16HV785: Programmable Lead Acid Battery Charger

Features

- User-configurable battery charger for 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 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>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VDD</td>
</tr>
<tr>
<td>2</td>
<td>LED2</td>
</tr>
<tr>
<td>3</td>
<td>VIN</td>
</tr>
<tr>
<td>4</td>
<td>RESET</td>
</tr>
<tr>
<td>5</td>
<td>CTRLOUT</td>
</tr>
<tr>
<td>6</td>
<td>CHGOUT</td>
</tr>
<tr>
<td>7</td>
<td>LOOPFBK</td>
</tr>
<tr>
<td>8</td>
<td>LOOPIN</td>
</tr>
<tr>
<td>9</td>
<td>CTRLIN</td>
</tr>
<tr>
<td>10</td>
<td>LED1</td>
</tr>
<tr>
<td>11</td>
<td>HVOUT</td>
</tr>
<tr>
<td>12</td>
<td>IFBINB</td>
</tr>
<tr>
<td>13</td>
<td>IFBINA</td>
</tr>
<tr>
<td>14</td>
<td>IFBOUT</td>
</tr>
<tr>
<td>15</td>
<td>BATID</td>
</tr>
<tr>
<td>16</td>
<td>CHGFBK</td>
</tr>
<tr>
<td>17</td>
<td>SHDN</td>
</tr>
<tr>
<td>18</td>
<td>VOVP</td>
</tr>
<tr>
<td>19</td>
<td>TEMP</td>
</tr>
<tr>
<td>20</td>
<td>VSS</td>
</tr>
</tbody>
</table>

20-Pin PDIP, SOIC, SSOP
USING THE 16HV785

Product Overview

The 16HV785 provides an unprecedented level of configurability for charging lead battery packs. Its precise, 10-bit Analog-to-Digital converter and high-frequency Pulse-Width Modulator enable the 16HV785 to provide optimum control of charging algorithms for lead battery chemistries. Special features include an internal voltage regulator and an internal clock oscillator that reduce external component count. The 16HV785 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 lead chemistries during extreme temperature and voltage conditions, multi-step charging is required. The 16HV785 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 16HV785 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 16HV785.

SPECIAL FEATURES

The 16HV785 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>VOP</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
16HV785 HARDWARE OVERVIEW

The 16HV785 is a configurable, Switch mode charger which is comprised of a PIC16F microcontroller core and precise analog circuitry. This section explores the hardware features in relation to generic Switch mode charging. The 16HV785 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 16HV785 features are well-suited for Switch mode battery charging. The 16HV785 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. Please 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
- IFBINB: to op amp
- IFBINA: 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 precise 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 16HV785 when voltage is applied to VDD.

- Brown-out Reset – The BOR is activated when the input voltage falls to 2.1V; the 16HV785 is reset.
FIGURE 1: 16HV785 BLOCK DIAGRAM

- Voltage Regulator
- Voltage Reference
- Current/Voltage Measurement Block
- Internal Oscillator
- Charge Control Module

Key Components:
- C1
- OA1
- OA2
- C2
- 10-bit ADC
- VDD, VSS
- VOVP
- SHDN
- CTRLIN
- LOOPIN
- LOOPFBK
- CHGFBK
- IFBINB
- IFBINA
- IFBOUT
- VIN
- BATID
- TEMP
- LED1
- LED2
- HVOUT
- CTRROUT
- CHGOUT
- RESET
- To Charge Control Module
REFERENCE SCHEMATIC

Theory of Operation

In this schematic, the 16HV675 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 16HV675.

FIGURE 2: BUCK CONVERTER

In Figure 2, when a constant voltage is applied to \( V_{IN} \), 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 \( V_{OUT} \) which is a fraction of \( V_{IN} \) equal to the duty cycle of the pulse.

The voltage drop across the inductor is:

**EQUATION 1:**

\[ VL = i \frac{di}{dt} \]

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

**EQUATION 2:**

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

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:

**EQUATION 3:**

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

or

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

\[ V_{CHARGE} * T_{ON} = V_{OUT} * T \]

\[ V_{OUT} = (T/TON) * V_{CHARGE} \]

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

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 \( V_{OUT} \). 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 \( V_{OUT} \).

**FIGURE 3: BUCK CONVERTER WAVEFORMS**

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.

FIGURE 4: SAWTOOTH GENERATOR

The voltage at CHGFBK will ramp up when the PWM output at CHGOUT triggers high. When the ramp at CHGFBK exceeds the internal comparator level of reference voltage, the PWM will trigger CHGOUT low. The constant frequency sawtooth will determine the pulse width as a function of internal reference voltage.

FIGURE 5: SAWTOOTH AND PWM WAVEFORMS

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.

FIGURE 6: FEEDBACK DIAGRAM

The actual circuit implementation, including op amp feedback RC networks, is shown in Figure 7.

FIGURE 7: FEEDBACK CIRCUIT
Power Supply Shunt Regulator

The 16HV785 has a built-in shunt regulator allowing 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 Equation 4:

\[
\begin{align*}
R_{\text{MAX}} &= \frac{V_s(\text{MIN}) - 5V}{1.05} \times 1000 \\
R_{\text{MIN}} &= \frac{V_s(\text{MAX}) - 5V}{0.95} \times 1000
\end{align*}
\]

Where:
- \( R_{\text{MAX}} \) = maximum value of series resistor (ohms)
- \( R_{\text{MIN}} \) = minimum value of series resistor (ohms)
- \( V_s(\text{MIN}) \) = minimum value of charger DC supply (VDC)
- \( V_s(\text{MAX}) \) = maximum value of charger DC supply (VDC)
- I(led) = total current drawn by all LEDs when illuminated simultaneously

Note: The 1.05 and 0.95 constants are included to compensate for the tolerance of 5% resistors. The 16 mA constant is the anticipated load presented by the 16HV785, 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 16HV786 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: 16HV785 SWITCHING CHARGER SCHEMATIC

- **RSENSE**: Resistance used to sense the current in the circuit.
- **VDD**: The power supply voltage for the charger.
- **C1**: Capacitor with a value of 100 nF.
- **R4**: Resistor with a value of 1.5K.
- **RIH**: Input resistor for the inverting input of the op-amp.
- **RIL**: Input resistor for the non-inverting input of the op-amp.
- **Rcomp**: Resistor used for compensation in the feedback loop.
- **Ccomp1/Ccomp2/Rcomp**: Components used for compensation in the feedback loop.
- **RT/CT/D6**: Components used for the timing of the PWM output.
- **VIN**: Input voltage for the charger.
- **LED**: LED used to indicate the status of the charger.
- **C2**: Capacitor with a value of 4.7 μF.
- **Q1**: Discrete MOSFET driver.
- **RG/RD**: Resistors used for driving the MOSFET.
- **Q3**: MOSFET used to block reverse current.
- **RG**: Resistor used for driving the MOSFET.
- **RD**: Resistor used for driving the MOSFET.
- **C9**: Capacitor with a value of 470 nF.

**FIRMWARE CURRENT CONTROL OUTPUT:**
- CTRLOUT is a compare output, which is a firmware-controlled PWM that sets the current level. R5/R9/C9 scale and filter the voltage from CTRLOUT, for application to OP1. R5/R9/C9 form a low-pass filter, with a time constant of 4.7 milliseconds. This allows firmware loop update rates of up to 50 times a second.

**OP2** is a current-sense amp. RIH, RIL, and RSENSE determine the current signal scaling. CIH is used in some applications to filter noise/spikes from the current sense signal.

**HARDWARE FEEDBACK TO PWM:**
- OP1 is an error amp. Ccomp1/Ccomp2/Rcomp are loop compensation components. RT/CT/D6 convert the PWM output to a ramp waveform, which is fed back to CP2 and compared to the error amp signal to determine the PWM duty cycle.

**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.**
FUNCTIONAL DESCRIPTION:
LEAD CHEMISTRY

Lead Charging
To ensure the proper treatment of lead chemistries during extreme temperature and voltage conditions, multi-step charging is required. The 16HV785 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 16HV785 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 16HV785 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 two possible next states.
1. If Mode3<0> is set to ‘1’, then skip to Float Charge state is selected. Charge Qualification will always jump directly to Float Charge state.
2. If Mode3<0> is set to ‘0’, then skip to Float Charge state is deselected. Charge Qualification will always progress to Current Regulation state.

CURRENT REGULATION STATE
The Current Regulation state is entered from Charge Qualification state. Battery charging is initiated. This state provides constant current, voltage limited charging. The charge current is referred to as ChargeCurr or the regulation current. While the current is applied, the battery’s voltage increases until it reaches a voltage limit referred to as the regulation voltage. For lead batteries, this charge voltage can vary with temperature. Colder temperatures can allow the battery to use higher charging voltages. To take advantage of this, Mode3<1> can be set to ‘0’. This uses a look-up table of charge voltages as a function of temperature from the parameters V_CHG_0..9 (voltage) and T_VLUT_0..8 (temperature). When Mode3<1> is set to ‘1’, a constant charge voltage is used from the parameter ChargeVol. Charging continues, during which battery voltage and temperature are monitored. There are two possible next states.
1. If the battery’s voltage reaches or exceeds the voltage limit, then the next state is Voltage Regulation.
2. If the time in the Current Regulation state exceeds the time limit (TimeoutCCState), then the next state is Charge Suspend.

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 (EOCCurrent). There are three possible next states.
1. When the charge current reaches the taper current threshold for End-of-Charge (EOCCurrent), the battery’s voltage remains at the regulated voltage value and Float mode is deselected (Mode3<3> = 0), then the battery has reached the Charge Cycle Complete state.
2. When the charge current reaches the taper current threshold for End-of-Charge (EOCCurrent), the battery’s voltage remains at the regulated voltage value and Float mode is selected (Mode3<3> = 1), then the battery has reached the Float Charge state.
3. If the time in the Voltage Regulation state exceeds the time limit (TimeoutCVState), then the next state is Charge Suspend.

FLOAT CHARGE STATE
In the Float Charge state, a lower charge target voltage is applied. As in Current Regulation state, the target voltage can be a constant or can vary with temperature. When Mode3<1> is set to ‘0’, the charger uses a look-up table of float charge voltages as a function of temperature from the parameters V_FLT_0..9 (voltage) and T_VLUT_0..8 (temperature). When Mode3<1> is set to ‘1’, a constant charge voltage is used from the parameter FloatVol. The resulting taper current is measured and compared against EOCCurrent. This helps to maintain a full charge. There is only one possible next state and that is Charge Cycle Complete. Charge Cycle Complete is entered when the voltage reaches the float voltage target and the current tapers to less than EOCCurrent, or the float timer, TimeoutFState, expires.

CHARGE SUSPEND STATE
In the Charge Suspend state, no current is applied to the battery pack. There is only one possible next state. If Mode3<5> is set to ‘1’, then suspend forever is selected. Suspend mode will be active until the battery is removed. If Mode3<5> is set to ‘0’, then Suspend mode will be active until the suspend timer, TimeoutRemSus, expires. Charge Suspend state always progresses to Charge Pending state.

CHARGE CYCLE COMPLETE STATE
When the current is less than the taper current threshold and the voltage is greater than the target voltage, End-of-Charge is triggered. At this threshold, charging is terminated and the End-of-Charge state is reached. If Mode3<4> is set to ‘1’, then refloat is enabled and after the refloat timer, TimeoutRIFloat, expires, Float Charge state will be re-entered.
CONFIGURABLE PARAMETERS

The 16HV785 device's configurable parameters allow for flexible changes in designing battery chargers. The parameters are categorized as follows:

- Configuration
- Lead 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 16HV785.

Lead Charging

The lead 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.

LED Display Configuration

The 16HV785 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.

