>
<td>75</td>
<td>4B</td>
<td>2</td>
<td>3000</td>
<td>0BB8 mV</td>
</tr>
<tr>
<td>CHG_V_MIN_BP</td>
<td>77</td>
<td>4D</td>
<td>2</td>
<td>50</td>
<td>0032 mV</td>
</tr>
<tr>
<td>CHG_T_MIN</td>
<td>79</td>
<td>4F</td>
<td>1</td>
<td>50</td>
<td>32 TCODE</td>
</tr>
<tr>
<td>CHG_T_MAXCHGI</td>
<td>80</td>
<td>50</td>
<td>1</td>
<td>175</td>
<td>AF TCODE</td>
</tr>
<tr>
<td>CHG_T_MAXCHG</td>
<td>81</td>
<td>51</td>
<td>1</td>
<td>200</td>
<td>C8 TCODE</td>
</tr>
<tr>
<td>CHG_T_PCHG</td>
<td>82</td>
<td>52</td>
<td>1</td>
<td>75</td>
<td>4B TCODE</td>
</tr>
<tr>
<td>CHG_C_PCHG</td>
<td>83</td>
<td>53</td>
<td>2</td>
<td>100</td>
<td>0064 mA</td>
</tr>
<tr>
<td>CHG_C_CREG</td>
<td>85</td>
<td>55</td>
<td>2</td>
<td>800</td>
<td>0320 mA</td>
</tr>
<tr>
<td>CHG_C_MIN</td>
<td>87</td>
<td>57</td>
<td>2</td>
<td>50</td>
<td>0032 mA</td>
</tr>
<tr>
<td>CHG_TI_PCHG</td>
<td>89</td>
<td>59</td>
<td>1</td>
<td>0</td>
<td>0 4 min.</td>
</tr>
<tr>
<td>CHG_TI_CREG</td>
<td>90</td>
<td>5A</td>
<td>1</td>
<td>0</td>
<td>0 4 min.</td>
</tr>
<tr>
<td>CHG_TI_VREG</td>
<td>91</td>
<td>5B</td>
<td>1</td>
<td>0</td>
<td>0 4 min.</td>
</tr>
<tr>
<td>CHG_TIME_0</td>
<td>92</td>
<td>5C</td>
<td>1</td>
<td>20</td>
<td>14 .25 sec.</td>
</tr>
<tr>
<td>CHG_TIME_1</td>
<td>93</td>
<td>5D</td>
<td>1</td>
<td>20</td>
<td>14 .25 sec.</td>
</tr>
<tr>
<td>CHG_TIME_2</td>
<td>94</td>
<td>5E</td>
<td>1</td>
<td>20</td>
<td>14 .25 sec.</td>
</tr>
<tr>
<td>CHG_TIME_3</td>
<td>95</td>
<td>5F</td>
<td>1</td>
<td>0</td>
<td>0 .25 sec.</td>
</tr>
<tr>
<td>CHG_TIME_4</td>
<td>96</td>
<td>60</td>
<td>1</td>
<td>0</td>
<td>0 .25 sec.</td>
</tr>
<tr>
<td>CHG_TIME_5</td>
<td>97</td>
<td>61</td>
<td>1</td>
<td>0</td>
<td>0 .25 sec.</td>
</tr>
<tr>
<td>BATTID_MIN</td>
<td>98</td>
<td>62</td>
<td>1</td>
<td>0</td>
<td>0 units</td>
</tr>
<tr>
<td>BATTID_MAX</td>
<td>99</td>
<td>63</td>
<td>1</td>
<td>128</td>
<td>128 units</td>
</tr>
<tr>
<td>Reserved</td>
<td>100</td>
<td>64</td>
<td>8</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
### TABLE 11: EEPROM MEMORY MAP (CONTINUED)

