AN1012

PIC16HV785: Programmable Lithium and Nickel Battery Charger

Author: Rich DelRossi
Microchip Technology Inc.

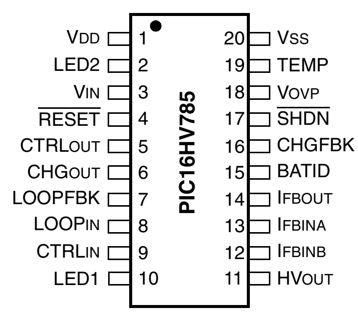
Features

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

Applications

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

Pin Description



20-Pin PDIP, SOIC, SSOP

USING THE PIC16HV785

Product Overview

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

MULTI-STEP CHARGING

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

USER-CONFIGURABLE PARAMETERS

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

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

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

SPECIAL FEATURES

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

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

TABLE 1: PINOUT DESCRIPTION

Pin	Pin Name	Pin Type	Input Type	Output Type	Description
1	V_{DD}	Supply	Power	—	Supply voltage
2	LED2	O	—	CMOS	Status indicator
3	V_{IN}	I	Analog	—	Battery voltage input
4	$\overline{RESET}$	I	ST	—	Reset
5	$CTRL_{OUT}$	O	—	CMOS	PWM output for setting current level
6	CHG_{OUT}	O	—	CMOS	PWM output to a buck converter for charge control
7	LOOPFBK	I	Analog	—	Current feedback loop
8	$LOOP_{IN}$	I	Analog	—	Current feedback loop input
9	$CTRL_{IN}$	I	Analog	—	Current level control
10	LED1	O	—	CMOS	Status indicator
11	HV_{OUT}	O	—	HVOD	High-voltage, open-drain output pin (optional)
12	I_{FBINB}	I	Analog	—	Current feedback input pin B used for current scaling
13	I_{FBINA}	I	Analog	—	Current feedback input pin A used for current scaling
14	I_{FBOUT}	O	—	Analog	Current feedback output
15	BATID	I	Analog	—	Battery ID select
16	CHGFBK	I	Analog	—	Charge control feedback
17	$\overline{SHDN}$	O	—	Analog	Shutdown signal, active-low
18	V_{OVP}	I	Analog	—	Overvoltage protection
19	TEMP	I	Analog	—	Battery temperature input
20	V_{SS}	Supply	Power	—	Supply ground

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

PIC16HV785 HARDWARE OVERVIEW

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

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

Hardware Features

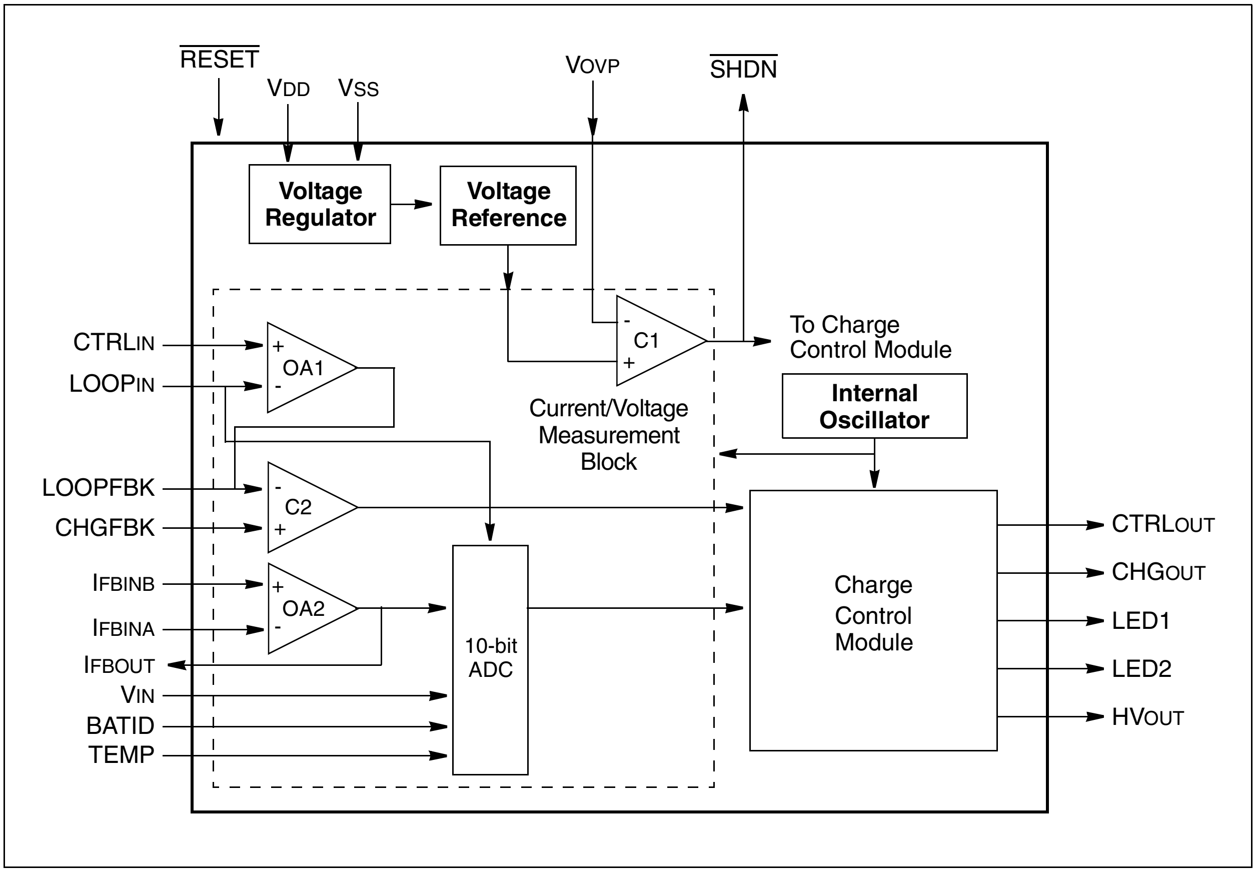
The PIC16HV785 features are well-suited for Switch mode battery charging. The PIC16HV785 device's block diagram (Figure 1) is to be used in conjunction with the Switch mode charger example (Figure 10, page 9).

- Current/Voltage Measurement Block – The Current/ Voltage Measurement Block consists of a 10-bit Analog-to-Digital converter, operational amplifiers and a comparator. The output of this block is fed into the charge control module (refer to Figure 1).

  The inputs into this block are to be connected as described in Figure 10. The following signals are inputs into this block:
  - LOOPFBK: to comparator
  - LOOPIN: to op amp and ADC
  - CTRLIN: to op amp
  - 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 precision internal 8 MHz clock oscillator eliminates the need for external oscillator circuits.
- In-circuit configurability utilizing 256 bytes of on-board EEPROM.
- Power on Reset – The POR insures the proper start-up of the PIC16HV785 when voltage is applied to VDD.
- Brown-out Reset – The BOR is activated when the input voltage falls to 2.1V; the PIC16HV785 is reset.

FIGURE 1: PIC16HV785 BLOCK DIAGRAM



REFERENCE SCHEMATIC

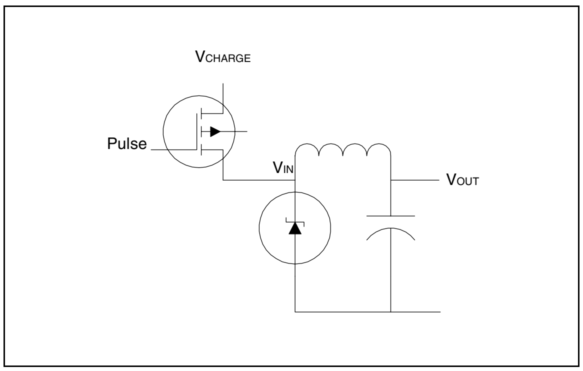
Theory of Operation

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

Buck Converter

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

FIGURE 2: BUCK CONVERTER



In the above diagram, when a constant voltage is applied to V_{IN} and a pulse train of constant frequency and duty cycle is applied to the age 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:

$$V_L = L\frac{di}{dt}$$

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

$$V_{CHARGE} * T_{ON} = V_{OUT} * T_{ON} + V_{OUT} * T_{OFF}$$

$$V_{CHARGE} * T_{ON} = V_{OUT} * T$$

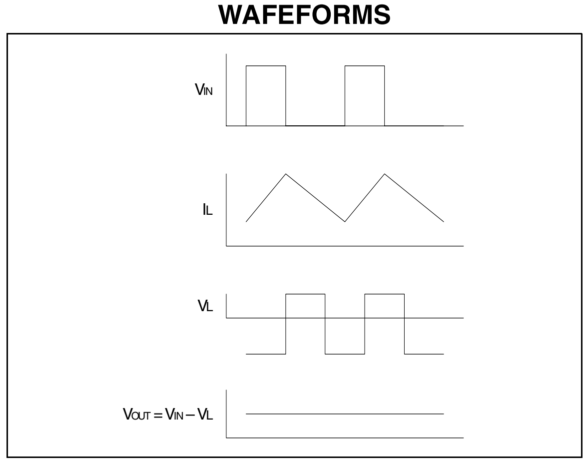
$$V_{OUT} = (T/T_{ON}) * V_{CHARGE}$$

$$V_{OUT} = V_{CHARGE} * \text{Duty Cycle}$$

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 (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 (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 WAFEFORMS



Feedback Circuits

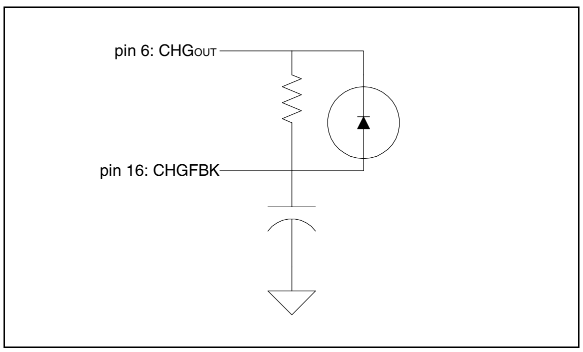
The circuit uses feedback for two purposes. One is to provide the ramp waveform that defines the PWM duty cycle. The other is the current sense that is compared to a reference voltage to determine if the current is being regulated at the correct level. This is also fed back into the PWM to modulate the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

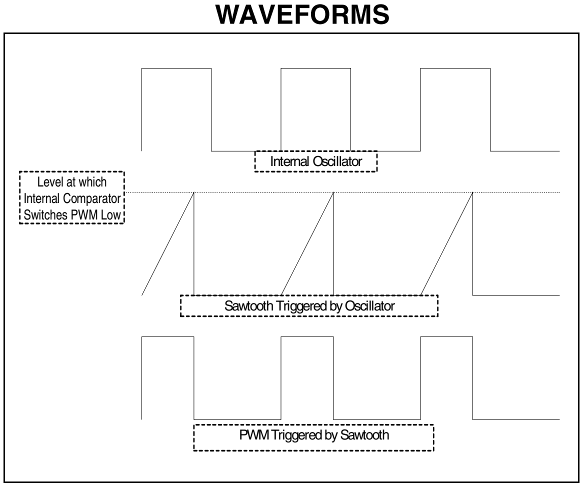
The RC circuit can be placed on the output of the PWM signal. A clamping diode can be used to control the total voltage drop.

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.

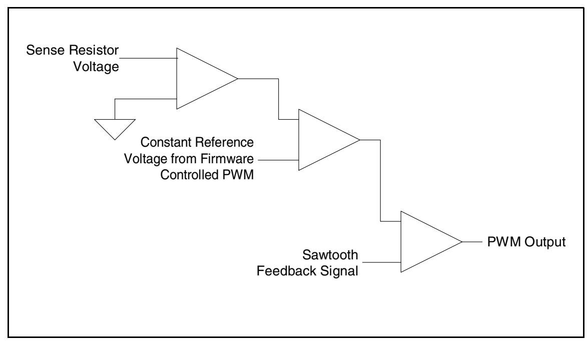
FIGURE 5: SAWTOOTH AND PWM WAVEFORMS



CURRENT FEEDBACK

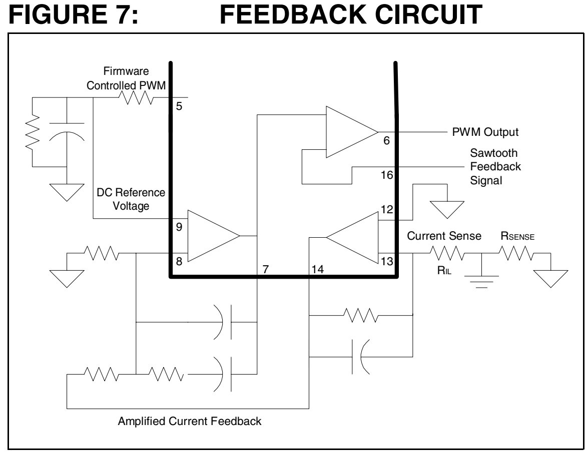
The aforementioned reference voltage is determined by current feedback in order to regulate the current. A second PWM, which is under firmware control, is used to create a DC level to which to compare the sensed current. The voltage drop across a current sense resistor is applied to pin 13 (IFBINA) and 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 below.

FIGURE 7: FEEDBACK CIRCUIT



Power Supply Shunt Regulator

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

EQUATION 4:

$$R_{MAX} = \frac{V_{S(MIN)} - 5V) * 1000}{1.05 * (16\ mA + I(led))}$$

$$R_{MIN} = \frac{V_{S(MAX)} - 5V) * 1000}{.95 * (50\ mA)}$$

Where:

R_{MAX} = maximum value of series resistor (ohms)

R_{MIN} = minimum value of series resistor (ohms)

$V_{S(MIN)}$ = minimum value of charger DC supply (V_{DC})

$V_{S(MAX)}$ = maximum value of charger DC supply (V_{DC})

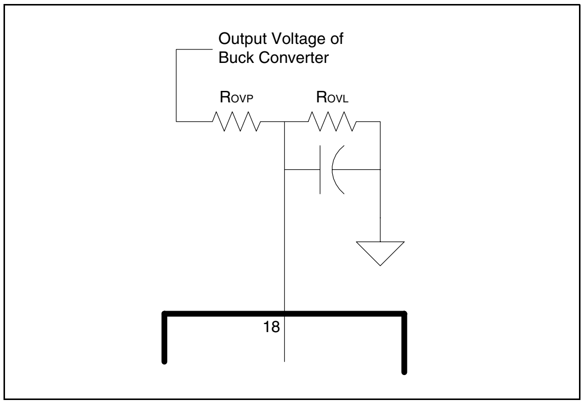
I(led) = total current drawn by all LEDs when illuminated simultaneously

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

Overvoltage Protection

The PIC16HV785 has a comparator that is gated to the PWM which compares the reference voltage to an external divided voltage applied to pin 18 (V_{OVP}). 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.

FIGURE 8: OVERVOLTAGE CIRCUIT



A/D Inputs

The internal A/D converter is used to measure the charging voltage on pin 3 (V_{IN}), the current on pin 13 (I_{FBINA}) 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 9: A/D INPUTS

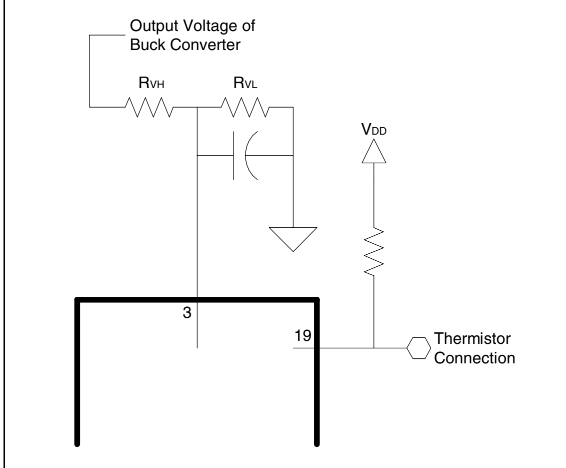
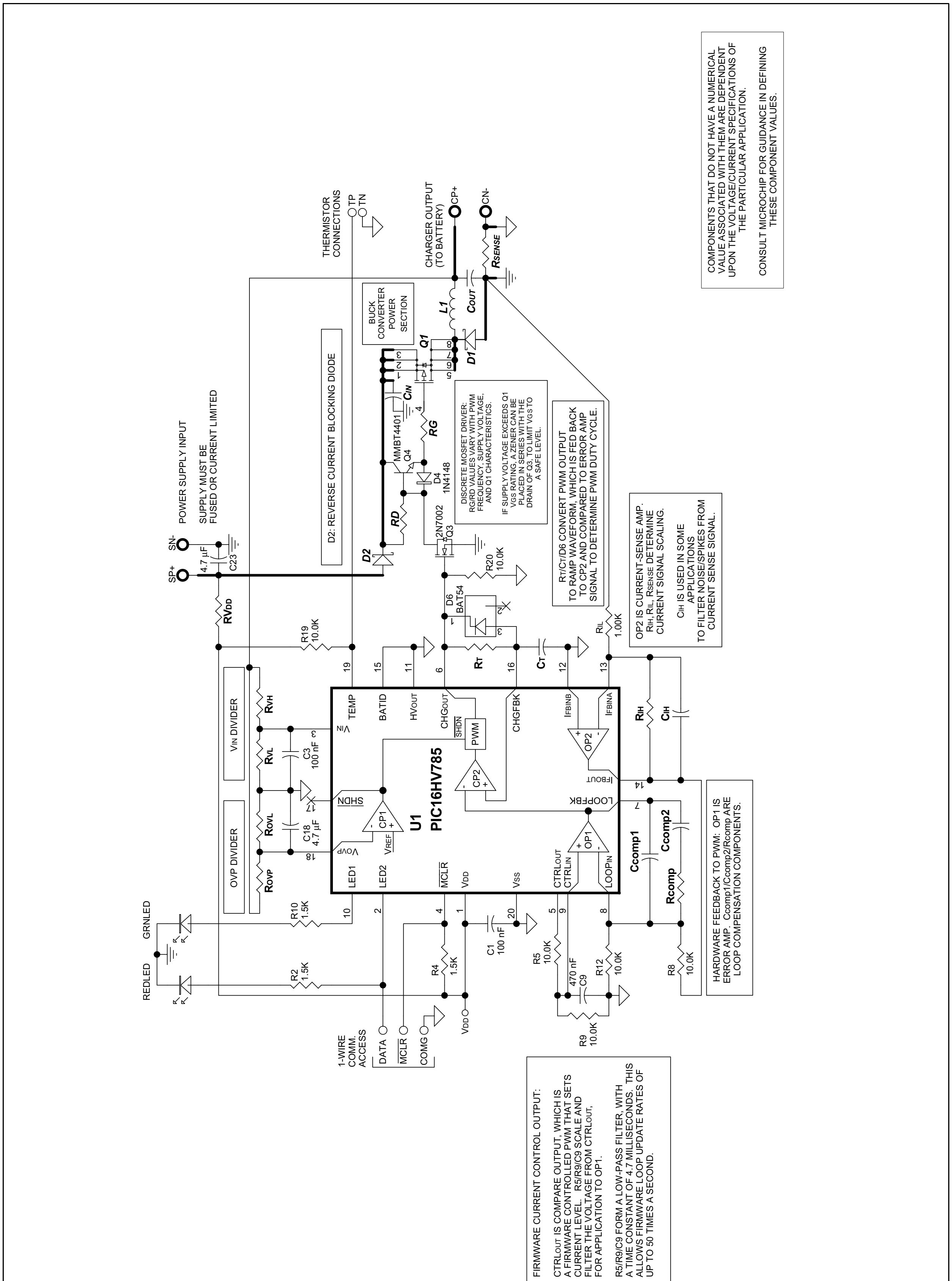


FIGURE 10: PIC16HV785 SWITCHING CHARGER SCHEMATIC



FUNCTIONAL DESCRIPTION: LITHIUM CHEMISTRY

Lithium Charging

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

CHARGE PENDING STATE – BEGINNING THE CHARGE CYCLE

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

CHARGE QUALIFICATION STATE

During Charge Qualification, the battery's temperature and voltage are measured to determine the next charging state. There are four possible next states.