### TABLE 2: 16HV785 LEAD CONFIGURATION 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>BandgapCF</td>
<td>2</td>
<td>248</td>
<td>integer</td>
<td>Internal band gap calibration factor.</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</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>16HV785</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>000000001b</td>
<td>binary</td>
<td>Configuration Register: bit 7: Unused; 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>
<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>Mode2</td>
<td>1</td>
<td>00100000b</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 = Disable auto-offset calibration</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 clock output on BATID pin after Reset</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 = Use constant temperature from EEPROM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: 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 = Disable voltage cutoff in regulator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: 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 (°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>
<tr>
<td>BattPresVolt</td>
<td>2</td>
<td>500</td>
<td>mV</td>
<td>Minimum voltage to set battery present when using battery voltage as a battery present determination.</td>
</tr>
<tr>
<td>ChargeCurr</td>
<td>2</td>
<td>2000</td>
<td>mA</td>
<td>Charging current during current regulation.</td>
</tr>
<tr>
<td>ChargeVolt</td>
<td>2</td>
<td>4200</td>
<td>mV</td>
<td>Target cell voltage in current regulation. This is set to the fully charged voltage of one cell, typically, as specified by the cell manufacturer.</td>
</tr>
<tr>
<td>EOCCurrent</td>
<td>2</td>
<td>200</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>FloatVolt</td>
<td>2</td>
<td>2275</td>
<td>mV</td>
<td>Target cell voltage during Float Charge state.</td>
</tr>
<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 7-6:</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 5:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = Suspend indefinitely – until Reset or battery removed</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 refloat – entered after Charge Cycle Complete 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 Float Charge state after Voltage Regulation state</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>1 = Use fixed float voltage (otherwise, use look-up table)</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 = Use fixed charge voltage (otherwise, use look-up table)</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 = Skip to Float Charge state immediately after Charge Qualification state</td>
</tr>
</tbody>
</table>
### TABLE 2: 16HV785 LEAD 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><strong>Lead Charging Parameters (Cont.)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TimeoutCCState</td>
<td>1</td>
<td>0</td>
<td>4 min.</td>
<td>Current regulation time limit.</td>
</tr>
<tr>
<td>TimeoutCVState</td>
<td>1</td>
<td>90</td>
<td>4 min.</td>
<td>Voltage regulation time limit.</td>
</tr>
<tr>
<td>TimeoutFLState</td>
<td>1</td>
<td>0</td>
<td>4 min.</td>
<td>Float charge time limit.</td>
</tr>
<tr>
<td>TimeoutRIFloat</td>
<td>1</td>
<td>0</td>
<td>4 min.</td>
<td>Re-enter float timer after Charge Cycle Complete state.</td>
</tr>
<tr>
<td>TimeoutRemSus</td>
<td>1</td>
<td>0</td>
<td>4 min.</td>
<td>Time to remain in Suspend mode.</td>
</tr>
<tr>
<td>V_CHG_0</td>
<td>2</td>
<td>2760</td>
<td>mV</td>
<td>Variable charge voltage. Used when TEMP &lt; T_VLUT_0.</td>
</tr>
<tr>
<td>V_CHG_1</td>
<td>2</td>
<td>2700</td>
<td>mV</td>
<td>Variable charge voltage. Used when T_VLUT_0 &lt; TEMP &lt; T_VLUT_1.</td>
</tr>
<tr>
<td>V_CHG_2</td>
<td>2</td>
<td>2650</td>
<td>mV</td>
<td>Variable charge voltage. Used when T_VLUT_1 &lt; TEMP &lt; T_VLUT_2.</td>
</tr>
<tr>
<td>V_CHG_3</td>
<td>2</td>
<td>2590</td>
<td>mV</td>
<td>Variable charge voltage. Used when T_VLUT_2 &lt; TEMP &lt; T_VLUT_3.</td>
</tr>
<tr>
<td>V_CHG_4</td>
<td>2</td>
<td>2530</td>
<td>mV</td>
<td>Variable charge voltage. Used when T_VLUT_3 &lt; TEMP &lt; T_VLUT_4.</td>
</tr>
<tr>
<td>V_CHG_5</td>
<td>2</td>
<td>2500</td>
<td>mV</td>
<td>Variable charge voltage. Used when T_VLUT_4 &lt; TEMP &lt; T_VLUT_5.</td>
</tr>
<tr>
<td>V_CHG_6</td>
<td>2</td>
<td>2470</td>
<td>mV</td>
<td>Variable charge voltage. Used when T_VLUT_5 &lt; TEMP &lt; T_VLUT_6.</td>
</tr>
<tr>
<td>V_CHG_7</td>
<td>2</td>
<td>2410</td>
<td>mV</td>
<td>Variable charge voltage. Used when T_VLUT_6 &lt; TEMP &lt; T_VLUT_7.</td>
</tr>
<tr>
<td>V_CHG_8</td>
<td>2</td>
<td>2350</td>
<td>mV</td>
<td>Variable charge voltage. Used when T_VLUT_7 &lt; TEMP &lt; T_VLUT_8.</td>
</tr>
<tr>
<td>V_CHG_9</td>
<td>2</td>
<td>2250</td>
<td>mV</td>
<td>Variable charge voltage. Used when T_VLUT_8 &lt; TEMP.</td>
</tr>
<tr>
<td>V_FLT_0</td>
<td>2</td>
<td>2380</td>
<td>mV</td>
<td>Variable float voltage. Used when TEMP &lt; T_VLUT_0.</td>
</tr>
<tr>
<td>V_FLT_1</td>
<td>2</td>
<td>2370</td>
<td>mV</td>
<td>Variable float voltage. Used when T_VLUT_0 &lt; TEMP &lt; T_VLUT_1.</td>
</tr>
<tr>
<td>V_FLT_2</td>
<td>2</td>
<td>2350</td>
<td>mV</td>
<td>Variable float voltage. Used when T_VLUT_1 &lt; TEMP &lt; T_VLUT_2.</td>
</tr>
<tr>
<td>V_FLT_3</td>
<td>2</td>
<td>2330</td>
<td>mV</td>
<td>Variable float voltage. Used when T_VLUT_2 &lt; TEMP &lt; T_VLUT_3.</td>
</tr>
<tr>
<td>V_FLT_4</td>
<td>2</td>
<td>2310</td>
<td>mV</td>
<td>Variable float voltage. Used when T_VLUT_3 &lt; TEMP &lt; T_VLUT_4.</td>
</tr>
<tr>
<td>V_FLT_5</td>
<td>2</td>
<td>2300</td>
<td>mV</td>
<td>Variable float voltage. Used when T_VLUT_4 &lt; TEMP &lt; T_VLUT_5.</td>
</tr>
<tr>
<td>V_FLT_6</td>
<td>2</td>
<td>2290</td>
<td>mV</td>
<td>Variable float voltage. Used when T_VLUT_5 &lt; TEMP &lt; T_VLUT_6.</td>
</tr>
<tr>
<td>V_FLT_7</td>
<td>2</td>
<td>2270</td>
<td>mV</td>
<td>Variable float voltage. Used when T_VLUT_6 &lt; TEMP &lt; T_VLUT_7.</td>
</tr>
<tr>
<td>V_FLT_8</td>
<td>2</td>
<td>2250</td>
<td>mV</td>
<td>Variable float voltage. Used when T_VLUT_7 &lt; TEMP &lt; T_VLUT_8.</td>
</tr>
<tr>
<td>V_FLT_9</td>
<td>2</td>
<td>2200</td>
<td>mV</td>
<td>Variable float voltage. Used when T_VLUT_8 &lt; TEMP.</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>VhhVh</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>VllVI</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>
</tbody>
</table>
### LUT 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>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>-10206</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>VLUT_N</td>
<td>1</td>
<td>10</td>
<td>integer</td>
<td>Number of entries in V_CHG and VFLT tables.</td>
</tr>
<tr>
<td>T_VLUT_0</td>
<td>1</td>
<td>0</td>
<td>coded</td>
<td>Temperature point for V_CHG and V FLT tables (°C * 10 + 200)/4.</td>
</tr>
<tr>
<td>T_VLUT_1</td>
<td>1</td>
<td>25</td>
<td>coded</td>
<td>Temperature point for V_CHG and V FLT tables (°C * 10 + 200)/4.</td>
</tr>
<tr>
<td>T_VLUT_2</td>
<td>1</td>
<td>50</td>
<td>coded</td>
<td>Temperature point for V_CHG and V FLT tables (°C * 10 + 200)/4.</td>
</tr>
<tr>
<td>T_VLUT_3</td>
<td>1</td>
<td>75</td>
<td>coded</td>
<td>Temperature point for V_CHG and V FLT tables (°C * 10 + 200)/4.</td>
</tr>
<tr>
<td>T_VLUT_4</td>
<td>1</td>
<td>100</td>
<td>coded</td>
<td>Temperature point for V_CHG and V FLT tables (°C * 10 + 200)/4.</td>
</tr>
<tr>
<td>T_VLUT_5</td>
<td>1</td>
<td>112</td>
<td>coded</td>
<td>Temperature point for V_CHG and V FLT tables (°C * 10 + 200)/4.</td>
</tr>
<tr>
<td>T_VLUT_6</td>
<td>1</td>
<td>125</td>
<td>coded</td>
<td>Temperature point for V_CHG and V FLT tables (°C * 10 + 200)/4.</td>
</tr>
<tr>
<td>T_VLUT_7</td>
<td>1</td>
<td>150</td>
<td>coded</td>
<td>Temperature point for V_CHG and V FLT tables (°C * 10 + 200)/4.</td>
</tr>
<tr>
<td>T_VLUT_8</td>
<td>1</td>
<td>175</td>
<td>coded</td>
<td>Temperature point for V_CHG and V FLT tables (°C * 10 + 200)/4.</td>
</tr>
</tbody>
</table>

### LED Parameters

| LED1State1     | 1       | 00000000b     | binary | LED1 display during state 1: Charge Pending. |
| LED1State2     | 1       | 00000000b     | binary | LED1 display during state 2: Charge Qualification. |
| LED1State3     | 1       | 00000000b     | binary | LED1 display during state 3: Current Regulation. |
| LED1State4     | 1       | 00000000b     | binary | LED1 display during state 4: Voltage Regulation. |
| LED1State5     | 1       | 00000000b     | binary | LED1 display during state 5: Float Charge. |
| LED1State6     | 1       | 00000000b     | binary | LED1 display during state 6: Charge Cycle Complete. |
| LED1State7     | 1       | 00000000b     | binary | LED1 display during state 7: Charge Suspend. |
| LED1State8     | 1       | 00000000b     | binary | LED1 display during state 8: Unused. |
| LED2State1     | 1       | 00000000b     | binary | LED2 display during state 1: Charge Pending. |
| LED2State2     | 1       | 00000000b     | binary | LED2 display during state 2: Charge Qualification. |
| LED2State3     | 1       | 00000000b     | binary | LED2 display during state 3: Current Regulation. |
| LED2State4     | 1       | 00000000b     | binary | LED2 display during state 4: Voltage Regulation. |
| LED2State5     | 1       | 00000000b     | binary | LED2 display during state 5: Float Charge. |
| LED2State6     | 1       | 00000000b     | binary | LED2 display during state 6: Charge Cycle Complete. |
| LED2State7     | 1       | 00000000b     | binary | LED2 display during state 7: Charge Suspend. |
| LED2State8     | 1       | 00000000b     | binary | LED2 display during state 8: Unused. |
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>F:</th>
<th>PWMP&lt;6:5&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>PER &lt;4:0&gt;</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>8000</td>
</tr>
<tr>
<td>1</td>
<td>4000</td>
</tr>
<tr>
<td>2</td>
<td>2667</td>
</tr>
<tr>
<td>3</td>
<td>2000</td>
</tr>
<tr>
<td>4</td>
<td>1600</td>
</tr>
<tr>
<td>5</td>
<td>1333</td>
</tr>
<tr>
<td>6</td>
<td>1143</td>
</tr>
<tr>
<td>7</td>
<td>1000</td>
</tr>
<tr>
<td>8</td>
<td>889</td>
</tr>
<tr>
<td>9</td>
<td>800</td>
</tr>
<tr>
<td>10</td>
<td>727</td>
</tr>
<tr>
<td>11</td>
<td>667</td>
</tr>
<tr>
<td>12</td>
<td>615</td>
</tr>
<tr>
<td>13</td>
<td>571</td>
</tr>
<tr>
<td>14</td>
<td>533</td>
</tr>
<tr>
<td>15</td>
<td>500</td>
</tr>
<tr>
<td>16</td>
<td>471</td>
</tr>
<tr>
<td>17</td>
<td>444</td>
</tr>
<tr>
<td>18</td>
<td>421</td>
</tr>
<tr>
<td>19</td>
<td>400</td>
</tr>
<tr>
<td>20</td>
<td>381</td>
</tr>
<tr>
<td>21</td>
<td>364</td>
</tr>
<tr>
<td>22</td>
<td>348</td>
</tr>
<tr>
<td>23</td>
<td>333</td>
</tr>
<tr>
<td>24</td>
<td>320</td>
</tr>
<tr>
<td>25</td>
<td>308</td>
</tr>
<tr>
<td>26</td>
<td>296</td>
</tr>
<tr>
<td>27</td>
<td>286</td>
</tr>
<tr>
<td>28</td>
<td>276</td>
</tr>
<tr>
<td>29</td>
<td>267</td>
</tr>
<tr>
<td>30</td>
<td>258</td>
</tr>
<tr>
<td>31</td>
<td>250</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 to report if current is flowing by the Mode parameter.

