The $Title UCM model

$eolcom //

Option threads=10;

Option IterLim=1000000000;

Option ResLim = 10000000000;

\*Option optca=0.0;

$onecho > cplex.opt

lpmethod 4

startalg 4

advind 0

scaind 1

submipscale 1

$offecho

OPTION MIP = cplex;

// Reduce .lst file size

// Turn off the listing of the input file

$offlisting

$offlog

// Turn off the listing and cross-reference of the symbols used

$offsymxref offsymlist

option

limrow = 0, // equations listed per block

limcol = 0, // variables listed per block

solprint = off, // solver's solution output printed

sysout = off; // solver's system output printed

\*===============================================================================

\*Definition of the dataset-related options

\*===============================================================================

\* Print results to excel files (0 for no, 1 for yes)

$set Verbose 0

\* Set debug mode. !! This breaks the loop and requires a debug.gdx file !!

\* (0 for no, 1 for yes)

$set Debug 0

\* Print results to excel files (0 for no, 1 for yes)

$set PrintResults 0

\* Name of the input file (Ideally, stick to the default Input.gdx)

\*$set InputFileName Input.gdx

$set InputFileName Inputs.gdx

\* Definition of the equations that will be present in LP or MIP

\* (1 for LP 0 for MIP)

$setglobal LPFormulation 0

\* Flag to retrieve status or not

\* (1 to retrieve 0 to not)

$setglobal RetrieveStatus 0

\*===============================================================================

\*Definition of sets and parameters

\*===============================================================================

SETS

mk Markets

n Nodes

l Lines

u Units

t Generation technologies

tr(t) Renewable generation technologies

f Fuel types

p Pollutants

s(u) Hydro Storage Units (with reservoir)

chp(u) CHP units

h Hours

i(h) Subset of simulated hours for one iteration

z(h) Subset of all simulated hours

;

\*$if %LookAhead%==1 SET i TimeStep /1\*48/ ;

\*$if %LookAhead%==0 SET i TimeStep /1\*24/ ;

Alias(mk,mkmk);

Alias(n,nn);

Alias(l,ll);

Alias(u,uu);

Alias(t,tt);

Alias(f,ff);

Alias(p,pp);

Alias(s,ss);

Alias(h,hh);

Alias(i,ii);

\*Parameters as defined in the input file

PARAMETERS

ReceiveNode(l)

SendNode(l)

AvailabilityFactor(u,h) [%] Availability factor

CHPPowerLossFactor(u) [%] Power loss when generating heat

CHPPowerToHeat(u) [%] Nominal power-to-heat factor

CHPMaxHeat(chp) [MW] Maximum heat capacity of chp plant

CommittedInitial(u) [n.a.] Initial committment status

Config

CostFixed(u) [EUR\h] Fixed costs

CostRampUp(u) [EUR\MW\h] Ramp-up costs

CostRampDown(u) [EUR\MW\h] Ramp-down costs

CostShutDown(u) [EUR] Shut-down costs

CostStartUp(u) [EUR] Start-up costs

CostVariable(u,h) [EUR\MW] Variable costs

CostLoadShedding(n,h) [EUR\MWh] Cost of load shedding

Demand(mk,n,h) [MW] Demand

Efficiency(u) [%] Efficiency

EmissionMaximum(n,p) [tP] Emission limit

EmissionRate(u,p) [tP\MWh] P emission rate

FlowMaximum(l,h) [MW] Line limits

FlowMinimum(l,h) [MW] Minimum flow

FuelPrice(n,f,h) [EUR\F] Fuel price

Fuel(u,f) [n.a.] Fuel type {1 0}

LineNode(l,n) [n.a.] Incidence matrix {-1 +1}

LoadShedding(n,h) [MW] Load shedding capacity

Location(u,n) [n.a.] Location {1 0}

Markup(u,h) [EUR\MW] Markup

OutageFactor(u,h) [%] Outage Factor (100% = full outage)

PartLoadMin(u) [%] Minimum part load

PowerCapacity(u) [MW] Installed capacity

PowerInitial(u) [MW] Power output before initial period

PowerMinStable(u) [MW] Minimum power output

PriceTransmission(l,h) [EUR\MWh] Transmission price

StorageChargingCapacity(u) [MW] Storage capacity

StorageChargingEfficiency(u) [%] Charging efficiency

StorageSelfDischarge(u) [%\day] Self-discharge of the storage units

RampDownMaximum(u) [MW\h] Ramp down limit

RampShutDownMaximum(u) [MW\h] Shut-down ramp limit

RampStartUpMaximum(u) [MW\h] Start-up ramp limit

RampUpMaximum(u) [MW\h] Ramp up limit

Reserve(t) [n.a.] Reserve technology {1 0}

StorageCapacity(u) [MWh] Storage capacity

StorageDischargeEfficiency(u) [%] Discharge efficiency

StorageOutflow(u,h) [MWh] Storage outflows

StorageInflow(u,h) [MWh] Storage inflows (potential energy)

StorageInitial(u) [MWh] Storage level before initial period

StorageProfile(u,h) [MWh] Storage level to be resepected at the end of each horizon

StorageMinimum(u) [MWh] Storage minimum

Technology(u,t) [n.a.] Technology type {1 0}

TimeDown(u,h) [h] Hours down

TimeDownLeft\_initial(u) [h] Required time down left at the beginning of the simulated time period

TimeDownLeft\_JustStopped(u,h) [h] Required time down left at hour h if the Unit has just been stopped

TimeDownInitial(u) [h] Hours down before initial period

TimeDownMinimum(u) [h] Minimum down time

TimeUpLeft\_initial(u) [h] Required time up left at the beginning of the simulated time period

TimeUpInitial(u) [h] Hours on before initial period

TimeUpMinimum(u) [h] Minimum up time

FlexibilityUp(u) [MW\h] Flexibility (up) of fast-starting power plants

FlexibilityDown(u) [MW\h] Flexibility (down) of a committed power plant

$If %RetrieveStatus% == 1 CommittedCalc(u,z) [n.a.] Committment status as for the MILP

;

\*Parameters as used within the loop

PARAMETERS

TimeUpLeft\_JustStarted(u,h) [h] Required time up left at hour h if the Unit has just been started

CostLoadShedding(n,h) [EUR\MW] Value of lost load

TimeUp(u,h) [h] Hours up

LoadMaximum(u,h) [%] Maximum load given AF and OF

PowerMustRun(u,h) [MW] Minimum power output

StorageFinalMin(s) [MWh] Minimum storage level at the end of the optimization horizon

;

\*===============================================================================

\*Data import

\*===============================================================================

