```
$Title  GAMS FlexNet Power Scheduler

$offlisting

$Ontext

(c) Iain Staffell & Richard Green ~ Imperial College London ~ 2011-2015

Please cite this model as:
  I. Staffell & R. Green, 2015. "Is There Still Merit In The Merit Order Stack?
  The Impact of Dynamic Constraints on Optimal Plant Mix". IEEE Transactions on
  Power Systems.

Please contact i.staffell@imperial.ac.uk with any questions or comments.

A set of power stations are committed to meet national electricity demand over
a period of time.  The model finds the dispatch pattern that gives the
lowest total cost of supply over the entire period.

This model was originally developed from the MAGIC Power Scheduling Problem
(http://www.gams.com/modlib/libhtml/magic.htm).

It shares the following features with MAGIC.GMS:
  *  Generators are modelled as having start-up, no-load and incremental costs
  *  Individual generators can be shut down or started up in each period
  *  They may operate between minimum and maximum output levels
  *  Together they must supply the total demand level in each period
  *  The total amount of spare capacity that is online and available (i.e. not
     generating) must exceed the spinning reserve requirement

Notable changes and additions include:
  *  The model can process one or several years of demand data in parallel
  *  Data input and output are made via Microsoft Excel 2007 files.
  *  Generators with storage capacity are included.  Run-of-river and pumped
     hydro storage are modelled as having fixed-capacity reservoirs which will
     be depleted and replenished to minimise the cost of electricity
  *  The shadow cost of demand (wholesale price of electricity) is calculated.

By default, nine types of generator are modelled: Nuclear, large and small CCGT,
large and small coal, run-of-river and pumped hydro, OCGT and oil.  Stochastic
output from renewables (i.e. wind) can be accounted by pre-processing the input
data - subtracting their production from demand before running the model.

$Offtext

*******************************************************************************
** define our options: global definitions for data input & output
*

* set the filename and cell range where plant data are read from
$set plantsFile          _plant_data.xlsx
$set plantsRange         Sheet1!A2:K17

* set the filename and cell ranges where demand data are read from
$set demandFile          _demand_data.xlsx
$set demandRange         2020Gross!A4:B8764
$set weightRange         2020Gross!B1:B2

* set the name of the results files we will produce
$set xlsResultsFile      results_filename_new.xlsx
$set txtResultsFile      results_filename_new.txt

* decide whether we will produce xls and txt files  (1 = yes, 0 = no)
$set xlsResults          1
$set txtResults          1

* scale factor for demand data - set to 1 if demand is given in GW, or 1000 if it is in MW
$set demandScale         1

* set the carbon tax - in £/T
$set carbonCost          30

* set the identifier for temporary (.gdx, .txt, ..) files
* if you want to run multiple programs in parallel, this must be unique!
$set tmpid                gc

* set a description for this run to be listed in the results
$set idString             "Example Scenario 1 - £30 carbon"
```

```
*****************************************************************************
** define all the data for our system
*

Sets
  d     demand profiles (individual days or years)       / Y1994 /

  dd(d) dynamic version of d

  t     demand periods (hours)                           / 1*8760 /

  g     generator types                                  / Nuclear, CCGTlarge, CCGTsmall,
                                                           CoalLarge, CoalSmall, OCGT, Oil,
                                                           Hydro, Pumped, WindSpilling,
                                                           Scarcity1, Scarcity2, Scarcity3,
                                                           Scarcity4, LoadShedding /

  rg(g)  generators that physically exist                / Nuclear, CCGTlarge, CCGTsmall,
                                                           CoalLarge, CoalSmall, OCGT, Oil,
                                                           Hydro, Pumped /

  cg(g)  pseudo generators on the consumer-side          / WindSpilling, Scarcity1, Scarcity2,
                                                           Scarcity3, Scarcity4, LoadShedding /

  sg(g)  generators with energy storage                  / Hydro, Pumped /
;


* data for energy storage systems
Parameters
  loadFactor(sg)  annual output relative to capacity     / Hydro 0.415, Pumped 0.150 /