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

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

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 in the “Functional Description: Lead Chemistry” section.

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 \( \text{PWMAdjust4} \), etc.

Table 4 shows the PWM adjustment factors as a function of current difference and voltage difference when comparing feedback to requirements:

Table: PWM Adjustment Factors

<table>
<thead>
<tr>
<th>Voltage Zones</th>
<th>Current Zones</th>
<th>-Cl</th>
<th>-Cl</th>
<th>Cl to Chl</th>
<th>Ch</th>
<th>Chh</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; Vhh</td>
<td>-PWMAdjust1</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>-PWMAdjust4</td>
<td>-PWMAdjust2</td>
</tr>
<tr>
<td>Vh to -VI</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></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>
<td></td>
</tr>
<tr>
<td>&lt; -VII</td>
<td>+PWMAdjust3</td>
<td>+PWMAdjust4</td>
<td>0</td>
<td>-PWMAdjust4</td>
<td>-PWMAdjust2</td>
<td></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>00</td>
<td>OFF</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>01</td>
<td>Flash N + 1 Times, Pause, Repeat</td>
<td>Flash Count = N + 1</td>
<td>On Time = Off Time = F + 1&lt;br&gt;Pause Time = (F + 1) * 5&lt;br&gt;Max = 3</td>
</tr>
<tr>
<td>10</td>
<td>On</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>11</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:

\[ VR = \frac{A/D_{RAW} \times 16384}{BandgapCF} \]

BandgapCF is typically around 248 since:

\[ \frac{VR}{VDD} = \frac{A/D_{RAW}(FULLSCALE)}{1212/5000 \times 1023} = 248 \]

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:

\[ \text{Current} = \frac{A/D_{RAW} \times \text{CurrentCF}}{65536} \]

\[ \text{Current} = \left( \frac{A/D_{RAW} \times VR/16384}{1212/5000 \times 1023} \right) \times \text{CurrentCF}/65536 \]

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

\[ \text{Current(full scale)} = \frac{V_{REF}/\text{AMPgain}/\text{SHUNT}}{5000/19.6/0.100} = 2551 \, \text{mA} \]

\[ 2551 = 1023 \times \text{CurrentCF} \]

\[ \text{CurrentCF} = 2.494 \]

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

\[ \text{CurrentCF Base} = 1024 \]

\[ \text{CurrentCF} = 2553 \]
**VOLTAGE**

The voltage reading is calibrated or translated from the raw A/D measurement \( (A_{\text{D\_raw}}) \) as follows:

**EQUATION 9:**

When referenced to VR:
\[ \text{Voltage} = A_{\text{D\_raw}} \times \text{VoltageCF}/1024 \]

When referenced to VDD:
\[ \text{Voltage} = (A_{\text{D\_raw}} \times VR/16384) \times \text{VoltageCF}/1024 \]

Where \( \text{VoltageCF} \) is determined as follows:
\[ \text{Voltage} = A_{\text{D\_raw}} \times \text{VoltageCF} \]
\[ A_{\text{D\_raw}} = (\text{Voltage} \times \text{Cells}) / \text{R/VREF} \times 1023 \]

Where:
\( R \) = Resistor Divider Ratio
\( \text{VREF} = 5000 \text{ mV} \)

This means:
\[ \text{Voltage} = \text{VoltageCF} \times \text{Voltage} \times \text{Cells} \times R/\text{VREF} \times 1023 \]

or
\[ \text{VoltageCF} = \text{VREF} / (\text{Cells} \times R \times 1023) \]

and using integer arithmetic:
\[ \text{VoltageCF} = \text{VoltageCF}/1024 \]

So that:
\[ \text{Voltage} = \text{VoltageCF} \times A_{\text{D\_raw}} / 1024 \]

Table 6 shows the typical \( \text{VoltageCF} \) 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>10.50</td>
<td>10.0</td>
<td>0.4878</td>
<td>5130</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>20.50</td>
<td>10.0</td>
<td>0.3279</td>
<td>5088</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>30.90</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 \( (A_{\text{D\_raw}}) \) as follows:

**EQUATION 10:**

\[ \text{Temperature} = A_{\text{D\_raw}} \times \text{TempCF}/8192 \]

Where temperature is in the internal units of:
\( (^\circ C + 20) \times 10 \)

\( \text{TempCF} \) 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 located. The look-up table was developed by rating raw A/D values; that is why \( \text{TempCF} \) 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. \( T_{\text{LUT\_N}} \) represents the number of entries in the table, in this case eight entries.

<table>
<thead>
<tr>
<th>A/D Reading</th>
<th>Slope</th>
<th>Y-intercept</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; T_{LUT_T_0}</td>
<td>T_{LUT_M_0}</td>
<td>T_{LUT_B_0}</td>
</tr>
<tr>
<td>&lt; T_{LUT_T_1}</td>
<td>T_{LUT_M_1}</td>
<td>T_{LUT_B_1}</td>
</tr>
<tr>
<td>&lt; T_{LUT_T_2}</td>
<td>T_{LUT_M_2}</td>
<td>T_{LUT_B_2}</td>
</tr>
<tr>
<td>&lt; T_{LUT_T_3}</td>
<td>T_{LUT_M_3}</td>
<td>T_{LUT_B_3}</td>
</tr>
<tr>
<td>&lt; T_{LUT_T_4}</td>
<td>T_{LUT_M_4}</td>
<td>T_{LUT_B_4}</td>
</tr>
<tr>
<td>&lt; T_{LUT_T_5}</td>
<td>T_{LUT_M_5}</td>
<td>T_{LUT_B_5}</td>
</tr>
<tr>
<td>&lt; T_{LUT_T_6}</td>
<td>T_{LUT_M_6}</td>
<td>T_{LUT_B_6}</td>
</tr>
<tr>
<td>&gt; T_{LUT_T_6}</td>
<td>T_{LUT_M_7}</td>
<td>T_{LUT_B_7}</td>
</tr>
</tbody>
</table>

The typical values are:

<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 16HV785. The command is eight bits long. After this, eight data bits are either written to the 16HV785, or read from the 16HV785.

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

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 ten 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 11: SINGLE PIN SERIAL TIMING

Break Timing
- Break bit
- Break Reset

Host to 16HV785
- Start bit
- Data bit
- Stop bit

16HV785 to Host
- Start bit
- Data bit
- Stop bit

CMD and Data Protocol
- Break
- LSB
- CMD Address
- MSB
- LSB
- Data to or from 16HV785

Communication Example
- CMD Addr = 04 hex
- Data = 25 hex
- Break
- 0 1 1 0 0 0 0 0 1 0 1 0 0 1 0 0
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>1 = EEPROM write is in progress; busy</td>
</tr>
<tr>
<td>6</td>
<td>EE_Err</td>
<td>1 = 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>1 = Regulation active</td>
</tr>
<tr>
<td>3</td>
<td>CHGCON</td>
<td>1 = Charge controller active</td>
</tr>
<tr>
<td>2</td>
<td>SIM_ACTIVE</td>
<td>1 = 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>1 = Suspend/skip all processing (used when writing EEPROM)</td>
</tr>
<tr>
<td>4</td>
<td>CHGCON_OFF</td>
<td>1 = 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>1 = Indirect memory addressing refers to EEPROM</td>
</tr>
<tr>
<td>0</td>
<td>MEMBANK_23</td>
<td>1 = 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 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 Register 0 and the bank indicated by Register 2 (bits&lt;1:0&gt;); Register 0 will be post-incremented</td>
</tr>
</tbody>
</table>
Host Driven Operations

Host driven operations refer to a host communicating with the 16HV785 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> = 1 0); 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>Table 10: EEPROM WRITE CONTROL PACKET</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Byte</strong></td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>N + 1</td>
</tr>
<tr>
<td>N + 2</td>
</tr>
</tbody>
</table>

The total procedure is a five step process:
1. Suspend normal operation: Set Communication Register 2 = 0x20 (set 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 description above. The names in the functional description match the names in 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 description. Variable names are defined for hardware interface registers like A/D control and data, timers, PWM configuration and GPIO.

```c
;=====================================================================
;--- 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 0x03
#include "p16f876.inc"

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

;--- registers: special function

r_indf        equ INDF
r_tmr0        equ TMR0
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 TMRL1
r_tmr1h       equ TMR1H
r_t1con       equ T1CON
r_tmr2        equ TMR2
r_t2con       equ T2CON
r_ccpr1l      equ CCPR1L
```
r_ccpr1h equ CCPR1H
r_ccplcon 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_piei equ PIE1
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 IOCA
r_refcon equ REFCON
r_vrcon equ VRCON
r_eedata equ EEDATA
r_eeadr equ EEADR
r_eedata equ EEDATA
r_eeadr equ EEADR
r_eecn1 equ EECN1
r_eecn2 equ EECN2
r_adresl equ ADRESL
r_adcon1 equ ADCON1
r_pcmcon1 equ PWCN1
r_pcmcon0 equ PWCN0
r_pwmclk equ PWCLK
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 ram1_start 0xa0
#define ram1_end 0xef
#define raml_length ram1_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 res 1 ; adc result - channel 0
r_adc_0_L res 1 ;

r_adc_0_H res 1 ;

r_adc_1 res 1 ; adc result - channel 1
r_adc_1_L res 1 ;

r_adc_1_H res 1 ;

r_adc_2 res 1 ; adc result - channel 2
r_adc_2_L res 1 ;

r_adc_2_H res 1 ;

r_adc_3 res 1 ; adc result - channel 3
r_adc_3_L res 1 ;

r_adc_3_H res 1 ;
r_adc_4     equ $               ;
r_adc_4_L    res 1               ; adc result - channel 4
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_contrl_1 res 1               ;
r_led_config_2 res 1               ;
r_led_contrl_2 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_tcode res 1               ;
;debug
r_not_used res 5               ;
r_mode3      res 1               ;
 r_timer_d2   res 1               ;
 ;-----------------------------------------------------------
 ;--- registers: bank0,1,2,3 (common)
 ;-----------------------------------------------------------
 org 0x70      ; *** bank 0 (common area)
 r_shadow_1   res 1               ; assorted bit flags
 r_shadow_2   res 1               ; assorted bit flags
 r_shadow_3   res 1               ; assorted bit flags
 ;r_shadow_4  res 1               ; assorted bit flags
 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               ; aassorted 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_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_cofs_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_vrchg_dis   r_mode2, 2   ; voltage recharge - disable
#define flag0_mode_vregco_dis  r_mode2, 1   ; regulation voltage cutoff - disable
#define flag0_mode_pwmas_dis   r_mode2, 0   ; pwm auto shutdown - disable
#define flag0_mode_suspend_4ever r_mode3, 5 ; suspend forever
#define flag0_mode_refloat     r_mode3, 4   ; re-float enable
#define flag0_mode_postfloat   r_mode3, 3   ; float after CC,CV cycle
#define flag0_mode_v_flt_k     r_mode3, 2   ; use constant v float (not vlut)
#define flag0_mode_v_reg_k     r_mode3, 1   ; use constant v reg/charge (not vlut)
#define flag0_mode_float       r_mode3, 0   ; skip to float state immediately