$gdxin %inputfilename%

$LOAD ReceiveNode

$LOAD SendNode

$LOAD mk

$LOAD n

$LOAD l

$LOAD u

$LOAD t

$LOAD tr

$LOAD f

$LOAD p

$LOAD s

$LOAD chp

$LOAD h

$LOAD z

$LOAD AvailabilityFactor

$LOAD CHPPowerLossFactor

$LOAD CHPPowerToHeat

$LOAD CHPMaxHeat

$LOAD Config

$LOAD CostFixed

$LOAD CostLoadShedding

$LOAD CostShutDown

$LOAD CostStartUp

$LOAD CostVariable

$LOAD Demand

$LOAD StorageDischargeEfficiency

$LOAD Efficiency

$LOAD EmissionMaximum

$LOAD EmissionRate

$LOAD FlowMaximum

$LOAD FlowMinimum

$LOAD FuelPrice

$LOAD Fuel

$LOAD LineNode

$LOAD LoadShedding

$LOAD Location

$LOAD Markup

$LOAD OutageFactor

$LOAD PowerCapacity

$LOAD PowerInitial

$LOAD PartLoadMin

$LOAD PriceTransmission

$LOAD StorageChargingCapacity

$LOAD StorageChargingEfficiency

$LOAD StorageSelfDischarge

$LOAD RampDownMaximum

$LOAD RampShutDownMaximum

$LOAD RampStartUpMaximum

$LOAD RampUpMaximum

$LOAD Reserve

$LOAD StorageCapacity

$LOAD StorageInflow

$LOAD StorageInitial

$LOAD StorageProfile

$LOAD StorageMinimum

$LOAD StorageOutflow

$LOAD Technology

$LOAD TimeDownInitial

$LOAD TimeDownMinimum

$LOAD TimeUpInitial

$LOAD TimeUpMinimum

$LOAD CostRampUp

$LOAD CostRampDown

$If %RetrieveStatus% == 1 $LOAD CommittedCalc

;

$If %Verbose% == 0 $goto skipdisplay

Display

mk,

n,

l,

u,

t,

tr,

f,

p,

s,

chp,

h,

AvailabilityFactor,

CHPPowerLossFactor,

CHPPowerToHeat,

CHPMaxHeat,

Config,

CostFixed,

CostShutDown,

CostStartUp,

CostRampUp,

CostVariable,

Demand,

StorageDischargeEfficiency,

Efficiency,

EmissionMaximum,

EmissionRate,

FlowMaximum,

FlowMinimum,

FuelPrice,

Fuel,

LineNode,

Location,

LoadShedding

Markup,

OutageFactor,

PartLoadMin,

PowerCapacity,

PowerInitial,

PriceTransmission,

StorageChargingCapacity,

StorageChargingEfficiency,

StorageSelfDischarge,

RampDownMaximum,

RampShutDownMaximum,

RampStartUpMaximum,

RampUpMaximum,

Reserve,

StorageCapacity,

StorageInflow,

StorageInitial,

StorageProfile,

StorageMinimum,

StorageOutflow,

Technology,

TimeDownInitial,

TimeDownMinimum,

TimeUpInitial,

TimeUpMinimum

$If %RetrieveStatus% == 1 , CommittedCalc

;

$label skipdisplay

\*===============================================================================

\*Definition of variables

\*===============================================================================

VARIABLES

Committed(u,h) [n.a.] Unit committed at hour h {1 0}

;

$If %LPFormulation% == 1 POSITIVE VARIABLES Committed (u,h) ; Committed.UP(u,h) = 1 ;

$If not %LPFormulation% == 1 BINARY VARIABLES Committed (u,h) ;

POSITIVE VARIABLES

CostStartUpH(u,h) [EUR] Cost of starting up

CostShutDownH(u,h) [EUR] cost of shutting down

CostRampUpH(u,h) [EUR] Ramping cost

CostRampDownH(u,h) [EUR] Ramping cost

Flow(l,h) [MW] Flow through lines

MaxRamp2U(u,h) [MW\h] Maximum 15-min Ramp-up capbility

MaxRamp2D(u,h) [MW\h] Maximum 15-min Ramp-down capbility

Power(u,h) [MW] Total power output for each power unit

\*\*\*\*\*\*\*

Power(u,n,h) [MW] The portion of power output for each power unit supplying certain node

PowerMaximum(u,h) [MW] Power output

PowerMinimum(u,h) [MW] Power output

ShedLoad(n,h) [MW] Shed load

StorageInput(u,h) [MWh] Charging input for storage units

StorageLevel(u,h) [MWh] Storage level of charge

LostLoad\_MaxPower(n,h) [MW] Deficit in terms of maximum power

LostLoad\_RampUp(u,h) [MW] Deficit in terms of ramping up for each plant

LostLoad\_RampDown(u,h) [MW] Deficit in terms of ramping down

LostLoad\_MinPower(n,h) [MW] Power exceeding the demand

LostLoad\_Reserve2U(n,h) [MW] Deficit in reserve up

LostLoad\_Reserve2D(n,h) [MW] Deficit in reserve down

SystemCost(h) [EUR] Hourly system cost

;

free variable

SystemCostD ![EUR] Total system cost for one optimization period

;

\*===============================================================================

\*Assignment of initial values

\*===============================================================================

\*Forecasted upwards reserve margin (UCTE). Only if not provided in the parameters

Demand("2U",n,h)$(Demand("2U",n,h)=0)=sqrt(10\*smax(hh,Demand("DA",n,hh))+150\*\*2)-150;

\*Forecasted downwards reserve margin (UCTE)

Demand("2D",n,h)$(Demand("2D",n,h)=0)=0.5\*Demand("2U",n,h);

\*Initial commitment status

CommittedInitial(u)=0;

CommittedInitial(u)$(PowerInitial(u)>0)=1;

\* Definition of the minimum stable load:

PowerMinStable(u) = PartLoadMin(u)\*PowerCapacity(u);

\* Start-up and Shutdown ramping constraints. This remains to be solved

RampStartUpMaximum(u) = max(RampStartUpMaximum(u),PowerMinStable(u));

RampShutDownMaximum(u) = max(RampShutDownMaximum(u),PowerMinStable(u));

\* If the plant is stopped, its 15-min ramp-up capability is RampStartUpMaximum if it can start in this timeframe:

FlexibilityUp(u) = RampStartUpMaximum(u)$(RampStartUpMaximum(u)>=PowerMinStable(u)\*4);

\* If the plant is started, its 15-min ramp-down capability is either RampShutDownMaximum if it is fast enough, or RampDownMaximum otherwise

FlexibilityDown(u) = RampShutDownMaximum(u)$(RampShutDownMaximum(u)>=PowerMinStable(u)\*4);

LoadMaximum(u,h)= AvailabilityFactor(u,h)\*(1-OutageFactor(u,h));

PowerMustRun(u,h)=PowerMinStable(u)\*LoadMaximum(u,h);

$If %Verbose% == 1 Display RampStartUpMaximum, RampShutDownMaximum, CommittedInitial, FlexibilityUp, FlexibilityDown;

$offorder

\*===============================================================================

\*Declaration and definition of equations

\*===============================================================================

