

PowerManager

1.0

Generated by Doxygen 1.8.1.2

Wed Mar 6 2013 13:02:23

Contents

1	PowerManager	1
1.1	Introduction	1
1.2	Licence	1
2	Documentation Page	3
2.1	First section	3
3	Data Type Index	5
3.1	Data Types List	5
4	File Index	7
4.1	File List	7
5	Data Type Documentation	9
5.1	interfaces::abortExecution Interface Reference	9
5.1.1	Detailed Description	9
5.1.2	Member Function/Subroutine Documentation	9
5.1.2.1	abortexecution	9
5.2	filetools::cFindEntry Interface Reference	10
5.2.1	Detailed Description	10
5.2.2	Member Function/Subroutine Documentation	10
5.2.2.1	cfindentry	10
5.3	filetools::cMatrixRead Interface Reference	10
5.3.1	Detailed Description	11
5.3.2	Member Function/Subroutine Documentation	11
5.3.2.1	cmatrixread	11
5.4	cmdvar Module Reference	11
5.4.1	Detailed Description	12
5.4.2	Member Data Documentation	12
5.4.2.1	debug	12
5.4.2.2	out	12
5.4.2.3	silent	12
5.4.2.4	verb	12

5.4.2.5	vsilent	12
5.5	cmdvarwr Module Reference	12
5.5.1	Detailed Description	13
5.5.2	Member Data Documentation	13
5.5.2.1	folder	13
5.6	command Module Reference	13
5.6.1	Detailed Description	14
5.6.2	Member Function/Subroutine Documentation	14
5.6.2.1	cmdvarwrite	14
5.6.3	Member Data Documentation	14
5.6.3.1	cartella	14
5.6.3.2	protected	14
5.7	graphtools::constraints Interface Reference	14
5.7.1	Detailed Description	14
5.7.2	Constructor & Destructor Documentation	15
5.7.2.1	constraints	15
5.8	heuristics::constraints Interface Reference	15
5.8.1	Detailed Description	15
5.8.2	Constructor & Destructor Documentation	15
5.8.2.1	constraints	15
5.9	filetools::dFindEntry Interface Reference	16
5.9.1	Detailed Description	16
5.9.2	Member Function/Subroutine Documentation	16
5.9.2.1	dfindentry	16
5.10	filetools::dMatrixRead Interface Reference	16
5.10.1	Detailed Description	17
5.10.2	Member Function/Subroutine Documentation	17
5.10.2.1	dmatrixread	17
5.11	economy Module Reference	17
5.11.1	Detailed Description	18
5.11.2	Member Function/Subroutine Documentation	18
5.11.2.1	chrev	18
5.11.2.2	currcost	18
5.11.2.3	elrev	19
5.11.2.4	firecost	20
5.11.2.5	fuelcost	20
5.11.2.6	grideconomy	21
5.11.2.7	maintenancecost	21
5.11.2.8	threv	21
5.12	energy Module Reference	22

5.12.1 Detailed Description	23
5.12.2 Member Function/Subroutine Documentation	23
5.12.2.1 chprod	23
5.12.2.2 cogthprod	23
5.12.2.3 elprod	24
5.12.2.4 elselfcons	24
5.12.2.5 energyexhaust	25
5.12.2.6 energyinput	25
5.12.2.7 fuelcons	25
5.12.2.8 limitrecovery	26
5.12.2.9 thprod	26
5.12.2.10 thselfcons	26
5.13 euristics Module Reference	27
5.13.1 Detailed Description	27
5.13.2 Member Function/Subroutine Documentation	27
5.13.2.1 chredundant	27
5.13.2.2 thredundant	28
5.14 filetools Module Reference	28
5.14.1 Detailed Description	29
5.15 filetools::findEntry Interface Reference	29
5.15.1 Detailed Description	29
5.15.2 Member Function/Subroutine Documentation	30
5.15.2.1 findentry	30
5.16 globalresults Module Reference	30
5.16.1 Detailed Description	31
5.16.2 Member Function/Subroutine Documentation	31
5.16.2.1 globchrev	31
5.16.2.2 globcost	31
5.16.2.3 globelrev	31
5.16.2.4 globfirecost	32
5.16.2.5 globfuelcost	33
5.16.2.6 globmaintcost	33
5.16.2.7 globprofit	34
5.16.2.8 globrevenues	34
5.16.2.9 globthrev	34
5.17 graphtools Module Reference	35
5.17.1 Detailed Description	35
5.17.2 Member Function/Subroutine Documentation	36
5.17.2.1 allcombin	36
5.17.2.2 grapharcs	36

5.17.2.3	graphpoints	36
5.17.2.4	minpathtopobw	37
5.17.2.5	minpathtopofw	37
5.17.2.6	uptimecalc	37
5.18	filetools::hCount Interface Reference	38
5.18.1	Detailed Description	38
5.18.2	Member Function/Subroutine Documentation	38
5.18.2.1	hcount	38
5.19	filetools::iFindEntry Interface Reference	38
5.19.1	Detailed Description	39
5.19.2	Member Function/Subroutine Documentation	39
5.19.2.1	ifindentry	39
5.20	filetools::iMatrixRead Interface Reference	39
5.20.1	Detailed Description	39
5.20.2	Member Function/Subroutine Documentation	39
5.20.2.1	imatrixread	39
5.21	inputvar Module Reference	40
5.21.1	Detailed Description	43
5.21.2	Member Data Documentation	43
5.21.2.1	altcorrb	43
5.21.2.2	altcorrc	43
5.21.2.3	altcorrt	44
5.21.2.4	altitude	44
5.21.2.5	cch	44
5.21.2.6	cel	44
5.21.2.7	cth	44
5.21.2.8	downtime0	44
5.21.2.9	dt1	44
5.21.2.10	etab	44
5.21.2.11	etac	44
5.21.2.12	etacht	44
5.21.2.13	etaelt	44
5.21.2.14	etatht	44
5.21.2.15	firecostb	45
5.21.2.16	firecostc	45
5.21.2.17	firecostt	45
5.21.2.18	fuelcostb	45
5.21.2.19	fuelcostt	45
5.21.2.20	fuellhvb	45
5.21.2.21	fuellhvt	45

5.21.2.22 global	45
5.21.2.23 gridbuycost	45
5.21.2.24 gridconnection	45
5.21.2.25 gridsellcost	45
5.21.2.26 ich	45
5.21.2.27 ichp	46
5.21.2.28 ideg	46
5.21.2.29 iel	46
5.21.2.30 ielp	46
5.21.2.31 ith	46
5.21.2.32 ithp	46
5.21.2.33 itime	46
5.21.2.34 lifeb	46
5.21.2.35 lifet	46
5.21.2.36 maintcostb	46
5.21.2.37 maintcostc	46
5.21.2.38 maintcostt	46
5.21.2.39 method	47
5.21.2.40 mindowntimeb	47
5.21.2.41 mindowntimec	47
5.21.2.42 mindowntimet	47
5.21.2.43 minuptimeb	47
5.21.2.44 minuptimec	47
5.21.2.45 minuptimet	47
5.21.2.46 nacb	47
5.21.2.47 nacc	47
5.21.2.48 nact	47
5.21.2.49 nboi	47
5.21.2.50 nchi	47
5.21.2.51 netab	48
5.21.2.52 netac	48
5.21.2.53 netacht	48
5.21.2.54 netaelt	48
5.21.2.55 netatht	48
5.21.2.56 nld	48
5.21.2.57 nload	48
5.21.2.58 nlp	48
5.21.2.59 npcb	48
5.21.2.60 npcc	48
5.21.2.61 npct	48

5.21.2.62 nsizec	48
5.21.2.63 nspb	49
5.21.2.64 nspc	49
5.21.2.65 nspt	49
5.21.2.66 ntc b	49
5.21.2.67 ntcc	49
5.21.2.68 ntct	49
5.21.2.69 ntime	49
5.21.2.70 ntimes	49
5.21.2.71 ntrig	49
5.21.2.72 obj	49
5.21.2.73 pamb	49
5.21.2.74 pmaxb	49
5.21.2.75 pmaxc	50
5.21.2.76 pmaxt	50
5.21.2.77 prescorrb	50
5.21.2.78 prescorrc	50
5.21.2.79 prescorrt	50
5.21.2.80 spb	50
5.21.2.81 spc	50
5.21.2.82 spt	50
5.21.2.83 startpoint	50
5.21.2.84 tamb	50
5.21.2.85 tecb	50
5.21.2.86 tecc	50
5.21.2.87 tect	51
5.21.2.88 tempcorrb	51
5.21.2.89 tempcorrc	51
5.21.2.90 tempcorrt	51
5.21.2.91 time	51
5.21.2.92 uch	51
5.21.2.93 uel	51
5.21.2.94 uptime0	51
5.21.2.95 useeuristics	51
5.21.2.96 uth	51
5.21.2.97 writeboi	51
5.21.2.98 writetechi	51
5.21.2.99 writetechillingrev	52
5.21.2.100writecosts	52
5.21.2.101writedemand	52