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_e 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_cregxEE_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_busy r_flags_1, 7
#define flag0_creg_rq r_flags_1, 6
#define flag0_creg_err r_flags_1, 5
#define flag0_creg_simdata_ready r_flags_1, 4
#define flag0_chg_state_timer r_flags_1, 3
#define flag0_math_temp r_flags_1, 2
#define flag0_timer_0 r_flags_1, 1
#define flag0_led_timer r_flags_1, 0

;--- trigger flags - lion
;#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_t_le_vreg r_flags_2, 4
;#define flag_t_le_tmin r_flags_2, 3
;#define flag_t_le_tmaxchgi r_flags_2, 2
;#define flag_t_le_tmaxchge r_flags_2, 1
;#define flag_t_le_tpchge r_flags_2, 0

;--- trigger flags - nimh
;#define flag_v_le_vpchg_nm r_flags_2, 7
;#define flag_t_le_tmaxchgi r_flags_2, 6
;#define flag_t_le_tmaxchge r_flags_2, 5
;#define flag_t_le_tmaxchge r_flags_2, 4
;#define flag_v_le_vpchg_nmm r_flags_2, 3
;#define flag_v_le_rchg_nm r_flags_2, 2
;#define flag_v_le_dchg_nm r_flags_2, 1

;#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_state_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

;*** WARNING: DO NOT MOVE: flag_led_2_save
;*** WARNING: DO NOT MOVE: flag_led_1_save
;#define flag_led_2_save r_flags_4, 7
;#define flag_led_1_save r_flags_4, 6
#define flag_adc_3_sim r_flags_4, 7

#define flag_adcset_2_rdy r_flags_4, 5
#define flag_adcset_2_rq r_flags_4, 4
#define flag_adcset_1_rdy r_flags_4, 3
#define flag_adcset_1_rq r_flags_4, 2
#define flag_adcset_0_rdy r_flags_4, 1
#define flag_adcset_0_rq 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_vreg2 r_flags_5, 1
#define flag_vreg1 r_flags_5, 0
#define BN_CHGN_TSEL 0
#define flag_chg_t1_done _r_flags_6, 7
#define flag_chg_t2_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

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;; option reg
#define CLOCK_8MHZ
clk_p equ .8000000 ; (mhz) clock frequency
OSCCON_DEFAULT equ 0x70

;;;; option
#define option_default equ 1<<NOT_RAPU | 0x02
tmrl_default       equ 0x10    ; 2:1 scale, lusec tic
TIME_COMM_USEC_T     equ 1

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)

;debug
;TIMER_D_SEC         equ .15      ; (sec) timer resolution (class d)
;debug
;TIMER_A1_MSEC       equ .2       ; (msec) regulation timer
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        1<<IOCA5

;--- WPua
#define       WPUA_DEFAULT        1<<WPUA5

;--- OPA1CON
;debug OVP
#define       OPA1CON_DEFAULT     1<<OPAON
#define       OPA1CON_DEFAULT     0<<OPAON
;--- OPA2CON
#define OPA2CON_DEFAULT 1<<OPAON

;--- CM1: INPUTS: RA1/C1Ref SPEED: NORM, OUTPUT: INT
#define CM1CON0_DEFAULT 1<<C1R | 1<<C1SP | 1<<C1ON
;debug OVP
#define CM1CON0_DEFAULT 1<<C1R | 1<<C1SP | 1<<C1ON | 1<<C1GE

#define CM2CON0_DEFAULT 1<<C2ON | 0<<C2POL | .0<<C2SP | 0<<C2R | .3<<C2CH0
#define CM2CON1_DEFAULT 0

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

;--- REFCON: ENABLED
#define REFCON_DEFAULT 1<<VREN | 0<<VRGE

#define ANSEL0_DEFAULT 1<<ANS0 | 1<<ANS1 | 0<<ANS2 | 1<<ANS3 | 1<<ANS4 | 0<<ANS5
#define ANSEL1_DEFAULT 1<<ANS8 | 1<<ANS9 | 1<<ANS10 | 1<<ANS11

#define PWMCON0_AS_DIS 0<<BLANK2 | 1<<PH2EN
#define PWMCON0_AS_EN 0<<BLANK2 | 1<<PH2EN | 1<<PASEN

#define PWMCON0_AS_DIS 0<<BLANK2 | 1<<PH2EN
#define PWMCON0_AS_EN 0<<BLANK2 | 1<<PH2EN | 1<<PASEN

#define PWMCLK_DEFAULT .0<<PWMP0 | .19<<PER0
#define PWMPH2_DEFAULT 0<<POL | 1<<C2EN | 0<<C1EN | .1<<PH0

#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
;debug
#define ADC_TAQ .51

#define CCP1CON_DEFAULT 0x0c

;--- constants: interrupts

#define INTCON_DEFAULT 1<<T0IE

#define PWM_DEFAULT equ .000
;--- EE MAP

EE_PATTERN equ .0
EE_NCELLS equ .2
EE_CAPACITY equ .19
EE_FWM_FREQ equ .21
EE_MODE equ .22
EE_MODE2 equ .23
EE_OSC_TRIM equ .24

EE_LBD1_CFG equ .32
EE_LBD2_CFG equ .40

EE_RGB_P1 equ .52
EE_RGB_P2 equ .53
EE_RGB_P3 equ .54
EE_RGB_P4 equ .55
EE_RGB_VHH_VH equ .56
EE_RGB_VH equ .57
EE_RGB_VL equ .58
EE_RGB_VLL_VL equ .59
EE_RGB_CNUL equ .60
EE_RGB_VSAFETY equ .61

EE_CHG_C equ .67
EE_CHG_C_FLOAT equ .69
EE_CHG_C_MIN equ .71
EE_CHG_TI_CC equ .73
EE_CHG_TI_CV equ .74
EE_CHG_TI_FLOAT equ .75
EE_CHG_TI_REFLOAT equ .76
EE_CHG_TI_SUSPEND equ .77
EE_CHG_V_CHG_K equ .78
EE_CHG_V_FLT_K equ .80
EE_CHG_V_MIN_BP equ .82

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 .128

EE_MODE3 equ .170
EE_VLUT_N equ .171
EE_VLUT_T equ .172
EE_VLUT_CHG equ .181
EE_VLUT_FLT equ .201

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

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

```assembly
org 0x00
vector_reset:                           
goto      start                    

org 0x0004                         
isr:                                    
vector_isr:                             
movwf     r_isr_w                  ; save context
swapf     r_status, w              
clf      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_pir1, TMR1IF           
goto      isr_t1_x                 
bcf       r_pir1, 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.  DS01015A-page 33
#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.

```assembly
; 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

; --- anssel
    movlw ANSEL0_DEFAULT
    movwf r_ansel0
    movlw ANSEL1_DEFAULT
    movwf r_ansel1
    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
    bset    r_tlcon, TMR1ON          ; enable timer 1

;--- clear ram
    bcf     r_status, IRP            ;
    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                      ;
    bset    r_status, RP1            ; *** bank=2
    movwf   r_pwmclk                 ;
    movlw   PWMPH2_DEFAULT           ;
    movwf   r_pwmph2                 ;
    bcf     r_status, RP1            ; *** bank=0
    movlw   PWMCN00_AS_DIS           ;
    btfss   flag0_mode_pwmas_dis     ;
    movlw   PWMCN00_AS_EN            ;
    movlw   PWMCN00_AS_DIS           ;
    bcf     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:
    btfss flag0_modeoscout
    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

---

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

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

;debug
; call     toggle

main_x:                                 ;
goto      main                     ;
```
FIGURE 12: MAIN LOOP FLOWCHART

- \( r_{\text{comm\_reg\_3}<2>} = 1? \) (Firmware Reset)
  - Yes: GOTO VECTOR_RESET

- \( r_{\text{comm\_reg\_2}<5>} = 1? \) ("Suspend" mode)
  - Yes

  - call \texttt{adc\_svc} (ADC Service)
  - call \texttt{adc\_start} (ADC Start)
  - call \texttt{check\_triggers} (Compare ADC Results to Threshold)
  - call \texttt{chg\_state\_svc} (Service Charge Control)
  - call \texttt{regulate} (Dynamic Voltage/Current Control)

  - call \texttt{led\_svc} (LED Service)
  - call \texttt{timer\_svc} (Service Timer(s))
  - call \texttt{ee\_write\_buf} (Execute EEPROM Control Packet)
  - call \texttt{ccmd\_svc} (Service Comm Control Commands)
  - call \texttt{status\_build} (Build Status Byte, i.e., "comm\_reg\_1")
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 description on communication are implemented below, including all of the functionality in each bit of the Communication registers. Functions like 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 bitmaps 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.

;---------------------------------------
;--- 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()
;---------------------------------------
pwm_config:                             
   bsf       r_status, RP0            ; *** bank=1
   movlw     0xff                     
   movwf     r_pr2                    
   bcf       r_status, RP0            ; *** 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_ccplcon                
   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.

```asm
;---------------------------------------
;--- pwm_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_ccp1con                ;
    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 description 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 ...
clrw      r_accB_L                  ; set result = 0
clrw      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                             
```

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 chksum 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[IRP] 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:
        ; add data byte
        addwf  r_indf, w
        incf  r_fsr, f
        decfsz r_tempc_1, f
        goto  chksum_loop
    return


Thermistor Temperature Processing (Subroutine of ADC_Service)

As described in functional description, 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_buf2

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

vlut_index_tcode:  
    movf      r_tcode, w               ; use 1-byte temperature as key

vlut_index:  
    movwf     r_temp_4                 ; save key

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

;--- setup read ee to buffer  
    movwf     r_temp_1                 ;
    movlw     EE_VLUT_T               ;
    call      ee_read_buf              ; read temperature vector into ram
```
AN1015

;--- index into temperature vector 
; goto  lut_index_buf2 

----------

;--- vlut_fetch() 
;--- vlut_fetch_flt() 
;--- vlut_fetch_chg()
/
call:
/   w = vlut index
----------

vlut_fetch_flt: 
   movwf  r_ee_addr 
   movlw  EE_VLUT_FLT 
   goto   vlut_fetch 

vlut_fetch_chg: 
   movwf  r_ee_addr 
   movlw  EE_VLUT_CHG 

vlut_fetch: 
   bcf    r_status, C 
   rlfs   r_ee_addr, f 
   addwf  r_ee_addr, f 
   call   ee_read_ia 
   movwf  r_reg_v 
   call   ee_read_ia 
   movwf  r_reg_v + 1 
   return 

----------

;--- ee_read_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 


----------

;--- lut_index() - index into arbitrary buffer 
;--- lut_index_buf2() - index into buffer: buf2 
/
determine index by scanning vector with key
index will be the last element of the vector
where key <= vector[index]
/
call:
/
   w   ram location
/   r_temp_3  vector length
/   r_temp_4  search key

return:
/
   w   vector index
/   r_temp_3  vector length
/   r_temp_4  search key
/
----------

lut_index_buf2: 
   movlw  r_buf2 
   ; buffer location
lut_index:                              ;
    movwf  r_fsr                       ;
    movf   r_temp_3, w                ;
    movwf  r_temp_2                   ; save copy of vector length
lut_index_loop:                         ;
    movf   r_temp_4, w                ;
    subwf  r_indf, w                  ;
    btfsc  r_status, C                ;
    goto  lut_index_x                 ;
    incf  r_fsr, 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.