1. If the battery's temperature is outside of the limits for charge initiation (**StartTMax**, **TempMin**), then the next state is Charge Suspend.
2. If the battery's voltage is less than the precharge threshold (**PCVolt**) and its temperature is within the limits for charge initiation (**StartTMax**, **TempMin**), then the next state is Precharge.
3. If the battery's voltage is above the precharge threshold (**PCVolt**) and its temperature is within the limits for charge initiation (**StartTMax**, **TempMin**), then the next state is Current Regulation.
4. If the battery's voltage is above the voltage at which charging will restart (**Vrchg**), then the next state is Charge Complete.

PRECHARGE STATE

The Precharge state allows for the recovery of a deeply discharged battery pack by applying a low charge rate. In this state, a user-configured precharge current is applied to the battery, resulting in an increase in the battery's voltage. There are three possible next states:

1. If the battery's voltage is above the precharge threshold (**PCVolt**) and the battery's temperature is within the limits for charge initiation (**StartTMax**, **TempMin**), then the next state is Current Regulation.
2. If the Precharge state time limit is exceeded (**PCTimeMax**) and the battery's voltage remains less than the precharge threshold (**PCVolt**), then the next state is Charge Suspend.

   If the Precharge state time limit is exceeded (**PCTimeMax**) and the battery's temperature is greater than the maximum temperature for charge initiation (**StartTMax**), then the next state is Charge Suspend.

   If the Precharge state time limit is exceeded (**PCTimeMax**) and the battery's temperature is less than the minimum temperature for charge initiation (**StartTMin**), then the next state is Charge Suspend.
3. If the battery pack is taken away, then the PIC16HV785 enters the Charge Pending state.

CHARGE SUSPEND STATE

In the Charge Suspend state, no current is applied to the battery pack. There are two possible next states:

1. If the battery's temperature is within the limits for charge initiation (**StartTMax**, **TempMin**) and its voltage is less than the voltage at which charging would restart (**VRVrech**), then the next state is Precharge.
2. If the battery pack is taken away, then the PIC16HV785 enters the Charge Pending state.

CURRENT REGULATION STATE

The Current Regulation state can be entered from the Precharge state or Charge Qualification state. Battery charging is initiated. This state provides constant current, voltage limited charging. The charge current is referred to as **CRCurrent** or the regulation current. While the current is applied, the battery's voltage increases until it reaches a voltage limit referred to as **CRVTarg** or regulation voltage. Charging continues, during which battery voltage and temperature are monitored. There are three possible next states.

1. If the battery's voltage reaches or exceeds the voltage limit (**CRVTarg**) and its temperature remains below the maximum allowable during current regulated charging (**TempMax**), then the next state is Voltage Regulation.
2. If the battery exhibits any one of the following conditions, then the next state is Charge Suspend:
   - Battery voltage exceeds upper voltage limit for charging (**Vmax**)
   - Battery temperature exceeds upper temperature limit for charging (**TempMax**)
   - Battery temperature is below the lower temperature limit for charging (**TMIN**)

   If the time in the Current Regulation state exceeds the time limit (**CRTimeMax**), then the next state is Charge Suspend.
3. If the battery pack is taken away, then the PIC16HV785 enters the Charge Pending state.

VOLTAGE REGULATION STATE

Voltage Regulation provides charging at a constant voltage while the charge current decreases (or tapers) to the user-specified minimum current threshold (**VRIMin**). There are three possible next states.

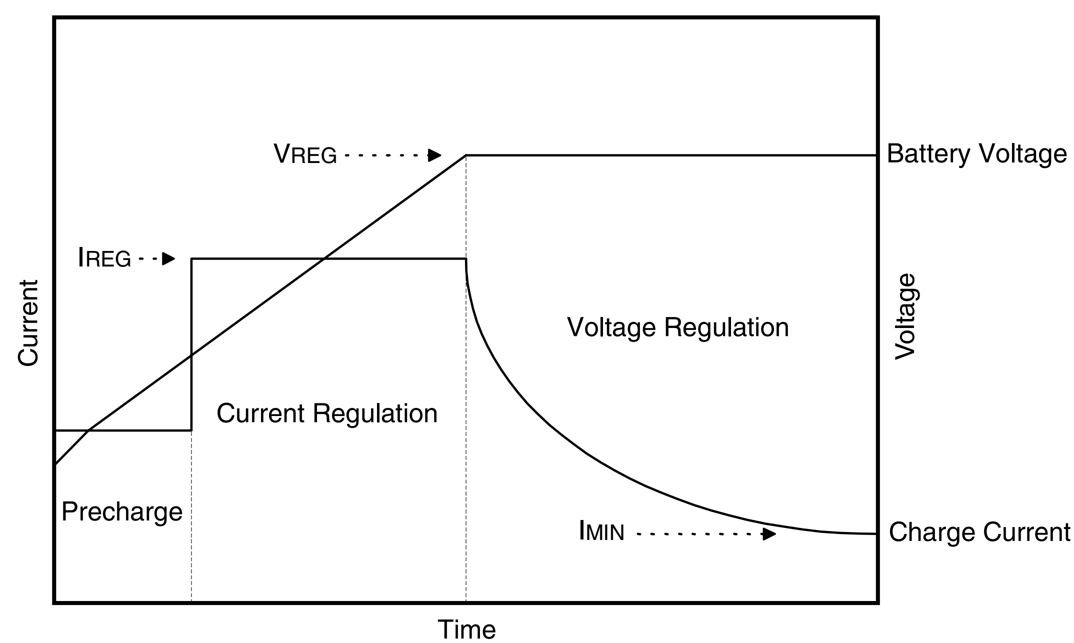
1. When the charge current reaches the taper current threshold for End-of-Charge (**VRIMin**) and the battery's voltage remains at the regulated voltage value (**CRVTarg**), then the battery has reached the Charge Complete state.
2. If the battery exhibits any one of the following conditions, then the next state is Charge Suspend.
   - Battery voltage exceeds upper voltage limit for charging (**Vmax**)
   - Battery temperature exceeds upper temperature limit for charging (**TempMax**)

   If the time in the Voltage Regulation state exceeds the time limit (**VRTimeMax**), then the next state is Charge Suspend.
3. If the battery pack is taken away, then the PIC16HV785 enters the Charge Pending state.

CHARGE CYCLE COMPLETE STATE

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

FIGURE 11: PIC16HV785 LITHIUM CHARGING PROFILE



FUNCTIONAL DESCRIPTION: NICKEL CHEMISTRY

Nickel Charging

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

CHARGE PENDING STATE – BEGINNING THE CHARGE CYCLE

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

CHARGE QUALIFICATION STATE

During charge qualification, the battery's temperature and voltage are measured to determine the next charging state. There are four possible next states.

1. If the battery's temperature is greater than the charge limit (**MaxTmp**), or the voltage is greater than the charge limit (**MaxVolt**), then the next state is Trickle Charge.
2. If the battery's voltage is less than the precharge threshold (**PCVolt**), or its temperature is outside of the limits for fast charge (**PCTempLo**, **PCTempHi**), then the next state is Precharge.
3. If the battery's voltage is above the precharge threshold (**PCVolt**) and its temperature is within the limits for fast charge (**PCTempLo**, **PCTempHi**), then the next state is Fast Charge.

PRECHARGE STATE

The Precharge state allows for the recovery of a deeply discharged battery pack by applying a low charge rate, or a slow recharge of a battery when the temperature is too high or too low to accept a fast charge. In this state, a user-configured precharge current is applied to the battery, resulting in an increase in the battery's voltage. There are three possible next states.

1. If the battery's voltage is above the precharge threshold (**PCVolt**) and the battery's temperature is within the limits for charge initiation (**PCTempLo**, **PCTempHi**), then the next state is Fast Charge.
2. If the Precharge state time limit is exceeded (**PCTimeMax**) while still in Precharge state, then the next state is Trickle Charge.
3. If the battery pack is taken away, then the PIC16HV785 enters the Charge Pending state.

FAST CHARGE STATE

In Fast Charge state, the full charging current is applied to the battery. There are three possible next states.

1. The charger will enter Trickle Charge state under any of the following conditions:
   - The voltage exceeds **MaxVolt**.
   - The temperature exceeds **MaxTmp**.
   - The time exceeds **FCTimeMax**.
   - -dV (minus delta V) full trigger is detected and time is greater than **FCTimeMin**.
   - dT/dt full trigger is detected and time is greater than **FCTimeMin** and Top Off state is disabled.
2. The charger will enter Top Off state when dT/dt full trigger is detected and time is greater than **FCTimeMin** and Top Off state is enabled.
3. If the battery pack is taken away, then the PIC16HV785 enters the Charge Pending state.

TOP OFF STATE

In the Top Off state, the dT/dt trigger has been reached and a lower amount of current is delivered to top off the battery. There are two possible next states.

1. The battery will exit Top Off state and enter Trickle Charge under the following conditions:
   - The voltage exceeds **MaxVolt**
   - The temperature exceeds **MaxTmp**
   - The time exceeds **TPTimeMax**
2. If the battery pack is taken away, then the PIC16HV785 enters the Charge Pending state.

TRICKLE CHARGE STATE

Trickle Charge state provides a very small constant current to maintain a full charge on the battery. There are two possible next states.

1. The battery will enter Charge Complete state if the voltage exceeds **MaxVolt** or the time exceeds **TPTimeMax**.
2. If the battery pack is taken away, then the PIC16HV785 enters the Charge Pending state.

CHARGE CYCLE COMPLETE STATE

When the charge is complete, all charge current terminates. The battery will re-enter Trickle Charge mode when the voltage is less than **Vrech** and voltage recharge is enabled. Charge complete is determined by the dT/dt method or the -dV method.

1. dT/dt: When the rate of change of the temperature increases suddenly, the temperature increases by greater than **dTDelta** in time **dTTime** and the charging can be declared complete.
2. -dV: When the voltage changes suddenly in the negative direction, drops by **dVDetect** suddenly after rising throughout the charge cycle, the charging can be declared complete.

CONFIGURABLE PARAMETERS

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

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

Configuration Parameters

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

Lithium Charging

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

Nickel Charging

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

LED Display Configuration

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

Look-up Tables

The look-up tables are grids of data that perform thermistor measurement linearization and PWM adjustment based on feedback measurements.

TABLE 2: PIC16HV785 LITHIUM/NICKEL CONFIGURATION PARAMETERS

Parameter Name	# Bytes	Typical Value	Units	Description
Configuration Parameters				
BandgapCF	2	248	integer	Internal band gap calibration factor.
BattDetectVolt	2	50	mV	Voltage threshold for battery detection.
BattIDMax	1	255	A/D full scale divided by 255	BATID input pin value maximum. When using BATID pin battery detection, voltage on BATID pin must be between **BattIDMax** and **BattIDMin** for battery present.
BattIDMin	1	0	A/D full scale divided by 255	BATID input pin value minimum. When using BATID pin battery detection, voltage on BATID pin must be between **BattIDMax** and **BattIDMin** for battery present.
Capacity (mAh)	2	2000	mAh	Full-charge capacity of the battery pack. For reference only.
CurrentCF	2	2553	integer	Current calibration factor.
DevName	—	PIC16HV785	ASCII	Device name. For reference only.
MfgName	—	Microchip	AXCII	Manufacturer's name. For reference only.
Mode	1	00000001b	binary	Configuration Register: bit 7: `1` = Always step through Precharge state bit 6: `1` = Enable GPIO cutoff logic bit5-3: Unused bit 2: `1` = Battery present on BATID bit 1: `1` = Battery present on voltage sense bit 0: `1` = Battery present always

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

Parameter Name	# Bytes	Typical Value	Units	Description
Configuration Parameters (Cont.)				
Mode2	1	00100000b	binary	Configuration Register: bit 7: 1 = Disable auto-offset calibration bit 6: 1 = Enable clock output on BATID pin after Reset bit 5: 1 = Use constant temperature from EEPROM bit 4: 1 = NiMh algorithm 0 = Li Ion algorithm bit 3: Unused bit 2: 1 = Disable voltage recharge trigger (**VRVrech**) bit 1: 1 = Disable voltage cutoff in regulator bit 0: 1 = Disable PWM auto-shutdown
OscTrim	1	0	integer	Oscillator trim calibration value.
PWMFreq	1	15	integer	LUT value which determines the PWM frequency.
PatternID	2	0x102	integer	ID for parameter set.
SHUNT	1	100	mOhms	Shunt resistor value.
SeriesCells	1	4	integer	Number of series connected cells in the battery pack.
Tdefault	1	112	code	Default temperature when using constant temperature in EEPROM (°C * 10 + 200)/4.
TempCF	2	8192	integer	Temperature calibration value.
TimerEOCRecheck	1	20	.25 sec.	Recheck timer for End-of-Charge condition.
TimerStChng	1	20	.25 sec.	Recheck timer for state change.
VoltageCF	2	5121	integer	Voltage calibration value.
Li Ion Charging Parameters				
CRCurrent	2	800	mA	Charging current during current regulation.
CRTimeMax	1	0	4 min.	Current regulation time limit.
CRVTarg	2	4200	mV	Target cell voltage in current regulation. This is set to the fully charged voltage of one cell, typically, as specified by the cell manufacturer.
PCCurrent	2	200	mA	Charging current during precharge.
PCTempThresh	1	175	code	Maximum temperature required to enable charging with precharge conditions (°C * 10 + 200)/4.
PCTimeMax	1	60	4 min.	Duration of precharge.
PCVolt	2	3000	mV	Cell voltage under which precharge occurs.
StartTMax	1		code	Maximum temperature under which charging may begin (°C * 10 + 200)/4.
TempMax	1	200	code	Maximum temperature under which charging is allowed to continue (°C * 10 + 200)/4.
TempMin	1	50	code	Minimum temperature over which charging is allowed to continue or start (°C * 10 + 200)/4.
Vmax	2	4300	mV	Voltage which triggers Charge Suspend mode when exceeded.

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

Parameter Name	# Bytes	Typical Value	Units	Description
Li Ion Charging Parameters (Cont.)				
VRIMin	2	50	mA	Voltage regulation fully charged current. This is the value of the taper current which will determine that the battery is fully charged.
VRTimeMax	1	90	4 min.	Voltage regulation time limit.
VRVrech	2	4100	mV	Voltage regulation recharge cell voltage. Charger will begin recharging if cell voltage of pack falls below **SeriesCells** * **VRVrech**.
Vsafety	2	4350	mV	Charger will shut down immediately when voltage exceeds **Vsafety**.
Nickel Parameters				
ChargeVolt	2	2000	mV	Charge voltage.
DischVolt	2	1000	mV	Discharge target voltage.
FCCurr	2	2000	mA	Fast charge current.
FCTimeMax	1	0	4 min.	Fast charge time limit.
FCTimeMin	1	0	4 min.	Fast charge time minimum.
MaxTmp	1	175	code	Maximum temperature for charging (°C * 10 + 200)/4.
MaxVolt	2	1800	mV	Maximum voltage for charging.
Mode3	1	00111010b	binary	Configuration Register: bit 7: `1` = Enable external discharge bit 6: `1` = Enable external trickle circuit bit 5: `1` = Bypass Trickle Charge state bit 4: `1` = Enable Top Off state bit 3: `1` = Enable Trickle Charge if voltage drops below recharge voltage bit 2: Unused bit 1: `1` = Enable dT/dt EOC method bit 0: `1` = Enable -dV EOC method
PCCurrent	2	100	mA	Precharge current.
PCTempHi	1	160	code	High temperature limit over which the Precharge state is entered (°C * 10 + 200)/4.
PCTempLo	1	75	code	Low temperature limit under which the Precharge state is entered (°C * 10 + 200)/4.
PCTimeMax	1	0	4 min.	Precharge time limit.
PCTimeMin	1	0	4 min.	Precharge time minimum.
PCVolt	2	800	mV	Voltage threshold under which Precharge state occurs.
ShutdownVolt	2	1900	mV	Voltage which causes immediate shutdown when exceeded.
TPCurr	2	200	mA	Top Off state current.
TPTimeMax	1	11	4 min.	Top Off state time limit.
TRCurr	2	80	mA	Trickle state current.
TRTimeMax	1	150	4 min.	Trickle state time limit.
VoltCal	2	2544	integer	Voltage calibration value.
Vrech	2	1300	mV	Voltage threshold under which Trickle Charge state is re-entered.
dTDelta	1	10	0.1°C	Change in temperature for dT/dt EOC condition.
dTTime	1	120	0.5 sec.	Change in time for dT/dt EOC condition.
dVDetect	2	10	mV	Negative change in voltage for -dV EOC condition.

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

Parameter Name	# Bytes	Typical Value	Units	Description
LUT Parameters				
PWMAdjust1	1	12	integer	PWM adjustment for regulation control.
PWMAdjust2	1	10	integer	PWM adjustment for regulation control.
PWMAdjust3	1	5	integer	PWM adjustment for regulation control.
PWMAdjust4	1	1	integer	PWM adjustment for regulation control.
Vhh	1	19	mV	Voltage PWM adjustment zone limit.
Vh	1	6	mV	Voltage PWM adjustment zone limit.
Vl	1	6	mV	Voltage PWM adjustment zone limit.
Vll	1	44	mV	Voltage PWM adjustment zone limit.
Chl	1	5	mA	Current PWM adjustment zone limit.
T_LUT_N	1	8	integer	Number of temperature linearization LUT entries.
T_LUT_T_0	1	38	integer	Temperature A/D reading axis point.
T_LUT_T_1	1	48	integer	Temperature A/D reading axis point.
T_LUT_T_2	1	61	integer	Temperature A/D reading axis point.
T_LUT_T_3	1	79	integer	Temperature A/D reading axis point.
T_LUT_T_4	1	105	integer	Temperature A/D reading axis point.
T_LUT_T_5	1	183	integer	Temperature A/D reading axis point.
T_LUT_T_6	1	207	integer	Temperature A/D reading axis point.
T_LUT_M_0	2	-23362	integer	Temperature linearization slope LUT entry.
T_LUT_B_0	2	1418	integer	Temperature linearization Y-intercept LUT entry.
T_LUT_M_1	2	-19864	integer	Temperature linearization slope LUT entry.
T_LUT_B_1	2	1352	integer	Temperature linearization Y-intercept LUT entry.
T_LUT_M_2	2	-15709	integer	Temperature linearization slope LUT entry.
T_LUT_B_2	2	1255	integer	Temperature linearization Y-intercept LUT entry.
T_LUT_M_3	2	-12572	integer	Temperature linearization slope LUT entry.
T_LUT_B_3	2	1162	integer	Temperature linearization Y-intercept LUT entry.
T_LUT_M_4	2	-10206	integer	Temperature linearization slope LUT entry.
T_LUT_B_4	2	1071	integer	Temperature linearization Y-intercept LUT entry.
T_LUT_M_5	2	-8631	integer	Temperature linearization slope LUT entry.
T_LUT_B_5	2	990	integer	Temperature linearization Y-intercept LUT entry.
T_LUT_M_6	2	-10154	integer	Temperature linearization slope LUT entry.
T_LUT_B_6	2	1127	integer	Temperature linearization Y-intercept LUT entry.
T_LUT_M_7	2	-12875	integer	Temperature linearization slope LUT entry.
T_LUT_B_7	2	1402	integer	Temperature linearization Y-intercept LUT entry.
LED Parameters				
LED1State1	1	00000000b	binary	LED1 display during state 1 Lithium: Charge Pending Nickel: Charge Pending
LED1State2	1	00000000b	binary	LED1 display during state 2 Lithium: Charge Qualification Nickel: Charge Qualification
LED1State3	1	00000000b	binary	LED1 display during state 3 Lithium: Precharge Nickel: Precharge

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

Parameter Name	# Bytes	Typical Value	Units	Description
LED Parameters (Cont.)				
LED1State4	1	00000000b	binary	LED1 display during state 4 Lithium: Current Regulation Nickel: Fast Charge
LED1State5	1	00000000b	binary	LED1 display during state 5 Lithium: Voltage Regulation Nickel: Top Off
LED1State6	1	00000000b	binary	LED1 display during state 6 Lithium: Charge Cycle Complete Nickel: Charge Cycle Complete
LED1State7	1	00000000b	binary	LED1 display during state 7 Lithium: Charge Suspended Nickel: *Unused*
LED1State8	1	00000000b	binary	LED1 display during state 8 Lithium: *Unused* Nickel: Trickle Charge
LED2State1	1	00000000b	binary	LED1 display during state 1 Lithium: Charge Pending Nickel: Charge Pending
LED2State2	1	00000000b	binary	LED1 display during state 2 Lithium: Charge Qualification Nickel: Charge Qualification
LED2State3	1	00000000b	binary	LED1 display during state 3 Lithium: Precharge Nickel: Precharge
LED2State4	1	00000000b	binary	LED1 display during state 4 Lithium: Current Regulation Nickel: Fast Charge
LED2State5	1	00000000b	binary	LED1 display during state 5 Lithium: Voltage Regulation Nickel: Top Off
LED2State6	1	00000000b	binary	LED1 display during state 6 Lithium: Charge Cycle Complete Nickel: Charge Cycle Complete
LED2State7	1	00000000b	binary	LED1 display during state 7 Lithium: Charge Suspend Nickel: *Unused*
LED2State8	1	00000000b	binary	LED1 display during state 8 Lithium: *Unused* Nickel: Trickle Charge

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 3: PWM FREQUENCY

F:	8.000	PWMP<6:5>			
		0	1	2	3
PWMP<4:0>	0	8000	4000	2000	1000
	1	4000	2000	1000	500
	2	2667	1333	667	333
	3	2000	1000	500	250
	4	1600	800	400	200
	5	1333	667	333	167
	6	1143	571	286	143
	7	1000	500	250	125
	8	889	444	222	111
	9	800	400	200	100
	10	727	364	182	91
	11	667	333	167	83
	12	615	308	154	77
	13	571	286	143	71
	14	533	267	133	67
	15	500	250	125	63
	16	471	235	118	59
	17	444	222	111	56
	18	421	211	105	53
	19	400	200	100	50
	20	381	190	95	48
	21	364	182	91	45
	22	348	174	87	43
	23	333	167	83	42
	24	320	160	80	40
	25	308	154	77	38
	26	296	148	74	37
	27	286	143	71	36
	28	276	138	69	34
	29	267	133	67	33
	30	258	129	65	32
	31	250	125	63	31

The BATID pin is used to determine if a battery is present by measuring the voltage on the pin and comparing it to the proper EEPROM parameters. Alternatively, after a Reset and during initialization, this pin can be configured by the **Mode2** parameter to output a single burst of 256 clocks in order to determine the frequency of the internal oscillator.