EQUATIONS

\*\*\*\*\*\*\*

EQ\_Power\_Output\_Split

EQ\_Objective\_function

EQ\_CHP\_extraction\_Pmax

EQ\_CHP\_extraction

EQ\_CHP\_backpressure

EQ\_CHP\_P2H

EQ\_CHP\_demand\_satisfaction

EQ\_CHP\_max\_heat

EQ\_Heat\_Storage\_balance

EQ\_Heat\_Storage\_minimum

EQ\_Heat\_Storage\_level

EQ\_CostStartUp

EQ\_CostShutDown

EQ\_CostRampUp

EQ\_CostRampDown

EQ\_Demand\_balance\_DA

EQ\_Demand\_balance\_2U

EQ\_Demand\_balance\_2D

EQ\_Power\_must\_run

EQ\_Power\_bound\_lower

EQ\_Power\_bound\_upper

EQ\_Power\_available

EQ\_Ramp\_up

EQ\_Ramp\_down

EQ\_Max\_RampUp1

EQ\_Max\_RampUp2

EQ\_Max\_RampDown1

EQ\_Max\_RampDown2

EQ\_Minimum\_time\_up\_A

EQ\_Minimum\_time\_up\_B

EQ\_Minimum\_time\_up\_C

EQ\_Minimum\_time\_up\_JustStarted

EQ\_Minimum\_time\_down\_A

EQ\_Minimum\_time\_down\_B

EQ\_Minimum\_time\_down\_C

EQ\_Minimum\_time\_down\_JustStopped

EQ\_Storage\_minimum

EQ\_Storage\_level

EQ\_Storage\_input

EQ\_Storage\_MaxDischarge

EQ\_Storage\_MaxCharge

EQ\_Storage\_balance

EQ\_Storage\_boundaries

EQ\_SystemCost

EQ\_Emission\_limits

EQ\_Flow\_limits\_lower

EQ\_Flow\_limits\_upper

EQ\_Flow\_limits\_upper1

EQ\_Flow\_limits\_upper1i

EQ\_Flow\_limits\_upper2

EQ\_Flow\_limits\_upper2i

EQ\_Flow\_limits\_upper3

EQ\_Flow\_limits\_upper3i

EQ\_Flow\_limits\_upper4

EQ\_Flow\_limits\_upper4i

EQ\_Flow\_limits\_upper5

EQ\_Flow\_limits\_upper5i

EQ\_Flow\_limits\_upper6

EQ\_Flow\_limits\_upper6i

EQ\_Flow\_limits\_upper7

EQ\_Flow\_limits\_upper7i

EQ\_Flow\_limits\_upper8

EQ\_Flow\_limits\_upper8i

EQ\_Flow\_limits\_upper9

EQ\_Flow\_limits\_upper9i

EQ\_Force\_Commitment

EQ\_Force\_DeCommitment

EQ\_LoadShedding

$If %RetrieveStatus% == 1 EQ\_CommittedCalc

;

$If %RetrieveStatus% == 0 $goto skipequation

EQ\_CommittedCalc(u,z)..

Committed(u,z)

=E=

CommittedCalc(u,z)

;

$label skipequation

\*\*\*\*\*\*\*

EQ\_Power\_Output\_Split(u,i)..

Power(u,i) =E= sum(n,Power(u,n,i))

;

EQ\_Flow\_Definition(l,i)..

Flow(l,i) =E= sum(u, Power(u,ReceiveNode(l),i)\*Location(u,SendNode(l))

;

\*Objective function

EQ\_SystemCost(i)..

SystemCost(i)

=E=

sum(u,CostFixed(u)\*Committed(u,i))

+sum(u,CostStartUpH(u,i) + CostShutDownH(u,i))

+sum(u,CostRampUpH(u,i) + CostRampDownH(u,i))

+sum(u,CostVariable(u,i) \* Power(u,i))

+sum(l,PriceTransmission(l,i)\*Flow(l,i))

+sum(n,CostLoadShedding(n,i)\*ShedLoad(n,i))

+100E3\*(sum(n,LostLoad\_MaxPower(n,i)+LostLoad\_MinPower(n,i)))

+80E3\*(sum(n,LostLoad\_Reserve2U(n,i)+LostLoad\_Reserve2D(n,i)))

+70E3\*sum(u,LostLoad\_RampUp(u,i)+LostLoad\_RampDown(u,i))

;

EQ\_Objective\_function..

SystemCostD

=E=

sum(i,SystemCost(i))

;

EQ\_CostStartUp(u,i)$(CostStartUp(u) <> 0)..

CostStartUpH(u,i)

=G=

CostStartUp(u)\*(Committed(u,i)-CommittedInitial(u)$(ord(i) = 1)-Committed(u,i-1)$(ord(i) > 1))

;

EQ\_CostShutDown(u,i)$(CostShutDown(u) <> 0)..

CostShutDownH(u,i)

=G=

CostShutDown(u)\*(CommittedInitial(u)$(ord(i) = 1)+Committed(u,i-1)$(ord(i) > 1)-Committed(u,i))

;

EQ\_CostRampUp(u,i)$(CostRampUp(u) <> 0)..

CostRampUpH(u,i)

=G=

CostRampUp(u)\*(Power(u,i)-PowerInitial(u)$(ord(i) = 1)-Power(u,i-1)$(ord(i) > 1))

;

EQ\_CostRampDown(u,i)$(CostRampDown(u) <> 0)..

CostRampDownH(u,i)

=G=

CostRampDown(u)\*(PowerInitial(u)$(ord(i) = 1)+Power(u,i-1)$(ord(i) > 1)-Power(u,i))

;

\*\*\*\*\*\*\*

\*Hourly demand balance in the day-ahead market for each node

EQ\_Demand\_balance\_DA(n,i)..

sum(u,Power(u,i)\*Location(u,n))

+sum(l,Flow(l,i)\*LineNode(l,n))

=E=

Demand("DA",n,i)

-ShedLoad(n,i)

-LostLoad\_MaxPower(n,i)

+LostLoad\_MinPower(n,i)

;

\* Maximum 15-min ramping up, in MW/h:

Eq\_Max\_RampUp1(u,i)$(sum(tr,Technology(u,tr))=0)..

MaxRamp2U(u,i)

=L=

RampUpMaximum(u)\*Committed(u,i)

+ FlexibilityUp(u)\*(1-Committed(u,i))

;

\* Maximum 15-min ramping up, in MW/h:

Eq\_Max\_RampUp2(u,i)$(sum(tr,Technology(u,tr))=0)..

MaxRamp2U(u,i)

=L=

(PowerCapacity(u)\*LoadMaximum(u,i) - Power(u,i))\*4

;

\* Maximum 15-min shutting down, in MW/h:

Eq\_Max\_RampDown1(u,i)$(sum(tr,Technology(u,tr))=0)..

MaxRamp2D(u,i)

=L=

max(RampDownMaximum(u),FlexibilityDown(u))\*Committed(u,i)

;

\* Maximum 15-min ramping down, in MW/h:

Eq\_Max\_RampDown2(u,i)$(sum(tr,Technology(u,tr))=0)..