5.21.2.102	writeefficiency	52
5.21.2.103	writeelectricrev	52
5.21.2.104	writeenergy	52
5.21.2.105	writeinput	52
5.21.2.106	writepower	52
5.21.2.107	writethermalrev	52
5.21.2.108	writetrig	52
5.22	interfaces Module Reference	53
5.22.1	Detailed Description	53
5.23	mathtools::interpolation Interface Reference	53
5.23.1	Detailed Description	54
5.23.2	Constructor & Destructor Documentation	54
5.23.2.1	interpolation	54
5.24	mathtools Module Reference	54
5.24.1	Detailed Description	54
5.24.2	Member Function/Subroutine Documentation	55
5.24.2.1	locaterow	55
5.25	filetools::matrixRead Interface Reference	55
5.25.1	Detailed Description	55
5.25.2	Member Function/Subroutine Documentation	55
5.25.2.1	matrixread	55
5.26	myarithmetic Module Reference	55
5.26.1	Detailed Description	56
5.26.2	Member Function/Subroutine Documentation	56
5.26.2.1	inan	56
5.26.2.2	isnan	57
5.26.2.3	rnan	57
5.27	graphtools::objFunction Interface Reference	57
5.27.1	Detailed Description	58
5.27.2	Member Function/Subroutine Documentation	58
5.27.2.1	objfunction	58
5.28	interfaces::performances Interface Reference	58
5.28.1	Detailed Description	58
5.28.2	Constructor & Destructor Documentation	58
5.28.2.1	performances	58
5.29	plantvar Module Reference	59
5.29.1	Detailed Description	60
5.29.2	Member Data Documentation	60
5.29.2.1	cf	60
5.29.2.2	cr	60

5.29.2.3	cref	61
5.29.2.4	dt	61
5.29.2.5	envcorr	61
5.29.2.6	esource	61
5.29.2.7	etab_	61
5.29.2.8	etac_	61
5.29.2.9	etach	61
5.29.2.10	etacht_	61
5.29.2.11	etael	61
5.29.2.12	etaelt_	61
5.29.2.13	etath	61
5.29.2.14	etatht_	61
5.29.2.15	ib	62
5.29.2.16	ic	62
5.29.2.17	ie	62
5.29.2.18	is	62
5.29.2.19	it	62
5.29.2.20	lhv	62
5.29.2.21	mindowntime	62
5.29.2.22	minuptime	62
5.29.2.23	nm	62
5.29.2.24	nsp	62
5.29.2.25	nsptot	62
5.29.2.26	ntv	62
5.29.2.27	oemcost	63
5.29.2.28	onoffcost	63
5.29.2.29	pes	63
5.29.2.30	pmax	63
5.29.2.31	sp	63
5.29.2.32	tec	63
5.29.2.33	timevinc	63
5.30	filetools::readKeyword Interface Reference	63
5.30.1	Detailed Description	64
5.30.2	Member Function/Subroutine Documentation	64
5.30.2.1	readkeyword	64
5.31	filetools::rewUnit Interface Reference	64
5.31.1	Detailed Description	64
5.31.2	Member Function/Subroutine Documentation	64
5.31.2.1	rewunit	64
5.32	shared Module Reference	65

5.32.1 Detailed Description	65
5.32.2 Member Data Documentation	65
5.32.2.1 prec	65
5.32.2.2 rval	65
5.32.2.3 zero	65
5.33 graphtools::timeConstr Interface Reference	66
5.33.1 Detailed Description	66
5.33.2 Member Function/Subroutine Documentation	66
5.33.2.1 timeconstr	66
5.34 filetools::vCount Interface Reference	66
5.34.1 Detailed Description	67
5.34.2 Member Function/Subroutine Documentation	67
5.34.2.1 vcount	67
5.35 interfaces::warning Interface Reference	67
5.35.1 Detailed Description	67
5.35.2 Constructor & Destructor Documentation	67
5.35.2.1 warning	67
6 File Documentation	69
6.1 /home/Codici/Blink/PowerManager/src/abortExecution.f90 File Reference	69
6.1.1 Detailed Description	69
6.1.2 Function/Subroutine Documentation	69
6.1.2.1 abortexecution	69
6.2 /home/Codici/Blink/PowerManager/src/aiuto.f90 File Reference	70
6.2.1 Detailed Description	70
6.2.2 Function/Subroutine Documentation	70
6.2.2.1 aiuto	70
6.3 /home/Codici/Blink/PowerManager/src/allocateVar.f90 File Reference	70
6.3.1 Detailed Description	70
6.3.2 Function/Subroutine Documentation	71
6.3.2.1 allocatevar	71
6.4 /home/Codici/Blink/PowerManager/src/buildPlant.f90 File Reference	71
6.4.1 Detailed Description	71
6.4.2 Function/Subroutine Documentation	71
6.4.2.1 buildplant	71
6.5 /home/Codici/Blink/PowerManager/src/checkPlant.f90 File Reference	72
6.5.1 Detailed Description	72
6.5.2 Function/Subroutine Documentation	72
6.5.2.1 checkplant	72
6.6 /home/Codici/Blink/PowerManager/src/cmdVar.f90 File Reference	72

6.6.1	Detailed Description	72
6.7	/home/Codici/Blink/PowerManager/src/cmdVarWr.f90 File Reference	73
6.8	/home/Codici/Blink/PowerManager/src/command.f90 File Reference	73
6.8.1	Detailed Description	73
6.9	/home/Codici/Blink/PowerManager/src/commandline.f90 File Reference	73
6.9.1	Detailed Description	73
6.9.2	Function/Subroutine Documentation	73
6.9.2.1	commandline	73
6.10	/home/Codici/Blink/PowerManager/src/constraints.f90 File Reference	74
6.10.1	Detailed Description	74
6.10.2	Function/Subroutine Documentation	74
6.10.2.1	constraints	74
6.10.2.2	timeconstr	74
6.11	/home/Codici/Blink/PowerManager/src/deallocateVar.f90 File Reference	74
6.11.1	Function/Subroutine Documentation	75
6.11.1.1	deallocatevar	75
6.12	/home/Codici/Blink/PowerManager/src/economy.f90 File Reference	75
6.12.1	Detailed Description	75
6.13	/home/Codici/Blink/PowerManager/src/endExecution.f90 File Reference	75
6.13.1	Detailed Description	75
6.13.2	Function/Subroutine Documentation	76
6.13.2.1	endexecution	76
6.14	/home/Codici/Blink/PowerManager/src/energy.f90 File Reference	76
6.14.1	Detailed Description	76
6.15	/home/Codici/Blink/PowerManager/src/euristics.f90 File Reference	76
6.15.1	Detailed Description	76
6.16	/home/Codici/Blink/PowerManager/src/fileTools.f90 File Reference	77
6.16.1	Detailed Description	77
6.17	/home/Codici/Blink/PowerManager/src/findEntry2.f90 File Reference	77
6.17.1	Detailed Description	77
6.17.2	Function/Subroutine Documentation	78
6.17.2.1	cfindentry	78
6.17.2.2	dfindentry	78
6.17.2.3	findentry	79
6.17.2.4	ifindentry	79
6.18	/home/Codici/Blink/PowerManager/src/globalResults.f90 File Reference	80
6.19	/home/Codici/Blink/PowerManager/src/graphTools.f90 File Reference	80
6.19.1	Detailed Description	80
6.20	/home/Codici/Blink/PowerManager/src/hCount.f90 File Reference	80
6.20.1	Detailed Description	80