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

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

Mode<6> = 1: Charge Current Switch

Used as an indication of charge current flowing.

HVOUT = 1: Charge current flowing

HVOUT = 0: No charge current flowing

Mode3<7> = 1: Enable External Discharge Circuit

An option in Nickel charging is to provide a load to discharge a battery immediately prior to charging. This can help in detection of End-of-Charge conditions.

HVOUT = 1: Default

HVOUT = 0: Discharge circuit enabled

Mode3<6> = 1: Enable External Trickle Charge Circuit

If the charger cannot satisfactorily regulate a low-level trickle charge current, an external trickle charge circuit may be used. The HVOUT pin, in this case, can control the trickle charge circuit when the Trickle Charge state is entered.

HVOUT = 1: Default

HVOUT = 0: Trickle charge circuit enabled

Main Loop

The main loop cycles through the following functions:

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

The actual subroutines are:

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

Triggers and Charge States

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

Regulating the PWM

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

As Table 4 shows, if the voltage feedback is no greater than **Vh** more than the requirement, and no less than **Vl** lower than the requirement, the PWM is unchanged. If the feedback voltage exceeds the required voltage by more than **Vl**, the PWM is decreased by **PWMAdjust4**, etc.

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

TABLE 4: PWM ADJUSTMENT FACTORS

Voltage Zones	Current Zones				
	< -Cll	**< -Cl**	**-Chl to +Chl**	**> Ch**	**> Chh**
> Vhh	**-PWMAdjust1**	**-PWMAdjust1**	**-PWMAdjust1**	**-PWMAdjust1**	**-PWMAdjust1**
> Vh	**-PWMAdjust4**	**-PWMAdjust4**	**-PWMAdjust4**	**-PWMAdjust4**	**-PWMAdjust2**
Vh to -Vl	0	0	0	**-PWMAdjust4**	**-PWMAdjust2**
< -Vl	**+PWMAdjust4**	**+PWMAdjust4**	0	**-PWMAdjust4**	**-PWMAdjust2**
< -Vll	**+PWMAdjust3**	**+PWMAdjust4**	0	**-PWMAdjust4**	**-PWMAdjust2**

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

Mode<7,3>	Mode Description	N<6:4>	F<2:0>
`00`	Off	N/A	N/A
`01`	Flash N + 1 Times, Pause, Repeat	Flash Count = N + 1	On Time = Off Time = F + 1 Pause Time = (F + 1) * 5 Max = 3
`10`	On	N/A	N/A
`11`	Flash Continuously	On Time = N + 1	Off Time = F + 1

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/D_{RAW}) as follows:

EQUATION 5:

$$VR = A/D_{RAW} * 16384/\mathbf{BandgapCF}$$

BandgapCF is typically around 248 since:

$$VR/V_{DD} * A/D_{RAW}(FULLSCALE) = 1212/5000 * 1023 = 248$$

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/D_{RAW}) as follows:

EQUATION 6:

When referenced to VR:

$$\text{Current} = A/D_{RAW} * \textbf{CurrentCF}/65536$$

When referenced to VDD:

$$\text{Current} = (A/D_{RAW} * VR/16384) * \textbf{CurrentCF}/65535$$

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

EQUATION 7:

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

$$2551 = 1023 * \textbf{CurrentCF}$$

$$\textbf{CurrentCF} = 2.494$$

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

EQUATION 8:

$$\textbf{CurrentCF} \text{ Base} = 1024$$

$$\textbf{CurrentCF} = 2553$$

VOLTAGE

The voltage reading is calibrated or translated from the raw A/D measurement (A/D_{RAW}) as follows:

EQUATION 9:

When referenced to VR:

$$\text{Voltage} = A/D_{RAW} * \textbf{VoltageCF}/1024$$

When referenced to VDD:

$$\text{Voltage} = (A/D_{RAW} * VR/16384) * \textbf{VoltageCF}/1024$$

Where **VoltageCF** is determined as follows:

$$\text{Voltage} = A/D_{RAW} * \textbf{VoltageCF}'$$

$$A/D_{RAW} = (\text{Voltage} * \text{Cells}) * R/V_{REF} * 1023$$

Where:

R = Resistor Divider Ratio

V_{REF} = 5000 mV

This means:

$$\text{Voltage} = \textbf{VoltageCF}' * \text{Voltage} * \text{Cells} * R/V_{REF} * 1023$$

or

$$\textbf{VoltageCF}' = V_{REF}/(\text{Cells} * R * 1023)$$

and using integer arithmetic:

$$\textbf{VoltageCF} = \textbf{VoltageCF}' * 1024$$

So that:

$$\text{Voltage} = \textbf{VoltageCF} * A/D_{RAW}/1024$$

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

TABLE 6: TYPICAL VoltageCF VALUES FOR PS2070

Cells	R#	R1	R2	R Ratio	VoltageCF
1	1	0.232	10.0	0.9773	5121
2	2	10.500	10.0	0.4878	5130
3	3	20.500	10.0	0.3279	5088
4	4	30.900	10.0	0.2445	5117

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/D_{RAW}) as follows:

EQUATION 10:

$$\text{Temperature} = A/D_{RAW} * \textbf{TempCF}/8192$$

Where temperature is in the internal units of:

$$(°C + 20) * 10$$

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

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

Thermistor Linearization

The thermistor reading is subjected to piecewise linear interpolation using a look-up table of line equations. Since the variance of voltage with temperature for the thermistor is not always along the same line (same slope and intercept), multiple line equations must be used for interpolation depending on where the measurement is. The look-up table was developed by rating raw A/D values; that is why **TempCF** can typically be set to 1.

The look-up table is a series of slopes and Y-intercepts corresponding to regions of temperature A/D readings. **T_LUT_N** represents the number of entries in the table, in this case eight entries.

TABLE 7: THERMISTOR LINEARIZATION

A/D Reading	Slope	Y-intercept
< **T_LUT_T_0**	**T_LUT_M_0**	**T_LUT_B_0**
< **T_LUT_T_1**	**T_LUT_M_1**	**T_LUT_B_1**
< **T_LUT_T_2**	**T_LUT_M_2**	**T_LUT_B_2**
< **T_LUT_T_3**	**T_LUT_M_3**	**T_LUT_B_3**
< **T_LUT_T_4**	**T_LUT_M_4**	**T_LUT_B_4**
< **T_LUT_T_5**	**T_LUT_M_5**	**T_LUT_B_5**
< **T_LUT_T_6**	**T_LUT_M_6**	**T_LUT_B_6**
> **T_LUT_T_6**	**T_LUT_M_7**	**T_LUT_B_7**

The typical values are:

TABLE 8: A/D TEMPERATURE READINGS

A/D Reading	Slope	Y-intercept
< 38	-23362	1418
< 48	-19864	1352
< 61	-15709	1255
< 79	-12572	1162
< 105	-10206	1071
< 183	-8631	990
< 207	-10154	1127
> 207	-12875	1402

Communication

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

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

A Break cycle is defined by a low period of time equal to or greater than time t_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 10 are implemented.

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

READ: BREAK, REG_ADDR, DATA

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

FIGURE 12: SINGLE PIN SERIAL TIMING

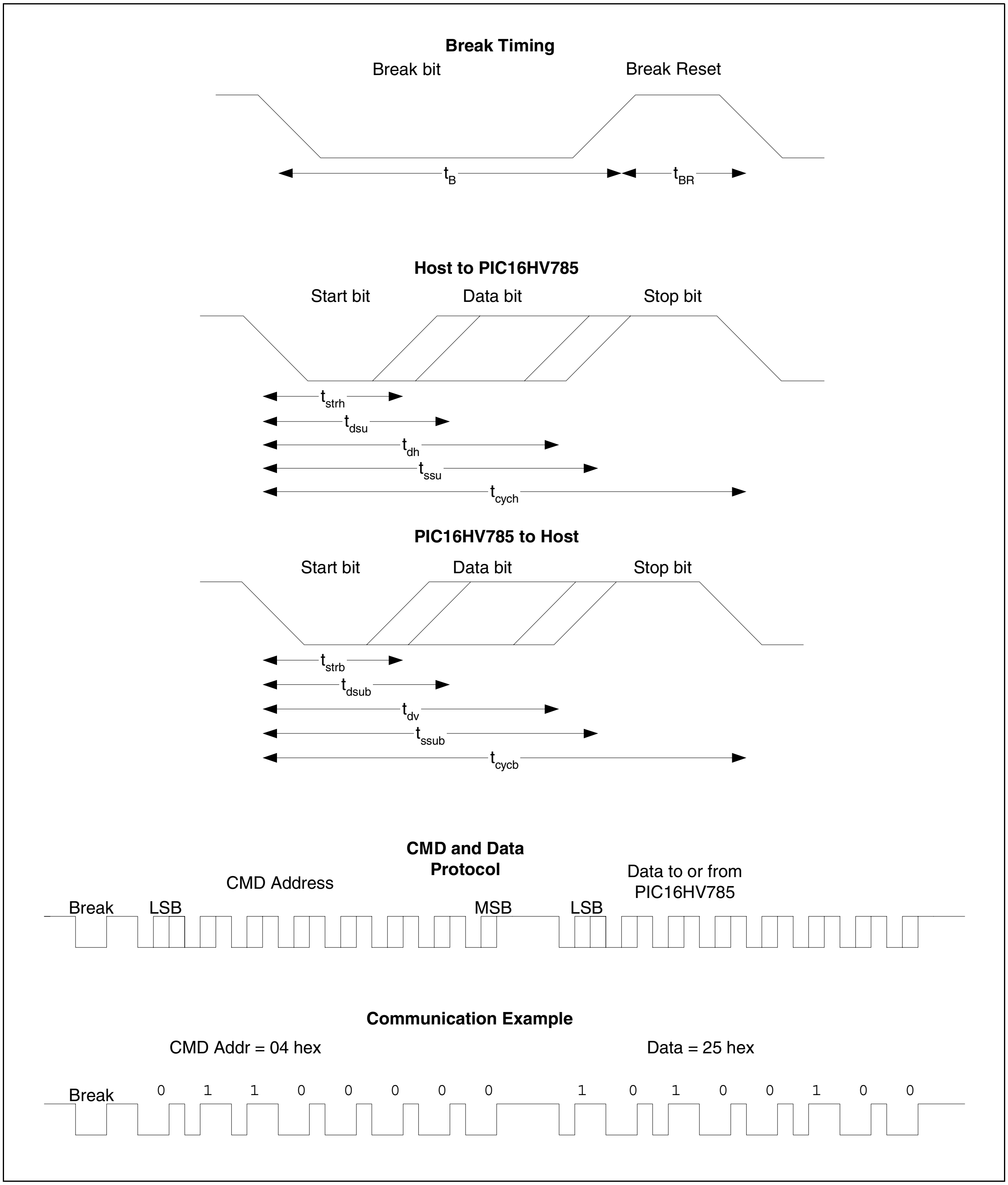


TABLE 9: REGISTER SUMMARY

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

REGISTER DESCRIPTIONS

REGISTER 0: MEM_ADDR

Bit	Name	Description
7:0	MEM_ADDR	Indirect memory address used for reading and writing data

REGISTER 1: STATUS

Bit	Name	Description
7	EE_Busy	`1` = EEPROM write is in progress; busy
6	EE_Err	`1` = Error encountered during last EEPROM write
5	Unused	
4	REG_ACTIVE	`1` = Regulation active
3	CHGCON	`1` = Charge controller active
2	SIM_ACTIVE	`1` = Data simulation active
1	Unused	
0	Unused	

REGISTER 2: CONFIG

Bit	Name	Description
7:6	Unused	
5	SUSPEND	`1` = Suspend/skip all processing (used when writing EEPROM)
4	CHGCON_OFF	`1` = Suspend charge controller
3:2	Unused	
1	MEMBANK_EE	`1` = Indirect memory addressing refers to EEPROM
0	MEMBANK_23	`1` = Indirect memory addressing refers to 2nd bank of RAM

REGISTER 3: CMND

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

REGISTER 4: DATA_LO

Bit	Name	Description
7:0	DATA_LO	Generic data used in memory reads and writes (LSB)

REGISTER 5: DATA_HI

Bit	Name	Description
7:0	DATA_HI	Generic data used in memory reads and writes (MSB)

REGISTER 6: UNUSED

Bit	Name	Description
7:0	Unused	

REGISTER 7: UNLOCK

Bit	Name	Description
7:0	UNLOCK	Unlock code is written here

REGISTER 8: MEM_ACCESS

Bit	Name	Description
7:0	MEM_ACCESS	Data written to Register 8 is actually sent to the memory address contained in Register 0 and the bank indicated by Register 2 (bits<1:0>)

REGISTER C: MEM_ACCESS_IA

Bit	Name	Description
7:0	MEM_ACCESS_IA	Data written to Register 8 is actually sent to the memory address contained in Register 0 and the bank indicated by Register 2 (bits<1:0>); Register 0 will be post-incremented

Host Driven Operations

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

RAM READ

There are three steps to the RAM read:

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

RAM WRITE

There are three steps to the RAM write:

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

EEPROM READ

There are three steps to the EEPROM read:

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

EEPROM WRITE

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

TABLE 10: EEPROM WRITE CONTROL PACKET

Byte	Name	Description
0	ADDR	Starting EEPROM Address to be Written
1	COUNT	Byte Count (N), Maximum = 29
2	DATA	Data[0]
...		
N + 1	DATA	Data[N – 1]
N + 2	CHKSUM	Checksum = Sum (byte[0]:byte[N + 1])

The total procedure is a five step process:

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

FIRMWARE SOURCE CODE

Define Constants, Registers and EEPROM Locations

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

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

```
;======================================================================
;======================================================================

;--- defines
;#define   CLOCK_4MHZ
#define    CLOCK_8MHZ
;#define   ENABLE_COMM_LOCK
;#define   DEBUG_ENABLE_TOGGLE

;--- firmware version
#define    FW_VERSION_LO  0x01
#define    FW_VERSION_HI  0x02

#include   "p16f785.inc"

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

;------------------------------------------------------------
;--- registers: special function
;------------------------------------------------------------
r_indf            equ INDF
r_tmr0            equ 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 TMR1L
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_ccp1con          equ CCP1CON
r_wdtcon           equ WDTCON
r_adresh           equ ADRESH
r_adcon0           equ ADCON0

r_option_reg       equ OPTION_REG
r_tris_a           equ TRISA
r_tris_b           equ TRISB
r_tris_c           equ TRISC
r_pie1             equ PIE1
r_pcon             equ PCON
r_osccon           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_eecon1           equ EECON1
r_eecon2           equ EECON2
r_adresl           equ ADRESL
r_adcon1           equ ADCON1
r_pwmcon1          equ PWMCON1
r_pwmcon0          equ PWMCON0
r_pwmclk           equ PWMCLK
r_pwmph1           equ PWMPH1
r_pwmph2           equ PWMPH2
r_cm1con0          equ CM1CON0
r_cm2con0          equ CM2CON0
r_cm2con1          equ CM2CON1
r_opa1con          equ OPA1CON
r_opa2con          equ OPA2CON

;*** register bank limits
#define    ram0_start            0x20
#define    ram0_end              0x7f
#define    ram0_length           ram0_end - ram0_start + 1
#define    ram1_start            0xa0
#define    ram1_end              0xef
#define    ram1_length           ram1_end - ram1_start + 1

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

```
r_adc_4           equ $                          ;
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_cmnd  res 1                          ;

r_mode2           res 1                          ;