MaxRamp2D(u,i)

=L=

(Power(u,i) - PowerMinStable(u)$(RampShutDownMaximum(u)<PowerMinStable(u)\*4)\*Committed(u,i))\*4

;

EQ\_Demand\_balance\_2U(n,i)..

sum((u,t),MaxRamp2U(u,i)\*Technology(u,t)\*Reserve(t)\*Location(u,n))

=G=

+Demand("2U",n,i)

-LostLoad\_reserve2U(n,i)

;

\*Hourly demand balance in the downwards reserve market for each node

EQ\_Demand\_balance\_2D(n,i)..

sum((u,t),MaxRamp2D(u,i)\*Technology(u,t)\*Reserve(t)\*Location(u,n))

=G=

Demand("2D",n,i)

-sum(s,(StorageChargingCapacity(s)-StorageInput(s,i)) )\*4

-LostLoad\_reserve2D(n,i)

;

\*Minimum power output is above the must-run output level for each unit in all periods

\*\*\*\*\*\*\*

EQ\_Power\_must\_run(u,i)..

PowerMustRun(u,i) \* Committed(u,i)

=L=

Power(u,i)

;

\*Maximum power output is below the available capacity

EQ\_Power\_available(u,i)..

Power(u,i)

=L=

PowerCapacity(u)

\*LoadMaximum(u,i)

\*Committed(u,i)

;

\*Maximum power output with respect to power output in the previous period (ramping up constraint).

EQ\_Ramp\_up(u,i)$(sum(tr,Technology(u,tr))=0)..

Power(u,i)

=L=

(PowerInitial(u)

+RampUpMaximum(u)

\*CommittedInitial(u)

+RampStartUpMaximum(u)

\*(1-CommittedInitial(u)))$(ord(i) = 1)

+(Power(u,i-1)

+RampUpMaximum(u)

\*Committed(u,i-1)

+RampStartUpMaximum(u)

\*(1-Committed(u,i-1)))$(ord(i) > 1)

+LostLoad\_RampUp(u,i)

;

\*If the unit keeps committed the reduction in power output is lower than the

\*ramp-down limit. If the unit is de-committed the reduction is lower than the

\*shut-down ramp limit

EQ\_Ramp\_down(u,i)$(sum(tr,Technology(u,tr))=0)..

(PowerInitial(u)-Power(u,i))$(ord(i) = 1)

+(Power(u,i-1)-Power(u,i))$(ord(i) > 1)

=L=

RampDownMaximum(u) \* Committed(u,i)

+ RampShutDownMaximum(u) \* (1-Committed(u,i))

+ LostLoad\_RampDown(u,i)

;

EQ\_Minimum\_time\_up\_A(u)..

sum(i$(ord(i) <= TimeUpLeft\_initial(u)),1-Committed(u,i))

=E=

0

;

EQ\_Minimum\_time\_up\_B(u,i)$((TimeUpLeft\_initial(u)+1 <= ord(i)) and (ord(i) <= card(i)-TimeUpMinimum(u)+1))..

sum(ii$((ord(i) <= ord(ii)) and (ord(ii) <= (ord(i)+TimeUpMinimum(u)-1))),Committed(u,ii))

=G=

TimeUpMinimum(u)

\*(Committed(u,i)-CommittedInitial(u)$(ord(i) = 1)-Committed(u,i-1)$(ord(i) > 1))

;

EQ\_Minimum\_time\_up\_C(u,i)$((card(i)-TimeUpMinimum(u)+2 <= ord(i)) and (ord(i)<=card(i)))..

sum(ii$((ord(i) <= ord(ii)) and (ord(ii) <= card(i))),Committed(u,ii)-(Committed(u,i)-CommittedInitial(u)$(ord(i) = 1)-Committed(u,i-1)$(ord(i) > 1)))

=G=

0

;

EQ\_Minimum\_time\_up\_JustStarted(u,i)$(ord(i) > 1)..

sum(ii$((ord(i) <= ord(ii)) and (ord(ii) <= (ord(i)+TimeUpLeft\_JustStarted(u,i)-1))),Committed(u,ii))

=G=

TimeUpLeft\_JustStarted(u,i)

\*(Committed(u,i)-CommittedInitial(u)$(ord(i) = 1)-Committed(u,i-1)$(ord(i) > 1))

;

EQ\_Minimum\_time\_down\_A(u)..

sum(i$(ord(i) <= TimeDownLeft\_initial(u)),Committed(u,i))

=E=

0

;

EQ\_Minimum\_time\_down\_B(u,i)$((TimeDownLeft\_initial(u)+1 <= ord(i)) and (ord(i) <= card(i)-TimeDownMinimum(u)+1))..

sum(ii$((ord(i) <= ord(ii)) and (ord(ii) <= (ord(i)+TimeDownMinimum(u)-1))),1-Committed(u,ii))

=G=

TimeDownMinimum(u)

\*(CommittedInitial(u)$(ord(i) = 1)+Committed(u,i-1)$(ord(i) > 1)-Committed(u,i))

;

EQ\_Minimum\_time\_down\_C(u,i)$((card(i)-TimeDownMinimum(u)+2 <= ord(i)) and (ord(i)<=card(i)))..

sum(ii$((ord(i) <= ord(ii)) and (ord(ii) <= card(i))),1-Committed(u,ii)-(CommittedInitial(u)$(ord(i) = 1)+Committed(u,i-1)$(ord(i) > 1)-Committed(u,i)))

=G=

0

;

\*IH: why do we need this equation?, to replace the two previous?

EQ\_Minimum\_time\_down\_JustStopped(u,i)$(TimeDownLeft\_initial(u)+1 <= ord(i))..

sum(ii$((ord(i) <= ord(ii)) and (ord(ii) <= (ord(i)+TimeDownLeft\_JustStopped(u,i)-1))),1-Committed(u,ii))

=G=

TimeDownLeft\_JustStopped(u,i)

\*(CommittedInitial(u)$(ord(i) = 1)+Committed(u,i-1)$(ord(i) > 1)-Committed(u,i))

;

\*Total emissions are capped

EQ\_Emission\_limits(n,i,p)..

sum(u,Power(u,i)\*EmissionRate(u,p)\*Location(u,n))

=L=

EmissionMaximum(n,p)

;

\*Flows are above minimum values

EQ\_Flow\_limits\_lower(l,i)..

FlowMinimum(l,i)

=L=

Flow(l,i)

;

\*Flows are below maximum values

EQ\_Flow\_limits\_upper(l,i)..

Flow(l,i)

=L=

FlowMaximum(l,i)

;

\*Flows are below maximum values of injection edges of each country

EQ\_Flow\_limits\_upper1(i)..

Flow("KW -> SA\_EOA",i) + Flow("BA -> SA\_EOA",i) + Flow("QA -> SA\_EOA",i) + Flow("UAE -> SA\_EOA",i) + Flow("OM -> SA\_EOA",i) =L= 1800

;

EQ\_Flow\_limits\_upper1i(i)..

Flow("SA\_EOA -> KW",i) + Flow("SA\_EOA -> BA",i) + Flow("SA\_EOA -> QA",i) + Flow("SA\_EOA -> UAE",i) + Flow("SA\_EOA -> OM",i) =L= 1800

;

EQ\_Flow\_limits\_upper2(i)..

Flow("SA\_EOA -> KW",i) + Flow("BA -> KW",i) + Flow("QA -> KW",i) + Flow("UAE -> KW",i) + Flow("OM -> KW",i) =L= 1550

;

EQ\_Flow\_limits\_upper2i(i)..