6.20.2	Function/Subroutine Documentation	81
6.20.2.1	hcount	81
6.21	/home/Codici/Blink/PowerManager/src/inputVar.f90 File Reference	81
6.21.1	Detailed Description	81
6.22	/home/Codici/Blink/PowerManager/src/interfaces.f90 File Reference	81
6.22.1	Detailed Description	82
6.23	/home/Codici/Blink/PowerManager/src/interpolation.f90 File Reference	82
6.23.1	Detailed Description	82
6.23.2	Function/Subroutine Documentation	82
6.23.2.1	interpolation	82
6.24	/home/Codici/Blink/PowerManager/src/main.f90 File Reference	82
6.24.1	Detailed Description	83
6.24.2	Function/Subroutine Documentation	83
6.24.2.1	main	83
6.25	/home/Codici/Blink/PowerManager/src/manual.f90 File Reference	83
6.26	/home/Codici/Blink/PowerManager/src/mathTools.f90 File Reference	83
6.26.1	Detailed Description	83
6.27	/home/Codici/Blink/PowerManager/src/matrixRead2.f90 File Reference	84
6.27.1	Detailed Description	84
6.27.2	Function/Subroutine Documentation	84
6.27.2.1	cmatrixread	84
6.27.2.2	dmatrixread	85
6.27.2.3	imatrixread	86
6.27.2.4	matrixread	87
6.28	/home/Codici/Blink/PowerManager/src/myArithmetic.f90 File Reference	88
6.28.1	Detailed Description	88
6.29	/home/Codici/Blink/PowerManager/src/objFunction.f90 File Reference	88
6.29.1	Detailed Description	88
6.29.2	Function/Subroutine Documentation	88
6.29.2.1	objfunction	88
6.30	/home/Codici/Blink/PowerManager/src/openUnit.f90 File Reference	89
6.30.1	Detailed Description	89
6.30.2	Function/Subroutine Documentation	89
6.30.2.1	openunit	89
6.31	/home/Codici/Blink/PowerManager/src/output.f90 File Reference	89
6.31.1	Detailed Description	89
6.31.2	Function/Subroutine Documentation	90
6.31.2.1	output	90
6.31.2.2	preparefile	90
6.32	/home/Codici/Blink/PowerManager/src/performances.f90 File Reference	90

6.32.1 Detailed Description	90
6.32.2 Function/Subroutine Documentation	90
6.32.2.1 performances	90
6.33 /home/Codici/Blink/PowerManager/src/plantVar.f90 File Reference	90
6.33.1 Detailed Description	91
6.34 /home/Codici/Blink/PowerManager/src/prototipo.f90 File Reference	91
6.34.1 Detailed Description	91
6.34.2 Function/Subroutine Documentation	91
6.34.2.1 implicit	91
6.35 /home/Codici/Blink/PowerManager/src/readBoiler.f90 File Reference	91
6.35.1 Detailed Description	91
6.35.2 Function/Subroutine Documentation	92
6.35.2.1 readboiler	92
6.36 /home/Codici/Blink/PowerManager/src/readChiller.f90 File Reference	92
6.36.1 Detailed Description	92
6.36.2 Function/Subroutine Documentation	93
6.36.2.1 readchillers	93
6.37 /home/Codici/Blink/PowerManager/src/readEnv.f90 File Reference	93
6.37.1 Function/Subroutine Documentation	93
6.37.1.1 readenv	93
6.38 /home/Codici/Blink/PowerManager/src/readGeneral.f90 File Reference	94
6.38.1 Detailed Description	94
6.38.2 Function/Subroutine Documentation	94
6.38.2.1 readgeneral	94
6.39 /home/Codici/Blink/PowerManager/src/readKeyword.f90 File Reference	95
6.39.1 Detailed Description	95
6.39.2 Function/Subroutine Documentation	95
6.39.2.1 readkeyword	95
6.40 /home/Codici/Blink/PowerManager/src/readLoad.f90 File Reference	96
6.40.1 Detailed Description	96
6.40.2 Function/Subroutine Documentation	96
6.40.2.1 readloads	96
6.41 /home/Codici/Blink/PowerManager/src/readSteamPlant.f90 File Reference	97
6.41.1 Function/Subroutine Documentation	97
6.41.1.1 readtrigen	97
6.42 /home/Codici/Blink/PowerManager/src/readTrigen.f90 File Reference	98
6.42.1 Detailed Description	98
6.42.2 Function/Subroutine Documentation	98
6.42.2.1 readtrigen	98
6.43 /home/Codici/Blink/PowerManager/src/rewUnit.f90 File Reference	99

6.43.1 Detailed Description	99
6.43.2 Function/Subroutine Documentation	99
6.43.2.1 rewunit	99
6.44 /home/Codici/Blink/PowerManager/src/shared.f90 File Reference	99
6.44.1 Detailed Description	100
6.45 /home/Codici/Blink/PowerManager/src/vCount.f90 File Reference	100
6.45.1 Detailed Description	100
6.45.2 Function/Subroutine Documentation	100
6.45.2.1 vcount	100
6.46 /home/Codici/Blink/PowerManager/src/warning.f90 File Reference	101
6.46.1 Detailed Description	101
6.46.2 Function/Subroutine Documentation	101
6.46.2.1 warning	101

Chapter 1

PowerManager

1.1 Introduction

PowerManager is a free software for optimization of energy dispatching, with particular regards to distributed generation.

PowerManager simulates the energy fluxes between the components of the plant and between the plant and energy loads and eventually the electric grid. The optimal plant state is determined using dynamic programming, in order to minimize the total costs (or maximize the profits), including, fuel, maintenance, plant ignition, and eventual energy purchase costs, and energy selling revenues. Constraints on the minimum duration of the operation intervals of the plant are also considered.

1.2 Licence

PowerManager is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

PowerManager is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with PowerManager; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Chapter 2

Documentation Page

This is a documentation page

2.1 First section

this is the first section

Chapter 3

Data Type Index

3.1 Data Types List

Here are the data types with brief descriptions:

interfaces::abortExecution	9
filetools::cFindEntry	10
filetools::cMatrixRead	10
cmdvar	
Collects the variable read from command line	11
cmdvarwr	
Collects the variable read from command line	12
command	13
graphtools::constraints	14
heuristics::constraints	15
filetools::dFindEntry	16
filetools::dMatrixRead	16
economy	17
energy	
Module for energy calculations	22
heuristics	
Module that contains the function to apply an euristics to the graph	27
filetools	
Interfaces of procedures to read from file	28
filetools::findEntry	29
globalresults	30
graphtools	35
filetools::hCount	38
filetools::iFindEntry	38
filetools::iMatrixRead	39
inputvar	
Input variables collection	40
interfaces	53
mathtools::interpolation	53
mathtools	
Collection of interfaces for basic mathematical tools	54
filetools::matrixRead	55
myarithmetic	
Creates and detects NaN	55
graphtools::objFunction	57
interfaces::performances	58
plantvar	
Collection of variables relative to the power plant structure	59

filetools::readKeyword	63
filetools::rewUnit	64
shared	65
graphtools::timeConstr	66
filetools::vCount	66
interfaces::warning	67

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

/home/Codici/Blink/PowerManager/src/ abortExecution.f90	69
Terminates the execution in case of error	
/home/Codici/Blink/PowerManager/src/ aiuto.f90	70
Prints a very short help	
/home/Codici/Blink/PowerManager/src/ allocateVar.f90	70
Variable allocation	
/home/Codici/Blink/PowerManager/src/ buildPlant.f90	71
Collects all the informations relative to the power plant	
/home/Codici/Blink/PowerManager/src/ checkPlant.f90	72
Check the power plant coherence with the energy demand	
/home/Codici/Blink/PowerManager/src/ cmdVar.f90	72
Collects the variable read from command line	
/home/Codici/Blink/PowerManager/src/ cmdVarWr.f90	73
/home/Codici/Blink/PowerManager/src/ command.f90	73
Reads the commad line	
/home/Codici/Blink/PowerManager/src/ commandline.f90	73
Reads the commad line	
/home/Codici/Blink/PowerManager/src/ constraints.f90	74
Static constraints	
/home/Codici/Blink/PowerManager/src/ deallocateVar.f90	74
/home/Codici/Blink/PowerManager/src/ economy.f90	75
Costs and revenues calulation prodedures	
/home/Codici/Blink/PowerManager/src/ endExecution.f90	75
Normal termination of the execution	
/home/Codici/Blink/PowerManager/src/ energy.f90	76
Collection of function for energy flow calculation	
/home/Codici/Blink/PowerManager/src/ heuristics.f90	76
Module that contains the function to apply an euristics to the graph	
/home/Codici/Blink/PowerManager/src/ fileTools.f90	77
Collection of proceture interfaces useful to read from files	
/home/Codici/Blink/PowerManager/src/ findEntry2.f90	77
Collection of procetures to find a specific entry inside a file	
/home/Codici/Blink/PowerManager/src/ globalResults.f90	80
/home/Codici/Blink/PowerManager/src/ graphTools.f90	80
Graph construction and minumum path	
/home/Codici/Blink/PowerManager/src/ hCount.f90	80
Count the number of elements of a vector in a text file	