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

```
r_adc_3_save_b     res 2                         ;

r_mode3            res 1                         ;
r_timer_d2         res 1                         ;
;------------------------------------------------------------
;--- registers: bank0,1,2,3 (common)
;------------------------------------------------------------
                   org 0x70                      ; *** bank 0 (common area)
r_shadow_1         res 1                         ;
r_shadow_2         res 1                         ;
r_shadow_3         res 1                         ;
;r_shadow_4        res 1                         ;
r_flags_1          res 1                         ; assorted bit flags
r_flags_2          res 1                         ; assorted bit flags
r_flags_3          res 1                         ; assorted bit flags
r_flags_4          res 1                         ; assorted bit flags
r_flags_5          res 1                         ; assorted bit flags
r_flags_6          res 1                         ; assorted bit flags
r_isr_w            res 1                         ; interrupt context
r_isr_status       res 1                         ; interrupt context
r_isr_pclath       res 1                         ; interrupt context
r_isr_fsr          res 1                         ; interrupt context
r_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_nm_extdchg_en  r_mode3, 7 ; enable discharge
#define    flag0_mode_nm_exttrik_en  r_mode3, 6 ; enable discharge
#define    flag0_mode_nm_trik_byp    r_mode3, 5 ; disable trickle stage
#define    flag0_mode_nm_topoff_en   r_mode3, 4 ; enable topoff stage
#define    flag0_mode_nm_vrchg_en    r_mode3, 3 ; enable trickle re-charge
#define    flag0_mode_nm_slochg      r_mode3, 2 ; enable slow charge mode
#define    flag0_mode_nm_eoc_dtdt    r_mode3, 1 ; enable eoc method - dtdt
#define    flag0_mode_nm_eoc_dv      r_mode3, 0 ; enable eoc method - minus dv

BN_CREG_EE_BUSY    equ .7                        ; ee write busy
BN_CREG_EE_ERR     equ .6                        ; error on last ee write
;BN_CREG_UNLOCKED  equ .5                        ; comm unlocked
BN_CREG_REG        equ .4                        ; regulation active
BN_CREG_CHGCON     equ .3                        ; charge controller enabled
BN_CREG_SIM        equ .2                        ; simulation active (>=1 channel)

#define    flag0_creg_st_ee_busy     r_comm_reg_1, BN_CREG_EE_BUSY
#define    flag0_creg_st_ee_err      r_comm_reg_1, BN_CREG_EE_ERR
;#define   flag0_creg_st_unlocked    r_comm_reg_1, BN_CREG_UNLOCKED
#define    flag0_creg_st_reg         r_comm_reg_1, BN_CREG_REG
#define    flag0_creg_st_chgcon      r_comm_reg_1, BN_CREG_CHGCON
#define    flag0_creg_st_sim         r_comm_reg_1, BN_CREG_SIM

#define    flag0_creg_suspend        r_comm_reg_2, 5
```

```
#define    flag0_creg_chgcon_off      r_comm_reg_2, 4
#define    flag0_creg_membank_ee      r_comm_reg_2, 1
#define    flag0_creg_membank_23      r_comm_reg_2, 0

#define    flag0_creg_version         r_comm_reg_3, 7
#define    flag0_creg_pwm_set         r_comm_reg_3, 6
#define    flag0_creg_reg_on          r_comm_reg_3, 5
#define    flag0_creg_ee_rq           r_comm_reg_3, 4
#define    flag0_creg_test            r_comm_reg_3, 3
#define    flag0_creg_reset           r_comm_reg_3, 2
#define    flag0_creg_fchgstate       r_comm_reg_3, 1
#define    flag0_creg_sim_rq          r_comm_reg_3, 0

#define    flag_ee_busy               r_flags_1, 7
#define    flag_ee_rq                 r_flags_1, 6
#define    flag_ee_err                r_flags_1, 5
#define    flag_simdata_ready         r_flags_1, 4
#define    flag_chg_state_timer       r_flags_1, 3
#define    flag_math_temp             r_flags_1, 2
#define    flag_timer_0               r_flags_1, 1
#define    flag_led_timer             r_flags_1, 0

;--- 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_v_le_vpchg            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_tmaxchg          r_flags_2, 1
#define    flag_t_le_tpchg            r_flags_2, 0
;--- trigger flags - nimh
#define    flag_v_le_vpchg_nm         r_flags_2, 7
#define    flag_t_le_tpchg_lo_nm      r_flags_2, 6
#define    flag_t_le_tpchg_hi_nm      r_flags_2, 5
#define    flag_t_le_tmaxchg_nm       r_flags_2, 4
#define    flag_v_le_vmaxchg_nm       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_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_rqq          r_flags_4, 4
#define    flag_adcset_1_rdy          r_flags_4, 3
#define    flag_adcset_1_rqq          r_flags_4, 2
#define    flag_adcset_0_rdy          r_flags_4, 1
#define    flag_adcset_0_rqq          r_flags_4, 0

#define    flag_reg_timer             r_flags_5, 7
#define    flag_battpres1             r_flags_5, 6
#define    flag_battpres              r_flags_5, 5
```

```
#define    flag_comm_active          r_flags_5, 4
#define    flag_reg_on               r_flags_5, 3
#define    flag_vreg                 r_flags_5, 2
#define    flag_vreg_2               r_flags_5, 1
#define    flag_vreg_1               r_flags_5, 0


#define    BN_CHGN_TSEL   0
#define    flag_chg_ti1_done         r_flags_6, 7
#define    flag_chg_ti2_done         r_flags_6, 6
#define    flag_chgn_tsel            r_flags_6, BN_CHGN_TSEL
#define    MASK_CHGN_TSEL            1<<BN_CHGN_TSEL
#define    REG_CHGN_TSEL             r_flags_6

#define    ADCH_4                    0x10
#define    ADCH_3                    0x08
#define    ADCH_2                    0x04
#define    ADCH_1                    0x02
#define    ADCH_0                    0x01

#define    ADCSET_0  ADCH_4 | ADCH_3 | ADCH_2 | ADCH_1 | ADCH_0
#define    ADCSET_1  ADCH_2 | ADCH_1
#define    ADCSET_2  ADCH_4 | ADCH_3 | ADCH_2 | ADCH_1

#define    COMM_UNLOCK_KEY           0x96

#define    flag_comm_pin             r_comm_flags, 5
#define    flag_comm_timeout         r_comm_flags, 4
#define    flag_comm_cmnd            r_comm_flags, 3
#define    flag_comm_bit             r_comm_flags, 2
#define    flag_comm_H2L             r_comm_flags, 1
#define    flag_comm_xmit            r_comm_flags, 0

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

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

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

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

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

```
;--- option
option_default         equ 1<<NOT_RAPU | 0x02

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

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

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

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

TIME_COMM_REPLY_USEC   equ .250
TIME_COMM_B1_LO_USEC   equ .20
TIME_COMM_B1_HI_USEC   equ .230
TIME_COMM_B0_LO_USEC   equ .170
TIME_COMM_B0_HI_USEC   equ .080

TIME_COMM_0_MAX_USEC   equ .175
TIME_COMM_1_MAX_USEC   equ .70
TIME_COMM_BREAK_USEC   equ .200

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

;------------------------------------------------------------
;--- constants: i/o configuration
;------------------------------------------------------------
#define     TRIS_A_COMM    b'11111011'
#define     TRIS_B_DEFAULT b'00111111'
#define     TRIS_C_BIOUT   b'11001101'
#defin      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<<C1OE

;--- CM2: INPUTS: RC3/AN4 SPEED: NORM, OUTPUT: INT
#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<<VROE

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

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

;--- PWMCON0
#define     PWMCON0_AS_DIS     0<<BLANK2 | 1<<PH2EN
#define     PWMCON0_AS_EN      0<<BLANK2 | 1<<PH2EN | 1<<PASEN

;--- PWMCLK
#define     PWMCLK_DEFAULT     .0<<PWMP0 | .19<<PER0

;--- PWMPH2
#define     PWMPH2_DEFAULT     0<<POL | 1<<C2EN | 0<<C1EN | .1<<PH0

;--- ADCON
#define     ADC_ADCON0_DEFAULT 1<<ADFM | 1<<ADON

#define     ADC_ADCON0_0       ADC_ADCON0_DEFAULT | .13<<CHS0 | 0<<VCFG
#define     ADC_ADCON0_1       ADC_ADCON0_DEFAULT | .06<<CHS0 | 0<<VCFG
#define     ADC_ADCON0_2       ADC_ADCON0_DEFAULT | .03<<CHS0 | 0<<VCFG
#define     ADC_ADCON0_3       ADC_ADCON0_DEFAULT | .00<<CHS0 | 0<<VCFG
#define     ADC_ADCON0_4       ADC_ADCON0_DEFAULT | .05<<CHS0 | 0<<VCFG
#define     ADC_CHANNEL_MASK   0x1F

#define     ADC_ADCON1_DEFAULT 0x05<<ADCS0

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

;--- CCP1CON
#define     CCP1CON_DEFAULT    0x0c

;------------------------------------------------------------
;--- constants: interrupts
;------------------------------------------------------------
#define     INTCON_DEFAULT     1<<T0IE
;------------------------------------------------------------
;--- constants:
;------------------------------------------------------------
PWM_DEFAULT            equ   .000
;------------------------------------------------------------
;--- EE MAP
;------------------------------------------------------------
EE_PATTERN             equ .0
```

```
EE_NCELLS              equ .2
EE_CAPACITY          equ .19
EE_PWM_FREQ          equ .21
EE_MODE              equ .22
EE_MODE2             equ .23
EE_OSC_TRIM          equ .24

EE_LED1_CFG          equ .32
EE_LED2_CFG          equ .40

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

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

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

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

EE_MODE3             equ .170
EE_CAL_ADC_2_NM      equ .171
EE_CHGN_V            equ .173
EE_CHGN_V_DCHG       equ .175
EE_CHGN_T_MAX        equ .177
EE_CHGN_V_MAX        equ .178
```

```
EE_CHGN_C_PCHG      equ .180
EE_CHGN_T_PCHG_LO     equ .182
EE_CHGN_T_PCHG_HI     equ .183
EE_CHGN_V_PCHG        equ .184
EE_CHGN_TI_PCHG_MN    equ .186
EE_CHGN_TI_PCHG_MX    equ .187
EE_CHGN_C_FCHG        equ .188
EE_CHGN_TI_FCHG_MN    equ .190
EE_CHGN_TI_FCHG_MX    equ .191
EE_CHGN_TI_DTDT       equ .192
EE_CHGN_T_DTDT        equ .193
EE_CHGN_V_DV          equ .194
EE_CHGN_C_TOPOFF      equ .196
EE_CHGN_TI_TOPOFF     equ .198
EE_CHGN_V_RCHG        equ .199
EE_CHGN_C_TRIK        equ .201
EE_CHGN_TI_TRIK       equ .203
EE_REG_VSAFETY_NM     equ .204
;======================================================================
```

Interrupt Service

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

```
;======================================================================
           org 0x00
vector_reset:                                ;
           goto       start                  ;

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

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

isr_t1:                                      ; isr: TMR1
           btfss      r_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                            ;
;======================================================================
```

```
;debug
#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 BATID pin if requested.

```
;    org 0x100                        ;
start:                                ;

     clrf      r_port_a               ;
     clrf      r_port_b               ;
     clrf      r_port_c               ;

;--- default gpio
     bsf       p_gpio                 ;

     bsf       r_status, RP0          ; *** bank=1

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

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

;--- option
     movlw     option_default         ;
     movwf     r_option_reg           ;

;--- vrcon
     movlw     VRCON_DEFAULT          ;
     movwf     r_vrcon                ;
;--- refcon
     movlw     REFCON_DEFAULT         ;
     movwf     r_refcon               ;
;--- ansel
     movlw     ANSEL0_DEFAULT         ;
     movwf     r_ansel0               ;
     movlw     ANSEL1_DEFAULT         ;
     movwf     r_ansel1               ;
     bcf       r_status, RP0          ; *** bank=0
     bsf       r_status, RP1          ; *** bank=2
;--- opamps
     movlw     OPA1CON_DEFAULT        ;
     movwf     r_opa1con              ;
     movlw     OPA2CON_DEFAULT        ;
     movwf     r_opa2con              ;
;--- comparators
```

```
    movlw     CM1CON0_DEFAULT          ;
    movwf     r_cm1con0                ;
    movlw     CM2CON0_DEFAULT          ;
    movwf     r_cm2con0                ;
    movlw     CM2CON1_DEFAULT          ;
    movwf     r_cm2con1                ;

    bcf       r_status, RP1            ; *** bank=0

    bsf       r_t1con, 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                     ;
    bsf       r_status, RP1            ; *** bank=2
    movwf     r_pwmclk                 ;
    movlw     PWMPH2_DEFAULT           ;
    movwf     r_pwmph2                 ;
    bcf       r_status, RP1            ; *** bank=0
    movlw     PWMCON0_AS_DIS           ;
    btfss     flag0_mode_pwmas_dis     ;
    movlw     PWMCON0_AS_EN            ;
    btfsc     flag0_mode_nm            ; nimh mode? - disable pwmas
    movlw     PWMCON0_AS_DIS           ;
    bsf       r_status, RP1            ; *** bank=2
    movwf     r_pwmcon0                ;
    bcf       r_status, RP1            ; *** bank=0

;--- config pwm
    call      pwm_config               ;

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

;--- init chg controller timer
    movlw     0xff                     ;
    movwf     r_chg_timer_a            ;
```

```
;--- comm?
comm_chk:                                   ;
     call      comm_pin_input_              ;
     movlw     .255                         ;
     movwf     r_temp_1                     ;
comm_chk_loop:                              ;
     btfsc     p_comm                       ;
     goto      comm_chk_on                  ;
     decfsz    r_temp_1, f                  ;
     goto      comm_chk_loop                ;
comm_chk_off:                               ;
     call      comm_off                     ;
     goto      comm_chk_x                   ;
comm_chk_on:                                ;
     call      comm_on                      ;
comm_chk_x:                                 ;

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

;--- init chg controller
     call      chg_state_0_init             ;

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

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

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

;------------------------------------------------------------
```

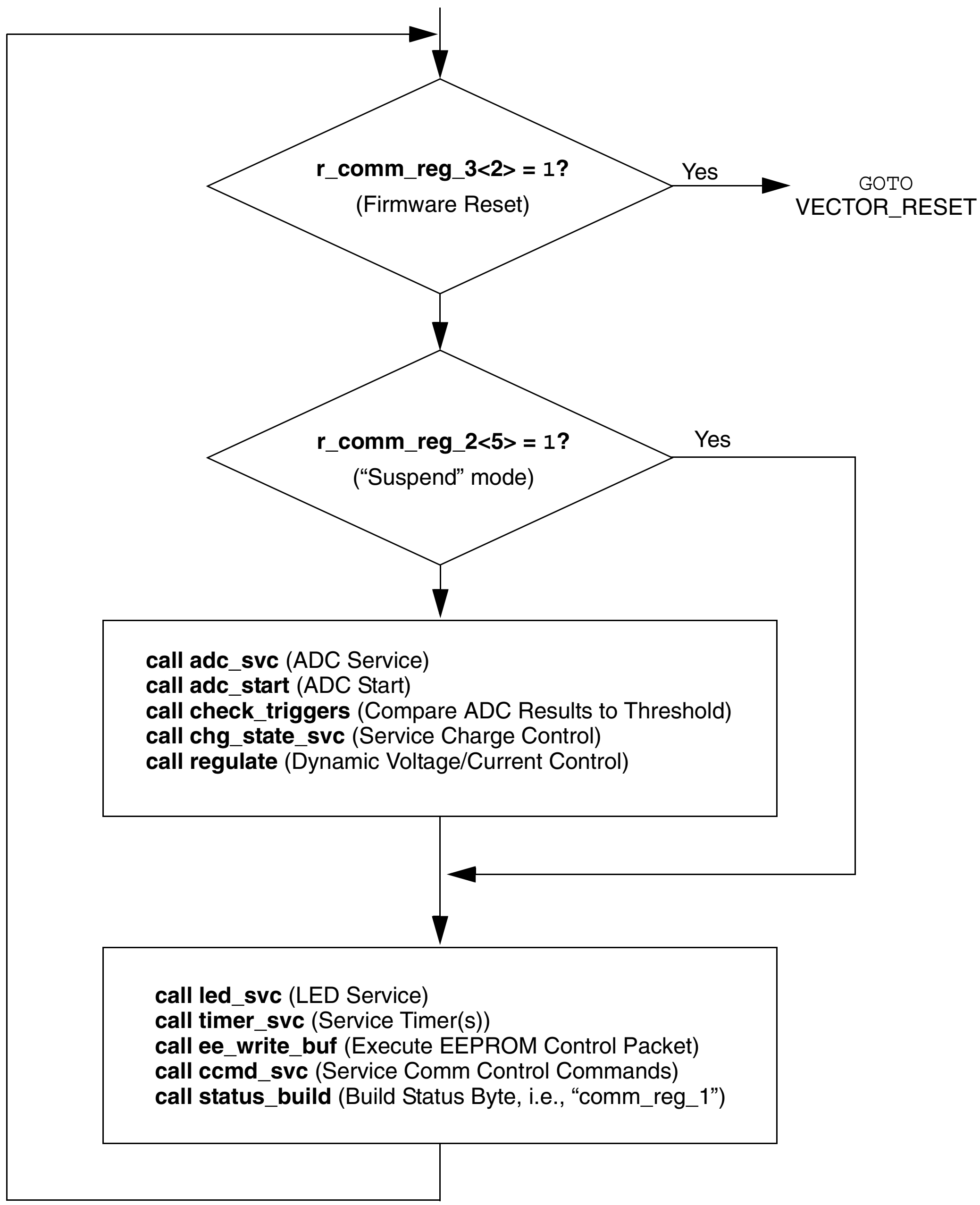
Main Loop

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

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

```
;------------------------------------------------------------
main:                                ;

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

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

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

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

main_x:                              ;
     goto      main                     ;
;----------------------------------------
```


FIGURE 13: MAIN LOOP FLOWCHART



Communication Command Service

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

```
;---------------------------------------
;--- comm command service
;---------------------------------------
ccmd_svc:                               ;

ccmd_ee_rq:                             ;
     btfss     flag0_creg_ee_rq         ;
     goto      ccmd_ee_rq_x             ;
     bcf       flag0_creg_ee_rq         ;
     bsf       flag_ee_rq               ;
ccmd_ee_rq_x:                           ;

ccmd_version:                           ;
     btfss     flag0_creg_version       ;
     goto      ccmd_version_x           ;
     bcf       flag0_creg_version       ;
     movlw     FW_VERSION_LO            ;
     movwf     r_comm_reg_4             ;
     movlw     FW_VERSION_HI            ;
     movwf     r_comm_reg_5             ;
ccmd_version_x:                         ;

ccmd_pwm_set:                           ;
     btfss     flag0_creg_pwm_set       ;
     goto      ccmd_pwm_set_x           ;
     bcf       flag0_creg_pwm_set       ;
     movf      r_comm_reg_4, w          ;
     movwf     r_pwm_L                  ;
     movf      r_comm_reg_5, w          ;
     movwf     r_pwm_H                  ;
     call      pwm_set                  ;
ccmd_pwm_set_x:                         ;

ccmd_reg_on:                            ;
     btfss     flag0_creg_reg_on        ;
     goto      ccmd_reg_on_x            ;
     bcf       flag0_creg_reg_on        ;
     call      reg_on                   ;
ccmd_reg_on_x:                          ;

ccmd_chg_state_force:                   ;
     btfss     flag0_creg_fchgstate     ;
     goto      ccmd_chg_state_force_x   ;
     bcf       flag0_creg_fchgstate     ;
     movf      r_comm_reg_4, w          ;
     movwf     r_chg_state              ;
     call      chg_state_svc_jumptable  ;
ccmd_chg_state_force_x:                 ;

ccmd_sim_rq:                            ;
     btfss     flag0_creg_sim_rq        ;
     goto      ccmd_sim_rq_x            ;
     bcf       flag0_creg_sim_rq        ;
     call      sim_rq_proc              ;
```



```
ccmd_sim_rq_x:                        ;

     return                           ;

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

Status Register Build

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

```
;---------------------------------------
;--- 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_ccp1con                 ;
     return                              ;
;--------------------------------------
```


PWM Set (Subroutine of Regulate)

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

```
;---------------------------------------
;--- 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 descriptions is followed to determine the PWM change value as a function of feedback measurements vs. requirements of voltage and current.

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

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.

```
;------------------------------------------------------------
;--- 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:                            ;
     addwf     r_indf, w                ; add data byte
     incf      r_fsr, f                 ; increment pointer
     decfsz    r_tempc_1, f             ; loop ...
     goto      chksum_loop              ;
     return                             ;

;---------------------------------------
```

Thermistor Temperature Processing (Subroutine of ADC_Service)

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

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

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

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

;--- index into temperature vector   ;
     goto      lut_index_buf        ;

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

;-------------------------------------
;--- lut index
;    call:
;         w         ram location
;         r_temp_3  vector length
;         r_temp_4  search key
;    return:
;         w         vector index
;-------------------------------------
lut_index_buf:                      ;
     movlw     r_buf2               ;
```

```
lut_index:                              ;
     movwf     r_fsr                    ;
     movf      r_temp_3, w              ;
     movwf     r_temp_2                 ;
lut_index_loop:                         ;
     movf      r_temp_4, w              ;
     subwf     r_indf, w                ;
     btfsc     r_status, C              ;
     goto      lut_index_x              ;
     incf      r_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:                               ;
     btfsc     flag_comm_xmit           ;
     goto      comm_isr_xmit            ;

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

comm_isr_rcv_1:                         ;
     btfss     flag_comm_bit            ;
     goto      comm_isr_x               ;
     call      comm_timer_off           ;
     bcf       flag_comm_bit            ;
     comf      r_tmr1l, w               ;
     sublw     TIME_COMM_BREAK_T        ;
     movwf     r_tempi_1                ;
     sublw     TIME_COMM_1_MAX_T        ;
     btfsc     r_status, C              ;
```

```
     goto       comm_isr_rcv_bit         ;
     movlw      TIME_COMM_0_MAX_T        ;
     subwf      r_tempi_1, w             ;
     btfsc      r_status, C              ;
     goto       comm_reset               ; break!