;-----------------------------------------------------------
;--- 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       flag_comm_bit           ;
  cmff      r_tmrl1, w             ;
  sublw     TIME_COMM_BREAK_T       ;
  movwf     r_templ_1               ;
  sublw     TIME_COMM_1_MAX_T       ;
  btfsb    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_isr_reg_write ;

comm_isr_rcv_data: ;
; movwf r_comm_data ;
movlw 0x70 ;
andwf r_comm_data_cmnd, w ;
btfss r_status, Z ;
goto comm_isr_x ;
btfsc r_comm_data_cmnd, 3 ;
goto comm_isr_reg_write_ia ;

comm_isr_reg_write: ;
movlw 0x07 ;
xorf 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 ;
goto comm_isr_reg_read ;

comm_isr_reg_read: ;
movf r_indf, w ;
movwf r_comm_data ;
goto comm_xmit_byte_prep ;

comm_isr_rcv_data_ia: ;
bsf r_status, IRP ;
btfs r_flag0_creg_membank_23 ;
bcf r_status, IRP ;
movf r_comm_reg_0, w ;
movwf r_fsr ;
btfss r_comm_data_cmnd, 7 ;
goto comm_isr_reg_read_ia ;

comm_isr_reg_write_ia: ;
movf r_comm_data, w ;
movwf r_indf ;
btfsc 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 r_comm_data_cmnd, 2           ;
  incf r_comm_reg_0, f                ;
  goto comm_xmit_byte_prep            ;

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_hi                    ;
  decfsz r_comm_count, f              ;
  goto $+2                            ;
  goto comm_rcv_byte_prep             ; setup for receive
  bsf flag_comm_H2L                   ;
  movlw TIME_COMM_B1_HI_T             ;
  btfss r_comm_data, 0                ;
  movlw TIME_COMM_B0_HI_T             ;
  call comm_timer_load                ;
  rrf r_comm_data, f                  ; rotate to next bit

comm_isr_x:                              ;
  return                              ;

;-----------------------------------------------------------
;--- comm: transmit byte prep
;-----------------------------------------------------------
comm_xmit_byte_prep:                    ;
  movwf r_comm_data                   ; load data byte
  movlw .8                            ;
  movwf r_comm_count                  ;
  bcf flag_comm_xmit                  ;
  bcf flag_comm_H2L                   ;
  call comm_pin_output                ;
  movlw TIME_COMM_REPLY_T             ;
  call comm_timer_load                ;
  return                              ;

;-----------------------------------------------------------
;--- comm timer load
;-----------------------------------------------------------
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 int(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
    bsf    r_status, RP0  ; *** bank=0
    return

;---------------------------------------
;--- set comm pin to input
;---------------------------------------
comm_pin_input:
    btfss  flag_comm_active
    return
    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 GPIO 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.

;-------------------------------------------------
;--- led_init_1() - configure/initialize 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() - configure/initialize 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
; [7:6] - mode
; [5:3] - 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_svc_1_x:                  ;

;--- 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_  ;
btfsr r_indf, 3  ;
goto led_svc_mode2  ;
goto led_svc_mode3  ;

led_svc_modex_

btfsr r_indf, 3  ;
goto led_svc_modex1  ;
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_model:

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

led_svc_mode1_on:

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

clrf r_indf  ; load long "off" time
decf r_fsr, f
movf r_indf, w
andlw 0x07
incf r_fsr, f
movwf r_indf

incf r_indf, f
bcf r_status, C
rlf r_indf, f
bcf r_status, C
rlf r_indf, f
addwf r_indf, f
movlw 0xf0
andwf r_indf, w
movlw 0x0f
btfsr r_status, Z
movwf r_indf

goto led_svc_x ;

led_svc_mode1_off:
    bsf r_indf, 7 ;
    movf r_indf, w ;
    andlw 0x70 ;
    btfs r_status, Z ;
    goto led_svc_mode1_off_ ; blink count<>0, decrement & go ...;
    decf r_far, f ; re-load blink count
    movf r_indf, w ;
    andlw 0x70 ;
    incf r_far, 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_far, f ; count==0 ?
    movlw 0x0f ;
    andwf r_indf, w ;
    btfs r_status, Z ;
    goto led_svc_dec_x ; --- no, decrement on exit ...
    btfs r_indf, 7 ; led currently on ?
    goto led_svc_mode3_off ; --- no, jump ...

led_svc_mode3_on:
    bcf r_indf, 7 ;

led_svc_mode3_on_:
    decf r_far, f ;
    movf r_indf, w ;
    goto led_svc_mode3_ ;

led_svc_mode3_off:
    bsf r_indf, 7 ; point to config register
    swapf r_indf, w ;

led_svc_mode3_:
    andlw 0x07 ;
    incf r_far, f ;
    iorwf r_indf, f ;
    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_x              
    clrf r_timer_a1               
    bsf flag_reg_timer           

;--- class "b" timer
    ;--- class "b" timer
    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     
    decf r_timer_b1, f            

;--- class "c" timer
    ;--- class "c" timer
    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                

;--- class "d" timer
    ;--- class "d" timer
    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_di:                            
    movf r_timer_di, f             
    btfsc r_status, Z              
    goto timer_svc_di_x            
    decf r_timer_di, f             
    btfsc r_status, Z              
    bsf flag_chg_tti_done          

    timer_svc_di_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.

```assembly
;-----------------------------------------------------------
;--- move_ee_ram() - move data from eeprom to ram
;
;    call:
;        r_fsr          ram address
;        r_ee_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_ee_addr                ; save address

ее_read_ia:                             ;
    call      ee_read               ;
    incf r_ee_addr, f             ;
return                             ;

ее_read_waddr:                          ;
    movwf r_ee_addr                ; save address

ее_read:                               ;
    bcf r_status, RP1             ; *** bank=1
    bcf r_status, RP0             ;
    movf r_ee_addr, w             ; load address
    movwf r_eeadr                  ;
    bsf r_eecom1, 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_ee_addr, f ;
;  return ;

; ee_write_waddr:
;  movwf r_ee_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 ;
;  bsf 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 description.

;---------------------------------------
;--- 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             ; 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_B              ; move result to register: avg
  movlw   r_adc_avg                ;
  call    math_load_D              ; accD = result
  clrf    r_adc_accum_count        ; reset accumlation registers
  clrf    r_adc_accum              ;
  clrf    r_adc_accum + 1          ;

;---------------------------------------
;--- 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
  movlw  low .16384               ;
  movwf   r_accA_L                ;
  movlw  high .16384              ;
  movwf   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 ER_CAL_ADC_0 ;
call ee_read_waddr_ia ;
movwf r_accC_L   ;
call ee_read_ia   ;
movwf r_accC_H   ;
call math_div_32  ;
movf r_accB_L, w  ;
movf r_accB_H, w  ;
movf r_adc_0_L   ;
movf r_adc_0_H   ;
adc_svc_0_x:      ;
go to adc_svc_x   ;
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 ER_CAL_ADC_1  ;
call loadA_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_load_A ; accA = result

;--- remove offset (option)
adc_svc_1_cofs_x: ;
call math_add_16 ; accA + accB = result - offset
btc r_raccB_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_cofs_x: ;
movlw r_adc_1_L ;
call math_move_B ;
go to 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 ; save result
    movlw EE_CAL_ADC_2
    call load_A_ee
    call math_mul_16_prep_
    movlw .2
    call math_shift_BC
    movf r_adc_2_H, w
    movwf r_adc_2_L
    movf r_accC_H, w
    movwf r_adc_2_L
    movf 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 ; constant temperature option
    goto adc_svc_3_k_x
    movlw EE_T_DEFAULT-1
    call load_B_ee ; accB_H = temp param
    clrf r_accB_L ; accB_L = 0
    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 ; save result
    movlw EE_CAL_ADC_3
    call load_A_ee
    call math_mul_16_prep_
    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
    call therm_index ; W = LUT TEMPERATURE INDEX
    movwf r_temp_1
    movwf r_ee_addr
    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 ;
btfss r_accB_H, 7 ;
goto $+3 ;
call math_neg_B ;
bsf flag_math_temp ;
movf r_accB_L, w ;
movwf r_accA_L ;
movf r_accB_H, w ;
movwf r_accA_H ;
call math_mul_16_prep_ ; accB,C = TSCALE' * M
movlw .5 ;
call math_shift_BC ;
movf r_accB_L, w ;
movwf r_accB_H ;
movf r_accC_H, w ;
movwf r_accB_L ; accB = TSCALE' * M / 8192
btfsc flag_math_temp ;
call math_neg_B ;
call ee_read_ia ;
call ee_read_ia ;
movwf r_accA_H ;
call math_add_16 ; accB = accB + yint = temperature

adc_svc_3_B: ;

adc_svc_3_under: ;
 btfs r_accB_H, 7 ; check limit: tcode < 0
goto adc_svc_3_under_x ;
clrfr r_accB_H ;
clrfr 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 ;
btfs r_status, Z ;
goto adc_svc_3_over_x ;
movlw 0x03 ;
movwf r_accB_H ;
movlw 0xFF ;
movwf r_accB_L ;
adc_svc_3_over_x: ;

movlw r_adc_3_L ;
call math_move_B ; save results

adc_svc_3_x_: ;
movlw r_adc_3_L ;
call math_load_B ; re-load results (needed for sim)
rrf r_accB_H, f ;
rrf r_accB_L, f ;
rrf r_accB_H, f ;
rrf r_accB_L, w ;
movwf r_tcode ;
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, C           
    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_mul_16_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                         

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.

