#ifndef \_BATTERY\_MODEL\_H

#define \_BATTERY\_MODEL\_H

#define BATTERY\_PROFILE\_LEN 3600

#define DEFAULT\_BATTERY\_CHARGE\_MONITORING\_INTERVAL 1\*MINUTE

#define BATTERY\_CHARGE\_UPDATE\_INTERVAL 1\*SECOND // DO not change

#define NUM\_TRX\_POWER\_STATES 15

#define DEFAULT\_FULL\_BATTERY\_CAPACITY 1200.0

enum {

 IFC\_NONE,

 IFC\_WINS,

 IFC\_MICA\_MOTES,

 IFC\_WAVELAN

};

enum {

 NO\_MODEL,

 LINEAR\_MODEL,

 SERVICE\_LIFE\_ACCURATE\_MODEL,

 RESIDUAL\_LIFE\_ACCURATE\_MODEL,

 USC\_MICRO\_MODEL

};

enum {

 BATTERY\_CHARGE\_MONITORING,

 BATTERY\_CHARGE\_UPDATE

};

enum {

 TX\_POWER\_min25\_dB,

 TX\_POWER\_min15\_dB,

 TX\_POWER\_min10\_dB,

 TX\_POWER\_min7\_dB,

 TX\_POWER\_min5\_dB,

 TX\_POWER\_min3\_dB,

 TX\_POWER\_min1\_dB,

 TX\_POWER\_0\_dB,

 TX\_POWER\_1\_dB,

 TX\_POWER\_3\_dB,

 TX\_POWER\_5\_dB,

 TX\_POWER\_7\_dB,

 TX\_POWER\_10\_dB,

 TX\_POWER\_15\_dB,

 TX\_POWER\_20\_dB,

 TX\_POWER\_25\_dB,

};

typedef struct {

 double V[2];

 double I[2];

 double Vint;

 double Vcut;

 double rf;

 double B[2];

 double lambda;

 double gammaC;

 double currLoad;

 double predictedRC;

 double Temp;

 int numCycle;

 clocktype lastTimeTrans;

}UscModelParameters;

struct BatteryUtilEntry {

 double I\_bat;

 double util;

};

 struct BatteryUtilTable {

 char fileName[MAX\_STRING\_LENGTH];

 int numEntries;

 BatteryUtilEntry\* entries;

 };

typedef struct {

 double ratedCap;//mASec

 double remainingCap;//mAsec

 double takenLoad;

 double cummulative;

 BatteryUtilTable\* loadUtilTable;

}RLAModelParameters;

typedef struct {

 float usage[BATTERY\_PROFILE\_LEN];

 float \*precomputed;

 double alpha;

 int index;

 float cummulative;

} AccurateBatteryData;

typedef struct {

 int model;

 int batteryId;

 int RuntimeId;

 double remaining;

 BOOL printBatteryStats;

 BOOL dead;

 clocktype deadTime;

 clocktype chargeMonitoringPeriod;

 UscModelParameters\* uscData;

 RLAModelParameters\* rlaData;

 AccurateBatteryData\* batData;

} Battery;

void

BatteryInit(

 Node \*node,

 const NodeInput

 \*nodeInput);

void

BatteryFinalize(Node \*node);

void

BatteryProcessEvent(

 Node \*node,

 Message \*msg);

void

BatteryDecCharge(

 Node \*node,

 double duration,

 double cost);

double

BatteryGetRemainingCharge(Node \*node);

void

BATTERY\_RunTimeStat(Node \*node);

#endif