To: LFEV-ESCM Team

From: Naing Minn Htet

Date: 27 March 2014

Subject: Configuration File Format

Abstract:

This memo describes the configuration file used by the PacMan Program to setup the system parameters. When the program starts up, the configuration file is loaded. The configuration file should be in the same directory as the program and in the proper format and uses correct syntax to be recognized by the program. The configuration file will include cell parameters, charger parameters and shutdown rules. Modifying the configuration file is accomplished using either an external SD card reader or the ethernet port to use an SSH client to access the TS-8160-4200 and edit the configuration file internally.

Libconfig (http://www.hyperrealm.com/libconfig/**):**

While a simple ini parser would work as a config file parser for the Pacman Program, libconfig is chosen for its extra functionalities and documentation. Its format is similar to XML style and will allow greater expandability for future revisions of the software.

File Overview

1. File name

   The file must be named "pm14.cfg"

2. File location

   The file must be in the same directory as the program which is "/root/PM14".

3. Syntax and format

   The config file will be written in ASCII. Comments should be prepended by '//' symbol. Please review http://www.hyperrealm.com/libconfig/ for exact syntax and format rules. The variable and its value should be separated by a tab or space. Please note that the comments will be deleted as the program will rewrite the configuration file without the comments.

4. Editing

The user is encouraged to edit the config file with a text editor. This can be done by physically removing the SD card from the TS-8160-4200, plugging it into an SD card reader in a PC, and editing the text file using any text editor on the PC. Additionally, the user can also edit the configuration file for the PacMan Program by connecting the TS-8160-4200 to a network using an ethernet cable. The TS-8160-4200 in each battery pack is designed to have a fixed IP address, so a user can use an SSH client to remotely connect to the filesystem of the TS-8160-4200. After an SSH connection is made, the pm14.cfg file can be edited using an onboard text editor such as VIM. After the user has finished editing the config file, the user is expected to save their changes and then restart the program through manual board reset to allow the configuration changes to take effect.

5. Error Handling

For the program to start correctly, pm14.cfg must exist. If the program cannot find it, lcd will display an error message stating that no configuration file could be found. Please create a new config file or copy the backup config file in such case. If the config file does not have a correct syntax, the program will not start and lcd will also display an error message. You can use the log files to check where the syntax is not correct. Most likely, you will be missing a parameter.

CONFIG FILE PARAMETERS

```
//Configuration file for PM14

//Threshold limits
safebounds =
{
	//voltage.high - In discharging mode, the program will deem SYSTEM UNSAFE if the
	//voltage is higher than specified. In charging mode, the program will deem that
	//the pack has finished charging if one of the cell reaches this voltage.

	//voltage.low - In discharging mode, the program will deem zero percent state of
	//charge if the program reaches this value.

	//The units are in V.
```

```
        voltage        = { high = 3.65; low = 2.5;};

        //temperature.normal_high - This is the high threshold limit for temperature in
        //discharging mode. The program will deem SYSTEM UNSAFE if this value is reached.

        //temperature.charging_high - this is the high threshold limit for temperature in
        //charging mode. The program will stop charging if this value is reached.

        //The units are in degree Celsius.
        temperature = { normal_high = 55.0; charging_high = 40.0;};
}

//This is the pack number. Each pack should have a unique number
pack_no = 1;

//Parameters for State of Charge algorithm. Refer to the SOC algorithm for more info.
state_of_charge_parameters = {

        //This is the initial SOC for the program. Provide an estimate in percentage. Use
        // SOC = -1.0 and the program will not set initial SOC until first charge or
        // discharging cycle.
        SOC = 98.0;

        //For gain, bias and learning rate, they are the adjustment values for SOC
        //conversion. Please refer to SOC algorithm for more info.
        gain = 0.00046;
        bias = 0.0;
        learning_rate = 0.2;
}

//This is the parameters for charging current sensor.
charging_current_parameters =
{
        //This is i2c address for charging current sensor.
        address                        = 0x69;
        //This is calibration parameters for charging current sensor. a = gain, b= bias.
        current_calib  = { a = 1.0; b = 0.0;};
}
```

```
//This is the parameters for discharging current sensor.
discharging_current_parameters =
{
	//This is i2c address for discharging current sensor.
	address                      = 0x6c;
	//This is calibration parameters for discharging current sensor. a = gain,b= bias.
	current_calib  = { a = 1.025; b = -0.007;};
}


//This is the parameters for individual cells. For 7 cells, add parameters for each cell.
i2c_parameters =
(
	//Cell 1
	{
		//I2C_address
		address                      = 0x11;
		//Calibration parameters for voltage sensor in cell 1.
		voltage_calib                = { a = 1.0025; b = 0.0030; };
		//Calibration parameters for temperature sensor in cell 1.
		temperature_calib            = { a = 1.0; b = 0.0; };
	}
}
```