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

     movf       r_comm_data, w           ;
     btfsc      flag_comm_cmnd           ;
     goto       comm_isr_rcv_data        ;
comm_isr_rcv_cmnd:                       ;
     bsf        flag_comm_cmnd           ;
     movwf      r_comm_data_cmnd         ;
     btfsc      r_comm_data, 7           ;
     goto       comm_rcv_byte_prep       ;
comm_isr_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_rcv_data_ia     ;
     movf       r_comm_data_cmnd, w      ;
     andlw      0x07                     ;
     addlw      r_comm_reg               ;
     movwf      r_fsr                    ;
     btfss      r_comm_data_cmnd, 7      ;
     goto       comm_isr_reg_read        ;
comm_isr_reg_write:                      ;
     movlw      0x07                     ;
     xorwf      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_rcv_byte_prep       ;
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            ;
     btfss      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             ;
     bsf       flag_comm_xmit           ;
     bsf       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              ;
     bcf       r_status, RP0            ; *** bank=0
     return                             ;

;---------------------------------------
;--- set comm pin to input
;---------------------------------------
comm_pin_input:                         ;
     btfss     flag_comm_active         ;
     return                             ;
comm_pin_input_:                        ;
     bsf       r_status, RP0            ; *** bank=1
     bsf       r_tris_a, 5              ;
     bcf       r_status, RP0            ; *** bank=0
     return                             ;

;---------------------------------------
;--- set comm pin lo
;---------------------------------------
comm_pin_lo:                            ;
     bcf       p_comm                   ;
     return                             ;

;---------------------------------------
;--- set comm pin hi
;---------------------------------------
comm_pin_hi:                            ;
     bsf       p_comm                   ;
     return                             ;
;---------------------------------------
```


LED Service

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

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

;--- led 1
led_svc_1:                                   ;
     movlw      r_led_config_1               ;
     movwf      r_fsr                        ;
     call       led_svc_modex                ;
     btfss      r_indf, 7                    ;
     bcf        p_led_1                      ;
     btfsc      r_indf, 7                    ;
     bsf        p_led_1                      ;
led_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_   ;
     btfss     r_indf, 3        ;
     goto      led_svc_mode2    ;
     goto      led_svc_mode3    ;
led_svc_modex_:                 ;
     btfsc     r_indf, 3        ;
     goto      led_svc_mode1    ;
     incf      r_fsr, f         ;

;----------------------------------------
;--- led service: mode 0
;----------------------------------------
led_svc_mode0:                  ;
     incf      r_fsr, f         ;
     bcf       r_indf, 7        ;
     goto      led_svc_x        ;

;----------------------------------------
;--- led service: mode 1
;----------------------------------------
led_svc_mode1:                  ;
     incf      r_fsr,f          ; timer==0 ?
     movlw     0x0f             ;
     andwf     r_indf, w        ;
     btfss     r_status, Z      ;
     goto      led_svc_dec_x    ; --- no, decrement & exit ...
     btfss     r_indf, 7        ; led currently on ?
     goto      led_svc_mode1_off ; --- no, jump ...

led_svc_mode1_on:               ;
     movf      r_indf, w        ;
     andlw     0x70             ;
     btfss     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             ;
     btfss     r_status, Z      ;
     movwf     r_indf           ;
     goto      led_svc_x        ;
```



```
led_svc_mode1_off:                      ;
     bsf       r_indf, 7                ;
     movf      r_indf, w                ;
     andlw     0x70                     ;
     btfss     r_status, Z              ;
     goto      led_svc_mode1_off_       ; blink count<>0, decrement & go ...
     decf      r_fsr, f                 ; re-load blink count
     movf      r_indf, w                ;
     andlw     0x70                     ;
     incf      r_fsr, f                 ;
     iorwf     r_indf, f                ;
     goto      led_svc_mode3_on_        ;

led_svc_mode1_off_:                     ;
     movlw     0x10                     ;
     subwf     r_indf, f                ;
     goto      led_svc_mode3_on_        ;

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

;---------------------------------------
;--- led svc mode: 3
;---------------------------------------
led_svc_mode3:                          ;
     incf      r_fsr,f                  ; count==0 ?
     movlw     0x0f                     ;
     andwf     r_indf, w                ;
     btfss     r_status, Z              ;
     goto      led_svc_dec_x            ; --- no, decrement on exit ...
     btfss     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_fsr, f                 ;
     movf      r_indf, w                ;
     goto      led_svc_mode3_           ;
led_svc_mode3_off:                      ;
     bsf       r_indf, 7                ;
     decf      r_fsr, f                 ; point to config register
     swapf     r_indf, w                ;
led_svc_mode3_:                         ;
     andlw     0x07                     ;
     incf      r_fsr, 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_a1_x               ;
     clrf      r_timer_a1                   ;
     bsf       flag_reg_timer               ;
timer_svc_a1_x:                             ;

;--- 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
     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
     incf      r_timer_d, f                 ;
     movlw     TIMER_D_TC                   ;
     subwf     r_timer_d, w                 ;
     btfss     r_status, C                  ;
     goto      timer_svc_x                  ;
     clrf      r_timer_d                    ;

timer_svc_d1:                               ;
     movf      r_timer_d1, f                ;
     btfsc     r_status, Z                  ;
     goto      timer_svc_d1_x               ;
     decf      r_timer_d1, f                ;
     btfsc     r_status, Z                  ;
     bsf       flag_chg_ti1_done            ;
timer_svc_d1_x:                             ;
```

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

timer_svc_x:                                ;

     return                                 ;
;------------------------------------------------------------
```

EEPROM Read and Write

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

```
;-----------------------------------------------------------
;--- move_ee_ram() - move data from eeprom to ram
;
;    call:
;         r_fsr          ram address
;         r_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
ee_read_ia:                             ;
     call      ee_read                  ;
     incf      r_ee_addr, f             ;
     return                             ;
ee_read_waddr:                          ;
     movwf     r_ee_addr                ; save address

ee_read:                                ;
     bcf       r_status, RP1            ; *** bank=1
     bsf       r_status, RP0            ;
     movf      r_ee_addr, w             ; load address
     movwf     r_eeadr                  ;
     bsf       r_eecon1, RD             ;
     movf      r_eedata, w              ; read data into WREG
     bcf       r_status, RP0            ; *** bank=0
     movwf     r_ee_data                ;
     return

;-----------------------------------------------------------
;--- eeprom write
;-----------------------------------------------------------
ee_write_wdata_ia:                      ;
     movwf     r_ee_data                ; save data
     call      ee_write                 ;
     incf      r_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 descriptions.

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

;----------------------------------------
;--- fetch "raw" result
;----------------------------------------
     bsf       r_status, RP0            ; *** bank=1
     movf      r_adresl, w              ;
     bcf       r_status, RP0            ; *** bank=0
     movwf     r_adc_raw_L              ;
     movf      r_adresh, w              ;
     movwf     r_adc_raw_H              ; load raw data

     movlw     r_adc_raw_L              ;
     call      math_load_B              ; accB = raw
     movlw     r_adc_accum              ; accA = running accum
     call      math_add_16_load_A       ; accB = raw + accum
     movlw     r_adc_accum              ;
     call      math_move_B              ; move result to register: accum
     incf      r_adc_accum_count, f     ;
     btfss     r_adc_accum_count, 4     ; accum complete ?
     goto      adc_svc_x                ; --- no,  exit ...

     movlw     .4                       ;
     call      math_shift_BC            ; accB = accum / count = avg
     movlw     r_adc_avg                ;
     call      math_move_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     EE_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             ;
     movwf     r_adc_0_L               ;
     movf      r_accB_H, w             ;
     movwf     r_adc_0_H               ;
adc_svc_0_x_:                          ;
     goto      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     EE_CAL_ADC_1            ;
     call      load_A_ee               ; accA = cal factor
     call      math_mul_16_prep_       ; accBC = result
     movlw     .2                      ;
     call      math_shift_BC           ; accB_L,accC_H = result/1024

     movlw     r_accC_H                ;
     call      math_load_A             ; accA = result

;--- remove offset (option)
adc_svc_1_cofs:                        ;
     clrf      r_accB_L                ;
     clrf      r_accB_H                ;
     btfsc     flag0_mode_cofs_dis     ;
     goto      adc_svc_1_cofs_x_       ;
     movlw     r_adc_1_ofs             ;
     call      math_load_B             ; accB = -offset
     clrf      r_accB_H                ;
     call      math_neg_B              ;
adc_svc_1_cofs_x_:                     ;
     call      math_add_16             ; accA + accB = result - offset
     btfss     r_accB_H, 7             ; negative ?
     goto      adc_svc_1_cofs_x        ; --- no,  skip ...
     clrf      r_accB_L                ; --- yes, make zero
     clrf      r_accB_H                ;
```

```
adc_svc_1_cofs_x:                        ;

     movlw     r_adc_1_L                 ;
     call      math_move_B               ;
     goto      adc_svc_x                 ;
adc_svc_1_x:                             ;

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

;debug
;    movlw     r_adc_avg_shadow          ;
;    call      math_move_D               ; save result

     movlw     EE_CAL_ADC_2              ;
     btfsc     flag0_mode_nm             ;
     movlw     EE_CAL_ADC_2_NM           ;
     call      load_A_ee                 ;
     call      math_mul_16_prep_         ;
     movlw     .2                        ;
     call      math_shift_BC             ;
     movf      r_accC_H, w               ;
     movwf     r_adc_2_L                 ;
     movf      r_accB_L, w               ;
     movwf     r_adc_2_H                 ;
     goto      adc_svc_x                 ;
adc_svc_2_x:                             ;

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

adc_svc_3_sim:                           ;
     btfsc     flag_adc_3_sim            ;
     goto      adc_svc_3_x               ;
adc_svc_3_sim_x:                         ;

adc_svc_3_k:                             ;
     btfss     flag0_mode_temp_k         ; 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_         ; accB,C = TSCALE = RAW * CF
     movlw     .5                        ;
```



```
     call      math_shift_BC          ;
     movlw     r_accC_H               ;
     call      math_load_D            ; accD = TSCALE' = TSCALE/8192
     movlw     .2                     ;
     call      math_shift_BC          ;
     movf      r_accC_H, w            ; W = TSCALE' / 4
     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               ;
     bcf       flag_math_temp         ;
     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             ;
     movwf     r_accA_L               ;
     call      ee_read_ia             ;
     movwf     r_accA_H               ;
     call      math_add_16            ; accB = accB + yint = temperature

adc_svc_3_B:                          ;

adc_svc_3_under:                      ;
     btfss     r_accB_H, 7            ; check limit: tcode < 0
     goto      adc_svc_3_under_x      ;
     clrf      r_accB_H               ;
     clrf      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            ;
     btfsc     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:                     ;
```

```
adc_svc_3_x_:                         ;
     movlw     r_adc_3_L              ;
     call      math_move_B            ; save results
     goto      adc_svc_x              ;
adc_svc_3_x:                          ;

;------------------------------
;--- adc_4: battid
;------------------------------
adc_svc_4:                            ;
     btfss     r_adc_control, 4       ;
     goto      adc_svc_4_x            ;
     bcf       r_adc_control, 4       ;

;debug
;    movlw     r_adc_avg_shadow       ;
;    call      math_move_D            ; save result

     movlw     .2                     ;
     call      math_shift_BC          ; accB = result/4 (i.e. 0->255)
     movlw     r_adc_4_L              ;
     call      math_move_B            ;

     bcf       flag_battpres1         ;
     movlw     EE_BATTID_MIN          ;
     call      ee_read_waddr_ia       ; w = lower limit
     subwf     r_adc_4_L, w           ;
     btfss     r_status, C            ;
     goto      adc_svc_4_x_           ;
     call      ee_read                ; w = upper limit
     subwf     r_adc_4_L, w           ;
     btfss     r_status, 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_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.

```
;------------------------------------------------------------
;--- 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          ;
     btfsc     r_status, Z               ;
     goto      adc_start_done            ;

     bsf       r_adc_control, 6          ;

     btfsc     r_adc_control, 4          ;
     movlw     ADC_ADCON0_4              ;
     btfsc     r_adc_control, 3          ;
     movlw     ADC_ADCON0_3              ;
     btfsc     r_adc_control, 2          ;
     movlw     ADC_ADCON0_2              ;
     btfsc     r_adc_control, 1          ;
     movlw     ADC_ADCON0_1              ;
     btfsc     r_adc_control, 0          ;
     movlw     ADC_ADCON0_0              ;

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

;--- done
adc_start_done:                          ;

;--- set "data ready" flag(s)
adc_start_done_0:                        ;
     btfss     flag_adcset_0_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 descriptions. Charge Suspend, Fast Charge, Trickle Charge, etc., are entered when appropriate. See Lithium and Nickel charge state descriptions in the functional description sections.

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

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

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

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

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

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

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

     call      check_chg_timer_a        ;
     btfss     r_status, Z              ;
     goto      chg_state_svc_x          ;
     goto      chg_state_svc_jumptable  ;
```

```
;---------------------------------------------------------
;--- charge state: 0 - reset
;---------------------------------------------------------
chg_state_0:                            ;
     call      reg_off                  ;
     btfsc     flag_battpres            ;
     goto      chg_state_1_init         ;

chg_state_0_x:                          ;
     goto      chg_state_x              ;


;---------------------------------------------------------
;--- charge state: 1 - charge qualifiation
;---------------------------------------------------------
chg_state_1:                            ;

     btfss     flag0_mode_nm            ;
     goto      chg_state_1_lion         ;

chg_state_1_nimh:                       ;

;--- discharge enabled ?
chg_state_1_nimh_dchg:                  ;
     btfss     flag0_mode_nm_extdchg_en ;
     goto      chg_state_1_nimh_dchg_x  ;
     btfsc     flag_v_le_dchg_nm        ;
     goto      chg_state_1_nimh_dchg_x_ ;
     bcf       p_gpio                   ;
     goto      chg_state_x              ;
chg_state_1_nimh_dchg_x_:               ;
     bsf       p_gpio                   ;
chg_state_1_nimh_dchg_x:                ;

;--- max exceeded?
     btfsc     flag0_mode_nm_slochg     ;
     goto      chg_state_7_init         ;
     btfss     flag_t_le_tmaxchg_nm     ;
     goto      chg_state_57_init        ;
     btfss     flag_v_le_vmaxchg_nm     ;
     goto      chg_state_57_init        ;

;--- pchg?
     btfsc     flag0_mode_pchg_always   ;
     goto      chg_state_2_init         ;
     btfsc     flag_v_le_vpchg_nm       ;
     goto      chg_state_2_init         ;
     btfsc     flag_t_le_tpchg_lo_nm    ;
     goto      chg_state_2_init         ;
     btfss     flag_t_le_tpchg_hi_nm    ;
     goto      chg_state_2_init         ;
     goto      chg_state_1_x            ;

chg_state_1_lion:                       ;
;--- suspend?
     btfss     flag_v_le_vmax           ;
     goto      chg_state_6_init         ;
     btfsc     flag_t_le_tmin           ;
     goto      chg_state_6_init         ;
     btfss     flag_t_le_tmaxchgi       ;
     goto      chg_state_6_init         ;
```



```
;--- pchg?
     btfsc     flag0_mode_pchg_always   ;
     goto      chg_state_2_init         ;
     btfsc     flag_v_le_vpchg          ;
     goto      chg_state_2_init         ;
     btfsc     flag_t_le_tpchg          ;
     goto      chg_state_2_init         ;

chg_state_1_x:                          ;
     goto      chg_state_3_init         ;

;------------------------------------------------------------
;--- charge state: 2 - precharge
;------------------------------------------------------------
chg_state_2:                            ;
     btfss     flag0_mode_nm            ;
     goto      chg_state_2_lion         ;

chg_state_2_nimh:                       ;
     btfss     flag_reg_on              ; regulation module shutdown ?
     goto      chg_state_0_init         ; --- yes, exit to state 0 ...

;--- max exceeded?
     btfss     flag_t_le_tmaxchg_nm     ;
     goto      chg_state_57_init        ;
     btfss     flag_v_le_vmaxchg_nm     ;
     goto      chg_state_57_init        ;
     btfsc     flag_chg_ti1_done        ;
     goto      chg_state_57_init        ;
;--- min/holdoff timer expired ?
     btfss     flag_chg_ti2_done        ;
     goto      chg_state_x              ;
;--- check pchg criteria
     btfsc     flag_v_le_vpchg_nm       ;
     goto      chg_state_x              ;
     btfsc     flag_t_le_tpchg_lo_nm    ;
     goto      chg_state_x              ;
     btfss     flag_t_le_tpchg_hi_nm    ;
     goto      chg_state_x              ;
     goto      chg_state_3_init         ;

chg_state_2_lion:                       ;
     btfss     flag_reg_on              ; regulation module shutdown ?
     goto      chg_state_6_init         ; --- yes, exit to state 6 ...
     btfss     flag_v_le_vmax           ;
     goto      chg_state_6_init         ;
     btfsc     flag_t_le_tmin           ;
     goto      chg_state_6_init         ;
     btfss     flag_t_le_tmaxchg        ;
     goto      chg_state_6_init         ;
     btfsc     flag_chg_ti1_done        ;
     goto      chg_state_6_init         ;

     btfsc     flag_t_le_tpchg          ;
     goto      chg_state_x              ;
     btfsc     flag_v_le_vpchg          ;
     goto      chg_state_x              ;

     goto      chg_state_3_init         ;

;------------------------------------------------------------
;--- charge state: 3
;------------------------------------------------------------
```

```
chg_state_3:                       ;
     btfss     flag0_mode_nm            ;
     goto      chg_state_3_lion         ;

chg_state_3_nimh:                  ;
     btfss     flag_reg_on              ; regulation module shutdown ?
     goto      chg_state_0_init         ; --- yes, exit to state 6 ...

;--- max exceeded?
     btfss     flag_t_le_tmaxchg_nm     ;
     goto      chg_state_57_init        ;
     btfss     flag_v_le_vmaxchg_nm     ;
     goto      chg_state_57_init        ;
     btfsc     flag_chg_ti1_done        ;
     goto      chg_state_57_init        ;

;--- minus delta V?
chg_state_3_nimh_dv:               ;
     btfss     flag0_mode_nm_eoc_dv     ;
     goto      chg_state_3_nimh_dv_x    ;
     btfss     flag_chg_ti2_done        ;
     goto      chg_state_x              ;
     btfss     r_adc_2_save+1, 7        ;
     goto      chg_state_3_nimh_dv_ok   ;
     movf      r_adc_2, w               ;
     movwf     r_adc_2_save             ;
     movf      r_adc_2+1, w             ;
     movwf     r_adc_2_save+1           ;
chg_state_3_nimh_dv_ok:            ;
     call      check_nm_dv              ;
     btfsc     flag_temp_1              ;
     goto      chg_state_7_init         ;
chg_state_3_nimh_dv_x:             ;

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

     goto      chg_state_3_x            ;

chg_state_3_lion:                  ;
     btfss     flag_reg_on              ; regulation module shutdown ?
     goto      chg_state_6_init         ; --- yes, exit to state 6 ...

     btfsc     flag_v_le_vpchg          ;
     goto      chg_state_6_init         ;
     btfss     flag_v_le_vmax           ;
     goto      chg_state_6_init         ;
     btfsc     flag_t_le_tpchg          ;
     goto      chg_state_6_init         ;
     btfss     flag_t_le_tmaxchg        ;
     goto      chg_state_6_init         ;
     btfsc     flag_chg_ti1_done        ;
     goto      chg_state_6_init         ;
     btfsc     flag_vreg                ;
```