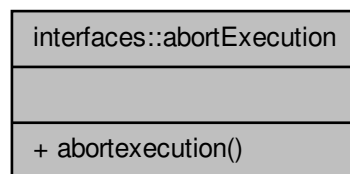
/home/Codici/Blink/PowerManager/src/ inputVar.f90	
Input variables collection	81
/home/Codici/Blink/PowerManager/src/ interfaces.f90	
Collection of general pourpose interfaces	81
/home/Codici/Blink/PowerManager/src/ interpolation.f90	
Remaps a discrete scalar field on a given 1d grid	82
/home/Codici/Blink/PowerManager/src/ main.f90	
This is the main driver	82
/home/Codici/Blink/PowerManager/src/ manual.f90	83
/home/Codici/Blink/PowerManager/src/ mathTools.f90	
Collection of interfaces for basic mathematical tools	83
/home/Codici/Blink/PowerManager/src/ matrixRead2.f90	
Reads 2D array of values from a file	84
/home/Codici/Blink/PowerManager/src/ myArithmetic.f90	
Creates and detects NaN	88
/home/Codici/Blink/PowerManager/src/ objFunction.f90	
Objective function	88
/home/Codici/Blink/PowerManager/src/ openUnit.f90	
Checks file presence and opens the unit	89
/home/Codici/Blink/PowerManager/src/ output.f90	
Print the output	89
/home/Codici/Blink/PowerManager/src/ performances.f90	
Calculates the performances of a single equipment	90
/home/Codici/Blink/PowerManager/src/ plantVar.f90	
Collection of variables relative to the power plant structure	90
/home/Codici/Blink/PowerManager/src/ prototipo.f90	
File prototype	91
/home/Codici/Blink/PowerManager/src/ readBoiler.f90	
Reads Boiler.inp file	91
/home/Codici/Blink/PowerManager/src/ readChiller.f90	
Reads Chiller.inp file	92
/home/Codici/Blink/PowerManager/src/ readEnv.f90	93
/home/Codici/Blink/PowerManager/src/ readGeneral.f90	
Reads General.inp file	94
/home/Codici/Blink/PowerManager/src/ readKeyword.f90	
Reads a line of a text file	95
/home/Codici/Blink/PowerManager/src/ readLoad.f90	
Reads Load.inp file	96
/home/Codici/Blink/PowerManager/src/ readSteamPlant.f90	97
/home/Codici/Blink/PowerManager/src/ readTrigen.f90	
Reads Trigeration.inp file	98
/home/Codici/Blink/PowerManager/src/ rewUnit.f90	
Rewind a unit	99
/home/Codici/Blink/PowerManager/src/ shared.f90	
File prototype	99
/home/Codici/Blink/PowerManager/src/ vCount.f90	
Counts the lines of a text matrix	100
/home/Codici/Blink/PowerManager/src/ warning.f90	
Prints warnings to standard output	101

Chapter 5

Data Type Documentation

5.1 interfaces::abortExecution Interface Reference

Collaboration diagram for interfaces::abortExecution:



Public Member Functions

- subroutine [abortexecution](#) (*i*, *j*, *line*, *word*, *r1*, *r2*, *iVec*)

5.1.1 Detailed Description

Definition at line 38 of file `interfaces.f90`.

5.1.2 Member Function/Subroutine Documentation

- 5.1.2.1 subroutine `interfaces::abortExecution::abortexecution` (*integer*, intent(in), optional *i*, *integer*, intent(in), optional *j*, *integer*, intent(in), optional *line*, *character*(len=*), intent(in), optional *word*, *real*(kind = *prec*), intent(in), optional *r1*, *real*(kind = *prec*), intent(in), optional *r2*, *real*(kind = *prec*), *dimension*(*), intent(in), optional *iVec*)

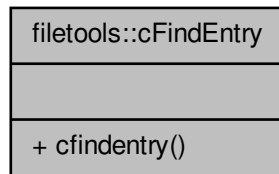
Definition at line 38 of file `interfaces.f90`.

The documentation for this interface was generated from the following file:

- `/home/Codici/Blink/PowerManager/src/interfaces.f90`

5.2 filetools::cFindEntry Interface Reference

Collaboration diagram for filetools::cFindEntry:



Public Member Functions

- subroutine [cfindentry](#) (entry, n, theUnit, rew, valore, isPresent, nRow)

5.2.1 Detailed Description

Definition at line 139 of file fileTools.f90.

5.2.2 Member Function/Subroutine Documentation

5.2.2.1 subroutine filetools::cFindEntry::cfindentry (character(len=*), intent(in) *entry*, integer, intent(in) *n*, integer, intent(in) *theUnit*, logical, intent(in), optional *rew*, character(len=100), dimension(n), intent(out), optional *valore*, logical, intent(out), optional *isPresent*, integer, intent(out), optional *nRow*)

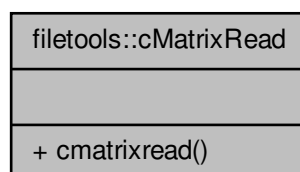
Definition at line 139 of file fileTools.f90.

The documentation for this interface was generated from the following file:

- [/home/Codici/Blink/PowerManager/src/fileTools.f90](#)

5.3 filetools::cMatrixRead Interface Reference

Collaboration diagram for filetools::cMatrixRead:



Public Member Functions

- `character(len=100) function,`
`dimension(nline, ncol) cmatrixread (theUnit, nline, ncol, first_, last_)`

5.3.1 Detailed Description

Definition at line 91 of file `fileTools.f90`.

5.3.2 Member Function/Subroutine Documentation

5.3.2.1 `character(len=100) function, dimension(nline,ncol) filetools::cMatrixRead::cmatrixread (integer, intent(in) theUnit, integer, intent(in) nline, integer, intent(in) ncol, character(len=1), optional first_, character(len=1), optional last_)`

Definition at line 91 of file `fileTools.f90`.

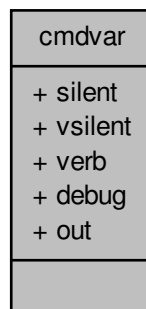
The documentation for this interface was generated from the following file:

- `/home/Codici/Blink/PowerManager/src/fileTools.f90`

5.4 cmdvar Module Reference

Collects the variable read from command line.

Collaboration diagram for `cmdvar`:



Public Attributes

- logical `silent`
Controls the input to the screen.
- logical `vsilent`
- logical `verb`
- logical `debug`
- `character(len=100) out`

5.4.1 Detailed Description

Collects the variable read from command line.

Author

Andrea Facci

Definition at line 39 of file cmdVar.f90.

5.4.2 Member Data Documentation

5.4.2.1 logical cmdvar::debug

Definition at line 42 of file cmdVar.f90.

5.4.2.2 character(len=100) cmdvar::out

Definition at line 43 of file cmdVar.f90.

5.4.2.3 logical cmdvar::silent

Definition at line 42 of file cmdVar.f90.

5.4.2.4 logical cmdvar::verb

Definition at line 42 of file cmdVar.f90.

5.4.2.5 logical cmdvar::vsilent

Definition at line 42 of file cmdVar.f90.

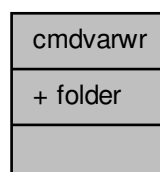
The documentation for this module was generated from the following file:

- /home/Codici/Blink/PowerManager/src/[cmdVar.f90](#)

5.5 cmdvarwr Module Reference

Collects the variable read from command line.

Collaboration diagram for cmdvarwr:



Public Attributes

- character(len=100) [folder](#)
Controls the input to the screen.

5.5.1 Detailed Description

Collects the variable read from command line.

Author

Andrea Facci

Definition at line 39 of file cmdVarWr.f90.

5.5.2 Member Data Documentation

5.5.2.1 character(len=100) cmdvarwr::folder

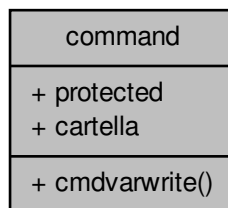
Definition at line 42 of file cmdVarWr.f90.

The documentation for this module was generated from the following file:

- /home/Codici/Blink/PowerManager/src/[cmdVarWr.f90](#)

5.6 command Module Reference

Collaboration diagram for command:



Public Member Functions

- subroutine [cmdvarwrite](#)

Public Attributes

- character(len=100) [protected](#)
- character(len=100) [cartella](#)

5.6.1 Detailed Description

Definition at line 35 of file command.f90.

5.6.2 Member Function/Subroutine Documentation

5.6.2.1 subroutine `command::cmdvarwrite ()`

Definition at line 41 of file command.f90.

5.6.3 Member Data Documentation

5.6.3.1 `character(len=100) command::cartella`

Definition at line 37 of file command.f90.

5.6.3.2 `character(len=100) command::protected`

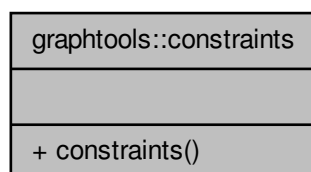
Definition at line 37 of file command.f90.