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Len</th>
<th>Default</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dec</td>
<td>Hex</td>
<td>Dec</td>
<td>Hex</td>
<td></td>
</tr>
<tr>
<td>SUB-CHG</td>
<td>108</td>
<td>6C</td>
<td>41</td>
<td></td>
<td>Calibration (reference)</td>
</tr>
<tr>
<td>ADC_CAL_0</td>
<td>108</td>
<td>6C</td>
<td>2</td>
<td>248</td>
<td>00F8 scaler Calibration (reference)</td>
</tr>
<tr>
<td>ADC_CAL_1</td>
<td>110</td>
<td>6E</td>
<td>2</td>
<td>2553</td>
<td>09F9 scaler Calibration (current)</td>
</tr>
<tr>
<td>ADC_CAL_2</td>
<td>112</td>
<td>70</td>
<td>2</td>
<td>5121</td>
<td>1401 scaler Calibration (voltage)</td>
</tr>
<tr>
<td>ADC_CAL_3</td>
<td>114</td>
<td>72</td>
<td>2</td>
<td>8192</td>
<td>2000 scaler Calibration (temperature)</td>
</tr>
<tr>
<td>ADC_CAL_4</td>
<td>116</td>
<td>74</td>
<td>2</td>
<td>6407</td>
<td>1907 scaler Calibration (BATID)</td>
</tr>
<tr>
<td>SHUNT</td>
<td>118</td>
<td>76</td>
<td>1</td>
<td>100</td>
<td>64 mOhm Shunt resistor value</td>
</tr>
<tr>
<td>T_DEFAULT</td>
<td>119</td>
<td>77</td>
<td>1</td>
<td>112</td>
<td>70 TCODE T CODE Default temperature</td>
</tr>
<tr>
<td>Reserved</td>
<td>120</td>
<td>78</td>
<td>4</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

### SUB-ADC

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Len</th>
<th>Default</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dec</td>
<td>Hex</td>
<td>Dec</td>
<td>Hex</td>
<td></td>
</tr>
<tr>
<td>T_LUT_N</td>
<td>124</td>
<td>7C</td>
<td>16</td>
<td></td>
<td>Temperature LUT – length (max. = 8)</td>
</tr>
<tr>
<td>T_LUT_T_0</td>
<td>125</td>
<td>7D</td>
<td>1</td>
<td>38</td>
<td>26 TCODE Temperature LUT – T0 (T_LUT_N – 1 entry)</td>
</tr>
<tr>
<td>T_LUT_T_1</td>
<td>126</td>
<td>7E</td>
<td>1</td>
<td>48</td>
<td>30 TCODE Temperature LUT – T1</td>
</tr>
<tr>
<td>T_LUT_T_2</td>
<td>127</td>
<td>7F</td>
<td>1</td>
<td>61</td>
<td>3D TCODE Temperature LUT – T2</td>
</tr>
<tr>
<td>T_LUT_T_3</td>
<td>128</td>
<td>80</td>
<td>1</td>
<td>79</td>
<td>4F TCODE Temperature LUT – T3</td>
</tr>
<tr>
<td>T_LUT_T_4</td>
<td>129</td>
<td>81</td>
<td>1</td>
<td>105</td>
<td>69 TCODE Temperature LUT – T4</td>
</tr>
<tr>
<td>T_LUT_T_5</td>
<td>130</td>
<td>82</td>
<td>1</td>
<td>183</td>
<td>B7 TCODE Temperature LUT – T5</td>
</tr>
<tr>
<td>T_LUT_T_6</td>
<td>131</td>
<td>83</td>
<td>1</td>
<td>207</td>
<td>CF TCODE Temperature LUT – T6</td>
</tr>
<tr>
<td>T_LUT_M_0</td>
<td>132</td>
<td>84</td>
<td>2</td>
<td>-23362 A4BE scaler Temperature LUT – T0 slope (T_LUT_N entries)</td>
<td></td>
</tr>
<tr>
<td>T_LUT_B_0</td>
<td>134</td>
<td>86</td>
<td>2</td>
<td>1418</td>
<td>058A scaler Temperature LUT – T0 Y-intercept</td>
</tr>
<tr>
<td>T_LUT_M_1</td>
<td>136</td>
<td>88</td>
<td>2</td>
<td>-19864 B268 scaler Temperature LUT – T1 slope</td>
<td></td>
</tr>
<tr>
<td>T_LUT_B_1</td>
<td>138</td>
<td>8A</td>
<td>2</td>
<td>1352</td>
<td>0548 scaler Temperature LUT – T1 Y-intercept</td>
</tr>
<tr>
<td>T_LUT_M_2</td>
<td>140</td>
<td>8C</td>
<td>2</td>
<td>-15709 C2A3 scaler Temperature LUT – T2 slope</td>
<td></td>
</tr>
<tr>
<td>T_LUT_B_2</td>
<td>142</td>
<td>8E</td>
<td>2</td>
<td>1255</td>
<td>04E7 scaler Temperature LUT – T2 Y-intercept</td>
</tr>
<tr>
<td>T_LUT_M_3</td>
<td>144</td>
<td>90</td>
<td>2</td>
<td>-12572 CEE4 scaler Temperature LUT – T3 slope</td>
<td></td>
</tr>
<tr>
<td>T_LUT_B_3</td>
<td>146</td>
<td>92</td>
<td>2</td>
<td>1162</td>
<td>048A scaler Temperature LUT – T3 Y-intercept</td>
</tr>
<tr>
<td>T_LUT_M_4</td>
<td>148</td>
<td>94</td>
<td>2</td>
<td>-10206 D822 scaler Temperature LUT – T4 slope</td>
<td></td>
</tr>
<tr>
<td>T_LUT_B_4</td>
<td>150</td>
<td>96</td>
<td>2</td>
<td>1071</td>
<td>042F scaler Temperature LUT – T4 Y-intercept</td>
</tr>
<tr>
<td>T_LUT_M_5</td>
<td>152</td>
<td>98</td>
<td>2</td>
<td>-8631</td>
<td>DE49 scaler Temperature LUT – T5 slope</td>
</tr>
<tr>
<td>T_LUT_B_5</td>
<td>154</td>
<td>9A</td>
<td>2</td>
<td>990</td>
<td>03DE scaler Temperature LUT – T5 Y-intercept</td>
</tr>
<tr>
<td>T_LUT_M_6</td>
<td>156</td>
<td>9C</td>
<td>2</td>
<td>-10154 D856 scaler Temperature LUT – T6 slope</td>
<td></td>
</tr>
<tr>
<td>T_LUT_B_6</td>
<td>158</td>
<td>9E</td>
<td>2</td>
<td>1127</td>
<td>0467 scaler Temperature LUT – T6 Y-intercept</td>
</tr>
<tr>
<td>T_LUT_M_7</td>
<td>160</td>
<td>A0</td>
<td>2</td>
<td>-12875 CDB5 scaler Temperature LUT – T7 slope</td>
<td></td>
</tr>
<tr>
<td>T_LUT_B_7</td>
<td>162</td>
<td>A2</td>
<td>2</td>
<td>1402</td>
<td>057A scaler Temperature LUT – T7 Y-intercept</td>
</tr>
</tbody>
</table>
### TABLE 11: EEPROM MEMORY MAP (CONTINUED)