```assembly
;-----------------------------------------------------------
;--- adc_start_reset() - reset adc scheduler
;-----------------------------------------------------------
adc_start_reset:                        
    clrf r_adc_control            ;
    return                         ;

;-----------------------------------------------------------
;--- adc_start() - start/initiate new coversion
;-----------------------------------------------------------
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                 ;
btfsr r_status, Z                       ;
goto adc_start_done                     ;
bsf r_adc_control, 6                    ;

btfsr r_adc_control, 4                  ;
movlw ADC_ADCON0_4                     ;
btfsp r_adc_control, 3                  ;
movlw ADC_ADCON0_3                     ;
btfsr r_adc_control, 2                  ;
movlw ADC_ADCON0_2                     ;
btfsr r_adc_control, 1                  ;
movlw ADC_ADCON0_1                     ;
btfsr 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_rqq         ;
    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_rqq
  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_rqq
  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_done_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 description. Charge Suspend, Fast Charge, Trickle Charge, etc., are entered when appropriate. See lead charge state descriptions in “Functional Description: Lead Chemistry”.

```assembly
;-----------------------------------------------------------
;--- charge state service
;-----------------------------------------------------------

chg_state_svc:                          
  btfs     flag_chg_state_timer     ; service timer expired ?
  goto     chg_state_svc_x_         ; --- no, exit ... 
  bcf      flag_chg_state_timer     
  btfs     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:                           
  btfs      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:                      
  btfs     flag0_mode_bpres_v       
  goto     chg_state_svc_bpv_x      
  movlw     EE_CHG_V_MIN_BP         
  call      check_voltage           
  btfs      r_status, C             
  bsf      flag_battpres          
chg_state_svc_bpv_x:                    

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

;--- battery present - battid
chg_state_svc_bpbi:                     
  btfs     flag0_mode_bpres_battid  
  goto     chg_state_svc_bpbi_x     
  btfs      flag_battpres1         
  bsf      flag_battpres          
chg_state_svc_bpbi_x:                   

chg_state_svc_bp:                       
  btfs      flag_battpres          ; battery present ?
  goto     chg_state_svc_bp_x      ; --- yes, skip ... 
  movf      r_chg_state, f         
  btfs      r_status, Z            ;chg state already zero ?
  goto     chg_state_0_init        ; --- no, go initialize state 0 
chg_state_svc_bp_x:                     
  call      check_chg_timer_a      
  btfs      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
;-----------------------------------------------------------
chg_state_1:                            ;
btfsc    flag0_mode_float             ;
goto    chg_state_7_init             ;
goto    chg_state_3_init             ;

;-----------------------------------------------------------
;--- charge state: 2 - precharge
;-----------------------------------------------------------
chg_state_2:                            ;
goto    chg_state_0_init             ;

;-----------------------------------------------------------
;--- charge state: 3
;-----------------------------------------------------------
chg_state_3:                            ;
btfss    flag_reg_on                  ; regulation module shutdown ?
goto    chg_state_0_init             ; --- yes, exit to state 6 ...
btfsc    flag_chg_ti1_done           ; timer expired ?
goto    chg_state_6_init             ; --- yes, suspend
btfsc    flag_vreg                    ; reached voltage ?
goto    chg_state_4_init             ; --- yes, goto CV

goto    chg_state_x                   ;

;-----------------------------------------------------------
;--- charge state: 4
;-----------------------------------------------------------
chg_state_4:                            ;
btfss    flag_reg_on                  ; regulation module shutdown ?
goto    chg_state_0_init             ; --- yes, exit to state 6 ...
btfsc    flag_chg_ti1_done           ; timer expired ?
goto    chg_state_6_init             ; --- yes, suspend ...
chg_state_4_a:                          ;
movlw    EE_CHG_C_MIN                 ; check current w/ cmin in ee
call    check_current                 ;
btfsf    r_status, C                  ;
goto    chg_state_4_a_1              ; current too hi - reset timer & exit
call    check_chg_timer_b            ; current good - check timer
btfsf    r_status, Z                  ;
goto    chg_state_4_x                 ;
btfss    flag0_mode_postfloat        ;
goto    chg_state_5_init             ;
goto chg_state_7_init ;

chg_state_4_a_1:;
   movlw EE_CHG_TIME_1 ; reset min current timer
call ee_read_waddr ;
movwf r_chg_timer_b ;

chg_state_4_x: ;
goto chg_state_x ;

;;;;;-----------------------------------------------------------
;;;;;--- charge state: 5
;;;;;-----------------------------------------------------------
chg_state_5: ;
   btfss flag0_mode_refloat ;
goto chg_state_5_x ;
   btfsc flag_chg_ti1_done ;
goto chg_state_7_init ;
goto chg_state_5_x ;

chg_state_5_x: ;
goto chg_state_x ;

;;;;;-----------------------------------------------------------
;;;;;--- charge state: 6
;;;;;-----------------------------------------------------------
chg_state_6: ;
   btfsc flag0_mode_suspend_4ever ;
goto chg_state_6_x ;
   btfsc flag_chg_ti2_done ;
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 ...
   btfsc flag_chg_ti1_done ;
goto chg_state_5_init ;
goto chg_state_x ;

;;;;;-----------------------------------------------------------
;;;;;--- 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_0_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 13: LEAD CHARGE STATE FLOWCHART

STATE-0
CHARGE PENDING

STATE-1
CHARGE QUAL

STATE-3
CURRENT REG

STATE-4
VOLTAGE REG

STATE-7
FLOAT

STATE-5
EOC

STATE-6
CHARGE SUSPEND
EXAMPLE 1: LEAD CHARGE STATE FLOWCHART EQUATIONS

State Transition Criteria:
[12] => Transition from state 1 to state 2
[1] => 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]*

[13] = MODE_FLOATONLY*
[17] = MODE_FLOATONLY

[36] = TI1 > TICC
[34] = [36]* and (V > VCHG)

[46] = TI1 > TICV
[45] = [46]* and MODE_POSTFLOAT* and (C < CMIN)
[47] = [46]* and MODE_POSTFLOAT and (C < CMIN)

[57] = MODE_REFLOAT and (TI > TI1)

[75] = TI1 > TIFLOAT

[60] = MODE_SUSPEND_4EVER* and TI2 > TISUSPEND

STATE-7
[75] = (V > VMAX) or (TI1 > TITRICKLE_MAX)
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.

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

;--------------------------
;--- chg_state_0_init()
;--------------------------
chg_state_0_init:       
call      reg_off           
clrfr    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:       
goto     chg_state_0_init

;--------------------------
;--- chg_state_3_init()
;--------------------------
chg_state_3_init:       
call      reg_load_chg      
movlw    EE_CHG_TI_CC       
call     load_timer_d1_ee    
movlw    .3                
goto     chg_state_init_x_gpio_lo

;--------------------------
;--- chg_state_4_init()
;--------------------------
chg_state_4_init:       
call      reg_load_chg      
movlw    EE_CHG_TI_CV       
call     load_timer_d1_ee    
movlw    .4                
goto     chg_state_init_x_gpio_lo

;--------------------------
;--- chg_state_5_init()
;--------------------------
chg_state_5_init:       
call      reg_off           
movlw    EE_CHG_TI_REFLOAT  
call     load_timer_d1_ee    
movlw    .5                
goto     chg_state_init_x_gpio_hi
```
;-----------------------------
;--- chg_state_6_init()
;-----------------------------
chg_state_6_init:
   call reg_off
   movlw EE_CHG_TI_SUSPEND
   call load_timer_d2_ee
   movlw .6
   goto chg_state_init_x_gpio_hi

;-----------------------------
;--- chg_state_7_init()
;-----------------------------
chg_state_7_init:
   call reg_load_flt
   movlw EE_CHG_TI_FLOAT
   call load_timer_d1_ee
   movlw .7
   goto chg_state_init_x_gpio_lo

;---
chg_state_init_x_gpio_hi:
   btfsc flag0_mode_gpio_cutoff
   bsf p_gpio
   goto chg_state_init_x_
chg_state_init_x_gpio_lo:
   btfsc flag0_mode_gpio_cutoff
   bcf p_gpio
   chg_state_init_x_
   movwf r_chg_state ; save state
   movf r_chg_state, w ; configure led1
   addlw EE_LED1_CFG
   call ee_read_waddr
   call led_init_1
   movf r_chg_state, w ; configure led2
   addlw EE_LED2_CFG
   call ee_read_waddr
   call led_init_2
   movlw EE_CHG_TIME_0
   call ee_read_waddr ; w = threshold from ee
   movwf 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.

```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
  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() - 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:
    clrf r_flags_2            ; pre-clear flags
    movlw EE_CHG_T_PCHG      ;
    call check_temperature   ;
    rlf r_flags_2, f         ;
    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 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               ;
    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                     ;
movlw  EE_REG_VH        ;
call   ee_read_waddr    ;
call   math_add_8b      ;
btfss 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      ;
btfss r_accB_H, 7       ; delta <= upper_limit ?
goto  regulate_v_x     ; --- yes, go check current ...  
bsf   flag_vreg_2       ; V > LEVEL 2 !!

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

==> (zones: A1,A2,A3,A4,A5)

regulate_v_lo:                       
call   math_neg_B        
movlw  EE_REG_VL        ;
call   ee_read_waddr    ;
call   math_add_8b      ;
btfss 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      ;
btfss 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_x    ;
btfsc  flag_vreg_2      ;

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         
  btfss 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)
    goto  regulate_adj_up_0          ;===(zones: D1,D2)
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_dn_2:                       
    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                    

AN1015
regulate_x:                  
  bcf flag_adcset_1_rdy    
  bsf flag_adcset_1_rq     

regulate_x:               
  return                   

;---------------------------------------
;--- reg_load()               
;---------------------------------------

;--- reg_load_flt() - load float voltage (K or LUT)
reg_load_flt:             
  movlw ER_CHG_V_FLT_K     
  btfsb flag0_mode_v_flt_k ; constant or lut ?
  goto reg_load_v          ; --- constant !
  call vlut_index_tcode    ; --- lut !
  call vlut_fetch_flt      
  goto reg_load_c          

;--- reg_load_flt() - load charge voltage (K or LUT)
reg_load_chg:             
  movlw ER_CHG_V_CHG_K     
  btfsb flag0_mode_v_reg_k ; constant or lut ?
  goto reg_load_v          ; --- constant !
  call vlut_index_tcode    ; --- lut !
  call vlut_fetch_chg      
  goto reg_load_c          

reg_load_v:               
  call load_B_ee           
  movlw r_reg_v            
  call math_move_B         

reg_load_c:               
  movlw ER_CHG_C           
  call load_B_ee           
  movlw r_reg_c            
  call math_move_B         
  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          
  clrfr r_reg_c            
  clrfr r_reg_c+1          
  clrfr r_reg_v            
  clrfr r_reg_v+1          
  clrfr r_pwm_L            
  clrfr 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                
    clrf      r_accB_L                
    clrf      r_accC_H                
    clrf      r_accC_L                
    math_mul_16_loop:                       
    rrf       r_accD_H, F             
    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             
    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, F
    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:
;      op1: accA
;      op2: 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:
    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:
    movwf r_fsr
    movf r_indf, w
    subwf r_accB_H, w
    btfss r_status, Z
    return
    decf r_fsr, f
    movf r_indf, w
    subwf r_accB_L, w
    return

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                ;
math_load_x:                            ;
  incf  r_fsr, f               ;
  return                        ;

;---  math_move_B:
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:
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
# 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 software names. The PowerTool 200 software names are used in the functional description section.

```asm
;-------------------------------------;
;--- EEPROM DEFAULT
;-------------------------------------;
org 0x2100
DE 0x02, 0x00 ; pattern
DE .1 ; ncells
DE "microchip" ; manuf name
DE "16HV785 " ; device name
DE low .2000, high .2000 ; capacity
DE .19 ; pwm
DE 0x01 ; mode
DE 0x21 ; 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 .6 ; regulation: current zone thresholds
DE low .2900, high .2900 ; regulation: v_safety (lion)
org 0x2143
DE low .1500, high .1500 ; chg_c
DE low .1000, high .1000 ; chg_c_float
DE low .150, high .150 ; chg_c_min
DE .90 ; chg_ti_cc (90=6hr)
DE .90 ; chg_ti_cv
DE .15 ; chg_ti_float
DE .15 ; chg_ti_refloat
DE .0 ; chg_ti_suspend
DE low .2450, high .2450 ; chg_v_chg
DE low .2275, high .2275 ; chg_v_flt
DE low .0500, high .0500 ; chg_v_min_bp
org 0x215c
DE .20, .20, .00, .00, .00, .00 ; chg_time
DE .000, .128 ; 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 .3412, high .3412 ; 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
DE  0x1E ; mode_3
DE  .10 ; vlut_n - vlut length

;--- vlut_t
DE  .025, .050, .075, .100 ; -10, 0, 10, 20
DE  .113, .125, .137, .150 ; 25, 30, 35, 40
DE  .175 ; 50

;--- vlut_chg
DE  low .2760, high .2760 ;
DE  low .2770, high .2770 ;
DE  low .2650, high .2650 ;
DE  low .2590, high .2590 ;
DE  low .2530, high .2530 ;
DE  low .2500, high .2500 ;
DE  low .2470, high .2470 ;
DE  low .2410, high .2410 ;
DE  low .2350, high .2350 ;
DE  low .2250, high .2250 ;