The documentation for this module was generated from the following file:

- [/home/Codici/Blink/PowerManager/src/command.f90](#)

5.7 `graphtools::constraints` Interface Reference

Collaboration diagram for `graphtools::constraints`:



Public Member Functions

- logical function [constraints](#) (c, t)

5.7.1 Detailed Description

Definition at line 70 of file graphTools.f90.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 logical function `graphTools::constraints::constraints` (`integer`, `dimension(nm)`, `intent(in) c`, `integer`, `intent(in) t`)

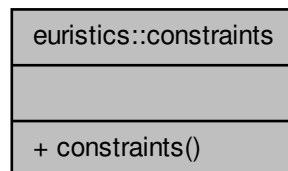
Definition at line 70 of file `graphTools.f90`.

The documentation for this interface was generated from the following file:

- `/home/Codici/Blink/PowerManager/src/graphTools.f90`

5.8 `heuristics::constraints` Interface Reference

Collaboration diagram for `heuristics::constraints`:



Public Member Functions

- logical function `constraints` (`c`, `t`)

5.8.1 Detailed Description

Definition at line 47 of file `heuristics.f90`.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 logical function `heuristics::constraints::constraints` (`integer`, `dimension(nm)`, `intent(in) c`, `integer`, `intent(in) t`)

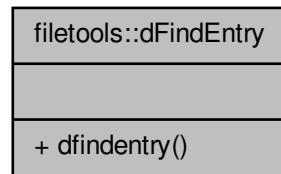
Definition at line 47 of file `heuristics.f90`.

The documentation for this interface was generated from the following file:

- `/home/Codici/Blink/PowerManager/src/heuristics.f90`

5.9 filetools::dFindEntry Interface Reference

Collaboration diagram for filetools::dFindEntry:



Public Member Functions

- subroutine [dfindentry](#) (entry, n, theUnit, rew, valore, isPresent, nRow)

5.9.1 Detailed Description

Definition at line 125 of file fileTools.f90.

5.9.2 Member Function/Subroutine Documentation

5.9.2.1 subroutine filetools::dFindEntry::dfindentry (character(len=*) intent(in) *entry*, integer intent(in) *n*, integer intent(in) *theUnit*, logical intent(in), optional *rew*, real(kind = prec), dimension(n), intent(out), optional *valore*, logical intent(out), optional *isPresent*, integer intent(out), optional *nRow*)

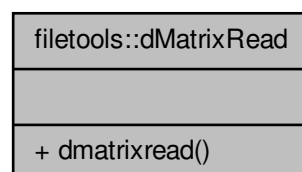
Definition at line 125 of file fileTools.f90.

The documentation for this interface was generated from the following file:

- [/home/Codici/Blink/PowerManager/src/fileTools.f90](#)

5.10 filetools::dMatrixRead Interface Reference

Collaboration diagram for filetools::dMatrixRead:



Public Member Functions

- `real(kind=prec) function,`
`dimension(nline, ncol) dmatrixread (theUnit, nline, ncol, first_, last_)`

5.10.1 Detailed Description

Definition at line 71 of file `fileTools.f90`.

5.10.2 Member Function/Subroutine Documentation

- 5.10.2.1 `real(kind = prec) function, dimension(nline,ncol) filetools::dMatrixRead::dmatrixread (integer, intent(in) theUnit, integer, intent(in) nline, integer, intent(in) ncol, character(len=1), optional first_, character(len=1), optional last_)`

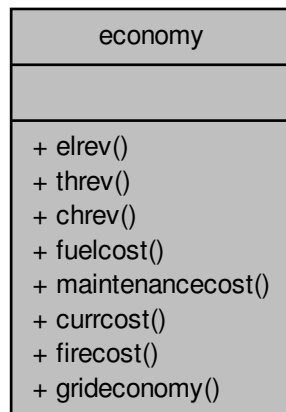
Definition at line 71 of file `fileTools.f90`.

The documentation for this interface was generated from the following file:

- `/home/Codici/Blink/PowerManager/src/fileTools.f90`

5.11 economy Module Reference

Collaboration diagram for economy:



Public Member Functions

- `real(kind=prec) function elrev (c, t)`
Electric energy revenues.
- `real(kind=prec) function threv (t)`
thermal energy revenues.
- `real(kind=prec) function chrev (t)`
Chilling energy revenues.

- real(kind=prec) function `fuelcost` (c, t)
Calculates the costs to buy the fuel.
- real(kind=prec) function `maintenancecost` (c, t)
Calculates the mintenance costs.
- real(kind=prec) function `currcost` (c, t)
Calculates the profit during a time step.
- real(kind=prec) function `firecost` (cNew, cOld)
Calculates the lighting cost.
- real(kind=prec) function,
dimension(2) `grideconomy` (c, t)

5.11.1 Detailed Description

Author

This module contains the definition of all the procedures that perform economic calculations for a give set-point and time-step, that are, fuel costs, O&M costs, and the revenues from thermal, electric, and chilling, energy selling.

Author

Definition at line 44 of file `economy.f90`.

5.11.2 Member Function/Subroutine Documentation

5.11.2.1 real(kind = prec) function `economy::chrev` (integer, intent(in) t)

Calculates the revenues (in euro or any other currency according to the one used in the input) from chilling energy selling to the various clients.

$$R_{ch} = \sum_{clients} U_{ch}(i) c_{ch}(i) dt$$

where $U_{ch}(i)$ is the power demand (in kW) of the i 'th client, $c_{ch}(i)$ is the price in euro/kJ of electric energy for each client and dt is the time-step duration.

Parameters

<code>in</code>	<code>c</code>	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
<code>in</code>	<code>t</code>	time step index. Note $t=x$ meas the x 'th time step from the simulation start.

Author

Andrea Facci

Definition at line 187 of file `economy.f90`.

Referenced by `currcost()`, and `globalresults::globchrev()`.

5.11.2.2 real(kind = prec) function `economy::currcost` (integer, dimension(nm), intent(in) c, integer, intent(in) t)

Calculates the profit of a time-step(in euro or any other currency according to the one used in the input) for a given set-point, all the costs, except for the costs associated to equipment ignition.

$$G = R_{el} + R_{th} + R_{ch} + C_f + C_m$$

where

- R_{el} are the electrical revenues;
- R_{th} are the thermal revenues;
- R_{ch} are the chilling revenues
- C_f are the fuel costs;
- C_m are the maintenance costs;
- dt is the time-step duration.

Having discarded lighting costs, this profit depends only on the state of the plant at a determined time-step and not on the state at pre previous or subsequent time-step and will be associated to a vertex of the graph.

Parameters

in	c	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
in	t	time step index. Note t=x meas the x'th time step from the simulation start.

Author

Andrea Facci

Definition at line 322 of file economy.f90.

References chrev(), elrev(), fuelcost(), maintenancencost(), and threv().

Referenced by objfunction().

5.11.2.3 real(kind = prec) function economy::elrev (integer, dimension(nm), intent(in) c, integer, intent(in) t)

Calculates the revenues (in euro or any other currency according to the one used in the input) from electric energy selling to the various clients and to the grid. Electric energy revenues are calculated in a different way for each kind of grid connection. Specifically:

- Stand Alone:

$$R_{el} = \sum_{clients} U_{el}(i) c_{el}(i) dt$$

where $U_{el}(i)$ is the power demand (in kW) of the i 'th client, $c_{el}(i)$ is the price in euro/kJ of electric energy for each client and dt is the time-step duration.

- Grid connected with net metering:

$$R_{el} = \sum_{clients} U_{el}(i) c_{el}(i) dt + P_{el} c_{s_{grid}} - U_{el}^t c_{b_{grid}}$$

where P_{el} is the total electric power produced by the power plant, $c_{s_{grid}}$ is the selling price to the grid, $U_{el}^t = \sum_{clients} U_{el}(i) + U_{el}^{self}$ is the total electric demand, including the power plant self-consumption U_{el}^{self}

- Grid Connected without net metering:

$$R_{el} = \sum_{clients} U_{el}(i) c_{el}(i) dt + (P_{el} - U_{el}^t) c_{grid}$$

where $c_{grid} = c_{s_{grid}}$ if $P_{el} \geq U_{el}^t$ and $c_{grid} = c_{b_{grid}}$ otherwise

Parameters

in	c	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
in	t	time step index. Note t=x meas the x'th time step from the simulation start.

Author

Andrea Facci

Definition at line 80 of file economy.f90.

References energy::elprod(), and energy::elseifcons().

Referenced by currcost(), and globalresults::globelrev().

5.11.2.4 `real(kind = prec) function economy::firecost (integer, dimension(nm), intent(in) cNew, integer, dimension(nm), intent(in) cOld)`

Calculates the cost associated to each lighting of a machinery.

$$C_l > 0 \quad \text{if} \quad sp(t, i) > 0 \text{ and } sp(t-1, i) = 0$$

this cost will be added to the operative profit to for the arc profit/cost.