<table>
<thead>
<tr>
<th>Name</th>
<th>Location Dec</th>
<th>Location Hex</th>
<th>Len Dec</th>
<th>Len Hex</th>
<th>Default Dec</th>
<th>Default Hex</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB-T LUT</td>
<td>164</td>
<td>A4</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE3</td>
<td>170</td>
<td>AA</td>
<td>1</td>
<td>26</td>
<td>1A</td>
<td></td>
<td>bits</td>
<td>Mode bits</td>
</tr>
<tr>
<td>ADC_CAL_2_NM</td>
<td>171</td>
<td>AB</td>
<td>2</td>
<td>2544</td>
<td>9F0</td>
<td></td>
<td>scaler</td>
<td>Voltage calibration</td>
</tr>
<tr>
<td>CHGN_V_CHG</td>
<td>173</td>
<td>AD</td>
<td>2</td>
<td>2000</td>
<td>07D0</td>
<td></td>
<td>mV</td>
<td>Charge voltage</td>
</tr>
<tr>
<td>CHGN_V_DCHG</td>
<td>175</td>
<td>AF</td>
<td>2</td>
<td>1000</td>
<td>03E8</td>
<td></td>
<td>mV</td>
<td>Voltage target for discharge</td>
</tr>
<tr>
<td>CHGN_T_MAX</td>
<td>177</td>
<td>B1</td>
<td>1</td>
<td>175</td>
<td>AF</td>
<td></td>
<td>TCODE</td>
<td>Max. temperature</td>
</tr>
<tr>
<td>CHGN_V_MAX</td>
<td>178</td>
<td>B2</td>
<td>2</td>
<td>1800</td>
<td>0708</td>
<td></td>
<td>mV</td>
<td>Max. voltage</td>
</tr>
<tr>
<td>CHGN_C_PCHG</td>
<td>180</td>
<td>B4</td>
<td>2</td>
<td>100</td>
<td>0064</td>
<td></td>
<td>mA</td>
<td>Precharge current</td>
</tr>
<tr>
<td>CHGN_T_PCHG_LO</td>
<td>182</td>
<td>B6</td>
<td>1</td>
<td>75</td>
<td>4B</td>
<td></td>
<td>TCODE</td>
<td>Precharge criteria – lo temp</td>
</tr>
<tr>
<td>CHGN_T_PCHG_HI</td>
<td>183</td>
<td>B7</td>
<td>1</td>
<td>162</td>
<td>A2</td>
<td></td>
<td>TCODE</td>
<td>Precharge criteria – hi temp</td>
</tr>
<tr>
<td>CHGN_V_PCHG</td>
<td>184</td>
<td>B8</td>
<td>2</td>
<td>800</td>
<td>0320</td>
<td></td>
<td>mV</td>
<td>Precharge criteria – voltage</td>
</tr>
<tr>
<td>CHGN_TI_PCHG_MN</td>
<td>186</td>
<td>BA</td>
<td>1</td>
<td>0</td>
<td>00</td>
<td></td>
<td>4 min.</td>
<td>Precharge stage – min. time</td>
</tr>
<tr>
<td>CHGN_TI_PCHG_MX</td>
<td>187</td>
<td>BB</td>
<td>1</td>
<td>0</td>
<td>00</td>
<td></td>
<td>4 min.</td>
<td>Precharge stage – max time</td>
</tr>
<tr>
<td>CHGN_C_FCHG</td>
<td>188</td>
<td>BC</td>
<td>2</td>
<td>2000</td>
<td>07D0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHGN_C_FCHG_MN</td>
<td>190</td>
<td>BE</td>
<td>1</td>
<td>0</td>
<td>00</td>
<td></td>
<td>4 min.</td>
<td>Fast Charge state – min. time</td>
</tr>
<tr>
<td>CHGN_TI_FCHG_MX</td>
<td>191</td>
<td>BF</td>
<td>1</td>
<td>0</td>
<td>00</td>
<td></td>
<td>4 min.</td>
<td>Fast Charge state – max. time</td>
</tr>
<tr>
<td>CHGN_TI_DTDT</td>
<td>192</td>
<td>C0</td>
<td>1</td>
<td>120</td>
<td>78</td>
<td></td>
<td>0.5 sec.</td>
<td>dttd detect – time window</td>
</tr>
<tr>
<td>CHGN_T_DTDT</td>
<td>193</td>
<td>C1</td>
<td>1</td>
<td>10</td>
<td>10</td>
<td></td>
<td>0.1°</td>
<td>dttd detect – temp delta</td>
</tr>
<tr>
<td>CHGN_V_DV</td>
<td>194</td>
<td>C2</td>
<td>2</td>
<td>10</td>
<td>10</td>
<td></td>
<td>mV</td>
<td>dv detect – voltage delta</td>
</tr>
<tr>
<td>CHGN_C_TOPOFF</td>
<td>196</td>
<td>C4</td>
<td>2</td>
<td>200</td>
<td>0064</td>
<td></td>
<td>mA</td>
<td>Top Off state – current</td>
</tr>
<tr>
<td>CHGN_V_TOPOFF</td>
<td>198</td>
<td>C6</td>
<td>1</td>
<td>11</td>
<td>0B</td>
<td></td>
<td>4 min.</td>
<td>Top Off state – max. time</td>
</tr>
<tr>
<td>CHGN_V_RCHG</td>
<td>199</td>
<td>C7</td>
<td>2</td>
<td>1300</td>
<td>0514</td>
<td></td>
<td>mV</td>
<td>Voltage threshold to re-init trickle</td>
</tr>
<tr>
<td>CHGN_TI_TRIK</td>
<td>201</td>
<td>C9</td>
<td>1</td>
<td>80</td>
<td>50</td>
<td></td>
<td>4 min.</td>
<td>Trickle Charge state – max. time</td>
</tr>
<tr>
<td>CHGN_C_TRIK</td>
<td>202</td>
<td>CA</td>
<td>2</td>
<td>150</td>
<td>0096</td>
<td></td>
<td>mA</td>
<td>Trickle Charge state – current</td>
</tr>
<tr>
<td>REG_VSAFETY</td>
<td>204</td>
<td>CC</td>
<td>2</td>
<td>1900</td>
<td>076C</td>
<td></td>
<td>mV</td>
<td>Regulation – shutdown voltage</td>
</tr>
</tbody>
</table>

