#ifndef ENERGY\_MODEL\_H

#define ENERGY\_MODEL\_H

//for enargy model

#define DEFAULT\_TRX\_CURRENT\_LOAD 280 //mA

#define DEFAULT\_RCV\_CURRENT\_LOAD 204 //mA

#define DEFAULT\_IDLE\_CURRENT\_LOAD 178 //mA

#define DEFAULT\_SLEEP\_CURRENT\_LOAD 14//mA

#define DEFAULT\_OPT\_VOLTAGE 3 //Volt

#define DEFAULT\_ALPHA\_AMP 6.5

#define DEFAULT\_PCT 100

#define DEFAULT\_PCR 130

#define DEFAULT\_PID 120

#define DEFAULT\_PSP 0

#define DEFAULT\_VS 6.5

//Contains identification od various energy models

enum EnergyModelType{

 NO\_ENERGY\_MODEL,

 TECHNOLOGY\_DEFINED\_ENERGY\_MODEL,

 USER\_DEFINED\_ENERGY\_MODEL,

 GENERIC\_ENERGY\_MODEL

};

/\*

The current loads consumed in different radio states

and the supply volatge applied to the radio

\*/

typedef struct{

 float sleep\_current\_load;

 float idle\_current\_load;

 float rcv\_current\_load;

 float trx\_current\_load;

 float\* trx\_current\_table;

 float voltage;

}PowerCosts;

//statistics parameters of energy models

typedef struct{

 double totalIdlePower;

 double totalSleepPower;

 double totalTxPower;

 double totalRxPower;

 clocktype totalSleepDuration;

 clocktype totalIdleDuration;

 clocktype totalTxDuration;

 clocktype totalRxDuration;

}PowerConsumpStats;

//Profile of the load being consumed by the radio

struct LoadProfile

{

 int RuntimeId;

 double load;

 clocktype startTime;

 clocktype lastUpdate;

 PowerConsumpStats powStats;

};

//Parameters specifiy generic energy model

struct EnergyModelGeneric

{

 double alpha\_amp;

 double Pct;

 double Pcr;

 double Pid;

 double Psp;

 double Vs;

 EnergyModelGeneric() : alpha\_amp(0.0), Pct(0.0), Pcr(0.0), Pid(0.), Psp(0.0), Vs(0.0) { ; }

};

/\*

// FUNCTION: ENERGY\_Init

// PURPOSE: This function declares energy model variables and initializes them.

// Moreover, the function read energy model specifications and configures

// the parameters which are configurable.

\*/

void

ENERGY\_Init(Node \*node,

 const int phyIndex,

 const NodeInput \*nodeInput);

// FUNCTION: ENERGY\_PrintStats

// PURPOSE: To print the statistic of Energy Model

void

ENERGY\_PrintStats(Node \*node,

 const int phyIndex);

// FUNCTION: Phy\_ReportStatusToEnergyModel

// PURPOSE: This function should be called whenever a state transition occurs

// in any place in PHY layer. As input parameters,

// the function reads the current state and the new state of PHY layer

// and based on the new sates calculates the cost of the load that should be taken off the battery.

// The function then interacts with battery model and updates the charge of battery.

void

Phy\_ReportStatusToEnergyModel(Node\* node, const int phyIndex,

 unsigned char prevStatus, unsigned char newStatus);

// FUNCTION: Generic\_UpdateCurrentLoad

// PURPOSE: To update the current load of generic energy model

void

Generic\_UpdateCurrentLoad(Node\* node, const int phyIndex);

#endif /\*ENERGY\_MODEL\_H\*/