Parameters

in	c	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
in	t	time step index. Note t=x meas the x'th time step from the simulation start.

Author

Andrea Facci

Definition at line 355 of file economy.f90.

Referenced by globalresults::globfirecost(), graphtools::grapharcs(), and output().

5.11.2.5 `real(kind = prec) function economy::fuelcost (integer, dimension(nm), intent(in) c, integer, intent(in) t)`

Calculates the costs to buy the fuel (in euro or any other currency according to the one used in the input)

$$C_f = \sum_{Trig+Boi} \frac{E_{in}(i)c_f(i)}{H_i(i)} dt$$

where $c_f(i)$ is the specific cost (per unit mass or volume) of the fuel for the i 'th machine, $H_i(i)$ is the fuel LHV (kJ per unit mass or volume), $E_{in}(i)$ is the primary energy input. $c_{ch}(i)$ is the price in euro/kJ of electric energy for each client and dt is the time-step duration.

Note that even though international units are strongly suggested, any units of mass and/or volume is valid for c_f and H_i provided that coherence between the units of these two variables is respected. Moreover prices and LVSS may may be expressend in different units for differend machines.

Parameters

in	c	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
in	t	time step index. Note t=x meas the x'th time step from the simulation start.

Author

Andrea Facci

Definition at line 233 of file economy.f90.

References energy::fuelcons().

Referenced by `currcost()`, `globalresults::globfuelcost()`, and `output()`.

5.11.2.6 `real(kind = prec) function, dimension(2) economy::grideconomy (integer, dimension(nm), intent(in) c, integer, intent(in) t)`

Definition at line 380 of file `economy.f90`.

References `energy::elprod()`, and `energy::elselfcons()`.

Referenced by `output()`.

5.11.2.7 `real(kind = prec) function economy::maintenancecost (integer, dimension(nm), intent(in) c, integer, intent(in) t)`

Calculates the maintenance costs (in euro or any other currency according to the one used in the input)

$$C_m = \sum c_m(i) dt \quad \text{if} \quad sp(i) > 0$$

where $c_m(i)$ is the maintenance cost per unit time and dt is the time-step duration.

Parameters

in	c	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
in	t	time step index. Note t=x meas the x'th time step from the simulation start.

Author

Andrea Facci

Definition at line 276 of file `economy.f90`.

Referenced by `currcost()`, `globalresults::globmaintcost()`, and `output()`.

5.11.2.8 `real(kind = prec) function economy::threv (integer, intent(in) t)`

Calculates the revenues (in euro or any other currency according to the one used in the input) from thermal energy selling to the various clients.

$$R_{th} = \sum_{clients} U_{th}(i) c_{th}(i) dt$$

where $U_{th}(i)$ is the power demand (in kW) of the i 'th client, $c_{th}(i)$ is the price in euro/kJ of electric energy for each client and dt is the time-step duration.

Parameters

in	c	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
in	t	time step index. Note t=x meas the x'th time step from the simulation start.

Author

Andrea Facci

Definition at line 147 of file `economy.f90`.

Referenced by `currcost()`, and `globalresults::globthrev()`.

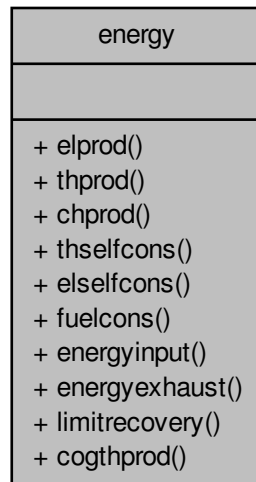
The documentation for this module was generated from the following file:

- `/home/Codici/Blink/PowerManager/src/economy.f90`

5.12 energy Module Reference

module for energy calculations.

Collaboration diagram for energy:



Public Member Functions

- real(kind=prec) function [elprod](#) (c_, t)
Electrical production.
- real(kind=prec) function [thprod](#) (c_, t)
Thermal production.
- real(kind=prec) function [chprod](#) (c_, t)
Chilling production.
- real(kind=prec) function [thselfcons](#) (c_, t)
Internal thermal consumption of the power plant.
- real(kind=prec) function [elselfcons](#) (c_, t)
Internal electrical consumption of the power plant.
- real(kind=prec) function,
dimension(nboi+ntrig) [fuelcons](#) (c_, t)
Primary energy input.
- real(kind=prec) function,
dimension(nm) [energyinput](#) (c_, t)
Primary energy input.
- real(kind=prec) function,
dimension(ntrig+nboi) [energyexhaust](#) (c_, t)
- real(kind=prec) function [limitrecovery](#) (i, valore, t)
- real(kind=prec) function [cogthprod](#) (c_, t)
Cogenerative Thermal production.

5.12.1 Detailed Description

This module contains all procedures to calculate the energy fluxes inside the power plant and between the power plant and the clients.

Author

Andrea Facci

Definition at line 38 of file energy.f90.

5.12.2 Member Function/Subroutine Documentation

5.12.2.1 `real(kind = prec) function energy::chprod (integer, dimension(nm), intent(in) c_, integer, intent(in) t)`

Calculates the chilling production in kW of the whole power plant, for a given set-point. Note that only trigeneration machines and Chillers produce chilling power so far. Thus chilling power is:

$$P_{ch} = \sum_{Trig} \frac{sp(i) \cdot P_{max}(i)}{\eta_{el}(i, sp(i))} \eta_{ch}(i, sp(i)) + \sum_{Chi} sp(i) \cdot P_{max}(i)$$

where $sp(i)$ is the set point of the i 'th machine, η_{ch} and η_{el} are the chilling and electrical efficiencies, respectively, and $P_{max}(i)$ is its rated power.

Parameters

in	c_	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_ (i))$
in	t	time index

Author

Andrea Facci

Definition at line 165 of file energy.f90.

Referenced by `constraints()`, and `output()`.

5.12.2.2 `real(kind = prec) function energy::cogthprod (integer, dimension(nm), intent(in) c_, integer, intent(in) t)`

Calculates the cogenerative Thermal production in kW of the whole power plant, for a given set-point. Note that only trigeneration machines and Boilers produce thermal power so far. Thus Thermal power is:

$$P_{th} = \sum_{Trig} \frac{sp(i) \cdot P_{max}(i)}{\eta_{el}(i, sp(i))} \eta_{th}(i, sp(i)) + \sum_{HRSG} sp(i) \cdot P_{max}(i)$$

where $sp(i)$ is the set point of the i 'th machine, η_{th} and η_{el} are the thermal and electrical efficiencies, respectively, and $P_{max}(i)$ is its rated power.

Parameters

in	c_	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_ (i))$
in	t	time index

Author

Andrea Facci

Definition at line 525 of file energy.f90.

References `limitrecovery()`.Referenced by `heuristics::thredundant()`.**5.12.2.3 `real(kind = prec) function energy::elprod (integer, dimension(nm), intent(in) c_, integer, intent(in) t)`**

Calculates the electrical production in kW of the whole power plant, for a given set-point Note that only trigeneration machines produce electrical power so far. Thus electrical power is:

$$P_{el} = \sum_{Trig} sp(i) \cdot P_{max}(i) k_{env}$$

where $sp(i)$ is the set point of the i 'th trigenerative machine and $P_{max}(i)$ is its rated power. $k_{env} = k_a \cdot k_T \cdot k_p$ is the environmental correction. And k_a , k_t , k_p are the altitude, temperature and pressure corrections.

Parameters

in	c_	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_(i))$
in	t	time index

Author

Andrea Facci

Definition at line 59 of file energy.f90.

Referenced by `economy::elrev()`, `economy::grideconomy()`, and `output()`.**5.12.2.4 `real(kind = prec) function energy::elselfcons (integer, dimension(nm), intent(in) c_, integer t)`**

Calculates the electrical self-consumption of the trigeneration plant, that is, the electrical power needed by the mechanical chillers, for a given set-point

$$U_{el}^{self} = \sum_{MecChi} \frac{sp(i) \cdot P_{max}(i)}{\eta_{ch}(i, sp(i))}$$

where $sp(i)$ is the set point of the i 'th machine, η_{ch} is the chilling efficiency, and $P_{max}(i)$ is its rated power. The summation is extended over the number of mechanical chillers.

Parameters

in	c_	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_(i))$
in	t	time index

Author

Andrea Facci

Definition at line 263 of file energy.f90.

Referenced by `economy::elrev()`, and `economy::grideconomy()`.

5.12.2.5 `real(kind = prec) function, dimension(ntrig+nboi) energy::energyexhaust (integer, dimension(ntrig+nboi), intent(in) c_, integer, intent(in) t)`

Definition at line 417 of file energy.f90.