| SUB-NM             |              |              | 24      |         |             |             |       |                                                       |

| TOTAL              |              |              | 188     |         |             |             |       |                                                       |
# Mode Registers

## TABLE 12: MODE REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>pchg_always</td>
<td>1 = Always precharge</td>
</tr>
<tr>
<td>6</td>
<td>gpio_cutoff</td>
<td>1 = Enable GPIO cutoff logic</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>bpres_battid</td>
<td>1 = Battery present on BATID</td>
</tr>
<tr>
<td>1</td>
<td>bpres_v</td>
<td>1 = Battery present on voltage sense</td>
</tr>
<tr>
<td>0</td>
<td>bpres_always</td>
<td>1 = Battery present always</td>
</tr>
</tbody>
</table>

## TABLE 13: MODE2 REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>cofs_dis</td>
<td>1 = Disable auto-offset current cancellation</td>
</tr>
<tr>
<td>6</td>
<td>osc_out</td>
<td>1 = Enable clock output (256) on BATID after Reset</td>
</tr>
<tr>
<td>5</td>
<td>temp_k</td>
<td>1 = Use constant temperature from EEPROM</td>
</tr>
<tr>
<td>4</td>
<td>nm_enable</td>
<td>1 = Nickel metal hydride algorithm</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>vrchg_dis</td>
<td>1 = Disable voltage recharge trigger</td>
</tr>
<tr>
<td>1</td>
<td>vregco_dis</td>
<td>1 = Disable voltage cutoff in regulator</td>
</tr>
<tr>
<td>0</td>
<td>pwmas_dis</td>
<td>1 = Disable PWM auto-shutdown</td>
</tr>
</tbody>
</table>

## TABLE 14: MODE3 REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>nm_extdchg_en</td>
<td>1 = Enable external discharge</td>
</tr>
<tr>
<td>6</td>
<td>nm_extrik_en</td>
<td>1 = Enable external trickle circuit</td>
</tr>
<tr>
<td>5</td>
<td>nm_trik_byp</td>
<td>1 = Bypass Trickle Charge state</td>
</tr>
<tr>
<td>4</td>
<td>nm_topoff_en</td>
<td>1 = Enable “Top Off” state</td>
</tr>
<tr>
<td>3</td>
<td>nm_vrchg_en</td>
<td>1 = Enable Trickle Charge if V drops below recharge voltage</td>
</tr>
<tr>
<td>2</td>
<td>nm_slochg</td>
<td>1 = Enable Slow Charge mode (Trickle Charge only)</td>
</tr>
<tr>
<td>1</td>
<td>nm_eoc_ddt</td>
<td>1 = Enable EOC method – dttd</td>
</tr>
<tr>
<td>0</td>
<td>nm_eoc_mdv</td>
<td>1 = Enable EOC method – minus dv</td>
</tr>
</tbody>
</table>
### RAM Registers