Flow("KW -> SA\_EOA",i) + Flow("KW -> BA",i) + Flow("KW -> QA",i) + Flow("KW -> UAE",i) + Flow("KW -> OM",i) =L= 1550

;

EQ\_Flow\_limits\_upper3(i)..

Flow("SA\_EOA -> BA",i) + Flow("KW -> BA",i) + Flow("QA -> BA",i) + Flow("UAE -> BA",i) + Flow("OM -> BA",i) =L= 1360

;

EQ\_Flow\_limits\_upper3i(i)..

Flow("BA -> SA\_EOA",i) + Flow("BA -> KW",i) + Flow("BA -> QA",i) + Flow("BA -> UAE",i) + Flow("BA -> OM",i) =L= 1360

;

EQ\_Flow\_limits\_upper4(i)..

Flow("SA\_EOA -> QA",i) + Flow("KW -> QA",i) + Flow("BA -> QA",i) + Flow("UAE -> QA",i) + Flow("OM -> QA",i) =L= 1500

;

EQ\_Flow\_limits\_upper4i(i)..

Flow("QA -> SA\_EOA",i) + Flow("QA -> KW",i) + Flow("QA -> BA",i) + Flow("QA -> UAE",i) + Flow("QA -> OM",i) =L= 1500

;

EQ\_Flow\_limits\_upper5(i)..

Flow("SA\_EOA -> UAE",i) + Flow("KW -> UAE",i) + Flow("BA -> UAE",i) + Flow("QA -> UAE",i) =L= 1550

;

EQ\_Flow\_limits\_upper5i(i)..

Flow("UAE -> SA\_EOA",i) + Flow("UAE -> KW",i) + Flow("UAE -> BA",i) + Flow("UAE -> QA",i) =L= 1550

;

EQ\_Flow\_limits\_upper6(i)..

Flow("SA\_EOA -> OM",i) + Flow("KW -> OM",i) + Flow("BA -> OM",i) + Flow("QA -> OM",i) + Flow("UAE -> OM",i) =L= 400

;

EQ\_Flow\_limits\_upper6i(i)..

Flow("OM -> SA\_EOA",i) + Flow("OM -> KW",i) + Flow("OM -> BA",i) + Flow("OM -> QA",i) + Flow("OM -> UAE",i) =L= 400

;

\*Flows are below maximum values of some edges (transmission lines) that are shared between GCC countries

\*transmission line between Salwa and UAE

EQ\_Flow\_limits\_upper7(i)..

Flow("SA\_EOA -> OM",i) + Flow("KW -> OM",i) + Flow("BA -> OM",i) + Flow("QA -> OM",i) + Flow("SA\_EOA -> UAE",i) + Flow("KW -> UAE",i) + Flow("BA -> UAE",i) + Flow("QA -> UAE",i) =L= 1550

;

EQ\_Flow\_limits\_upper7i(i)..

Flow("UAE -> QA",i) + Flow("OM -> QA",i) + Flow("UAE -> BA",i) + Flow("OM -> BA",i) + Flow("UAE -> SA\_EOA",i) + Flow("OM -> SA\_EOA",i) + Flow("UAE -> KW",i) + Flow("OM -> KW",i) =L= 1550

;

\*transmission line between Ghunan and Salwa

EQ\_Flow\_limits\_upper8(i)..

Flow("SA\_EOA -> QA",i) + Flow("KW -> QA",i) + Flow("BA -> QA",i) + Flow("SA\_EOA -> UAE",i) + Flow("KW -> UAE",i) + Flow("BA -> UAE",i) + Flow("SA\_EOA -> OM",i) + Flow("KW -> OM",i) + Flow("BA -> OM",i) =L= 1500

;

EQ\_Flow\_limits\_upper8i(i)..

Flow("QA -> KW",i) + Flow("UAE -> KW",i) + Flow("OM -> KW",i) + Flow("QA -> SA\_EOA",i) + Flow("UAE -> SA\_EOA",i) + Flow("OM -> SA\_EOA",i) + Flow("QA -> BA",i) + Flow("UAE -> BA",i) + Flow("OM -> BA",i) =L= 1500

;

\*transmission line between Alfadhili and Ghunan

EQ\_Flow\_limits\_upper9(i)..

Flow("BA -> KW",i) + Flow("QA -> KW",i) + Flow("UAE -> KW",i) + Flow("OM -> KW",i) + Flow("BA -> SA\_EOA",i) + Flow("QA -> SA\_EOA",i) + Flow("UAE -> SA\_EOA",i) + Flow("OM -> SA\_EOA",i) =L= 1500

;

EQ\_Flow\_limits\_upper9i(i)..

Flow("SA\_EOA -> BA",i) + Flow("KW -> BA",i) + Flow("SA\_EOA -> QA",i) + Flow("KW -> QA",i) + Flow("SA\_EOA -> UAE",i) + Flow("KW -> UAE",i) + Flow("SA\_EOA -> OM",i) + Flow("KW -> OM",i) =L= 1500

;

\*Force Unit commitment/decommitment:

\* E.g: renewable units with AF>0 must be committed

EQ\_Force\_Commitment(u,i)$((sum(tr,Technology(u,tr))>=1 and LoadMaximum(u,i)>0) or (ord(i)=4 and ord(u)=129))..

Committed(u,i)

=E=

1;

\* E.g: renewable units with AF=0 must be decommitted

EQ\_Force\_DeCommitment(u,i)$(LoadMaximum(u,i)=0 or ord(u)=200)..

Committed(u,i)

=E=

0;

\*Load shedding

EQ\_LoadShedding(n,i)..

ShedLoad(n,i)

=L=

LoadShedding(n,i)

;

\*===============================================================================

\*Definition of models

\*===============================================================================

MODEL UCM\_SIMPLE /

EQ\_Objective\_function,

$If not %LPFormulation% == 1 EQ\_CostStartUp,

$If not %LPFormulation% == 1 EQ\_CostShutDown,

$If %LPFormulation% == 1 EQ\_CostRampUp,

$If %LPFormulation% == 1 EQ\_CostRampDown,

EQ\_Demand\_balance\_DA,

EQ\_Demand\_balance\_2U,

EQ\_Demand\_balance\_2D,

$If not %LPFormulation% == 1 EQ\_Power\_must\_run,

EQ\_Power\_available,

EQ\_Ramp\_up,

EQ\_Ramp\_down,

$If not %LPFormulation% == 1 EQ\_Minimum\_time\_up\_A,

$If not %LPFormulation% == 1 EQ\_Minimum\_time\_up\_JustStarted,

$If not %LPFormulation% == 1 EQ\_Minimum\_time\_down\_A,

$If not %LPFormulation% == 1 EQ\_Minimum\_time\_down\_JustStopped,

EQ\_Max\_RampUp1,

EQ\_Max\_RampUp2,

EQ\_Max\_RampDown1,

EQ\_Max\_RampDown2,

EQ\_SystemCost

EQ\_Flow\_limits\_lower,

EQ\_Flow\_limits\_upper,

EQ\_Flow\_limits\_upper1,

EQ\_Flow\_limits\_upper1i,

EQ\_Flow\_limits\_upper2,

EQ\_Flow\_limits\_upper2i,

EQ\_Flow\_limits\_upper3,

EQ\_Flow\_limits\_upper3i,

EQ\_Flow\_limits\_upper4,

EQ\_Flow\_limits\_upper4i,

EQ\_Flow\_limits\_upper5,

EQ\_Flow\_limits\_upper5i,

EQ\_Flow\_limits\_upper6,

EQ\_Flow\_limits\_upper6i,

EQ\_Flow\_limits\_upper7,

EQ\_Flow\_limits\_upper7i,

EQ\_Flow\_limits\_upper8,

EQ\_Flow\_limits\_upper8i,

EQ\_Flow\_limits\_upper9,

EQ\_Flow\_limits\_upper9i,

EQ\_Force\_Commitment,

EQ\_Force\_DeCommitment,

EQ\_LoadShedding,

$If %RetrieveStatus% == 1 EQ\_CommittedCalc

/

;