  storageCap(sg)  GWh of energy storage in reservoirs    / Hydro 25, Pumped 25 /
;


* reserve margins
Scalars
  opReserve      spinning reserve requirement (GW)       / 4.00 /
  sysReserve     system margin (% of demand)             / 0.20 /
;







* read in the data on each type of generator, producing:
*
* plant(g, "numPlant")     -  the number of plants of that type
* plant(g, "minPower")     -  the minimum output level per plant (GW)
* plant(g, "maxPower")     -  the maximum output level per plant (GW)
*
* plant(g, "fixedCost")    -  the annualised capital cost of building plant plus other fixed costs (£/kW/year)
*
* plant(g, "noLoadCost")   -  the cost of spinning at no load (£/hour per plant)
* plant(g, "incCost")      -  the incremental cost of power generation at zero carbon cost (£/MWh)
* plant(g, "startCost")    -  the cost of starting a plant from cold at zero carbon cost (£)
*
* plant(g, "noLoadCarbon") -  the carbon emissions from spinning (kg/hour)
* plant(g, "incCarbon")    -  the carbon emissions from producing power (kg/MWh)
* plant(g, "startCarbon")  -  the carbon emissions from starting up (T)
*
Parameter plant(g, *)  data for each generator type;

$CALL GDXXRW.EXE input="%plantsFile%" output="gdx.%tmpid%.plants.gdx" par=plant rng="%plantsRange%"
$GDXIN gdx.%tmpid%.plants.gdx
$LOAD plant
$GDXIN

* recalculate the no-load, incremental and start costs to factor in the cost of carbon
plant(g, "noLoadCost") = plant(g, "noLoadCost") + %carbonCost% * plant(g, "noLoadCarbon") / 1000;
plant(g, "incCost")    = plant(g, "incCost")    + %carbonCost% * plant(g, "incCarbon") / 1000;
plant(g, "startCost")  = plant(g, "startCost")  + %carbonCost% * plant(g, "startCarbon");


* convert capital cost to £m per plant per year
* (£ / kW) * (GW / plant)  =  (£m / plant)
plant(g, "fixedCost") = plant(g, "fixedCost") * plant(g, "maxPower");
```

```
* read in the demand to be met - net of any generation that cannot be
* dispatched e.g. wind, CHP, waste incinerators - and inclusive of demand
* from pumped storage (pumping), imports and exports
*
* a weighting factor is also read in for each demand profile, representing
* how many days that profile represents
Parameter demand(t, d)   GW - national electricity demand;
Parameter weight(d)      numbers of days represented;

$CALL GDXXRW.EXE input="%demandFile%" output="gdx.%tmpid%.demand.gdx" par=demand rng="%demandRange%" par=weight
rng="%weightRange%" CDim=1 RDim=0
$GDXIN gdx.%tmpid%.demand.gdx
$LOAD demand
$LOAD weight
$GDXIN


* convert from MW to GW if necessary..
if ((%demandScale% <> 1), demand(t, d) = demand(t, d) / %demandScale%);

display weight;
display demand;



*** examples of testing input data ***
*
* individual plant numbers can be tweaked to test different grid mixes..
* plant("Nuclear", "numPlant") = 40;

* individual demand values can be tweaked to see the impact on marginal costs..
* demand("2", "day1") = 32.526;
* demand("24", "day3") = 12.668;









*****************************************************************************************************
* final parameters that make it into the aggregated results
parameters

  myTotalCapacity(g)            MW

  myTotalDemand                 TWh
  myDuration                    hours

  myTotalCost                   £m
  myTotalPrice                  £m
  myTotalCarbon                 MT

  myAverageCost                 £ per MWh
  myAveragePrice                £ per MWh

  myCarbonInc                   MT CO2 from power generation
  myCarbonStart                 MT CO2 from plant start-ups

  myAverageCarbon               kg per MWh
  myMarginalCarbon              kg per MWh
  myAverageCarbonSU             kg per MWh (w starts)
  myMarginalCarbonSU            kg per MWh (w starts)

  myAverageGridMix(g)           percent of total generation
  myMarginalGridMix(g)          percent of time marginal

  myAverageUptime(g)            average plant up-time (based on # online)
  myAverageLoadFactor(g)        average load factor (based on MWh)

  myStartsPerYear(g)            average number of start-ups per year

  myGenCost(g)                  average cost of generation per plant type (£ per MWh)
  myRevenue(g)                  total revenue for plant type g (£m)
  myProfit(g)                   total profit for plant type g (£m)

  myExcessProfit(g)             excess profit for plant type g (net of capital costs) (£m)
;
```

```
* intermediate parameters that need to be calculated
parameters

  pTotalOutput(g)        total output from each plant type (TWh)

  pTotalStarts(g)        number of start ups for each plant type

  pHoursOnline(g)        number of plant-hours online