**TABLE 15: RAM**

<table>
<thead>
<tr>
<th>Name</th>
<th>Dec</th>
<th>Hex</th>
<th>Len</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_mode</td>
<td>32</td>
<td>20</td>
<td>1</td>
<td>bits</td>
<td>Operational mode register</td>
</tr>
<tr>
<td>r_chg_state</td>
<td>33</td>
<td>21</td>
<td>1</td>
<td>int</td>
<td>Charge Controller “state”</td>
</tr>
<tr>
<td>r_adc_0_L</td>
<td>34</td>
<td>22</td>
<td>1</td>
<td>units</td>
<td>ADC result – channel 0 (VREF)</td>
</tr>
<tr>
<td>r_adc_0_H</td>
<td>35</td>
<td>23</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r_adc_1_L</td>
<td>36</td>
<td>24</td>
<td>1</td>
<td>mA</td>
<td>ADC result – channel 1 (CURRENT)</td>
</tr>
<tr>
<td>r_adc_1_H</td>
<td>37</td>
<td>25</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r_adc_2_L</td>
<td>38</td>
<td>26</td>
<td>1</td>
<td>mV</td>
<td>ADC result – channel 2 (VOLTAGE)</td>
</tr>
<tr>
<td>r_adc_2_H</td>
<td>39</td>
<td>27</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r_adc_3_L</td>
<td>40</td>
<td>28</td>
<td>1</td>
<td>TCODE</td>
<td>ADC result – channel 3 (TEMP)</td>
</tr>
<tr>
<td>r_adc_3_H</td>
<td>41</td>
<td>29</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r_adc_4_L</td>
<td>42</td>
<td>2A</td>
<td>1</td>
<td>units</td>
<td>ADC result – channel 4 (BATID)</td>
</tr>
<tr>
<td>r_adc_4_H</td>
<td>43</td>
<td>2B</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r_pwm_L</td>
<td>44</td>
<td>2C</td>
<td>1</td>
<td>int</td>
<td>PWM setting</td>
</tr>
<tr>
<td>r_pwm_H</td>
<td>45</td>
<td>2D</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r_reg_c</td>
<td>46</td>
<td>2E</td>
<td>2</td>
<td>mA</td>
<td>Regulation target: current (mA)</td>
</tr>
<tr>
<td>r_reg_v</td>
<td>48</td>
<td>30</td>
<td>2</td>
<td>mV</td>
<td>Regulation target: voltage (mV)</td>
</tr>
<tr>
<td>r_comm_reg_0</td>
<td>50</td>
<td>32</td>
<td>1</td>
<td>N/A</td>
<td>Indirect Address register</td>
</tr>
<tr>
<td>r_comm_reg_1</td>
<td>51</td>
<td>33</td>
<td>1</td>
<td>N/A</td>
<td>Status</td>
</tr>
<tr>
<td>r_comm_reg_2</td>
<td>52</td>
<td>34</td>
<td>1</td>
<td>N/A</td>
<td>Configuration flags</td>
</tr>
<tr>
<td>r_comm_reg_3</td>
<td>53</td>
<td>35</td>
<td>1</td>
<td>N/A</td>
<td>Command flags</td>
</tr>
<tr>
<td>r_comm_reg_4</td>
<td>54</td>
<td>36</td>
<td>1</td>
<td>N/A</td>
<td>Data lo</td>
</tr>
<tr>
<td>r_comm_reg_5</td>
<td>55</td>
<td>37</td>
<td>1</td>
<td>N/A</td>
<td>Data hi</td>
</tr>
<tr>
<td>r_comm_reg_6</td>
<td>56</td>
<td>38</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_comm_reg_7</td>
<td>57</td>
<td>39</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_sim</td>
<td>58</td>
<td>3A</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_chg_timer_a</td>
<td>59</td>
<td>3B</td>
<td>1</td>
<td>.25 sec.</td>
<td>Hysteresis timer</td>
</tr>
<tr>
<td>r_chg_timer_b</td>
<td>60</td>
<td>3C</td>
<td>1</td>
<td>.25 sec.</td>
<td>Hysteresis timer</td>
</tr>
<tr>
<td>r_chg_timer_c</td>
<td>61</td>
<td>3D</td>
<td>1</td>
<td>.25 sec.</td>
<td>Hysteresis timer</td>
</tr>
<tr>
<td>r_chg_timer_d</td>
<td>62</td>
<td>3E</td>
<td>1</td>
<td>.25 sec.</td>
<td>Hysteresis timer</td>
</tr>
<tr>
<td>r_temp_1</td>
<td>63</td>
<td>3F</td>
<td>1</td>
<td>N/A</td>
<td>Location sensitive (init ram clear)</td>
</tr>
<tr>
<td>r_temp_2</td>
<td>64</td>
<td>40</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_temp_3</td>
<td>65</td>
<td>41</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_temp_4</td>
<td>66</td>
<td>42</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_tempi_1</td>
<td>67</td>
<td>43</td>
<td>1</td>
<td>N/A</td>
<td>Temporary register for ISR</td>
</tr>
<tr>
<td>r_timer_a1</td>
<td>68</td>
<td>44</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_timer_b</td>
<td>69</td>
<td>45</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_timer_b1</td>
<td>70</td>
<td>46</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_timer_c</td>
<td>71</td>
<td>47</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_timer_d</td>
<td>72</td>
<td>48</td>
<td>1</td>
<td>sec.</td>
<td></td>
</tr>
<tr>
<td>r_timer_d1</td>
<td>73</td>