UCM\_SIMPLE.optcr = 0.01;

UCM\_SIMPLE.optfile = 1;

UCM\_SIMPLE.holdfixed = 1;

\*UCM\_SIMPLE.epgap = 0.005

\*UCM\_SIMPLE.probe = 3

\*UCM\_SIMPLE.optfile=1;

\*===============================================================================

\*Solving loop

\*===============================================================================

\* Scalar variables necessary to the loop:

scalar FirstHour,LastHour,LastKeptHour,day,ndays,failed;

ndays = floor(card(h)/24);

if (Config("RollingHorizon LookAhead","day") > ndays -1, abort "The look ahead period is longer than the simulation length";);

\* Some parameters used for debugging:

failed=0;

parameter TimeUpInitial\_dbg(u), TimeDownInitial\_dbg(u), CommittedInitial\_dbg(u), PowerInitial\_dbg(u) ;

\* Fixing the initial guesses:

\*PowerH.L(u,i)=PowerInitial(u);

\*Committed.L(u,i)=CommittedInitial(u);

\* Defining a parameter that records the solver status:

set tmp "tpm" / "model", "solver" / ;

parameter status(tmp,h);

$if %Debug% == 1 $goto DebugSection

FOR(day = 1 TO ndays-Config("RollingHorizon LookAhead","day") by Config("RollingHorizon Length","day"),

FirstHour = (day-1)\*24+1;

LastHour = min(card(h),FirstHour + (Config("RollingHorizon Length","day")+Config("RollingHorizon LookAhead","day")) \* 24 - 1);

LastKeptHour = LastHour - Config("RollingHorizon LookAhead","day") \* 24;

i(h) = no;

i(h)$(ord(h)>=firsthour and ord(h)<=lasthour)=yes;

display day,FirstHour,LastHour,LastKeptHour;

TimeUpLeft\_initial(u)=min(card(i),(TimeUpMinimum(u)-TimeUpInitial(u))\*CommittedInitial(u));

TimeUpLeft\_JustStarted(u,i) = min(card(i)-ord(i)+1,TimeUpMinimum(u));

TimeDownLeft\_initial(u)=min(card(i),(TimeDownMinimum(u)-TimeDownInitial(u))\*(1-CommittedInitial(u)));

TimeDownLeft\_JustStopped(u,i) = min(card(i)-ord(i)+1,TimeDownMinimum(u));

$If %Verbose% == 1 Display TimeUpLeft\_initial,TimeUpLeft\_JustStarted,TimeDownLeft\_initial,TimeDownLeft\_JustStopped,TimeUpInitial,TimeDownInitial,PowerInitial,CommittedInitial,StorageFinalMin;

$If %LPFormulation% == 1 SOLVE UCM\_SIMPLE USING LP MINIMIZING SystemCostD;

$If not %LPFormulation% == 1 SOLVE UCM\_SIMPLE USING MIP MINIMIZING SystemCostD;

$If %LPFormulation% == 1 Display EQ\_Objective\_function.M, EQ\_CostRampUp.M, EQ\_CostRampDown.M, EQ\_Demand\_balance\_DA.M, EQ\_Power\_available.M, EQ\_Ramp\_up.M, EQ\_Ramp\_down.M, EQ\_Max\_RampUp1.M, EQ\_Max\_RampUp2.M,EQ\_Max\_RampDown1.M, EQ\_Max\_RampDown2.M, EQ\_Flow\_limits\_lower.M ;

$If not %LPFormulation% == 1 Display EQ\_Objective\_function.M, EQ\_CostStartUp.M, EQ\_CostShutDown.M, EQ\_Demand\_balance\_DA.M, EQ\_Power\_must\_run.M, EQ\_Power\_available.M, EQ\_Ramp\_up.M, EQ\_Ramp\_down.M, EQ\_Minimum\_time\_up\_A.M, EQ\_Minimum\_time\_up\_JustStarted.M, EQ\_Minimum\_time\_down\_A.M, EQ\_Minimum\_time\_down\_JustStopped.M, EQ\_Max\_RampUp1.M, EQ\_Max\_RampUp2.M, EQ\_Max\_RampDown1.M, EQ\_Max\_RampDown2.M, EQ\_Flow\_limits\_lower.M ;

status("model",i) = UCM\_SIMPLE.Modelstat;

status("solver",i) = UCM\_SIMPLE.Solvestat;

\*\*\*\*\*\*\*\*

if(UCM\_SIMPLE.Modelstat <> 1 and UCM\_SIMPLE.Modelstat <> 8 and not failed, TimeUpInitial\_dbg(u) = TimeUpInitial(u); TimeDownInitial\_dbg(u) = TimeDownInitial(u); CommittedInitial\_dbg(u) = CommittedInitial(u); PowerInitial\_dbg(u) = PowerInitial(u);

\*\*\*\*\*\*\*\*

EXECUTE\_UNLOAD "debug.gdx" day, status, TimeUpInitial\_dbg, TimeDownInitial\_dbg, CommittedInitial\_dbg, PowerInitial\_dbg;

failed=1;);

\*Time counters

Loop(i,

TimeUp(u,i)$(ord(i) = 1 and Committed.L(u,i) = 1)=TimeUpInitial(u)+1;

TimeUp(u,i)$(ord(i) = 1 and Committed.L(u,i) = 0)=0;

TimeUp(u,i)$(ord(i) > 1 and Committed.L(u,i) = 1) = TimeUp(u,i-1)+1;

TimeUp(u,i)$(ord(i) > 1 and Committed.L(u,i) = 0) = 0;

TimeDown(u,i)$(ord(i) = 1 and Committed.L(u,i) = 0) = TimeDownInitial(u)+1;

TimeDown(u,i)$(ord(i) = 1 and Committed.L(u,i) = 1) = 0;

TimeDown(u,i)$(ord(i) > 1 and Committed.L(u,i) = 0) = TimeDown(u,i-1)+1;

TimeDown(u,i)$(ord(i) > 1 and Committed.L(u,i) = 1) = 0;

);

TimeUpInitial(u)=sum(i$(ord(i)=LastKeptHour-FirstHour+1),TimeUp(u,i));