  pHoursMarginal(g)      time that each plant type was marginal (hours)
;




*****************************************************************************
** calculate some statistics about the demand and available supply
*
Parameters
  peakDemand     GW - maximum demand
  troughDemand   GW - minimum demand
  peakSupply     GW - maximum generation capacity

  totalDemand    GWh - total energy demand
  loadFactorD    load factor of the demand (relative to peak)
  loadFactorS    load factor of the demand (relative to maximum supply)

  mySysReserve   peak supply relative to peak demand
;


  peakDemand   = smax((t, d), demand(t, d));
  troughDemand = smin((t, d), demand(t, d));
  peakSupply   = sum(rg, (plant(rg, "maxPower") * plant(rg, "numPlant")));

  totalDemand  = sum(d, (weight(d) * sum(t, demand(t, d))));
  loadFactorD  = totalDemand / (peakDemand * sum(d, weight(d)) * card(t));
  loadFactorS  = totalDemand / (peakSupply * sum(d, weight(d)) * card(t));

  mySysReserve = peakSupply / peakDemand;


  display peakDemand, peakSupply, mySysReserve;




*****************************************************************************
** define the model equations
*
Variables
  output(g, t, d)        GW - generator output

  numOnline(g, t, d)     number of generators in use
  numStarts(g, t, d)     number of generators that started up for this period
  numStops(g, t, d)      number of generators that stopped for this period

  storageLevel(sg, t, d) GWh - available storage in each energy reservoir

  cost(g, t, d)          £m - total operating cost for each plant type and period
  dailyCost(d)           £m - total operating cost for each demand cluster (day)
  totalCost              £m - total operating cost over the entire set
;



* limit the number of operating plants to be an integer between 0 and the
* number of plants that are specified to be in each group
  Integer Variable numOnline;
  numOnline.up(g, t, d) = plant(g, "numPlant");

  Positive Variable numStarts;
  Positive Variable numStops;

* limit the storage level to be between 0 and the capacity limit
  storageLevel.lo(sg, t, d) = 0;
  storageLevel.up(sg, t, d) = storageCap(sg);
```

```
Equations
  eqDemandMet(t, d)      the sum of output from all plant types must be greater or equal to demand at all times
  eqSpinningMet(t, d)    the maximum possible output from all the plants online must exceed the requirement for spinning
                         reserve at all times

  eqMinLimit(g, t, d)    each plant's output must exceed its minimum rated power at all times
  eqMaxLimit(g, t, d)    each plant's output must not exceed its maximum rated power at any time

  calcStarts(g, t, d)    define the number of generators that started up for this period
  calcStops(g, t, d)     define the number of generators that stopped for this period

  calcStorage(sg, t, d)  calculate the storage level of the energy reservoirs

  calcCost(g, t, d)      calculate the cost of generation from each group in each period
  calcDayCost(d)         calculate the cost of generation for each day
  calcTotalCost          the objective function - define total cost

;


* the output of all types of plant (physical, consumer-side and storage) must equal demand
 eqDemandMet(t, dd)..     sum(g, output(g, t, dd)) =E= demand(t, dd);

* the physical capacity available must exceed demand... demand will be reduced by consumer-side
* load shedding, so we must reduce demand by their combined output
 eqSpinningMet(t, dd)..   sum(rg, plant(rg, "maxPower") * numOnline(rg, t, dd)) =G=
                          demand(t, dd) + opReserve - sum(cg, output(cg, t, dd));

 eqMinLimit(g, t, dd)..   output(g, t, dd) =G= plant(g, "minPower") * numOnline(g, t, dd);
 eqMaxLimit(g, t, dd)..   output(g, t, dd) =L= plant(g, "maxPower") * numOnline(g, t, dd);

 calcStarts(g, t, dd)..   numStarts(g, t, dd) =G= numOnline(g, t, dd) - numOnline(g, t--1, dd);
 calcStops(g, t, dd)..    numStops(g, t, dd) =G= numOnline(g, t--1, dd) - numOnline(g, t, dd);

 calcStorage(sg, t, dd).. storageLevel(sg, t, dd) =E= storageLevel(sg, t--1, dd) +
                          (loadFactor(sg) * plant(sg, "maxPower")) - output(sg, t, dd);

 calcCost(g, t, dd)..     cost(g, t, dd) =E= ( (plant(g, "noLoadCost") * numOnline(g, t, dd)) +
                                               (plant(g, "startCost") * numStarts(g, t, dd)) +
                                               (1000 * plant(g, "incCost") * output(g, t, dd))
                                             ) / 1000000;

 calcDayCost(dd)..        dailyCost(dd) =E= sum((g, t), cost(g, t, dd));

 calcTotalCost..          totalCost =E= sum(dd, weight(dd) * dailyCost(dd));