<td>49</td>
<td>1</td>
<td>4 min.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Dec</td>
<td>Hex</td>
<td>Len</td>
<td>Units</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>r_led_config_1</td>
<td>74</td>
<td>4A</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_led_config_2</td>
<td>76</td>
<td>4C</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_adc_control</td>
<td>78</td>
<td>4E</td>
<td>1</td>
<td>N/A</td>
<td>ADC control</td>
</tr>
<tr>
<td>r_adc_raw_L</td>
<td>79</td>
<td>4F</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_adc_raw_H</td>
<td>80</td>
<td>50</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_count_1</td>
<td>81</td>
<td>51</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_accD_L</td>
<td>82</td>
<td>52</td>
<td>1</td>
<td>N/A</td>
<td>Math – accumulator – D</td>
</tr>
<tr>
<td>r_accD_H</td>
<td>83</td>
<td>53</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_accC_L</td>
<td>84</td>
<td>54</td>
<td>1</td>
<td>N/A</td>
<td>Math – accumulator – C</td>
</tr>
<tr>
<td>r_accC_H</td>
<td>85</td>
<td>55</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_accB_L</td>
<td>86</td>
<td>56</td>
<td>1</td>
<td>N/A</td>
<td>Math – accumulator – B</td>
</tr>
<tr>
<td>r_accB_H</td>
<td>87</td>
<td>57</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_accA_L</td>
<td>88</td>
<td>58</td>
<td>1</td>
<td>N/A</td>
<td>Math – accumulator – A</td>
</tr>
<tr>
<td>r_accA_H</td>
<td>89</td>
<td>59</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_comm_count</td>
<td>90</td>
<td>5A</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_comm_data</td>
<td>91</td>
<td>5B</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_comm_flags</td>
<td>92</td>
<td>5C</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_comm_data_cmnd</td>
<td>93</td>
<td>5D</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_mode2</td>
<td>94</td>
<td>5E</td>
<td>1</td>
<td>bits</td>
<td>Operational mode flags</td>
</tr>
<tr>
<td>unused</td>
<td>95</td>
<td>5F</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_adc_accum</td>
<td>96</td>
<td>60</td>
<td>2</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_adc_accum_count</td>
<td>98</td>
<td>62</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_adc_avg</td>
<td>99</td>
<td>63</td>
<td>2</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>r_adc_shadow</td>
<td>101</td>
<td>65</td>
<td>2</td>
<td>units</td>
<td></td>
</tr>
<tr>
<td>r_adc_1_ofs</td>
<td>103</td>
<td>67</td>
<td>2</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>r_adc_2_save</td>
<td>104</td>
<td>68</td>
<td>2</td>
<td>mV</td>
<td>NiMh -dv reference</td>
</tr>
<tr>
<td>r_adc_3_save_a</td>
<td>106</td>
<td>6A</td>
<td>2</td>
<td>TCODE</td>
<td>NiMh dtdt reference</td>
</tr>
<tr>
<td>r_adc_3_save_b</td>
<td>108</td>
<td>6C</td>
<td>2</td>
<td>TCODE</td>
<td>NiMh dtdt reference</td>
</tr>
<tr>
<td>r_mode3</td>
<td>110</td>
<td>6E</td>
<td>1</td>
<td>bits</td>
<td>Operational mode bits</td>
</tr>
<tr>
<td>r_timer_d2</td>
<td>111</td>
<td>6F</td>
<td>1</td>
<td>4 min.</td>
<td>Timer – min. timer – NiMh algorithm</td>
</tr>
<tr>
<td>r_shadow_1</td>
<td>112</td>
<td>70</td>
<td>1</td>
<td>N/A</td>
<td>Debug</td>
</tr>
<tr>
<td>r_shadow_2</td>
<td>113</td>
<td>71</td>
<td>1</td>
<td>N/A</td>
<td>Debug</td>
</tr>
<tr>
<td>r_shadow_3</td>
<td>114</td>
<td>72</td>
<td>1</td>
<td>N/A</td>
<td>Debug</td>
</tr>
<tr>
<td>r_flags_1</td>
<td>115</td>
<td>73</td>
<td>1</td>
<td>bits</td>
<td>Assorted bit flags</td>
</tr>
<tr>
<td>r_flags_2</td>
<td>116</td>
<td>74</td>
<td>1</td>
<td>bits</td>
<td>Assorted bit flags</td>
</tr>
<tr>
<td>r_flags_3</td>
<td>117</td>
<td>75</td>
<td>1</td>
<td>bits</td>
<td>Assorted bit flags</td>
</tr>
<tr>
<td>r_flags_4</td>
<td>118</td>
<td>76</td>
<td>1</td>
<td>bits</td>
<td>Assorted bit flags</td>
</tr>
<tr>
<td>r_flags_5</td>
<td>119</td>
<td>77</td>
<td>1</td>
<td>bits</td>
<td>Assorted bit flags</td>
</tr>
<tr>
<td>r_flags_6</td>
<td>120</td>
<td>78</td>
<td>1</td>
<td>bits</td>
<td>Assorted bit flags</td>
</tr>
</tbody>
</table>
TABLE 15: RAM (CONTINUED)