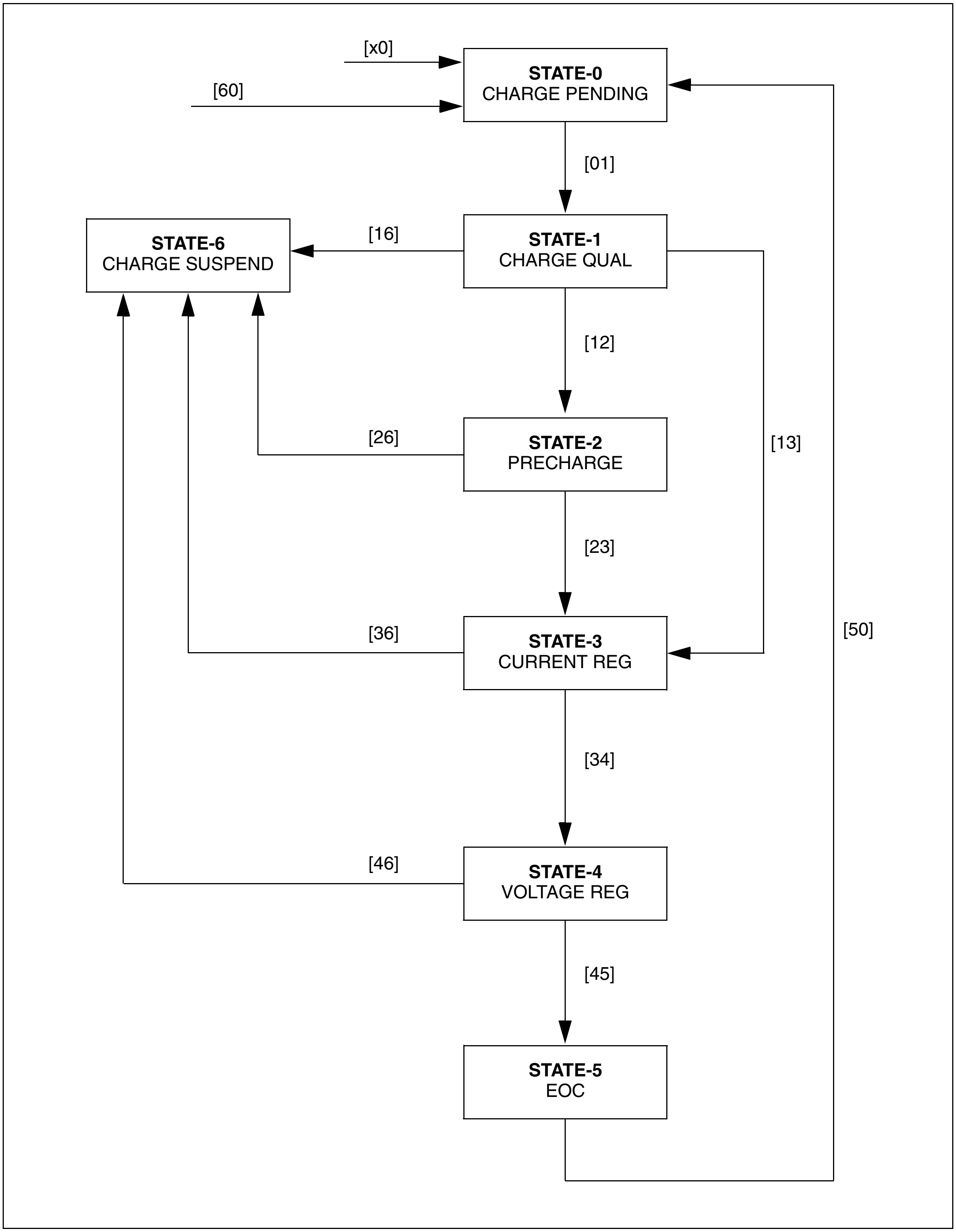
```
     goto      chg_state_4_init          ;

chg_state_3_x:                           ;
     goto      chg_state_x               ;

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

     call      math_neg_B                ;
     movlw     EE_CHGN_T_DTDT            ;
     call      load_A_ee                 ;
     clrf      r_accA_H                  ;
     call      math_add_16               ; accB = deltaT_threshold - deltaT
     btfsc     r_accB_H, 7               ; deltaT > deltaT_threshold ?
     bsf       flag_temp_1               ; --- yes, set detect flag ...

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

check_nm_dtdt_x:                         ;
     return                              ;

;-------------------------------------------------
;--- check_nm_dv(): check minus delta V
;    flag_temp_1=1 deltaV > threshold
;    flag_temp_1=0 deltaV positive or < threshold
;-------------------------------------------------
check_nm_dv:                             ;
     bcf       flag_temp_1               ; clear flag
     movlw     r_adc_2_save              ;
     call      math_load_B               ;
     call      math_neg_B                ; accB = -(adc_2_save) [i.e. -VMAX]
     movlw     r_adc_2                   ; accA = V
     call      math_add_16_load_A        ; accB = dV = (V - VMAX)
     btfss     r_accB_H, 7               ; delta V negative
     goto      check_nm_dv_newmax        ; --- no,  skip ...
```

```
check_nm_dv_negative:                      ;
     movlw     EE_CHGN_V_DV                ;
     call      load_A_ee                   ;
     call      math_add_16                 ; add dV threshold to dV result from above
     btfss     r_accB_H, 7                 ; still negative ?
     goto      check_nm_dv_x               ; --- yes, change not enough - exit
     bsf       flag_temp_1                 ; --- no,  change is enough - set carry
     goto      check_nm_dv_x               ;

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

check_nm_dv_x:                             ;
     return                                ;


;------------------------------------------------------------
;--- charge state: 4
;------------------------------------------------------------
chg_state_4:                               ;
     btfss     flag0_mode_nm               ;
     goto      chg_state_4_lion            ;

chg_state_4_nimh:                          ;
     btfss     flag_reg_on                 ; regulation module shutdown ?
     goto      chg_state_0_init            ; --- yes, exit to state 0 ...

;--- max exceeded?
     btfss     flag_t_le_tmaxchg_nm        ;
     goto      chg_state_57_init           ;
     btfss     flag_v_le_vmaxchg_nm        ;
     goto      chg_state_57_init           ;
     btfsc     flag_chg_ti1_done           ;
     goto      chg_state_57_init           ;
     goto      chg_state_x                 ;

chg_state_4_lion:                          ;
     btfss     flag_reg_on                 ; regulation module shutdown ?
     goto      chg_state_6_init            ; --- yes, exit to state 6 ...

     btfsc     flag_v_le_vpchg             ;
     goto      chg_state_6_init            ;
     btfss     flag_v_le_vmax              ;
     goto      chg_state_6_init            ;
     btfsc     flag_t_le_tpchg             ;
     goto      chg_state_6_init            ;
     btfss     flag_t_le_tmaxchg           ;
     goto      chg_state_6_init            ;
     btfsc     flag_chg_ti1_done           ;
     goto      chg_state_6_init            ;

chg_state_4_a:                             ;
     movlw     EE_CHG_C_MIN                ; check current w/ cmin in ee
     call      check_current               ;
     btfss     r_status, C                 ;
     goto      chg_state_4_a_1             ;
     call      check_chg_timer_b           ;
     btfsc     r_status, Z                 ;
     goto      chg_state_5_init            ;
     goto      chg_state_4_a_x             ;
```

```
chg_state_4_a_1:                    ;
     movlw     EE_CHG_TIME_1        ;
     call      ee_read_waddr        ;
     movwf     r_chg_timer_b        ;
chg_state_4_a_x:                    ;

     goto      chg_state_x          ;

;------------------------------------------------------------
;--- charge state: 5
;------------------------------------------------------------
chg_state_5:                        ;
     call      reg_off              ;

     btfss     flag0_mode_nm        ;
     goto      chg_state_5_lion     ;

chg_state_5_nimh:                   ;
;    btfss     flag_t_le_tmaxchg_nm     ;
;    goto      chg_state_5_x        ;
     btfss     flag_v_le_vmaxchg_nm     ;
     goto      chg_state_5_x        ;
     btfss     flag0_mode_nm_vrchg_en   ;
     goto      chg_state_5_x        ;
     btfsc     flag_v_le_rchg_nm    ;
     goto      chg_state_7_init     ;
     goto      chg_state_5_x        ;

chg_state_5_lion:                   ;
     movlw     EE_CHG_V_RCHG        ;
     call      check_voltage        ;
     btfsc     r_status, C          ;
     goto      chg_state_0_init     ;

chg_state_5_x:                      ;
     goto      chg_state_x          ;


;------------------------------------------------------------
;--- charge state: 6
;------------------------------------------------------------
chg_state_6:                        ;
     call      reg_off              ;

     btfss     flag0_mode_nm        ;
     goto      chg_state_6_lion     ;

chg_state_6_nimh:                   ;
     btfss     flag_t_le_tmaxchg_nm     ;
     goto      chg_state_6_x        ;
     goto      chg_state_6_x_       ;

chg_state_6_lion:                   ;
     btfss     flag_v_le_vmax       ;
     goto      chg_state_6_x        ;
     btfsc     flag_t_le_tmin       ;
     goto      chg_state_6_x        ;
     btfss     flag_t_le_tmaxchgi   ;
     goto      chg_state_6_x        ;

chg_state_6_x_:                     ;
     goto      chg_state_0_init     ;

chg_state_6_x:                      ;
     goto      chg_state_x          ;
```

```
;----------------------------------------------------------
;--- charge state: 7
;----------------------------------------------------------
chg_state_7:                            ;
     btfss     flag_reg_on              ; regulation module shutdown ?
     goto      chg_state_0_init         ; --- yes, exit to state 0 ...

;--- max exceeded?
;    btfss     flag_t_le_tmaxchg_nm     ;
;    goto      chg_state_7_x2           ;
     btfss     flag_v_le_vmaxchg_nm     ;
     goto      chg_state_7_x2           ;
     btfsc     flag_chg_ti1_done        ;
     goto      chg_state_7_x2           ;
chg_state_7_x1:                         ;
     goto      chg_state_x              ;
chg_state_7_x2:                         ;
     btfsc     flag0_mode_nm_exttrik_en ;
     bsf       p_gpio                   ;
     goto      chg_state_5_init         ;

;---------------------------------------
;--- charge state exit(s) ...
;---------------------------------------

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

;--- exit -
chg_state_svc_x:                        ;
     bcf       flag_adcset_0_rdy        ;
     bcf       flag_adcset_2_rdy        ;
     btfss     flag_reg_on              ;
     bsf       flag_adcset_0_rq         ;
     btfsc     flag_reg_on              ;
     bsf       flag_adcset_2_rq         ;

chg_state_svc_x_:                       ;
     return                             ;
;----------------------------------------------------------
```

FIGURE 14: Li Ion CHARGE STATE FLOWCHART



EXAMPLE 1: Li Ion CHARGE STATE FLOWCHART EQUATIONS

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

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

[01] = [x0]*

[16] = ($V > V_{MAX}$) or ($T < T_{MIN}$) or ($T > T_{MAXCHGI}$)
[12] = [16]* and (($V < V_{PCHG}$) or ($T < T_{PCHG}$))
[13] = [16]* and [12]*

[26] = ($V > V_{MAX}$) or ($T < T_{MIN}$) or ($T > T_{MAXCHG}$) or ($V > V_{SAFETY}$) or ($TI > TI_{PCHG}$)
[23] = [26]* and ($T > T_{PCHG}$) and ($V > V_{PCHG}$)

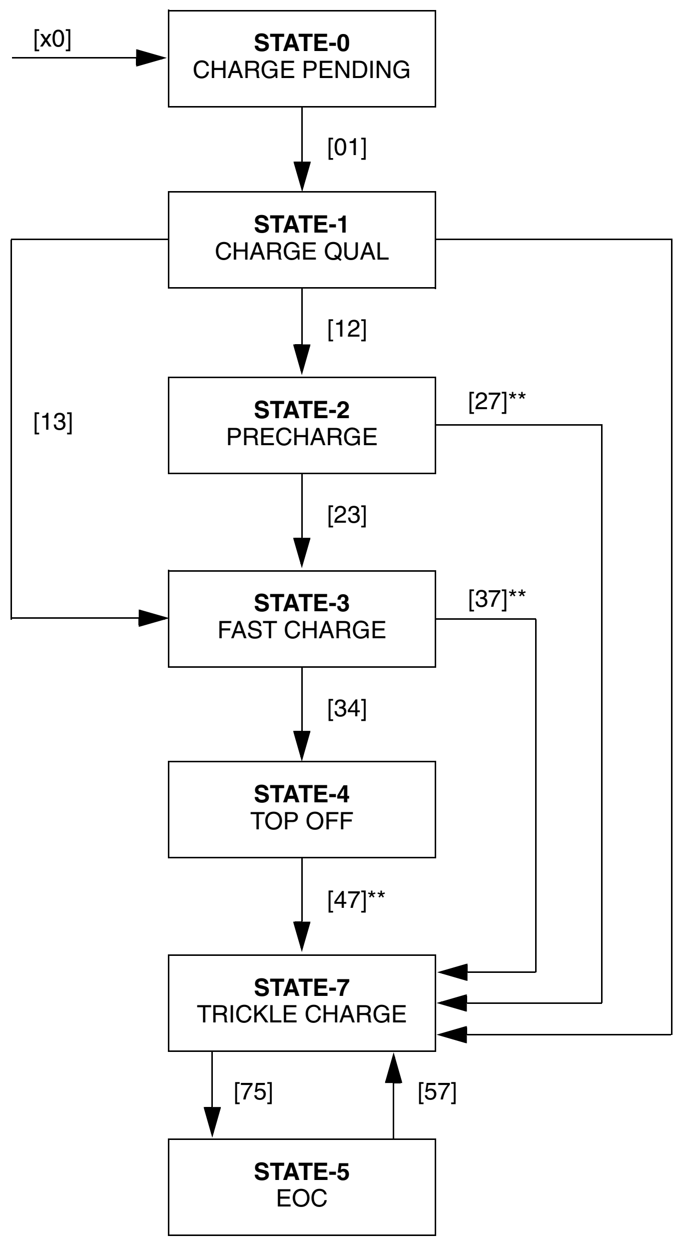
[36] = ($V < V_{PCHG}$) or ($V > V_{MAX}$) or ($T < T_{PCHG}$) or ($T > T_{MAXCHG}$) or ($V > V_{SAFETY}$) or ($TI > TI_{CREG}$)
[34] = [36]* and ($V > V_{REG}$)

[46] = ($V < V_{PCHG}$) or ($V > V_{MAX}$) or ($T < T_{PCHG}$) or ($T > T_{MAXCHG}$) or ($V > V_{SAFETY}$) or ($TI > TI_{VREG}$)
[45] = [46]* and ($C < C_{MIN}$)

[50] = [x0] or ($V < V_{RCHG}$)

[60] = [x0] or (($V < V_{MAX}$) and ($T > T_{MIN}$) and ($T < T_{MAXCHGI}$))

FIGURE 15: NICKEL CHARGE STATE SEQUENCE FLOWCHART



**Mode bit, TRICKLE_BYP, will cause STATE-7 to be bypassed, entering STATE-5 directly.

EXAMPLE 2: NICKEL CHARGE STATE SEQUENCE FLOWCHART EQUATIONS

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

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

STATE-0
[01] = [x0]*

STATE-1
AA = (MODE_EXTDCHG_EN and (V > V_{DCHG}))
[17] = AA* and ((V > V_{MAX}) or (T > T_{MAX}) or MODE_SLOCHG)
[12] = AA* and [17]* and ((T < T_{PCHG_LO}) or (T > T_{PCHG_HI}) or (V < V_{PCHG}))
[13] = AA* and [17]* and [12]*

STATE-2
[27] = ((V > V_{MAX}) or (T > T_{MAX}) or (TI_1 > TI_{PCHG_MAX}))
[23] = [27]* and (TI_2 > TI_{PCHG_MIN}) and (T > T_{PCHG_LO}) and (T < T_{PCHG_HI}) and (V > V_{PCHG})

STATE-3
AA = (V > V_{MAX}) or (T > T_{MAX}) or (TI_1 > TI_{FCHG_MAX})
[37] = AA or ((TI_2 > TI_{FCHG_MIN}) and ((DTDT_DETECT and MODE_TOPOFF_EN*) or (MDV_DETECT)))
[34] = AA* and (TI_2 > TI_{FCHG_MIN}) and (DTDT_DETECT and MODE_TOPOFF_EN)

STATE-4
[47] = (V > V_{MAX}) or (T > T_{MAX}) or (TI_1 > TI_{TOPOFF_MAX})

STATE-5
[57] = MODE_VRCHG_EN and (V < V_{MAX}) and (T < T_{MAX}) and (V < V_{RCHG})

STATE-7
[75] = (V > V_{MAX}) or (TI_1 > $TI_{TRICKLE\_MAX}$)

FIGURE 16: NICKEL END-OF-CHARGE dT/dt METHOD FLOWCHART

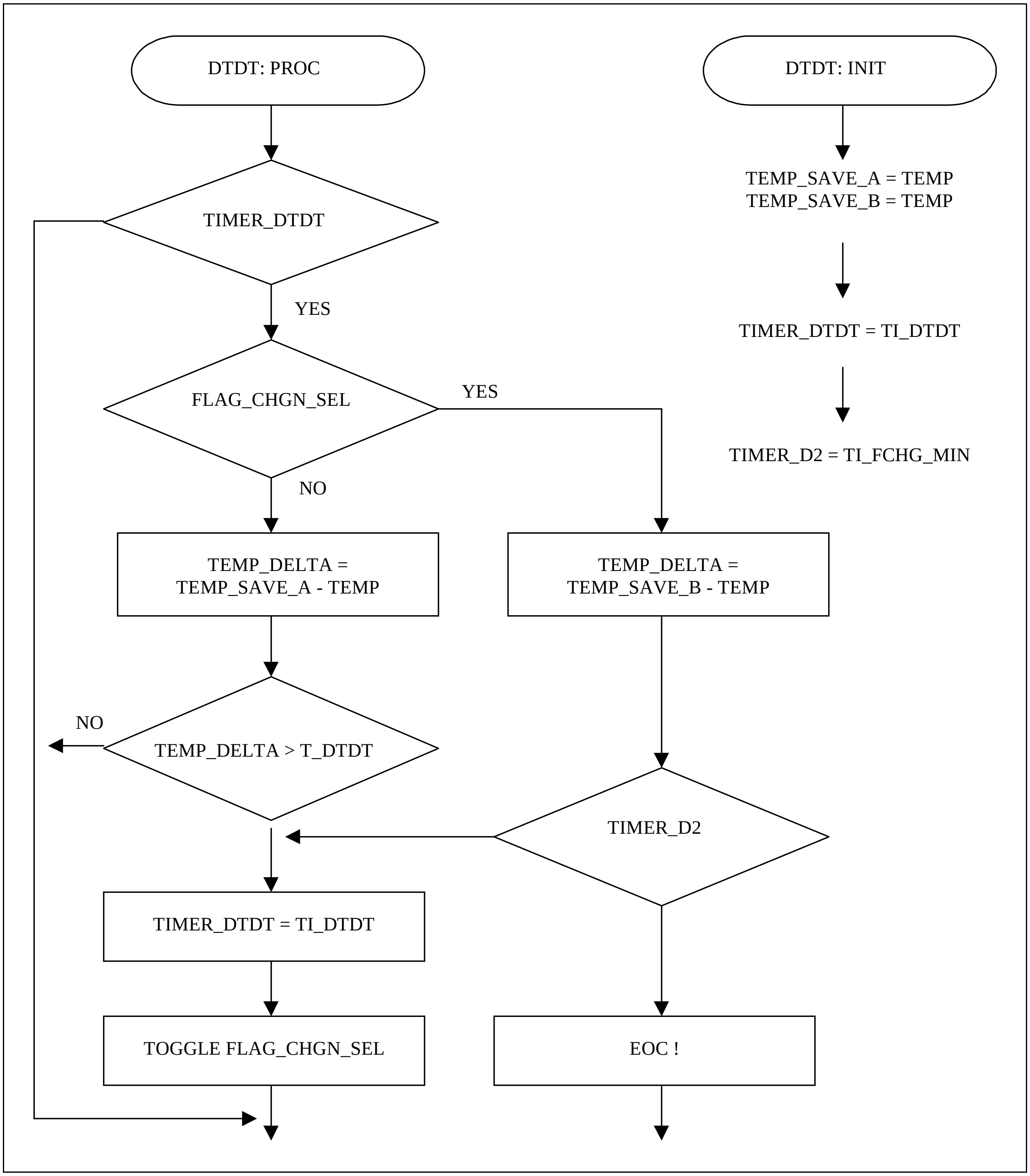
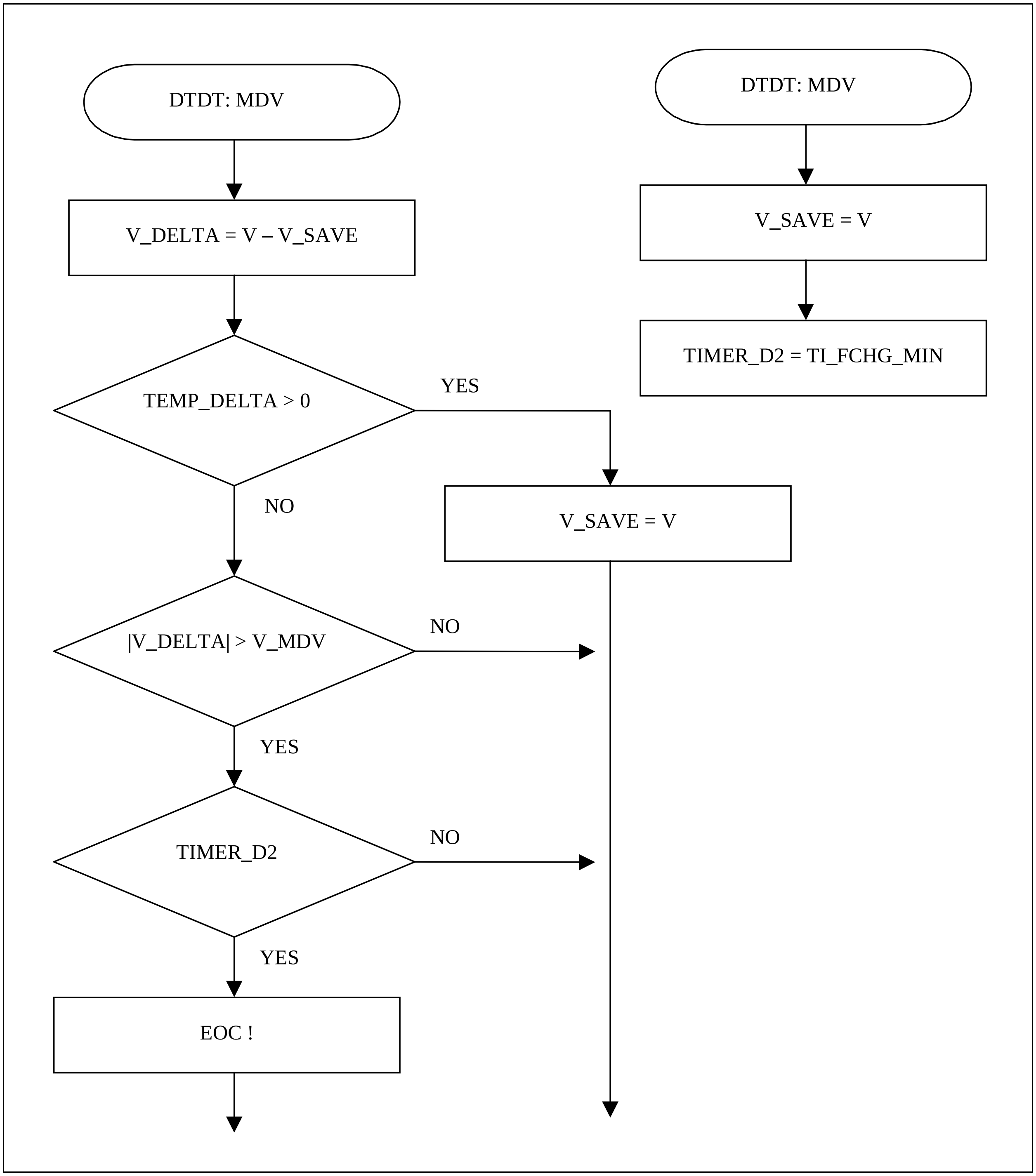