TimeDownInitial(u)=sum(i$(ord(i)=LastKeptHour-FirstHour+1),TimeDown(u,i));

CommittedInitial(u)=sum(i$(ord(i)=LastKeptHour-FirstHour+1),Committed.L(u,i));

PowerInitial(u) = sum(i$(ord(i)=LastKeptHour-FirstHour+1),Power.L(u,i));

\*Loop variables to display after solving:

$If %Verbose% == 1 Display LastKeptHour,PowerInitial,TimeUp,TimeDown,MaxRamp2D.L,MaxRamp2U.L,CostStartUpH.L,CostShutDownH.L,CostRampUpH.L;

);

$If %Verbose% == 1 Display Flow.L,Power.L,Committed.L,ShedLoad.L,SystemCost.L,MaxRamp2U.L,MaxRamp2D.L,LostLoad\_MaxPower.L,LostLoad\_MinPower.L,LostLoad\_reserve2U.L,LostLoad\_reserve2D.L,LostLoad\_RampUP.L,LostLoad\_RampDown.L;

\*===============================================================================

\*Result export

\*===============================================================================

PARAMETER

OutputCommitted(u,h)

OutputFlow(l,h)

OutputPower(u,h)

\*\*\*\*\*\*\*

OutputPowerLocal(n,h)

ImportedPower(n,h)

ExportedPower(n,h)

OutputPowerInside(u,h)

OutputPowerOutside(u,h)

OutputSystemCost(h)

OutputShedLoad(n,h)

ShadowPrice(n,h)

FixedCost(u,h)

StarUpCost(u,h)

ShutDownCost(u,h)

RampUpCost(u,h)

RampDownCost(u,h)

VariableCost(u,h)

OperationCost(u,h)

PowerTradeValue(l,h)

PowerTradeMin(l,h)

;

FixedCost(u,z)= CostFixed(u)\*Committed.L(u,z);

StarUpCost(u,z)= CostStartUpH.L(u,z);

ShutDownCost(u,z)= CostShutDownH.L(u,z);

RampUpCost(u,z)= CostRampUpH.L(u,z);

RampDownCost(u,z)= CostRampDownH.L(u,z);

VariableCost(u,z)= CostVariable(u,z) \* Power.L(u,z);

OperationCost(u,z)= CostFixed(u)\*Committed.L(u,z) + CostStartUpH.L(u,z) + CostShutDownH.L(u,z) + CostRampUpH.L(u,z) + CostRampDownH.L(u,z) + CostVariable(u,z) \* Power.L(u,z);

PowerTradeValue(l,z)= EQ\_Flow\_limits\_upper.m(l,z);

PowerTradeMin(l,z)= EQ\_Flow\_limits\_lower.m(l,z);

OutputCommitted(u,z)=Committed.L(u,z);

OutputFlow(l,z)=Flow.L(l,z);

OutputPower(u,z)=Power.L(u,z);

\*\*\*\*\*\*\*

OutputPowerLocal(n,z)= sum(u, Power(u,n,z)\*Location(u,n));

ImportedPower(n,z)= sum(u,Power(u,n,z)\*(1-Location(u,n));

ExportedPower(n,z)= sum(u,(Power(u,z)-Power(u,n,z))\*Location(u,n));

OutputPowerInside(u,z)= sum(n,Power(u,n,z)\*Location(u,n));

OutputPowerOutside(u,z)= sum(n,Power(u,n,z)\*(1-Location(u,n));

OutputSystemCost(z)=SystemCost.L(z);

OutputShedLoad(n,z) = ShedLoad.L(n,z);

ShadowPrice(n,z) = EQ\_Demand\_balance\_DA.m(n,z);

EXECUTE\_UNLOAD "Results.gdx"

OutputCommitted,

OutputFlow,

OutputPower,

OutputSystemCost,

OutputShedLoad,

LostLoad\_MaxPower,

LostLoad\_MinPower,

LostLoad\_Reserve2D,

LostLoad\_Reserve2U,

LostLoad\_RampUp,

LostLoad\_RampDown,

ShadowPrice,

status,

FixedCost,

StarUpCost,

ShutDownCost,

RampUpCost,

RampDownCost,

VariableCost,

OperationCost,

PowerTradeValue,

PowerTradeMin

;

$onorder

\* Exit here if the PrintResult option is set to 0:

$if not %PrintResults%==1 $exit

EXECUTE 'GDXXRW.EXE "%inputfilename%" O="Results.xlsx" Squeeze=N par=Technology rng=Technology!A1 rdim=2 cdim=0'

EXECUTE 'GDXXRW.EXE "%inputfilename%" O="Results.xlsx" Squeeze=N par=PowerCapacity rng=PowerCapacity!A1 rdim=1 cdim=0'

EXECUTE 'GDXXRW.EXE "%inputfilename%" O="Results.xlsx" Squeeze=N par=PowerInitial rng=PowerInitialA1 rdim=1 cdim=0'

EXECUTE 'GDXXRW.EXE "%inputfilename%" O="Results.xlsx" Squeeze=N par=RampDownMaximum rng=RampDownMaximum!A1 rdim=1 cdim=0'

EXECUTE 'GDXXRW.EXE "%inputfilename%" O="Results.xlsx" Squeeze=N par=RampShutDownMaximum rng=RampShutDownMaximum!A1 rdim=1 cdim=0'

EXECUTE 'GDXXRW.EXE "%inputfilename%" O="Results.xlsx" Squeeze=N par=RampStartUpMaximum rng=RampStartUpMaximum!A1 rdim=1 cdim=0'

EXECUTE 'GDXXRW.EXE "%inputfilename%" O="Results.xlsx" Squeeze=N par=RampUpMaximum rng=RampUpMaximum!A1 rdim=1 cdim=0'

EXECUTE 'GDXXRW.EXE "%inputfilename%" O="Results.xlsx" Squeeze=N par=TimeUpInitial rng=TimeUpInitial!A1 rdim=1 cdim=0'

EXECUTE 'GDXXRW.EXE "%inputfilename%" O="Results.xlsx" Squeeze=N par=TimeDownInitial rng=TimeDownInitial!A1 rdim=1 cdim=0'

EXECUTE 'GDXXRW.EXE "%inputfilename%" O="Results.xlsx" Squeeze=N par=TimeUpMinimum rng=TimeUpMinimum!A1 rdim=1 cdim=0'

EXECUTE 'GDXXRW.EXE "%inputfilename%" O="Results.xlsx" Squeeze=N par=TimeDownMinimum rng=TimeDownMinimum!A1 rdim=1 cdim=0'

EXECUTE 'GDXXRW.EXE "%inputfilename%" O="Results.xlsx" Squeeze=N par=Reserve rng=Reserve!A1 rdim=1 cdim=0'

EXECUTE 'GDXXRW.EXE "%inputfilename%" O="Results.xlsx" Squeeze=N par=LoadShedding rng=LoadShedding!A1 rdim=1 cdim=0'

EXECUTE 'GDXXRW.EXE "%inputfilename%" O="Results.xlsx" Squeeze=N par=FlowMaximum rng=FlowMaximum!A1 rdim=1 cdim=1'