<table>
<thead>
<tr>
<th>Name</th>
<th>Dec</th>
<th>Hex</th>
<th>Len</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>r_isr_w</code></td>
<td>121</td>
<td>79</td>
<td>1</td>
<td>N/A</td>
<td>Interrupt context</td>
</tr>
<tr>
<td><code>r_isr_status</code></td>
<td>122</td>
<td>7A</td>
<td>1</td>
<td>N/A</td>
<td>Interrupt context</td>
</tr>
<tr>
<td><code>r_isr_pclath</code></td>
<td>123</td>
<td>7B</td>
<td>1</td>
<td>N/A</td>
<td>Interrupt context</td>
</tr>
<tr>
<td><code>r_isr_fsr</code></td>
<td>124</td>
<td>7C</td>
<td>1</td>
<td>N/A</td>
<td>Interrupt context</td>
</tr>
<tr>
<td><code>r_ee_data</code></td>
<td>125</td>
<td>7D</td>
<td>1</td>
<td>N/A</td>
<td>EEPROM data</td>
</tr>
<tr>
<td><code>r_ee_addr</code></td>
<td>126</td>
<td>7E</td>
<td>1</td>
<td>N/A</td>
<td>EEPROM address</td>
</tr>
<tr>
<td><code>r_tempc_1</code></td>
<td>127</td>
<td>7F</td>
<td>1</td>
<td>N/A</td>
<td>Temporary register</td>
</tr>
</tbody>
</table>
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.

