Cell_Balance_solution1.cpp File Reference

```
#include <iostream>
#include <bits/stdc++.h>
#include <string>
```

Macros

#define **max_batt_vol** 15
Maximum Battery Voltage (Assumption).
More...

#define **nodeVoltage** 8.0
Node Voltage To Be Maintained in Volts (Assumption).
More...

#define **min_limit** 8.0
Cutoff Voltage For a Cell in Volts.
More...

#define **b1** 14.1
Initial Cell 1 Voltage.

More...

#define **b2** 12.2
Initial Cell 2 Voltage.

More...

#define **b3** 10.3
Initial Cell 3 Voltage.

More...

#define **R_load** 10.0
Load Resistance in ohms.

More...

#define **R_1** 1.0
Branch Resistance 1.

More...

#define **R_2** 2.0
Branch Resistance 2.

More...

#define **R_3** 3.0
Branch Resistance 3.

More...

#define **dt** 0.01
Stepping time in hours (Assumption).

More...

#define **bat_cap** 0.5
Batary Capacity in Ampere hours (Assumption).

More...

#define **sim_time** 10
Simulation Time in Hours (Assumption).

More...

Functions

bool **sortbyforth** (const tuple< float, float, float, float, string, float, float > &a, const tuple< float, float, float, float, string, float, float > &b)

int **main** ()

Macro Definition Documentation

◆ b1

#define b1 14.1

Initial Cell 1 Voltage.

◆ b2

#define b2 12.2

Initial Cell 2 Voltage.

◆ b3

#define b3   10.3

Initial Cell 3 Voltage.

◆ bat_cap

#define bat_cap   0.5

Batary Capacity in Ampere hours (Assumption).

◆ dt

#define dt   0.01

Stepping time in hours (Assumption).

◆ max_batt_vol

#define max_batt_vol   15

Maximum Battery Voltage (Assumption).

◆ min_limit

#define min_limit   8.0

Cutoff Voltage For a Cell in Volts.

◆ nodeVoltage

#define nodeVoltage   8.0

Node Voltage To Be Maintained in Volts (Assumption).

◆ R_1

```
#define R_1   1.0
```

Branch Resistance 1.

◆ R_2

```
#define R_2   2.0
```

Branch Resistance 2.

◆ R_3

```
#define R_3   3.0
```

Branch Resistance 3.

◆ R_load

```
#define R_load   10.0
```

Load Resistance in ohms.

◆ sim_time

```
#define sim_time   10
```

Simulation Time in Hours (Assumption).

Function Documentation

◆ main()

int main ( )

=========================.

Cell Balancing Algorithm.

---—.

Node Voltage will be Maintained at:8 Volts.
Each Cell Can Discharge UpTo: 8 Volts.
Branch Resistances are Respectively: R_1, R_2, R_3 in Ohm.
Load Resistance is: R_load in Ohm.
Initial Cell Voltages are Respectively: b1, b2, b3 in Volts.

---—.

Assumptions:.

---—.

Battery Capacity is: 0.5 Ah.
Simulation Time is: 10 Hours.
Stepping Time in Simulation is: 0.01 Hours.
Linear Relationship Between State of Charge and Voltage.

---—.

Initialized Time Remaining with Simulation Time
Initiliaze float time_rem = sim_time;

Initializing Duty Cycles With Zero.
float d1 = 0;
float d2 = 0;
float d3 = 0;
float current = 0;

Defining Vector.
vector<tuple<float, float, float, float, string, float,float>>bat;

Inserting Branch Parameters in Vector Tuple.
bat.push_back(make_tuple(b1, R_1, d1, 1, "OFF", bat_cap,current));
bat.push_back(make_tuple(b2, R_2, d2, 2, "OFF", bat_cap,current));
bat.push_back(make_tuple(b3, R_3, d3, 3, "OFF", bat_cap,current));

To Display Initial State of Switches
cout << "Sr No.\tBat Vol\tState\tCurrent\n";

```
for (int i = 0; i < bat.size(); i++)
{

cout << get<3>(bat[i]) << "\t";
<< get<0>(bat[i]) << "\t";
<< get<4>(bat[i]) << "\n";
<<get<6>(bat[i]) << "\n";

}

cout << endl;
```

Algorithm Starts...................

while (time_rem > 0)

{

Sorting With Respect To Cell Voltage.
sort(bat.begin(), bat.end());

Assigning V_max= Highest Voltage.
float V_max = get<0>(bat[2]);

Assigning V_mid= Middle Voltage
. float V_mid = get<0>(bat[1]);

Assigning V_low= Lowest Voltage.
float V_low = get<0>(bat[0]);

Assigning R_max= Branch resistance of Highest Voltage.
float R_max = get<1>(bat[2]);

Assigning Capacity= Capacity of Highest Voltage cell.
float capacity = get<5>(bat[2]);

Assigning ON State to Highest Voltage Cell and Keeping other two OFF.
get<4>(bat[2]) = "ON";.
get<4>(bat[1]) = "OFF";.
get<4>(bat[0]) = "OFF";.

If Any Cell Voltage < 8 V, then Simulation Stops.
if (V_max <= min_limit || V_mid <= min_limit || V_low <= min_limit)
{
break;
}

Effective Resistance= R_max / d_max.
b> Calculating Node Voltage Based on Effective Resistance.
float node_vol = (V_max - curr * (R_max / d_max));

Remaining capacity of Highest Cell Voltage After Each Simulation Step.
capacity = capacity - (d_max*curr*dt);

Calculating Percent Capacity.
float percent_cap = (capacity / bat_cap) * 100;

New Cell Voltage of Highest Cell Voltage After Each Simulation Step.
V_max = max_batt_vol * ((percent_cap * 0.6 / 100) + 0.4); //Linear relationship

Assigning New Cell Voltage.
get<0>(bat[2]) = V_max;

Assigning New Value of Duty Cycle.
get<2>(bat[2]) = d_max;

Assigning Remaining Capacity.

get<5>(bat[2]) = capacity;

Sorting Accoring to Serial Number.

sort(bat.begin(), bat.end(), sortbyforth);

Displaying Serial Number, Battery Voltage and, Switch State of Each Branch.

```
for (int i = 0; i < bat.size(); i++)
{
cout << get<3>(bat[i]) << "\t";
<< get<0>(bat[i]) << "\t";
<< get<4>(bat[i]) << "\n";
<<get<6>(bat[i]) << "\n";
}
```

Decrement The Time By One Step.

time_rem = time_rem -dt;

}

return 0;

}

◆ sortbyforth()

```
bool sortbyforth ( const tuple< float, float, float, float, string, float, float > &  a,
                   const tuple< float, float, float, float, string, float, float > &  b
                 )
```

Generated by doxygen 1.8.16