;--- vlut_flt
DE  low .2380, high .2380 ;
DE  low .2370, high .2370 ;
DE  low .2350, high .2350 ;
DE  low .2330, high .2330 ;
DE  low .2310, high .2310 ;
DE  low .2300, high .2300 ;
DE  low .2290, high .2290 ;
DE  low .2270, high .2270 ;
DE  low .2250, high .2250 ;
DE  low .2200, high .2200 ;

end
### MEMORY MAP

#### 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>PATTERN</td>
<td>0001</td>
<td>2</td>
<td>1</td>
<td>coded</td>
<td>Pattern ID (arbitrary)</td>
</tr>
<tr>
<td>NCELLS</td>
<td>01</td>
<td>1</td>
<td>1</td>
<td>cells</td>
<td>Number of cells</td>
</tr>
<tr>
<td>MANUF_NAME</td>
<td>03</td>
<td>8</td>
<td>—</td>
<td>—</td>
<td>ASCII Manufacturer’s name “Microchip”</td>
</tr>
<tr>
<td>DEV_NAME</td>
<td>0B</td>
<td>8</td>
<td>—</td>
<td>—</td>
<td>ASCII Device name “16HV785”</td>
</tr>
<tr>
<td>CAPACITY</td>
<td>13</td>
<td>2</td>
<td>800</td>
<td>320</td>
<td>mAh Capacity</td>
</tr>
<tr>
<td>PWM_FREQ</td>
<td>15</td>
<td>1</td>
<td>19</td>
<td>13</td>
<td>coded PWM frequency (e.g. 19 = 400 kHz)</td>
</tr>
<tr>
<td>MODE</td>
<td>16</td>
<td>1</td>
<td>1</td>
<td>01</td>
<td>coded Mode – control flags</td>
</tr>
<tr>
<td>MODE2</td>
<td>17</td>
<td>1</td>
<td>32</td>
<td>20</td>
<td>coded Mode – control flags</td>
</tr>
<tr>
<td>OSC_TRIM</td>
<td>18</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>coded Oscillator trim – signed value +/-10%</td>
</tr>
<tr>
<td>Reserved</td>
<td>19</td>
<td>7</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>LED1_CFG_0</td>
<td>20</td>
<td>32</td>
<td>8</td>
<td>08</td>
<td>coded LED1 configuration – state 0</td>
</tr>
<tr>
<td>LED1_CFG_1</td>
<td>21</td>
<td>1</td>
<td>24</td>
<td>18</td>
<td>coded LED1 configuration – state 1</td>
</tr>
<tr>
<td>LED1_CFG_2</td>
<td>22</td>
<td>1</td>
<td>40</td>
<td>28</td>
<td>coded LED1 configuration – state 2</td>
</tr>
<tr>
<td>LED1_CFG_3</td>
<td>23</td>
<td>1</td>
<td>56</td>
<td>38</td>
<td>coded LED1 configuration – state 3</td>
</tr>
<tr>
<td>LED1_CFG_4</td>
<td>24</td>
<td>1</td>
<td>72</td>
<td>48</td>
<td>coded LED1 configuration – state 4</td>
</tr>
<tr>
<td>LED1_CFG_5</td>
<td>25</td>
<td>1</td>
<td>88</td>
<td>58</td>
<td>coded LED1 configuration – state 5</td>
</tr>
<tr>
<td>LED1_CFG_6</td>
<td>26</td>
<td>1</td>
<td>104</td>
<td>68</td>
<td>coded LED1 configuration – state 6</td>
</tr>
<tr>
<td>LED1_CFG_7</td>
<td>27</td>
<td>1</td>
<td>120</td>
<td>78</td>
<td>coded LED1 configuration – state 7</td>
</tr>
<tr>
<td>LED2_CFG_0</td>
<td>28</td>
<td>1</td>
<td>8</td>
<td>08</td>
<td>coded LED2 configuration – state 0</td>
</tr>
<tr>
<td>LED2_CFG_1</td>
<td>29</td>
<td>1</td>
<td>24</td>
<td>18</td>
<td>coded LED2 configuration – state 1</td>
</tr>
<tr>
<td>LED2_CFG_2</td>
<td>30</td>
<td>1</td>
<td>40</td>
<td>28</td>
<td>coded LED2 configuration – state 2</td>
</tr>
<tr>
<td>LED2_CFG_3</td>
<td>31</td>
<td>1</td>
<td>56</td>
<td>38</td>
<td>coded LED2 configuration – state 3</td>
</tr>
<tr>
<td>LED2_CFG_4</td>
<td>32</td>
<td>1</td>
<td>72</td>
<td>48</td>
<td>coded LED2 configuration – state 4</td>
</tr>
<tr>
<td>LED2_CFG_5</td>
<td>33</td>
<td>1</td>
<td>88</td>
<td>58</td>
<td>coded LED2 configuration – state 5</td>
</tr>
<tr>
<td>LED2_CFG_6</td>
<td>34</td>
<td>1</td>
<td>104</td>
<td>68</td>
<td>coded LED2 configuration – state 6</td>
</tr>
<tr>
<td>LED2_CFG_7</td>
<td>35</td>
<td>1</td>
<td>120</td>
<td>78</td>
<td>coded LED2 configuration – state 7</td>
</tr>
<tr>
<td>Reserved</td>
<td>36</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>REG_P1</td>
<td>34</td>
<td>1</td>
<td>12</td>
<td>0C</td>
<td>count Regulation – control PWM adj</td>
</tr>
<tr>
<td>REG_P2</td>
<td>35</td>
<td>1</td>
<td>10</td>
<td>0A</td>
<td>count Regulation – control PWM adj</td>
</tr>
<tr>
<td>REG_P3</td>
<td>36</td>
<td>1</td>
<td>5</td>
<td>05</td>
<td>count Regulation – control PWM adj</td>
</tr>
<tr>
<td>REG_P4</td>
<td>37</td>
<td>1</td>
<td>1</td>
<td>01</td>
<td>count Regulation – control PWM adj</td>
</tr>
<tr>
<td>REG_VHH_VH</td>
<td>38</td>
<td>1</td>
<td>19</td>
<td>13</td>
<td>mV Regulation – voltage zone limit</td>
</tr>
<tr>
<td>REG_VH</td>
<td>39</td>
<td>1</td>
<td>6</td>
<td>06</td>
<td>mV Regulation – voltage zone limit</td>
</tr>
<tr>
<td>REG_VL</td>
<td>3A</td>
<td>1</td>
<td>6</td>
<td>06</td>
<td>mV Regulation – voltage zone limit</td>
</tr>
<tr>
<td>REG_VLL_VL</td>
<td>3B</td>
<td>1</td>
<td>44</td>
<td>2C</td>
<td>mV Regulation – voltage zone limit</td>
</tr>
<tr>
<td>REG_CNULL</td>
<td>3C</td>
<td>1</td>
<td>5</td>
<td>05</td>
<td>mA Regulation – current null limit</td>
</tr>
<tr>
<td>REG_VSAFETY</td>
<td>3D</td>
<td>2</td>
<td>4350</td>
<td>10FE</td>
<td>mV Regulation – voltage limit – shutdown</td>
</tr>
<tr>
<td>Reserved</td>
<td>3F</td>
<td>4</td>
<td>x</td>
<td>x</td>
<td></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>mA</td>
</tr>
<tr>
<td>SUB-REG</td>
<td>67</td>
<td>43</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHG_C</td>
<td>67</td>
<td>43</td>
<td>2</td>
<td>2000</td>
<td>7D0</td>
</tr>
<tr>
<td>CHG_C_FLOAT</td>
<td>69</td>
<td>45</td>
<td>2</td>
<td>2000</td>
<td>7D0</td>
</tr>
<tr>
<td>CHG_C_MIN</td>
<td>71</td>
<td>47</td>
<td>2</td>
<td>200</td>
<td>C8</td>
</tr>
<tr>
<td>CHG_Q_CC</td>
<td>73</td>
<td>49</td>
<td>1</td>
<td>75</td>
<td>4B</td>
</tr>
<tr>
<td>CHG_Q_CV</td>
<td>74</td>
<td>4A</td>
<td>1</td>
<td>75</td>
<td>4B</td>
</tr>
<tr>
<td>CHG_Q_FLOAT</td>
<td>75</td>
<td>4B</td>
<td>1</td>
<td>75</td>
<td>4B</td>
</tr>
<tr>
<td>CHG_Q_REFLOAT</td>
<td>76</td>
<td>4C</td>
<td>1</td>
<td>15</td>
<td>0F</td>
</tr>
<tr>
<td>CHG_Q_SUSPEND</td>
<td>77</td>
<td>4D</td>
<td>1</td>
<td>15</td>
<td>0F</td>
</tr>
<tr>
<td>CHG_V_CHG_K</td>
<td>78</td>
<td>4E</td>
<td>2</td>
<td>2450</td>
<td>0992</td>
</tr>
<tr>
<td>CHG_V_FLT_K</td>
<td>80</td>
<td>50</td>
<td>2</td>
<td>2275</td>
<td>08E3</td>
</tr>
<tr>
<td>CHG_V_MIN_BP</td>
<td>82</td>
<td>52</td>
<td>2</td>
<td>500</td>
<td>01F4</td>
</tr>
<tr>
<td>Reserved</td>
<td>84</td>
<td>54</td>
<td>8</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SUB-CHG-PB</td>
<td>92</td>
<td>5C</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHG_TIME_0</td>
<td>92</td>
<td>5C</td>
<td>1</td>
<td>20</td>
<td>14</td>
</tr>
<tr>
<td>CHG_TIME_1</td>
<td>93</td>
<td>5D</td>
<td>1</td>
<td>20</td>
<td>14</td>
</tr>
<tr>
<td>CHG_TIME_2</td>
<td>94</td>
<td>5E</td>
<td>1</td>
<td>20</td>
<td>14</td>
</tr>
<tr>
<td>CHG_TIME_3</td>
<td>95</td>
<td>5F</td>
<td>1</td>
<td>20</td>
<td>14</td>
</tr>
<tr>
<td>CHG_TIME_4</td>
<td>96</td>
<td>60</td>
<td>1</td>
<td>20</td>
<td>14</td>
</tr>
<tr>
<td>CHG_TIME_5</td>
<td>97</td>
<td>61</td>
<td>1</td>
<td>20</td>
<td>14</td>
</tr>
<tr>
<td>BATTID_MIN</td>
<td>98</td>
<td>62</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BATTID_MAX</td>
<td>99</td>
<td>63</td>
<td>1</td>
<td>128</td>
<td>80</td>
</tr>
<tr>
<td>Reserved</td>
<td>100</td>
<td>64</td>
<td>8</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SUB-CHG</td>
<td>108</td>
<td>6C</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC_CAL_0</td>
<td>108</td>
<td>6C</td>
<td>2</td>
<td>248</td>
<td>00F8</td>
</tr>
<tr>
<td>ADC_CAL_1</td>
<td>110</td>
<td>6E</td>
<td>2</td>
<td>2553</td>
<td>09F9</td>
</tr>
<tr>
<td>ADC_CAL_2</td>
<td>112</td>
<td>70</td>
<td>2</td>
<td>5121</td>
<td>1401</td>
</tr>
<tr>
<td>ADC_CAL_3</td>
<td>114</td>
<td>72</td>
<td>2</td>
<td>8192</td>
<td>2000</td>
</tr>
<tr>
<td>ADC_CAL_4</td>
<td>116</td>
<td>74</td>
<td>2</td>
<td>6407</td>
<td>1907</td>
</tr>
<tr>
<td>SHUNT</td>
<td>118</td>
<td>76</td>
<td>1</td>
<td>100</td>
<td>64</td>
</tr>
<tr>
<td>T_DEFAULT</td>
<td>119</td>
<td>77</td>
<td>1</td>
<td>112</td>
<td>70</td>
</tr>
<tr>
<td>Reserved</td>
<td>120</td>
<td>78</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-ADC</td>
<td>124</td>
<td>7C</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_LUT_N</td>
<td>124</td>
<td>7C</td>
<td>1</td>
<td>8</td>
<td>08 entries</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></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</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Temperature LUT – T0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(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</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Temperature LUT – T6</td>
</tr>
<tr>
<td>T_LUT_M_0</td>
<td>132</td>
<td>84</td>
<td>2</td>
<td>-23362</td>
<td>A4BE scaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Temperature LUT – T0 slope</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(T_LUT_N entries)</td>
</tr>
<tr>
<td>T_LUT_B_0</td>
<td>134</td>
<td>86</td>
<td>2</td>
<td>1418</td>
<td>058A scaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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</td>
<td>B268 scaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Temperature LUT – T1 slope</td>
</tr>
<tr>
<td>T_LUT_B_1</td>
<td>138</td>
<td>8A</td>
<td>2</td>
<td>1352</td>
<td>0548 scaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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</td>
<td>C2A3 scaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Temperature LUT – T2 slope</td>
</tr>
<tr>
<td>T_LUT_B_2</td>
<td>142</td>
<td>8E</td>
<td>2</td>
<td>1255</td>
<td>04E7 scaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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</td>
<td>CEE4 scaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Temperature LUT – T3 slope</td>
</tr>
<tr>
<td>T_LUT_B_3</td>
<td>146</td>
<td>92</td>
<td>2</td>
<td>1162</td>
<td>048A scaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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</td>
<td>D822 scaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Temperature LUT – T4 slope</td>
</tr>
<tr>
<td>T_LUT_B_4</td>
<td>150</td>
<td>96</td>
<td>2</td>
<td>1071</td>
<td>042F scaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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</td>
<td>D856 scaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Temperature LUT – T6 slope</td>
</tr>
<tr>
<td>T_LUT_B_6</td>
<td>158</td>
<td>9E</td>
<td>2</td>
<td>1127</td>
<td>0467 scaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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</td>
<td>CDB5 scaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Temperature LUT – T7 slope</td>
</tr>
<tr>
<td>T_LUT_B_7</td>
<td>162</td>
<td>A2</td>
<td>2</td>
<td>1402</td>
<td>057A scaler</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Temperature LUT – T7 Y-intercept</td>
</tr>
<tr>
<td>Name</td>
<td>Location</td>
<td>Len</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
<td>-----</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dec</td>
<td>Hex</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUB-T LUT</td>
<td>164</td>
<td>A4</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE_3</td>
<td>170</td>
<td>AA</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VLUT_N</td>
<td>171</td>
<td>AB</td>
<td>1</td>
<td>10</td>
<td>A</td>
</tr>
<tr>
<td>T VLUT_0</td>
<td>172</td>
<td>AC</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T VLUT_1</td>
<td>173</td>
<td>AD</td>
<td>1</td>
<td>25</td>
<td>19</td>
</tr>
<tr>
<td>T VLUT_2</td>
<td>174</td>
<td>AE</td>
<td>1</td>
<td>50</td>
<td>32</td>
</tr>
<tr>
<td>T VLUT_3</td>
<td>175</td>
<td>AF</td>
<td>1</td>
<td>75</td>
<td>4B</td>
</tr>
<tr>
<td>T VLUT_4</td>
<td>176</td>
<td>B0</td>
<td>1</td>
<td>100</td>
<td>64</td>
</tr>
<tr>
<td>T VLUT_5</td>
<td>177</td>
<td>B1</td>
<td>1</td>
<td>113</td>
<td>70</td>
</tr>
<tr>
<td>T VLUT_6</td>
<td>178</td>
<td>B2</td>
<td>1</td>
<td>125</td>
<td>7D</td>
</tr>
<tr>
<td>T VLUT_7</td>
<td>179</td>
<td>B3</td>
<td>1</td>
<td>150</td>
<td>96</td>
</tr>
<tr>
<td>T VLUT_8</td>
<td>180</td>
<td>B4</td>
<td>1</td>
<td>175</td>
<td>AF</td>
</tr>
<tr>
<td>V VLUT CHG_0</td>
<td>181</td>
<td>B5</td>
<td>2</td>
<td>2760</td>
<td>AC8</td>
</tr>
<tr>
<td>V VLUT CHG_1</td>
<td>183</td>
<td>B7</td>
<td>2</td>
<td>2700</td>
<td>A8C</td>
</tr>
<tr>
<td>V VLUT CHG_2</td>
<td>185</td>
<td>B9</td>
<td>2</td>
<td>2650</td>
<td>A5A</td>
</tr>
<tr>
<td>V VLUT CHG_3</td>
<td>187</td>
<td>BB</td>
<td>2</td>
<td>2590</td>
<td>A1E</td>
</tr>
<tr>
<td>V VLUT CHG_4</td>
<td>189</td>
<td>BD</td>
<td>2</td>
<td>2530</td>
<td>9E2</td>
</tr>
<tr>
<td>V VLUT CHG_5</td>
<td>191</td>
<td>BF</td>
<td>2</td>
<td>2500</td>
<td>9C4</td>
</tr>
<tr>
<td>V VLUT CHG_6</td>
<td>193</td>
<td>C1</td>
<td>2</td>
<td>2470</td>
<td>9A6</td>
</tr>
<tr>
<td>V VLUT CHG_7</td>
<td>195</td>
<td>C3</td>
<td>2</td>
<td>2410</td>
<td>96A</td>
</tr>
<tr>
<td>V VLUT CHG_8</td>
<td>197</td>
<td>C5</td>
<td>2</td>
<td>2350</td>
<td>92E</td>
</tr>
<tr>
<td>V VLUT CHG_9</td>
<td>199</td>
<td>C7</td>
<td>2</td>
<td>2250</td>
<td>8CA</td>
</tr>
<tr>
<td>V VLUT FLT_0</td>
<td>201</td>
<td>C9</td>
<td>2</td>
<td>2380</td>
<td>94C</td>
</tr>
<tr>
<td>V VLUT FLT_1</td>
<td>203</td>
<td>CB</td>
<td>2</td>
<td>2370</td>
<td>942</td>
</tr>
<tr>
<td>V VLUT FLT_2</td>
<td>205</td>
<td>CD</td>
<td>2</td>
<td>2350</td>
<td>92E</td>
</tr>
<tr>
<td>V VLUT FLT_3</td>
<td>207</td>
<td>CF</td>
<td>2</td>
<td>2330</td>
<td>91A</td>
</tr>
<tr>
<td>V VLUT FLT_4</td>
<td>209</td>
<td>D1</td>
<td>2</td>
<td>2310</td>
<td>906</td>
</tr>
<tr>
<td>V VLUT FLT_5</td>
<td>211</td>
<td>D3</td>
<td>2</td>
<td>2300</td>
<td>8FC</td>
</tr>
<tr>
<td>V VLUT FLT_6</td>
<td>213</td>
<td>D5</td>
<td>2</td>
<td>2290</td>
<td>8F2</td>
</tr>
<tr>
<td>V VLUT FLT_7</td>
<td>215</td>
<td>D7</td>
<td>2</td>
<td>2270</td>
<td>8DE</td>
</tr>
<tr>
<td>V VLUT FLT_8</td>
<td>217</td>
<td>D9</td>
<td>2</td>
<td>2250</td>
<td>8CA</td>
</tr>
<tr>
<td>V VLUT FLT_9</td>
<td>219</td>
<td>DB</td>
<td>2</td>
<td>2200</td>
<td>898</td>
</tr>
<tr>
<td>TOTAL</td>
<td>221</td>
<td>DD</td>
<td>51</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE 11: EEPROM MEMORY MAP (CONTINUED)
## MODE Registers