*****************************************************************************
** model solvers
*

* primary model
Model william / all /;
william.optcr = 0;
Parameter MIPmarginals(t, d);

* shadow model
Model robert / all /;
robert.optcr = 0;
Parameter LINmarginals(t, d);



* stats files
Parameter stats1 summary of supply and demand;
Parameter stats2 summary of load factors;
Parameter stats3 summary of total demand;
Parameter stats4 summary of total cost;
Parameter stats5 summary of general parameters per time period;
Parameter statsOnline summary of plants online per time period;
Parameter statsStarts summary of plant start-ups per time period;
Parameter statsOutput summary of plant output per time period;
Parameter stats6 summary of profit and revenue for each plant type;
file myFile / %txtResultsFile% /;
```

```
* cluster repetitions
Set i / rep1*rep365 /
some_i(i);



* we loop over our clustered days, setting each element of dd to yes in turn
* so that all of the above equations focus on one day at a time...
dd(d) = no;

loop(d,

        dd(d) = yes;

*       solve the integer problem to get the primary solution..
        Solve william minimising totalCost using mip;

*       store the integer shadow prices so that we can identify marginal plants (from inc cost)
        MIPmarginals(t, d) = (eqDemandMet.m(t, d) + eqSpinningMet.m(t, d)) / weight(d);

*       fix the number of plants online and solve as a linear problem to get the duals..
        numOnline.fx(g, t, d) = numOnline.l(g, t, d);

*       solve again as a relaxed integer problem to get the marginals..
*       use rmip instead of lp so that we can recycle our existing equations
        Solve robert minimising totalCost using rmip;
        LINmarginals(t, d) = (eqDemandMet.m(t, d) + eqSpinningMet.m(t, d)) / weight(d);

        dd(d) = no;
);









*******************************************************************************
** generate some output for GAMS
*

* total capacity of each plant type (MW)
myTotalCapacity(g) = plant(g, "numPlant") * plant(g, "maxPower") * 1000;

* total output from each plant type (TWh)
pTotalOutput(g) = sum((t, d), weight(d) * output.l(g, t, d)) / 1000;
display pTotalOutput;

* number of start-ups for each plant type
pTotalStarts(g) = sum((t, d), weight(d) * numStarts.l(g, t, d));
display pTotalStarts;

* number of plant-hours online
pHoursOnline(g) = sum((t, d), weight(d) * numOnline.l(g, t, d));
display pHoursOnline;

* time that each plant type was marginal (hours)
* if the incremental cost of an hour equals the generator's incremental cost, it was marginal...
pHoursMarginal(g) = 0;
loop(d,
  loop(t,
    loop(g,
      pHoursMarginal(g) $ (abs((MIPmarginals(t, d) * 1000) - plant(g, "incCost")) < 0.01) = pHoursMarginal(g) +
weight(d);
    )
  )
);

display pHoursMarginal;


myTotalDemand = sum((t, d), weight(d) * demand(t, d)) / 1000;
myDuration = sum(d, weight(d)) * card(t);

display myTotalDemand, myDuration;

myStartsPerYear(g) $ (plant(g, "numPlant") > 0) = pTotalStarts(g) / plant(g, "numPlant") * 8760 / myDuration;
display myStartsPerYear;

myTotalCost = sum(d, weight(d) * dailyCost.l(d));
```

```
myTotalPrice = sum((t, d), weight(d) * demand(t, d) * LINmarginals(t, d));

display myTotalCost, myTotalPrice;

myAverageCost = myTotalCost / myTotalDemand;
myAveragePrice = myTotalPrice / myTotalDemand;

display myAverageCost, myAveragePrice;

*                          TWh       *         kg/MWh         / 1000  =  billions of kg = MT
myCarbonInc = sum(g, pTotalOutput(g) * plant(g, "incCarbon")) / 1000;
*                           #          *            T           / 1e6  = MT
myCarbonStart = sum(g, pTotalStarts(g) * plant(g, "startCarbon")) / 1e6;
myTotalCarbon = myCarbonInc + myCarbonStart;

display myCarbonInc, myCarbonStart;

myAverageCarbon = 1000 * myCarbonInc / myTotalDemand;
myMarginalCarbon = sum(g, pHoursMarginal(g) * plant(g, "incCarbon")) / myDuration;
myAverageCarbonSU = 1000 * (myCarbonInc + myCarbonStart) / myTotalDemand;


display myAverageCarbon, myMarginalCarbon, myAverageCarbonSU;

myAverageGridMix(g) = pTotalOutput(g) / myTotalDemand;
myMarginalGridMix(g) = pHoursMarginal(g) / myDuration;

display myAverageGridMix, myMarginalGridMix;

myAverageUptime(g) $ (plant(g, "numPlant") > 0) = pHoursOnline(g) / plant(g, "numPlant") / myDuration;
loop (g,
  myAverageLoadFactor(g) = 0;

  if (plant(g, "numPlant") * plant(g, "maxPower") > 0,
    myAverageLoadFactor(g) = (pTotalOutput(g) * 1000) / (plant(g, "numPlant") * plant(g, "maxPower") * myDuration);
  );
);

display myAverageUptime, myAverageLoadFactor;




* calculate generating cost for plants with non-zero output.