5.12.2.6 `real(kind = prec) function, dimension(nm) energy::energyinput (integer, dimension(nm), intent(in) c_, integer, intent(in) t)`

Calculates the eprimary energy input of the trigeneration plant, for a given set-point

$$E_{in}(i) = \frac{sp(i) \cdot P_{max}(i)}{\eta(i, sp(i))}$$

where $sp(i)$ is the set point of the i 'th machine, $\eta(i, sp(i)) = \eta_{el}(i, sp(i))$ for trigenerative equipment and $\eta(i, sp(i)) = \eta_{th}(i, sp(i))$ for boilers and, and $P_{max}(i)$ is their rated power.

Parameters

in	c_	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_ (i))$
in	t	time index

Author

Andrea Facci

Definition at line 366 of file energy.f90.

Referenced by output().

5.12.2.7 `real(kind = prec) function, dimension(nboi+ntrig) energy::fuelcons (integer, dimension(nm), intent(in) c_, integer, intent(in) t)`

Calculates the eprimary energy input of the trigeneration plant, for a given set-point

$$E_{in}(i) = \frac{sp(i) \cdot P_{max}(i)}{\eta(i, sp(i))}$$

where $sp(i)$ is the set point of the i 'th machine, $\eta(i, sp(i)) = \eta_{el}(i, sp(i))$ for trigenerative equipment and $\eta(i, sp(i)) = \eta_{th}(i, sp(i))$ for boilers and, and $P_{max}(i)$ is their rated power.

Parameters

in	c_	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_ (i))$
in	t	time index

Author

Andrea Facci

Definition at line 310 of file energy.f90.

Referenced by economy::fuelcost(), and output().

5.12.2.8 **real(kind = prec) function energy::limitrecovery (integer, intent(in) *i*, real(kind = prec), intent(in) *valore*, integer, intent(in) *t*)**

Definition at line 461 of file energy.f90.

Referenced by cogthprod(), and thprod().

5.12.2.9 **real(kind = prec) function energy::thprod (integer, dimension(nm), intent(in) *c_*, integer, intent(in) *t*)**

Calculates the Thermal production in kW of the whole power plant, for a given set-point Note that only trigeneration machines and Boilers produce thermal power so far. Thus Thermal power is:

$$P_{th} = \sum_{Trig} \frac{sp(i) \cdot P_{max}(i)}{\eta_{el}(i, sp(i))} \eta_{th}(i, sp(i)) + \sum_{Boi} sp(i) \cdot P_{max}(i)$$

where $sp(i)$ is the set point of the i 'th machine, η_{th} and η_{el} are the thermal and electrical efficiencies, respectively, and $P_{max}(i)$ is its rated power.

Parameters

in	<i>c_</i>	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_-(i))$
in	<i>t</i>	time index

Author

Andrea Facci

Definition at line 103 of file energy.f90.

References limitrecovery().

Referenced by constraints(), and output().

5.12.2.10 **real(kind = prec) function energy::thselfcons (integer, dimension(nm), intent(in) *c_*, integer, intent(in) *t*)**

Calculates the thermal self-consumption of the trigeneration plant, that is, the thermal power needed by the absorption chillers.

$$U_{th}^{self} = \sum_{AbsChi} \frac{sp(i) \cdot P_{max}(i)}{\eta_{ch}(i, sp(i))}$$

where $sp(i)$ is the set point of the i 'th machine, η_{ch} is the chilling efficiency, and $P_{max}(i)$ is its rated power.

Parameters

in	<i>c_</i>	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_-(i))$
in	<i>t</i>	time index

Author

Andrea Facci

Definition at line 215 of file energy.f90.

Referenced by constraints(), and euristics::thredundant().

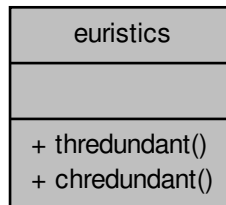
The documentation for this module was generated from the following file:

- /home/Codici/Blink/PowerManager/src/energy.f90

5.13 euristics Module Reference

Module that contains the function to apply an euristics to the graph.

Collaboration diagram for euristics:



Data Types

- interface [constraints](#)

Public Member Functions

- logical function [thredundant](#) (c, t)
Rejects solutions with excess of thermal production.
- logical function [chredundant](#) (c, t)
Rejects solutions with excess of chilling production.

5.13.1 Detailed Description

Module that contains the function to apply an euristics to the graph. in order to reduce the number of points and arcs of the graph.

Author

Andrea Facci

Definition at line 38 of file euristics.f90.

5.13.2 Member Function/Subroutine Documentation

5.13.2.1 logical function `euristics::chredundant (integer, dimension(nm), intent(in) c, integer, intent(in) t)`

Rejects the points where an excess chilling production is achieved using mechanical chillers. If there is a constraint on the duration of the on or off intervals the possibility to operate the chillers at minimum load even when not strictly necessary is considered.

Parameters

in	c	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
in	t	time-step index

Author

Andrea Facci

Definition at line 138 of file `heuristics.f90`.

Referenced by `graphtools::graphpoints()`.

5.13.2.2 logical function `heuristics::thredundant (integer, dimension(nm), intent(in) c, integer, intent(in) t)`

Rejects the points where an excess thermal production is achieved using fuel boilers. If there is a constraint on the duration of the on or off intervals the possibility to operate the boilers at minimum load even when not strictly necessary is considered.

Parameters

<code>in</code>	<code>c</code>	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
<code>in</code>	<code>t</code>	time-step index

Author

Andrea Facci

Definition at line 70 of file `heuristics.f90`.

References `energy::cogthprod()`, and `energy::thselfcons()`.

Referenced by `graphtools::graphpoints()`.

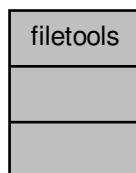
The documentation for this module was generated from the following file:

- `/home/Codici/Blink/PowerManager/src/`[heuristics.f90](#)

5.14 filetools Module Reference

Interfaces of procedures to read from file.

Collaboration diagram for `filetools`:

**Data Types**

- interface [cFindEntry](#)
- interface [cMatrixRead](#)

- interface [dFindEntry](#)
- interface [dMatrixRead](#)
- interface [findEntry](#)
- interface [hCount](#)
- interface [iFindEntry](#)
- interface [iMatrixRead](#)
- interface [matrixRead](#)
- interface [readKeyword](#)
- interface [rewUnit](#)
- interface [vCount](#)

5.14.1 Detailed Description

This module collects all the interfaces of the procedures useful to read data from files.

Author

Andrea Facci.

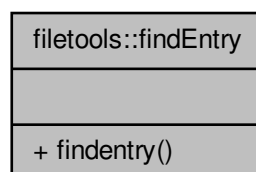
Definition at line 39 of file fileTools.f90.

The documentation for this module was generated from the following file:

- [/home/Codici/Blink/PowerManager/src/fileTools.f90](#)

5.15 filetools::findEntry Interface Reference

Collaboration diagram for filetools::findEntry:



Public Member Functions

- subroutine [findentry](#) (entry, theUnit, rew, valore, isPresent, nRow)

5.15.1 Detailed Description

Definition at line 153 of file fileTools.f90.

5.15.2 Member Function/Subroutine Documentation

5.15.2.1 subroutine `filetools::findEntry::findentry` (`character(len=*)`, `intent(in)` *entry*, `integer`, `intent(in)` *theUnit*, `logical`, `intent(in)`, optional *rew*, `character(len=100)`, `intent(out)`, optional *valore*, `logical`, `intent(out)`, optional *isPresent*, `integer`, `intent(out)`, optional *nRow*)

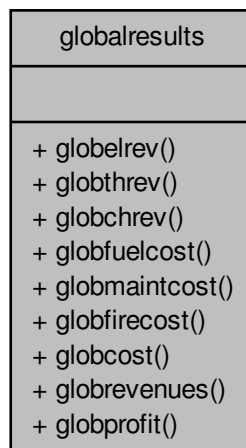
Definition at line 153 of file `fileTools.f90`.

The documentation for this interface was generated from the following file:

- `/home/Codici/Blink/PowerManager/src/fileTools.f90`

5.16 globalresults Module Reference

Collaboration diagram for `globalresults`:



Public Member Functions

- `real(kind=prec)` function `globelrev` (setPoint)
Electric energy revenues.
- `real(kind=prec)` function `globthrev` (setPoint)
thermal energy revenues.
- `real(kind=prec)` function `globchrev` (setPoint)
Chilling energy revenues.
- `real(kind=prec)` function `globfuelcost` (setPoint)
Calculates the costs to buy the fuel.
- `real(kind=prec)` function `globmaintcost` (setPoint)
Calculates the maintenance costs.
- `real(kind=prec)` function `globfirecost` (setPoint)
Calculates the lighting cost.
- `real(kind=prec)` function `globcost` (setPoint)
- `real(kind=prec)` function `globrevenues` (setPoint)
- `real(kind=prec)` function `globprofit` (setPoint)

5.16.1 Detailed Description

Author

This module contains the definition of all the procedures that perform global economic and energetic calculation for the whole duration of this simulation

Author

Definition at line 42 of file globalResults.f90.