Trademarks
The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELoC, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, PowerSmart, rfPIC, and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AmpLab, FilterLab, Migratable Memory, MXDEV, MXLAB, SEEVAL, SmartSensor 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, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Linear Active Thermistor, Mindi, MiWi, MPASM, MPLIB, MPLINK, PICkit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rLAB, rfPICDEM, Select Mode, Smart Serial, SmartTel, Total Endurance, UNI/O, 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.


Printed on recycled paper.

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona, Gresham, Oregon and Mountain View, California. The Company's quality system processes and procedures are for its PICmicro® 8-bit MCUs, KEELoC® 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://support.microchip.com  
Web Address: www.microchip.com

<table>
<thead>
<tr>
<th>City</th>
<th>Address</th>
<th>Tel</th>
<th>Fax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlanta</td>
<td>Alpharetta, GA</td>
<td>770-640-0034</td>
<td>770-640-0307</td>
</tr>
<tr>
<td>Boston</td>
<td>Westborough, MA</td>
<td>774-760-0087</td>
<td>774-760-0088</td>
</tr>
<tr>
<td>Chicago</td>
<td>Itasca, IL</td>
<td>630-285-0071</td>
<td>630-285-0075</td>
</tr>
<tr>
<td>Dallas</td>
<td>Addison, TX</td>
<td>972-818-7423</td>
<td>972-818-2924</td>
</tr>
<tr>
<td>Detroit</td>
<td>Farmington Hills, MI</td>
<td>248-538-2250</td>
<td>248-538-2260</td>
</tr>
<tr>
<td>Kokomo</td>
<td>Kokomo, IN</td>
<td>765-864-8360</td>
<td>765-864-8387</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>Mission Viejo, CA</td>
<td>949-462-9523</td>
<td>949-462-9608</td>
</tr>
<tr>
<td>San Jose</td>
<td>Mountain View, CA</td>
<td>650-215-1444</td>
<td>650-961-0286</td>
</tr>
<tr>
<td>Toronto</td>
<td>Mississauga, Ontario, Canada</td>
<td>905-673-0699</td>
<td>905-673-6509</td>
</tr>
</tbody>
</table>

## ASIA/PACIFIC
Australia - Sydney  
Tel: 61-2-9868-6733  
Fax: 61-2-9868-6755

China - Beijing  
Tel: 86-10-8528-2100  
Fax: 86-10-8528-2104

China - Chengdu  
Tel: 86-28-8676-6200  
Fax: 86-28-8676-6599

China - Fuzhou  
Tel: 86-591-8750-3506  
Fax: 86-591-8750-3521

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

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: 86-24-2334-2828  
Fax: 86-24-2334-2393

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

India - Bangalore  
Tel: 91-80-4182-8400  
Fax: 91-80-4182-8422

India - New Delhi  
Tel: 91-11-5160-8631  
Fax: 91-11-5160-8632

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 - Gumi  
Tel: 82-54-473-4301  
Fax: 82-54-473-4302

Korea - Seoul  
Tel: 82-2-554-7200  
Fax: 82-2-558-5922 or 82-2-558-5934

Malaysia - Penang  
Tel: 60-4-646-8870  
Fax: 60-4-646-5086

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: 886-3-572-9526  
Fax: 886-3-572-6459

Taiwan - Kaohsiung  
Tel: 886-7-536-4818  
Fax: 886-7-536-4803

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: 43-7242-2244-3910  
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

06/08/06