EXECUTE 'GDXXRW.EXE "%inputfilename%" O="Results.xlsx" Squeeze=N par=AvailabilityFactor rng=AvailabilityFactor!A1 rdim=1 cdim=1'

EXECUTE 'GDXXRW.EXE "%inputfilename%" O="Results.xlsx" Squeeze=Y par=OutageFactor rng=OutageFactor!A1 rdim=1 cdim=1'

EXECUTE 'GDXXRW.EXE "%inputfilename%" O="Results.xlsx" Squeeze=N par=Demand rng=Demand!A1 rdim=2 cdim=1'

EXECUTE 'GDXXRW.EXE "%inputfilename%" O="Results.xlsx" Squeeze=N par=PartLoadMin rng=PartLoadMin!A1 rdim=1 cdim=0'

EXECUTE 'GDXXRW.EXE "Results.gdx" O="Results.xlsx" Squeeze=N var=ShedLoad rng=ShedLoad!A1 rdim=1 cdim=1'

EXECUTE 'GDXXRW.EXE "Results.gdx" O="Results.xlsx" Squeeze=N par=OutputCommitted rng=Committed!A1 rdim=1 cdim=1'

EXECUTE 'GDXXRW.EXE "Results.gdx" O="Results.xlsx" Squeeze=N par=OutputFlow rng=Flow!A1 rdim=1 cdim=1'

EXECUTE 'GDXXRW.EXE "Results.gdx" O="Results.xlsx" Squeeze=N par=OutputPower rng=Power!A5 epsout=0 rdim=1 cdim=1'

EXECUTE 'GDXXRW.EXE "Results.gdx" O="Results.xlsx" Squeeze=N par=OutputSystemCost rng=SystemCost!A1 rdim=1 cdim=0'

EXECUTE 'GDXXRW.EXE "Results.gdx" O="Results.xlsx" Squeeze=Y var=LostLoad\_MaxPower rng=LostLoad\_MaxPower!A1 rdim=1 cdim=1'

EXECUTE 'GDXXRW.EXE "Results.gdx" O="Results.xlsx" Squeeze=Y var=LostLoad\_MinPower rng=LostLoad\_MinPower!A1 rdim=1 cdim=1'

EXECUTE 'GDXXRW.EXE "Results.gdx" O="Results.xlsx" Squeeze=Y var=LostLoad\_Reserve2D rng=LostLoad\_Reserve2D!A1 rdim=1 cdim=1'

EXECUTE 'GDXXRW.EXE "Results.gdx" O="Results.xlsx" Squeeze=Y var=LostLoad\_Reserve2U rng=LostLoad\_Reserve2U!A1 rdim=1 cdim=1'

EXECUTE 'GDXXRW.EXE "Results.gdx" O="Results.xlsx" Squeeze=Y var=LostLoad\_RampUp rng=LostLoad\_RampUp!A1 rdim=1 cdim=1'

EXECUTE 'GDXXRW.EXE "Results.gdx" O="Results.xlsx" Squeeze=Y var=LostLoad\_RampDown rng=LostLoad\_RampDown!A1 rdim=1 cdim=1'

$exit

$Label DebugSection

$gdxin debug.gdx

$LOAD day

$LOAD PowerInitial\_dbg

$LOAD CommittedInitial\_dbg

$LOAD TimeDownInitial\_dbg

$LOAD TimeUpInitial\_dbg

;

PowerInitial(u) = PowerInitial\_dbg(u); CommittedInitial(u) = CommittedInitial\_dbg(u); TimeDownInitial(u) = TimeDownInitial\_dbg(u); TimeUpInitial(u) = TimeUpInitial\_dbg(u);

FirstHour = (day-1)\*24+1;

LastHour = min(card(h),FirstHour + (Config("RollingHorizon Length","day")+Config("RollingHorizon LookAhead","day")) \* 24 - 1);

LastKeptHour = LastHour - Config("RollingHorizon LookAhead","day") \* 24;

i(h) = no;

i(h)$(ord(h)>=firsthour and ord(h)<=lasthour)=yes;

TimeUpLeft\_initial(u)=min(card(i),(TimeUpMinimum(u)-TimeUpInitial(u))\*CommittedInitial(u));

TimeUpLeft\_JustStarted(u,i) = min(card(i)-ord(i)+1,TimeUpMinimum(u));

TimeDownLeft\_initial(u)=min(card(i),(TimeDownMinimum(u)-TimeDownInitial(u))\*(1-CommittedInitial(u)));

TimeDownLeft\_JustStopped(u,i) = min(card(i)-ord(i)+1,TimeDownMinimum(u));

$If %Verbose% == 1 Display TimeUpLeft\_initial,TimeUpLeft\_JustStarted,TimeDownLeft\_initial,TimeDownLeft\_JustStopped,TimeUpInitial,TimeDownInitial,PowerInitial,CommittedInitial,StorageFinalMin;

$If %LPFormulation% == 1 SOLVE UCM\_SIMPLE USING LP MINIMIZING SystemCostD;

$If not %LPFormulation% == 1 SOLVE UCM\_SIMPLE USING MIP MINIMIZING SystemCostD;

$If %LPFormulation% == 1 Display EQ\_Objective\_function.M, EQ\_CostRampUp.M, EQ\_CostRampDown.M, EQ\_Demand\_balance\_DA.M, EQ\_Power\_available.M, EQ\_Ramp\_up.M, EQ\_Ramp\_down.M, EQ\_Max\_RampUp1.M, EQ\_Max\_RampUp2.M,EQ\_Max\_RampDown1.M, EQ\_Max\_RampDown2.M, EQ\_Flow\_limits\_lower.M ;

$If not %LPFormulation% == 1 Display EQ\_Objective\_function.M, EQ\_CostStartUp.M, EQ\_CostShutDown.M, EQ\_Demand\_balance\_DA.M, EQ\_Power\_must\_run.M, EQ\_Power\_available.M, EQ\_Ramp\_up.M, EQ\_Ramp\_down.M, EQ\_Minimum\_time\_up\_A.M, EQ\_Minimum\_time\_up\_JustStarted.M, EQ\_Minimum\_time\_down\_A.M, EQ\_Minimum\_time\_down\_JustStopped.M, EQ\_Max\_RampUp1.M, EQ\_Max\_RampUp2.M, EQ\_Max\_RampDown1.M, EQ\_Max\_RampDown2.M, EQ\_Flow\_limits\_lower.M ;

display day,FirstHour,LastHour,LastKeptHour;

Display TimeUpLeft\_initial,TimeUpLeft\_JustStarted,TimeDownLeft\_initial,TimeDownLeft\_JustStopped,TimeUpInitial,TimeDownInitial,PowerInitial,CommittedInitial;

Display Flow.L,Power.L,Committed.L,ShedLoad.L,SystemCost.L,MaxRamp2U.L,MaxRamp2D.L,LostLoad\_MaxPower.L,LostLoad\_MinPower.L,LostLoad\_reserve2U.L,LostLoad\_reserve2D.L,LostLoad\_RampUP.L,LostLoad\_RampDown.L;