* (cost * 1000) gives cost in £k, output * time is in GWh, £k/GWh = £/MWh...
loop(g,
  if (sum((t, d), output.l(g, t, d)) <> 0,
      myGenCost(g) = 1000 * sum(d, weight(d) * sum(t, cost.l(g, t, d))) / sum(d, weight(d) * sum(t, output.l(g, t,
d)));
      );
);

* (output * time) is in GWh, marginals are in £/kWh, the result is in £m
myRevenue(g) = sum(d, weight(d) * sum(t, output.l(g, t, d) * LINmarginals(t, d)));
myProfit(g) = myRevenue(g) - sum(d, weight(d) * sum(t, cost.l(g, t, d)));

* plant fixedCost has been converted from £/kW-year to £m/plant...
* super-profit = revenue - variable costs - (fixed cost x number of plant x duration in years)
myExcessProfit(g) = myProfit(g) - (plant(g, "fixedCost") * plant(g, "numPlant") * myDuration / 8760);

display myGenCost, myRevenue, myProfit, myExcessProfit;
```

```
*******************************************************************************
** generate some output for the text file
*
if (%txtResults% = 1,

  myFile.ap = 1;
  myFile.lw = 15;
  myFile.nw = 15;

  put myFile;

  put / '-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-'/;
  put '%idString%'/;

  put myTotalDemand      '  Total demand (TWh)'/;
  put myDuration         '  Duration (hours)'/;
  put /;
  put myTotalCost        '  Total cost (£m)'/;
  put myTotalPrice       '  Total price (£m)'/;
  put myTotalCarbon      '  Total carbon (MT)'/;
  put /;
  put myAverageCost      '  Average cost (£/MWh)'/;
  put myAveragePrice     '  Average price (£/MWh)'/;
  put /;
  put myAverageCarbon    '  Average carbon intensity (kg/MWh)'/;
  put myMarginalCarbon   '  Marginal carbon intensity (kg/MWh)'/;
  put myAverageCarbonSU  '  Average carbon intensity (kg/MWh - including start-ups)'/;
  put /;

  loop(g, put g.tl:>); put /;
  myFile.nd = 0; loop(g, put myTotalCapacity(g));     put '   Total capacity (MW)'/;
  myFile.nd = 2; loop(g, put pTotalOutput(g));        put '   Total ouptut (TWh)'/;
  myFile.nd = 4; loop(g, put myAverageGridMix(g));    put '   Average grid mix (%)'/;
  put /;

  myFile.nd = 2; loop(g, put pHoursMarginal(g));      put '   Time spent marginal (hours)'/;
  myFile.nd = 4; loop(g, put myMarginalGridMix(g));   put '   Marginal grid mix (%)'/;
  put /;

  myFile.nd = 4; loop(g, put myAverageUptime(g));     put '   Average uptime (%)'/;
  myFile.nd = 4; loop(g, put myAverageLoadFactor(g)); put '   Average load factor (%)'/;
  put /;

  myFile.nd = 0; loop(g, put pTotalStarts(g));        put '   Total number of start-ups'/;
  myFile.nd = 4; loop(g, put myStartsPerYear(g));     put '   Starts per plant per year'/;
  put /;

  myFile.nd = 2; loop(g, put myGenCost(g));           put '   Average gen cost (£/MWh)'/;
  myFile.nd = 3; loop(g, put myRevenue(g));           put '   Total revenue (£m)'/;
  myFile.nd = 3; loop(g, put myProfit(g));            put '   Total profit (£m)'/;
  myFile.nd = 3; loop(g, put myExcessProfit(g));      put '   Super-normal profit £m)'/;

  put /;
  putclose myFile;

);
* end %txtResults% condition..