FIGURE 17: NICKEL END-OF-CHARGE MINUS dV METHOD FLOWCHART



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                  ;
     clrf      r_adc_1_ofs              ;
     movlw     .0                       ;
     goto      chg_state_init_x_gpio_hi ;

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

;-----------------------------
;--- chg_state_2_init()
;-----------------------------
chg_state_2_init:                       ;
     btfss     flag0_mode_nm            ;
     goto      chg_state_2_init_lion    ;

chg_state_2_init_nimh:                  ;
     movlw     EE_CHGN_C_PCHG           ;
     call      reg_load_nimh            ;
     movlw     EE_CHGN_TI_PCHG_MX       ;
     call      load_timer_d1_ee         ;
     movlw     EE_CHGN_TI_PCHG_MN       ;
     call      load_timer_d2_ee         ;
     goto      chg_state_2_init_x       ;

chg_state_2_init_lion:                  ;
     movlw     EE_CHG_C_PCHG            ;
     call      reg_load_lion_           ;
     movlw     EE_CHG_TI_PCHG           ;
     call      load_timer_d1_ee         ;

chg_state_2_init_x:                     ;
     movlw     .2                       ;
     goto      chg_state_init_x_gpio_lo ;

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

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

chg_state_3_init_lion:                 ;
     call      reg_load_lion           ;
     movlw     EE_CHG_TI_CREG          ;
     call      load_timer_d1_ee        ;
chg_state_3_init_:                     ;
     movlw     .3                      ;
     goto      chg_state_init_x_gpio_lo ;

;----------------------------
;--- chg_state_4_init()
;----------------------------
chg_state_4_init:                      ;
     btfss     flag0_mode_nm           ;
     goto      chg_state_4_init_lion   ;

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

chg_state_4_init_lion:                 ;
     call      reg_load_lion           ;
     movlw     EE_CHG_TI_VREG          ;

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

;----------------------------
;--- chg_state_57_init()
;    i.e. 5 or 7
;----------------------------
chg_state_57_init:
     btfss     flag0_mode_nm_trik_byp  ;
     goto      chg_state_7_init        ;

;----------------------------
;--- chg_state_5_init()
;----------------------------
chg_state_5_init:                      ;
     call      reg_off                 ;
     movlw     .5                      ;
     goto      chg_state_init_x_gpio_hi ;
```

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

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

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

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

;----------------------------------------
;--- check timer a
;--- check timer b
;--- check timer c
;--- check timer d
;    status[Z]=1 if timer=0
;----------------------------------------
check_chg_timer_a:                      ;
     movlw     r_chg_timer_a            ;
     goto      check_chg_timer_         ;
check_chg_timer_b:                      ;
     movlw     r_chg_timer_b            ;
     goto      check_chg_timer_         ;
check_chg_timer_c:                      ;
     movlw     r_chg_timer_c            ;
     goto      check_chg_timer_         ;
check_chg_timer_d:                      ;
     movlw     r_chg_timer_d            ;
     goto      check_chg_timer_         ;

check_chg_timer_:                       ;
     movwf     r_fsr                    ;
     movf      r_indf, f                ;
     btfss     r_status, Z              ;
     decfsz    r_indf, f                ;
     nop                                ;
     return                             ;

;------------------------------------------------------------
;--- check_current() - compare current to threshold
;
;    call:
;         w    ee address of theshold
;    uses:
;         r_accB_L
;         r_accB_H
;         r_fsr
;    return:
;         r_status[C]=1  current <= threshold
;         r_status[C]=0  current > threshold
;------------------------------------------------------------
check_current:                          ;
     call      load_B_ee                ;
check_current_:
     movlw     r_adc_1_H                ;
     goto      math_cmp_16              ;

;------------------------------------------------------------
```

```
;--- check_voltage() - compare voltage to threshold
;
;    call:
;         w    ee address of theshold
;    uses:
;         r_accB_L
;         r_accB_H
;         r_fsr
;    return:
;         r_status[C]=1  voltage <= threshold
;         r_status[C]=0  voltage > threshold
;------------------------------------------------------------
check_voltage:                            ;
     call      load_B_ee                  ;
     movlw     r_adc_2_H                  ;
     goto      math_cmp_16                ;

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

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

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

;------------------------------------------------------------
;--- check_triggers() - compare a/d values to trigger levels
;------------------------------------------------------------
```

```
check_triggers:                          ;
     clrf      r_flags_2                 ; pre-clear flags

     btfss     flag0_mode_nm             ; nickel algorithm ?
     goto      check_triggers_lion       ; --- no,  skip ...

check_triggers_nimh:                     ;
     movlw     EE_CHGN_V_PCHG            ;
     call      check_voltage             ;
     rlf       r_flags_2, f              ;
     movlw     EE_CHGN_T_PCHG_LO         ;
     call      check_temperature         ;
     rlf       r_flags_2, f              ;
     movlw     EE_CHGN_T_PCHG_HI         ;
     call      check_temperature         ;
     rlf       r_flags_2, f              ;
     movlw     EE_CHGN_T_MAX             ;
     call      check_temperature         ;
     rlf       r_flags_2, f              ;
     movlw     EE_CHGN_V_MAX             ;
     call      check_voltage             ;
     rlf       r_flags_2, f              ;
     movlw     EE_CHGN_V_RCHG            ;
     call      check_voltage             ;
     rlf       r_flags_2, f              ;
     movlw     EE_CHGN_V_DCHG            ;
     call      check_voltage             ;
     rlf       r_flags_2, f              ;
     rlf       r_flags_2, f              ;

     goto      check_triggers_x          ;

check_triggers_lion:                     ;
     movlw     EE_CHG_V_MIN              ;
     call      check_voltage             ;
     rlf       r_flags_2, f              ;
     movlw     EE_CHG_V_MAX              ;
     call      check_voltage             ;
     rlf       r_flags_2, f              ;
     movlw     EE_CHG_V_REG              ;
     call      check_voltage             ;
     rlf       r_flags_2, f              ;
     movlw     EE_CHG_V_PCHG             ;
     call      check_voltage             ;
     rlf       r_flags_2, f              ;

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

check_triggers_x:                        ;
     return                              ;
;------------------------------------------------------------
```

Check Triggers

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

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

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

;----------------------------------------
;--- check timer a
;--- check timer b
;--- check timer c
;--- check timer d
;    status[Z]=1 if timer=0
;----------------------------------------
check_chg_timer_a:                        ;
     movlw     r_chg_timer_a              ;
     goto      check_chg_timer_           ;
check_chg_timer_b:                        ;
     movlw     r_chg_timer_b              ;
     goto      check_chg_timer_           ;
check_chg_timer_c:                        ;
     movlw     r_chg_timer_c              ;
     goto      check_chg_timer_           ;
check_chg_timer_d:                        ;
     movlw     r_chg_timer_d              ;
     goto      check_chg_timer_           ;

check_chg_timer_:                         ;
     movwf     r_fsr                      ;
     movf      r_indf, f                  ;
     btfss     r_status, Z                ;
     decfsz    r_indf, f                  ;
     nop                                  ;
     return                               ;
```

```
;----------------------------------------------------------
;--- check_current() - compare current to threshold
;
;    call:
;         w    ee address of theshold
;    uses:
;         r_accB_L
;         r_accB_H
;         r_fsr
;    return:
;         r_status[C]=1  current <= threshold
;         r_status[C]=0  current > threshold
;----------------------------------------------------------
check_current:                        ;
     call      load_B_ee              ;
check_current_:
     movlw     r_adc_1_H              ;
     goto      math_cmp_16            ;

;----------------------------------------------------------
;--- check_voltage() - compare voltage to threshold
;
;    call:
;         w    ee address of theshold
;    uses:
;         r_accB_L
;         r_accB_H
;         r_fsr
;    return:
;         r_status[C]=1  voltage <= threshold
;         r_status[C]=0  voltage > threshold
;----------------------------------------------------------
check_voltage:                        ;
     call      load_B_ee              ;
     movlw     r_adc_2_H              ;
     goto      math_cmp_16            ;

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

;--------------------------------------------------
;--- load_A_ee() - load accA from EE
;--------------------------------------------------
```

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

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

;------------------------------------------------------------
;--- check_triggers() - compare a/d values to trigger levels
;------------------------------------------------------------
check_triggers:                         ;
     clrf      r_flags_2                ; pre-clear flags

     btfss     flag0_mode_nm            ; nickel algorithm ?
     goto      check_triggers_lion      ; --- no,  skip ...

check_triggers_nimh:                    ;
     movlw     EE_CHGN_V_PCHG           ;
     call      check_voltage            ;
     rlf       r_flags_2, f             ;
     movlw     EE_CHGN_T_PCHG_LO        ;
     call      check_temperature        ;
     rlf       r_flags_2, f             ;
     movlw     EE_CHGN_T_PCHG_HI        ;
     call      check_temperature        ;
     rlf       r_flags_2, f             ;
     movlw     EE_CHGN_T_MAX            ;
     call      check_temperature        ;
     rlf       r_flags_2, f             ;
     movlw     EE_CHGN_V_MAX            ;
     call      check_voltage            ;
     rlf       r_flags_2, f             ;
     movlw     EE_CHGN_V_RCHG           ;
     call      check_voltage            ;
     rlf       r_flags_2, f             ;
     movlw     EE_CHGN_V_DCHG           ;
     call      check_voltage            ;
     rlf       r_flags_2, f             ;
     rlf       r_flags_2, f             ;

     goto      check_triggers_x         ;

check_triggers_lion:                    ;
     movlw     EE_CHG_V_MIN             ;
     call      check_voltage            ;
     rlf       r_flags_2, f             ;
     movlw     EE_CHG_V_MAX             ;
     call      check_voltage            ;
     rlf       r_flags_2, f             ;
     movlw     EE_CHG_V_REG             ;
     call      check_voltage            ;
     rlf       r_flags_2, f             ;
     movlw     EE_CHG_V_PCHG            ;
     call      check_voltage            ;
     rlf       r_flags_2, f             ;
```

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

check_triggers_x:                        ;
     return                              ;
;------------------------------------------------------------
```

Regulate

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

```
;------------------------------------------------------------
;--- regulate()
;------------------------------------------------------------
#define   REG_V_H2   .25               ; voltage - upper limit 2
#define   REG_V_H1   .6                ; voltage - upper limit 1
#define   REG_V_L1   .6                ; voltage - lower limit 1
#define   REG_V_L2   .50               ; voltage - upper limit 2

#define   REG_C_NULL     .5            ; current - null limit

#define   REG_ADJ_P1     .12           ; pwm adjust
#define   REG_ADJ_P2     .10           ; pwm adjust
#define   REG_ADJ_P3     .5            ; pwm adjust
#define   REG_ADJ_P4     .1            ; pwm adjust

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

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

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

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

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

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

;--- V > VREG
regulate_v_hi:                         ;
     bsf       flag_vreg               ;
     bsf       flag_neg                ;
```

```
     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)
;--- V <= VREG
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_             ;
     btfss     flag_vreg_2               ;
     bsf       flag_vreg                 ;
regulate_v_x_:                           ;

;----------------------------------------
;--- current
;----------------------------------------
regulate_c:                              ;
     movlw     r_adc_1_L                 ;
     call      math_load_B               ;
     call      math_neg_B                ;
     movlw     r_reg_c                   ;
     call      math_add_16_load_A        ; accB = delta = reg_c - current
     bcf       flag_neg                  ;
     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:                         ;
     bsf       flag_neg                  ;
     goto      regulate_adj              ;

regulate_adj_up_1:                       ;
     movlw     EE_REG_P2                 ;
     call      ee_read_waddr             ;
     goto      regulate_adj_up           ;
regulate_adj_up_0:                       ;
     movlw     EE_REG_P4                 ;
     call      ee_read_waddr             ;
regulate_adj_up:                         ;
     bcf       flag_neg                  ;

regulate_adj:                            ;
     call      pwm_adj                   ;
     call      pwm_set                   ;
```

```
regulate_x:                          ;
     bcf       flag_adcset_1_rdy      ;
     bsf       flag_adcset_1_rq       ;

regulate_x_:                         ;
     return                          ;


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

reg_load_v:                          ;
     call      load_B_ee              ;
     movlw     r_reg_v                ;
     call      math_move_B            ;
     return                          ;

reg_load_nimh:                       ;
     call      reg_load_c             ;
     movlw     EE_CHGN_V              ;
     call      reg_load_v             ;
     goto      reg_on                 ;

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

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

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


Math Functions

These are random math routines used by the firmware.

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

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

;-----------------------------------------------------------
;--- math_mul_16
;
;    function:
;         16x16 multiplication
;
;    call:
;         WREG  - count/shift ops
;         op1: accA
;         op2: accD
;
;    result:
;         accB,accC = accA * accD
;-----------------------------------------------------------
math_mul_16:                          ;
     movwf     r_count_1              ;
     clrf      r_accB_H               ; clear result accumulator
     clrf      r_accB_L               ;
     clrf      r_accC_H               ;
     clrf      r_accC_L               ;

math_mul_16_loop:                     ;
     rrf       r_accD_H, F            ; shift operand2 lsb into C
     rrf       r_accD_L, F            ;
     btfss     r_status, C            ; C = 1?
     goto      math_mul_16_shift      ; --- no,  go shift ...

math_mul_16_add:                      ;
     call      math_add_16            ;

math_mul_16_shift:                    ;
     rrf       r_accB_H, F            ; SHIFT result accumulator
     rrf       r_accB_L, F            ;
     rrf       r_accC_H, F            ;
     rrf       r_accC_L, F            ;
     decfsz    r_count_1, F           ;
     goto      math_mul_16_loop       ; loop

     retlw     0                      ;
```

```
;---------------------------------------------------------
;--- div32
;    operands:
;         dividend  - accA,accB
;         divisor   - accC
;
;    result:
;         quotient  - accB
;         remainder - accA
;         overflow  - WREG=0 else WREG=1 ?
;---------------------------------------------------------
math_div_32:                            ;
     movf      r_accC_L, W              ;
     subwf     r_accA_L, W              ;
     movf      r_accC_H, W              ;
     btfss     r_status, C              ;
     incf      r_accC_H, W              ;
     subwf     r_accA_H, W              ;
     btfsc     r_status, C              ;
     retlw     0                        ; overflow or division by zero

     movlw     .16                      ;
     movwf     r_count_1                ;
math_div_32_loop:                       ;
     bcf       r_status, C              ;
     rlf       r_accB_L, F              ; shift dividend (accA,accB << 1)
     rlf       r_accB_H, F              ;
     rlf       r_accA_L, F              ;
     rlf       r_accA_H, F              ;

     btfsc     r_status, C              ; if carry, go subtract
     goto      math_div_32_sub          ;

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

math_div_32_sub:                        ;
     movf      r_accC_L, W              ; subtract divisor from high
     subwf     r_accA_L, F              ;
     movf      r_accC_H, W              ;
     btfss     r_status, C              ;
     incf      r_accC_H, W              ;
     subwf     r_accA_H, F              ;
     bsf       r_accB_L, 0              ;

math_div_32_next:                       ;
     decfsz    r_count_1, F             ;
     goto      math_div_32_loop         ;
     retlw     1                        ; no more overflow possible

;---------------------------------------------------------
;--- math_add_16()
;
;    function:
;         add 16-bit operands
;
;    call:
;         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()
;
;    function:
;         accB = -accB
;------------------------------------------------------------
math_neg_B:                             ;
     comf      r_accB_L, f              ;
     incf      r_accB_L, f              ;
     btfsc     r_status, Z              ;
     decf      r_accB_H, f              ;
     comf      r_accB_H, f              ;
     retlw     0                        ;

;------------------------------------------------------------
;--- math_cmp_16()
;
;    function:
;         compare 2 16-bit values
;         1: accB  2: *WREG
;
;    call:
;         *WREG:    operand A (msb)
;         accB:     operand B
;
;    result:
;         status[C]=1:   accB >= *WREG
;         status[C]=0:   accB <  *WREG
;------------------------------------------------------------
math_cmp_16:                            ;
     movwf     r_fsr                    ;
     movf      r_indf, w                ;
     subwf     r_accB_H, w              ;
     btfss     r_status, Z              ;
     return                             ;
     decf      r_fsr, f                 ;
     movf      r_indf, w                ;
     subwf     r_accB_L, w              ;
     return                             ;

;------------------------------------------------------------
;--- math_load_ utilities
;         *WREG = lsb of source data
;------------------------------------------------------------

;--- math_load_D()
math_load_D:                            ;
     movwf     r_fsr                    ;
```

```
math_load_D_:                           ;
     movf      r_indf, w                ;
     movwf     r_accD_L                 ;
     incf      r_fsr, f                 ;
     movf      r_indf, w                ;
     movwf     r_accD_H                 ;
     goto      math_load_x              ;