### TABLE 12: MODE

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

<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></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>suspend_4ever</td>
<td>$Suspend$ indefinitely – until Reset or battery removed, etc.</td>
</tr>
<tr>
<td>4</td>
<td>refloat</td>
<td>$Enable$ “refloat” – entered from EOC state</td>
</tr>
<tr>
<td>3</td>
<td>postfloat</td>
<td>$Enable$ float after CC, CV states</td>
</tr>
<tr>
<td>2</td>
<td>v_flt_k</td>
<td>$Use$ fixed float voltage (otherwise use VLUT)</td>
</tr>
<tr>
<td>1</td>
<td>v_chg_k</td>
<td>$Use$ fixed charging voltage (otherwise use VLUT)</td>
</tr>
<tr>
<td>0</td>
<td>float</td>
<td>$Skip$ to float state immediately</td>
</tr>
</tbody>
</table>
### RAM

#### 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>PWM setting</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>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_contr1</td>
<td>75</td>
<td>4B</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>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_contrl_2</td>
<td>77</td>
<td>4D</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>1</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>r_tcode</td>
<td>104</td>
<td>68</td>
<td>1</td>
<td>mV</td>
<td>NiMh -dv reference</td>
</tr>
<tr>
<td>unused</td>
<td>106</td>
<td>6A</td>
<td>5</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>
<tr>
<td>r_isr_w</td>
<td>121</td>
<td>79</td>
<td>1</td>
<td>N/A</td>
<td>Interrupt context</td>
</tr>
<tr>
<td>r_isr_status</td>
<td>122</td>
<td>7A</td>
<td>1</td>
<td>N/A</td>
<td>Interrupt context</td>
</tr>
<tr>
<td>r_isr_pclath</td>
<td>123</td>
<td>7B</td>
<td>1</td>
<td>N/A</td>
<td>Interrupt context</td>
</tr>
<tr>
<td>r_isr_fsr</td>
<td>124</td>
<td>7C</td>
<td>1</td>
<td>N/A</td>
<td>Interrupt context</td>
</tr>
<tr>
<td>r_ee_data</td>
<td>125</td>
<td>7D</td>
<td>1</td>
<td>N/A</td>
<td>EEPROM data</td>
</tr>
<tr>
<td>r_ee_addr</td>
<td>126</td>
<td>7E</td>
<td>1</td>
<td>N/A</td>
<td>EEPROM address</td>
</tr>
<tr>
<td>r_tempc_1</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.

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

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

AmpLab, FilterLab, Migratable Memory, MXDEV, MXLAB, PICMASTER, 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, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Linear Active Thermistor, MPASM, MPLIB, MPLINK, MPSIM, PICkit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, Real ICE, rLAB, rTPICDEM, 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.

© 2006, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.

QUALITY MANAGEMENT SYSTEM CERTIFIED BY DNV

ISO/TS 16949:2002

Microchip received ISO/TS-16949:2002 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona and Mountain View, California in October 2003. The Company’s quality system processes and procedures are for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001:2000 certified.
# WORLDWIDE SALES AND SERVICE

**AMERICAS**

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

**Atlanta**  
Alpharetta, GA  
Tel: 770-640-0034  
Fax: 770-640-0307

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

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

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

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

**Kokomo**  
Kokomo, IN  
Tel: 765-864-8360  
Fax: 765-864-8387

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

**San Jose**  
Mountain View, CA  
Tel: 650-215-1444  
Fax: 650-961-0286

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

**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: 86-24-2334-2829  
Fax: 86-24-2334-2393

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

**China - Shunde**  
Tel: 86-757-2839-5507  
Fax: 86-757-2839-5571

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

**China - Xian**  
Tel: 86-29-8833-7250  
Fax: 86-29-8833-7256

**ASIA/PACIFIC**

**India - Bangalore**  
Tel: 91-80-2229-0061  
Fax: 91-80-2229-0062

**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: 82-2-558-5932 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: 43-7242-2244-399  
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

10/31/05