```

```
*******************************************************************************
** generate some output for excel
*
if (%xlsResults% = 1,

  stats1("Min Demand", "(GW)") = troughDemand;
  stats1("Peak Demand", "(GW)") = peakDemand;
  stats1("Peak Supply", "(GW)") = peakSupply;

  stats2("Relative to Peak Demand", "Load Factor") = loadFactorD;
  stats2("Relative to Supply", "Load Factor") = loadFactorS;

  stats3("Total Demand", "(TWh)") = totalDemand / 1000;

  stats4("Cost of Supply", "(£m)") = myTotalCost;
  stats4("Price of Supply", "(£m)") = myTotalPrice;
```

```
* convert all zeroes into epsilon values, so that they
* can be passed into the gdx file and on into excel...
  numOnline.l(g, t, d) $ (numOnline.l(g, t, d) = 0) = eps;

  numStarts.l(g, t, d) $ (numStarts.l(g, t, d) = 0) = eps;

  output.l(g, t, d) $ (output.l(g, t, d) = 0) = eps;


* for the stats that are listed for each time period, we repeat each day the
* number of times listed in weight(d), so that overall 8760 time periods will
* be printed (if the set of days represents a year).
*
* this isn't the most efficient method, but it allows for easier processing in excel
loop (d,

*  clear our dynamic set, then activate all the elements from i to weight(d)
   some_i(i) = no;
   loop(i, some_i(i)$(ord(i) <= weight(d)) = yes; );

   stats5(d, some_i, t, "Demand (GW)") = demand(t, d);
*  available spinning capacity is the sum of all physical generators online, plus any output from load shedding
   stats5(d, some_i, t, "Spinning Capacity (GW)")    = sum(rg, plant(rg, "maxPower") * numOnline.l(rg, t, d)) +
                                                       sum(cg, output.l(cg, t, d));
   stats5(d, some_i, t, "Start ups")                 = sum(g, numStarts.l(g, t, d));
   stats5(d, some_i, t, "Shut downs")                = sum(g, numStops.l(g, t, d));
   stats5(d, some_i, t, "Total Cost (£m)")           = sum(g, cost.l(g, t, d));
   stats5(d, some_i, t, "Incremental Cost (£/MWh)")  = MIPmarginals(t, d) * 1000;
   stats5(d, some_i, t, "Marginal Cost (£/MWh)")     = LINmarginals(t, d) * 1000;

   statsOnline(d, some_i, t, g) = numOnline.l(g, t, d);
   statsStarts(d, some_i, t, g) = numStarts.l(g, t, d);
   statsOutput(d, some_i, t, g) = output.l(g, t, d);
);


stats6(g, "Gen Cost (£/MWh)") = myGenCost(g);
stats6(g, "Revenue (£m)") = myRevenue(g);
stats6(g, "Profit (£m)") = myProfit(g);



* generate a call to gdxxrw to send all this junk to excel
$onecho > gdx.%tmpid%.results.txt

par=stats1                      rng=Sheet1!A3
par=stats2                      rng=Sheet1!A8
par=stats3                      rng=Sheet1!A12
par=stats4                      rng=Sheet1!A15

epsout=0    par=stats6          rng=Sheet1!A19

text="Stats per period"         rng=Sheet1!F1
epsout=0    par=stats5          rng=Sheet1!F3

text="Number of plants online"  rng=Plants!A1
epsout=0    par=statsOnline     rng=Plants!A3

text="Number of start ups"      rng=Plants!S1
epsout=0    par=statsStarts     rng=Plants!S3

text="Total generation (GW)"    rng=Output!A1
epsout=0    par=statsOutput     rng=Output!A3

text="General stats"            rng=Sheet1!A1

$offecho

execute_unload "gdx.%tmpid%.results.gdx"
execute 'gdxxrw.exe input="gdx.%tmpid%.results.gdx" output="%xlsResultsFile%" @"gdx.%tmpid%.results.txt"';

);
* end %xlsResults% condition...
```