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

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

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

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

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

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

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

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

Default EEPROM Values

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

```
;---------------------------------------
;--- EEPROM DEFAULT
;---------------------------------------

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

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

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

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

     org  0x216c
     DE   low .0248, high .0248      ; adc_cal_0 (reference)
     DE   low .2553, high .2553      ; adc_cal_1 (current)
     DE   low .5121, high .5121      ; adc_cal_2 (voltage)
     DE   low .8192, high .8192      ; adc_cal_3 (temperature)
     DE   low .6407, high .6407      ; adc_cal_4 (battid)
     DE   .100                       ; shunt
     DE   .112                       ; temperature default
```

```
     org  0x217C
     DE   .8                            ; TLUT - length
     DE   .38, .48, .61, .79            ; TLUT - temp axis
     DE   .105, .183, .207              ; TLUT - temp axis (cont)
     DE   low -.23362, high -.23362     ; TLUT - slope - 0
     DE   low .1418, high .1418         ; TLUT - yint  - 0
     DE   low -.19864, high -.19864     ; TLUT - slope - 1
     DE   low .1352, high .1352         ; TLUT - yint  - 1
     DE   low -.15709, high -.15709     ; TLUT - slope - 2
     DE   low .1255, high .1255         ; TLUT - yint  - 2
     DE   low -.12572, high -.12572     ; TLUT - slope - 3
     DE   low .1162, high .1162         ; TLUT - yint  - 3
     DE   low -.10206, high -.10206     ; TLUT - slope - 4
     DE   low .1071, high .1071         ; TLUT - yint  - 4
     DE   low -.8631, high -.8631       ; TLUT - slope - 5
     DE   low .990, high .990           ; TLUT - yint  - 5
     DE   low -.10154, high -.10154     ; TLUT - slope - 6
     DE   low .1127, high .1127         ; TLUT - yint  - 6
     DE   low -.12875, high -.12875     ; TLUT - slope - 7
     DE   low .1402, high .1402         ; TLUT - yint  - 7

     org  0x21AA
;debug
;    DE   0x1A                          ; mode_3
     DE   0x3A                          ; mode_3
     DE   low .2544, high .2544         ; adc_cal_2 (voltage) nimh
     DE   low .2000, high .2000         ; chgn_v
     DE   low .1000, high .1000         ; chgn_v_dchg
     DE   .175                          ; chgn_t_max (175=50degC, 200=60degC)
     DE   low .1800, high .1800         ; chgn_v_max
     DE   low .100, high .100           ; chgn_c_pchg
     DE   .75                           ; chgn_t_pchg_lo (75=10degC)
     DE   .162                          ; chgn_t_pchg_hi (162=44.8degC)
     DE   low .0800, high .0800         ; chgn_v_pchg
     DE   .0                            ; chgn_ti_pchg_mn
     DE   .0                            ; chgn_ti_pchg_mx
     DE   low .2000, high .2000         ; chgn_c_fchg
     DE   .0                            ; chgn_ti_fchg_mn
     DE   .0                            ; chgn_ti_fchg_mx
     DE   .120                          ; chgn_ti_dtdt (temp rise time criteria)
     DE   .10                           ; chgn_t_dtdt (temp rise criteria)
     DE   low .10, high .10             ; chgn_v_dv (-delta voltage)
     DE   low .200, high .200           ; chgn_c_topoff
     DE   .11                           ; chgn_ti_topoff (11=45min, i.e. 66% eff, 5% cap, c/10)
     DE   low .1300, high .1300         ; chgn_v_rchg (recharge voltage)
     DE   low .80, high .80             ; chgn_c_trik
     DE   .150                          ; chgn_ti_trik (timeout for trickle charge)
     DE   low .1900, high .1900         ; chgn_v_safety

     end
```

Memory Map

EEPROM

TABLE 11: EEPROM MEMORY MAP

Name	Location		Len	Default		Units	Description
	Dec	Hex		Dec	Hex		
PATTERN	1	0001	2	1	1	coded	Pattern ID (arbitrary)
NCELLS	1	01	1	1	1	cells	Number of cells
MANUF_NAME	3	03	8	—	—	ASCII	Manufacturer’s name “Microchip”
DEV_NAME	11	0B	8	—	—	ASCII	Device name “PIC16HV785”
CAPACITY	19	13	2	800	320	mAh	Capacity
PWM_FREQ	21	15	1	19	13	coded	PWM frequency (e.g., 19 = 400 kHz)
MODE	22	16	1	1	01	coded	Mode – control flags
MODE2	23	17	1	32	20	coded	Mode – control flags
OSC_TRIM	24	18	1	0	0	coded	Oscillator trim – signed value +/-10%
Reserved	25	19	7	x	x	x	
SUB-CONFIG	**32**	**20**	**32**				
LED1_CFG_0	32	20	1	8	08	coded	LED1 configuration – state 0
LED1_CFG_1	33	21	1	24	18	coded	LED1 configuration – state 1
LED1_CFG_2	34	22	1	40	28	coded	LED1 configuration – state 2
LED1_CFG_3	35	23	1	56	38	coded	LED1 configuration – state 3
LED1_CFG_4	36	24	1	72	48	coded	LED1 configuration – state 4
LED1_CFG_5	37	25	1	88	58	coded	LED1 configuration – state 5
LED1_CFG_6	38	26	1	104	68	coded	LED1 configuration – state 6
LED1_CFG_7	39	27	1	120	78	coded	LED1 configuration – state 7
LED2_CFG_0	40	28	1	8	08	coded	LED2 configuration – state 0
LED2_CFG_1	41	29	1	24	18	coded	LED2 configuration – state 1
LED2_CFG_2	42	2A	1	40	28	coded	LED2 configuration – state 2
LED2_CFG_3	43	2B	1	56	38	coded	LED2 configuration – state 3
LED2_CFG_4	44	2C	1	72	48	coded	LED2 configuration – state 4
LED2_CFG_5	45	2D	1	88	58	coded	LED2 configuration – state 5
LED2_CFG_6	46	2E	1	104	68	coded	LED2 configuration – state 6
LED2_CFG_7	47	2F	1	120	78	coded	LED2 configuration – state 7
Reserved	48	30	4	x	x	x	

TABLE 11: EEPROM MEMORY MAP (CONTINUED)

Name	Location		Len	Default		Units	Description
	Dec	Hex		Dec	Hex		
SUB-LED	**52**	**34**	**20**				
REG_P1	52	34	1	12	0C	count	Regulation – control PWM adj
REG_P2	53	35	1	10	0A	count	Regulation – control PWM adj
REG_P3	54	36	1	5	05	count	Regulation – control PWM adj
REG_P4	55	37	1	1	01	count	Regulation – control PWM adj
REG_VHH_VH	56	38	1	19	13	mV	Regulation – voltage zone limit
REG_VH	57	39	1	6	06	mV	Regulation – voltage zone limit
REG_VL	58	3A	1	6	06	mV	Regulation – voltage zone limit
REG_VLL_VL	59	3B	1	44	2C	mV	Regulation – voltage zone limit
REG_CNULL	60	3C	1	5	05	mA	Regulation – current null limit
REG_VSAFETY	61	3D	2	4350	10FE	mV	Regulation – voltage limit – shutdown
Reserved	63	3F	4	x	x	x	
SUB-REG	**67**	**43**	**15**				
CHG_V_MIN	67	43	2	2000	07D0	mV	Min. voltage for charging
CHG_V_MAX	69	45	2	4250	109A	mV	Max. voltage for charging
CHG_V_RCHG	71	47	2	4000	FA0	mV	Voltage trip point – recharge
CHG_V_VREG	73	49	2	4200	1068	mV	Regulation target – voltage
CHG_V_PCHG	75	4B	2	3000	0BB8	mV	Voltage threshold – precharge
CHG_V_MIN_BP	77	4D	2	50	0032	mV	Voltage threshold – battery detect
CHG_T_MIN	79	4F	1	50	32	TCODE	Min. temperature for charging
CHG_T_MAXCHGI	80	50	1	175	AF	TCODE	Max. temperature for charge init
CHG_T_MAXCHG	81	51	1	200	C8	TCODE	Max. temperature for charge
CHG_T_PCHG	82	52	1	75	4B	TCODE	Temperature threshold – precharge
CHG_C_PCHG	83	53	2	100	0064	mA	Regulation target – precharge current
CHG_C_CREG	85	55	2	800	0320	mA	Regulation target – current
CHG_C_MIN	87	57	2	50	0032	mA	Current threshold for EOC detect
CHG_TI_PCHG	89	59	1	0	0	4 min.	Time limit – precharge (0 = off)
CHG_TI_CREG	90	5A	1	0	0	4 min.	Time limit – current regulation (0 = off)
CHG_TI_VREG	91	5B	1	0	0	4 min.	Time limit – voltage regulation (0 = off)
CHG_TIME_0	92	5C	1	20	14	.25 sec.	Timer – state change
CHG_TIME_1	93	5D	1	20	14	.25 sec.	Timer – EOC recheck
CHG_TIME_2	94	5E	1	20	14	.25 sec.	Timer – not used
CHG_TIME_3	95	5F	1	0	0	.25 sec.	Timer – not used
CHG_TIME_4	96	60	1	0	0	.25 sec.	Timer – not used
CHG_TIME_5	97	61	1	0	0	.25 sec.	Timer – not used
BATTID_MIN	98	62	1	0	0	units	BATID minimum/lower threshold
BATTID_MAX	99	63	1	128	128	units	BATID maximum/upper threshold
Reserved	100	64	8	x	x	x	

TABLE 11: EEPROM MEMORY MAP (CONTINUED)

Name	Location		Len	Default		Units	Description
	Dec	Hex		Dec	Hex		
SUB-CHG	**108**	**6C**	**41**				
ADC_CAL_0	108	6C	2	248	00F8	scaler	Calibration (reference)
ADC_CAL_1	110	6E	2	2553	09F9	scaler	Calibration (current)
ADC_CAL_2	112	70	2	5121	1401	scaler	Calibration (voltage)
ADC_CAL_3	114	72	2	8192	2000	scaler	Calibration (temperature)
ADC_CAL_4	116	74	2	6407	1907	scaler	Calibration (BATID)
SHUNT	118	76	1	100	64	mOhm	Shunt resistor value
T_DEFAULT	119	77	1	112	70	TCODE	Default temperature
Reserved	120	78	4	x	x	x	
SUB-ADC	**124**	**7C**	**16**				
T_LUT_N	124	7C	1	8	08	entries	Temperature LUT – length (max. = 8)
T_LUT_T_0	125	7D	1	38	26	TCODE	Temperature LUT – T0 (T_LUT_N – 1 entry)
T_LUT_T_1	126	7E	1	48	30	TCODE	Temperature LUT – T1
T_LUT_T_2	127	7F	1	61	3D	TCODE	Temperature LUT – T2
T_LUT_T_3	128	80	1	79	4F	TCODE	Temperature LUT – T3
T_LUT_T_4	129	81	1	105	69	TCODE	Temperature LUT – T4
T_LUT_T_5	130	82	1	183	B7	TCODE	Temperature LUT – T5
T_LUT_T_6	131	83	1	207	CF	TCODE	Temperature LUT – T6
T_LUT_M_0	132	84	2	-23362	A4BE	scaler	Temperature LUT – T0 slope (T_LUT_N entries)
T_LUT_B_0	134	86	2	1418	058A	scaler	Temperature LUT – T0 Y-intercept
T_LUT_M_1	136	88	2	-19864	B268	scaler	Temperature LUT – T1 slope
T_LUT_B_1	138	8A	2	1352	0548	scaler	Temperature LUT – T1 Y-intercept
T_LUT_M_2	140	8C	2	-15709	C2A3	scaler	Temperature LUT – T2 slope
T_LUT_B_2	142	8E	2	1255	04E7	scaler	Temperature LUT – T2 Y-intercept
T_LUT_M_3	144	90	2	-12572	CEE4	scaler	Temperature LUT – T3 slope
T_LUT_B_3	146	92	2	1162	048A	scaler	Temperature LUT – T3 Y-intercept
T_LUT_M_4	148	94	2	-10206	D822	scaler	Temperature LUT – T4 slope
T_LUT_B_4	150	96	2	1071	042F	scaler	Temperature LUT – T4 Y-intercept
T_LUT_M_5	152	98	2	-8631	DE49	scaler	Temperature LUT – T5 slope
T_LUT_B_5	154	9A	2	990	03DE	scaler	Temperature LUT – T5 Y-intercept
T_LUT_M_6	156	9C	2	-10154	D856	scaler	Temperature LUT – T6 slope
T_LUT_B_6	158	9E	2	1127	0467	scaler	Temperature LUT – T6 Y-intercept
T_LUT_M_7	160	A0	2	-12875	CDB5	scaler	Temperature LUT – T7 slope
T_LUT_B_7	162	A2	2	1402	057A	scaler	Temperature LUT – T7 Y-intercept

TABLE 11: EEPROM MEMORY MAP (CONTINUED)

Name	Location		Len	Default		Units	Description
	Dec	Hex		Dec	Hex		
SUB-T LUT	**164**	**A4**	**40**				
MODE3	170	AA	1	26	1A	bits	Mode bits
ADC_CAL_2_NM	171	AB	2	2544	9F0	scaler	Voltage calibration
CHGN_V_CHG	173	AD	2	2000	07D0	mV	Charge voltage
CHGN_V_DCHG	175	AF	2	1000	03E8	mV	Voltage target for discharge
CHGN_T_MAX	177	B1	1	175	AF	TCODE	Max. temperature
CHGN_V_MAX	178	B2	2	1800	0708	mV	Max. voltage
CHGN_C_PCHG	180	B4	2	100	0064	mA	Precharge current
CHGN_T_PCHG_LO	182	B6	1	75	4B	TCODE	Precharge criteria – lo temp
CHGN_T_PCHG_HI	183	B7	1	162	A2	TCODE	Precharge criteria – hi temp
CHGN_V_PCHG	184	B8	2	800	0320	mV	Precharge criteria – voltage
CHGN_TI_PCHG_MN	186	BA	1	0	00	4 min.	Precharge stage – min. time
CHGN_TI_PCHG_MX	187	BB	1	0	00	4 min.	Precharge stage – max time
CHGN_C_FCHG	188	BC	2	2000	07D0		
CHGN_TI_FCHG_MN	190	BE	1	0	00	4 min.	Fast Charge state – min. time
CHGN_TI_FCHG_MX	191	BF	1	0	00	4 min.	Fast Charge state – max. time
CHGN_TI_DTDT	192	C0	1	120	78	0.5 sec.	dtdt detect – time window
CHGN_T_DTDT	193	C1	1	10	10	0.1°	dtdt detect – temp delta
CHGN_V_DV	194	C2	2	10	10	mV	dv detect – voltage delta
CHGN_C_TOPOFF	196	C4	2	200	C8	mA	Top Off state – current
CHGN_TI_TOPOFF	198	C6	1	11	0B	4 min.	Top Off state – max. time
CHGN_V_RCHG	199	C7	2	1300	0514	mV	Voltage threshold to re-init trickle
CHGN_TI_TRIK	201	C9	1	80	50	4 min.	Trickle Charge state – max. time
CHGN_C_TRIK	202	CA	2	150	0096	mA	Trickle Charge state – current
REG_VSAFETY	204	CC	2	1900	076C	mV	Regulation – shutdown voltage
SUB-NM			**24**				
TOTAL			**188**				

Mode Registers

TABLE 12: MODE REGISTER

Bit	Name	Description
7	pchg_always	1 = Always precharge
6	gpio_cutoff	1 = Enable GPIO cutoff logic
5		
4		
3		
2	bpres_battid	1 = Battery present on BATID
1	bpres_v	1 = Battery present on voltage sense
0	bpres_always	1 = Battery present always

TABLE 13: MODE2 REGISTER

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

TABLE 14: MODE3 REGISTER

Bit	Name	Description
7	nm_extdchg_en	1 = Enable external discharge
6	nm_exttrik_en	1 = Enable external trickle circuit
5	nm_trik_byp	1 = Bypass Trickle Charge state
4	nm_topoff_en	1 = Enable “Top Off” state
3	nm_vrchg_en	1 = Enable Trickle Charge if V drops below recharge voltage
2	nm_slochg	1 = Enable Slow Charge mode (Trickle Charge only)
1	nm_eoc_dtdt	1 = Enable EOC method – dtdt
0	nm_eoc_mdv	1 = Enable EOC method – minus dv

RAM Registers

TABLE 15: RAM

Name	Dec	Hex	Len	Units	Description
r_mode	32	20	1	bits	Operational mode register
r_chg_state	33	21	1	int	Charge Controller “state”
r_adc_0_L	34	22	1	units	ADC result – channel 0 (VREF)
r_adc_0_H	35	23	1		
r_adc_1_L	36	24	1	mA	ADC result – channel 1 (CURRENT)
r_adc_1_H	37	25	1		
r_adc_2_L	38	26	1	mV	ADC result – channel 2 (VOLTAGE)
r_adc_2_H	39	27	1		
r_adc_3_L	40	28	1	TCODE	ADC result – channel 3 (TEMP)
r_adc_3_H	41	29	1		
r_adc_4_L	42	2A	1	units	ADC result – channel 4 (BATID)
r_adc_4_H	43	2B	1		
r_pwm_L	44	2C	1	int	PWM setting
r_pwm_H	45	2D	1		PWM setting
r_reg_c	46	2E	2	mA	Regulation target: current (mA)
r_reg_v	48	30	2	mV	Regulation target: voltage (mV)
r_comm_reg_0	50	32	1	N/A	Indirect Address register
r_comm_reg_1	51	33	1	N/A	Status
r_comm_reg_2	52	34	1	N/A	Configuration flags
r_comm_reg_3	53	35	1	N/A	Command flags
r_comm_reg_4	54	36	1	N/A	Data lo
r_comm_reg_5	55	37	1	N/A	Data hi
r_comm_reg_6	56	38	1	N/A	
r_comm_reg_7	57	39	1	N/A	
r_sim	58	3A	1	N/A	
r_chg_timer_a	59	3B	1	.25 sec.	Hysteresis timer
r_chg_timer_b	60	3C	1	.25 sec.	Hysteresis timer
r_chg_timer_c	61	3D	1	.25 sec.	Hysteresis timer
r_chg_timer_d	62	3E	1	.25 sec.	Hysteresis timer
r_temp_1	63	3F	1	N/A	Location sensitive (init ram clear)
r_temp_2	64	40	1	N/A	
r_temp_3	65	41	1	N/A	
r_temp_4	66	42	1	N/A	
r_tempi_1	67	43	1	N/A	Temporary register for ISR
r_timer_a1	68	44	1	N/A	
r_timer_b	69	45	1	N/A	
r_timer_b1	70	46	1	N/A	
r_timer_c	71	47	1	N/A	
r_timer_d	72	48	1	sec.	
r_timer_d1	73	49	1	4 min.	

TABLE 15: RAM (CONTINUED)

Name	Dec	Hex	Len	Units	Description
r_led_config_1	74	4A	1	N/A	
r_led_contrl_1	75	4B	1	N/A	
r_led_config_2	76	4C	1	N/A	
r_led_contrl_2	77	4D	1	N/A	
r_adc_control	78	4E	1	N/A	ADC control
r_adc_raw_L	79	4F	1	N/A	
r_adc_raw_H	80	50	1	N/A	
r_count_1	81	51	1	N/A	
r_accD_L	82	52	1	N/A	Math – accumulator – D
r_accD_H	83	53	1	N/A	
r_accC_L	84	54	1	N/A	Math – accumulator – C
r_accC_H	85	55	1	N/A	
r_accB_L	86	56	1	N/A	Math – accumulator – B
r_accB_H	87	57	1	N/A	
r_accA_L	88	58	1	N/A	Math – accumulator – A
r_accA_H	89	59	1	N/A	
r_comm_count	90	5A	1	N/A	
r_comm_data	91	5B	1	N/A	
r_comm_flags	92	5C	1	N/A	
r_comm_data_cmnd	93	5D	1	N/A	
r_mode2	94	5E	1	bits	Operational mode flags
unused	95	5F	1	N/A	
r_adc_accum	96	60	2	N/A	
r_adc_accum_count	98	62	1	N/A	
r_adc_avg	99	63	2	N/A	
r_adc_shadow	101	65	2	units	
r_adc_1_ofs	103	67	2	mA	
r_adc_2_save	104	68	2	mV	NiMh -dv reference
r_adc_3_save_a	106	6A	2	TCODE	NiMh dtdt reference
r_adc_3_save_b	108	6C	2	TCODE	NiMh dtdt reference
r_mode3	110	6E	1	bits	Operational mode bits
r_timer_d2	111	6F	1	4 min.	Timer – min. timer – NiMh algorithm
r_shadow_1	112	70	1	N/A	Debug
r_shadow_2	113	71	1	N/A	Debug
r_shadow_3	114	72	1	N/A	Debug
r_flags_1	115	73	1	bits	Assorted bit flags
r_flags_2	116	74	1	bits	Assorted bit flags
r_flags_3	117	75	1	bits	Assorted bit flags
r_flags_4	118	76	1	bits	Assorted bit flags
r_flags_5	119	77	1	bits	Assorted bit flags
r_flags_6	120	78	1	bits	Assorted bit flags

TABLE 15: RAM (CONTINUED)

Name	Dec	Hex	Len	Units	Description
r_isr_w	121	79	1	N/A	Interrupt context
r_isr_status	122	7A	1	N/A	Interrupt context
r_isr_pclath	123	7B	1	N/A	Interrupt context
r_isr_fsr	124	7C	1	N/A	Interrupt context
r_ee_data	125	7D	1	N/A	EEPROM data
r_ee_addr	126	7E	1	N/A	EEPROM address
r_tempc_1	127	7F	1	N/A	Temporary register

NOTES:

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

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

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: 91-80-4182-8400
Fax: 91-80-4182-8422

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

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

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

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

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 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: 43-7242-2244-3910
Fax: 43-7242-2244-393

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

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

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

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

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

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

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

06/08/06