5.16.2 Member Function/Subroutine Documentation

5.16.2.1 `real(kind = prec) function globalresults::globchrev (integer, dimension(0:ntime+1, nm), intent(in) setPoint)`

Calculates the revenues (in euro or any other currency according to the one used in the input) from chilling energy selling to the various clients.

$$R_{ch} = \sum_{clients} U_{ch}(i) c_{ch}(i) dt$$

where $U_{ch}(i)$ is the power demand (in kW) of the i 'th client, $c_{ch}(i)$ is the price in euro/kJ of electric energy for each client and dt is the time-step duration.

Parameters

in	c	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
in	t	time step index. Note t=x meas the x'th time step from the simulation start.

Author

Andrea Facci

Definition at line 159 of file globalResults.f90.

References `economy::chrev()`.

Referenced by `globrevenues()`, and `output()`.

5.16.2.2 `real(kind=prec) function globalresults::globcost (integer, dimension(0:ntime+1,nm), intent(in) setPoint)`

Definition at line 290 of file globalResults.f90.

References `globfirecost()`, `globfuelcost()`, and `globmaintcost()`.

Referenced by `globprofit()`, and `output()`.

5.16.2.3 `real(kind = prec) function globalresults::globelrev (integer, dimension(0:ntime+1, nm), intent(in) setPoint)`

Calculates the revenues (in euro or any other currency according to the one used in the input) from electric energy selling to the various clients and to the grid. Electric energy revenues are calculated in a different way for each kind of grid connection. Specifically:

- Stand Alone:

$$R_{el} = \sum_{clients} U_{el}(i) c_{el}(i) dt$$

where $U_{el}(i)$ is the power demand (in kW) of the i 'th client, $c_{el}(i)$ is the price in euro/kJ of electric energy for each client and dt is the time-step duration.

- Grid connected with net metering:

$$R_{el} = \sum_{clients} U_{el}(i)c_{el}(i)dt + P_{el}c_{s_{grid}} - U_{el}^t c_{b_{grid}}$$

where P_{el} is the total electric power produced by the power plant, $c_{s_{grid}}$ is the selling price to the grid, $U_{el}^t = \sum_{clients} U_{el}(i) + U_{el}^{self}$ is the total electric demand, including the power plant self-consumption U_{el}^{self}

- Grid Connected without net metering:

$$R_{el} = \sum_{clients} U_{el}(i)c_{el}(i)dt + (P_{el} - U_{el}^t)c_{grid}$$

where $c_{grid} = c_{s_{grid}}$ if $P_{el} \geq U_{el}^t$ and $c_{grid} = c_{b_{grid}}$ otherwise

Parameters

in	c	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
in	t	time step index. Note t=x meas the x'th time step from the simulation start.

Author

Andrea Facci

Definition at line 84 of file globalResults.f90.

References economy::elrev().

Referenced by globrevenues(), and output().

5.16.2.4 real(kind = prec) function globalresults::globfirecost (integer, dimension(0:ntime+1,nm), intent(in) setPoint)

Calculates the cost associated to each lighting of a machinery.

$$C_l > 0 \quad \text{if} \quad sp(t,i) > 0 \text{ and } sp(t-1,i) = 0$$

this cost will be added to the operative profit to for the arc profit/cost.

Parameters

in	c	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
in	t	time step index. Note t=x meas the x'th time step from the simulation start.

Author

Andrea Facci

Definition at line 270 of file globalResults.f90.

References economy::firecost().

Referenced by globcost(), and output().

5.16.2.5 `real(kind = prec) function globalresults::globfuelcost (integer, dimension(0:ntime+1, nm), intent(in) setPoint)`

Calculates the costs to buy the fuel (in euro or any other currency according to the one used in the input)

$$C_f = \sum_{Trig+Boi} \frac{E_{in}(i)c_f(i)}{H_i(i)} dt$$

where $c_f(i)$ is the specific cost (per unit mass or volume) of the fuel for the i 'th machine, $H_i(i)$ is the fuel LHV (kJ per unit mass or volume), $E_{in}(i)$ is the primary energy input. $c_{ch}(i)$ is the price in euro/kJ of electric energy for each client and dt is the time-step duration.

Note that even though international units are strongly suggested, any units of mass and/or volume is valid for c_f and H_i provided that coherence between the units of these two variables is respected. Moreover prices and LVSS may be expressed in different units for different machines.

Parameters

in	c	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
in	t	time step index. Note t=x means the x'th time step from the simulation start.

Author

Andrea Facci

Definition at line 202 of file globalResults.f90.

References economy::fuelcost().

Referenced by globcost(), and output().

5.16.2.6 `real(kind = prec) function globalresults::globmaintcost (integer, dimension(0:ntime+1, nm), intent(in) setPoint)`

Calculates the maintenance costs (in euro or any other currency according to the one used in the input)

$$C_m = \sum c_m(i) dt \quad \text{if} \quad sp(i) > 0$$

where $c_m(i)$ is the maintenance cost per unit time and dt is the time-step duration.

Parameters

in	c	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
in	t	time step index. Note t=x means the x'th time step from the simulation start.

Author

Andrea Facci

Definition at line 237 of file globalResults.f90.

References `economy::maintenancecost()`.

Referenced by `globcost()`, and `output()`.

5.16.2.7 `real(kind=prec) function globalresults::globprofit (integer, dimension(0:ntime+1,nm), intent(in) setPoint)`

Definition at line 314 of file `globalResults.f90`.

References `globcost()`, and `globrevenues()`.

Referenced by `output()`.

5.16.2.8 `real(kind=prec) function globalresults::globrevenues (integer, dimension(0:ntime+1,nm), intent(in) setPoint)`

Definition at line 302 of file `globalResults.f90`.

References `globchrev()`, `globelrev()`, and `globthrev()`.

Referenced by `globprofit()`, and `output()`.

5.16.2.9 `real(kind = prec) function globalresults::globthrev (integer, dimension(0:ntime+1, nm), intent(in) setPoint)`

Calculates the revenues (in euro or any other currency according to the one used in the input) from thermal energy selling to the various clients.

$$R_{th} = \sum_{clients} U_{th}(i) c_{th}(i) dt$$

where $U_{th}(i)$ is the power demand (in kW) of the i 'th client, $c_{th}(i)$ is the price in euro/kJ of electric energy for each client and dt is the time-step duration.

Parameters

<code>in</code>	<code>c</code>	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
<code>in</code>	<code>t</code>	time step index. Note $t=x$ meas the x 'th time step from the simulation start.

Author

Andrea Facci

Definition at line 121 of file `globalResults.f90`.

References `economy::threv()`.

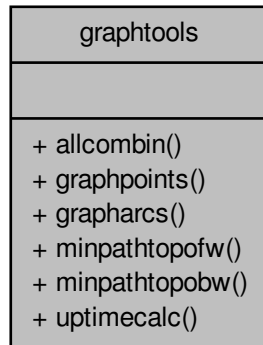
Referenced by `globrevenues()`, and `output()`.

The documentation for this module was generated from the following file:

- [/home/Codici/Blink/PowerManager/src/globalResults.f90](#)

5.17 graphtools Module Reference

Collaboration diagram for graphtools:



Data Types

- interface [constraints](#)
- interface [objFunction](#)
- interface [timeConstr](#)

Public Member Functions

- subroutine [allcombin](#) (icm, dcm, imax, m, targ)
Generates all the combinations of the set points, staring from the set-point vectors that are stored in the columns of "cm".
- subroutine [graphpoints](#)
Generates the graph veteces, starting from the array of the set-point combinations.
- subroutine [grapharcs](#)
Generates the grapsh arcs.
- subroutine [minpathtopofw](#) (ottLoad, minCost)
Miniumum path detemination.
- subroutine [minpathtopobw](#) (ottLoad, minCost, upTime, minPath)
Miniumum path detemination.
- real(kind=prec) function,
 dimension(2 *nm) [uptimecalc](#) (upT, c, t)

5.17.1 Detailed Description

Definition at line 34 of file graphTools.f90.

5.17.2 Member Function/Subroutine Documentation

5.17.2.1 subroutine `graphtools::allcombin` (integer, dimension(maxval(imax),m), intent(in), optional *icm*, real(kind = prec), dimension(maxval(imax),m), intent(in), optional *dcm*, integer, dimension(m), intent(in) *imax*, integer, intent(in) *m*, character(len=100), intent(in), optional *targ*)

Generates all the combinations of the set points, starting from the set-point vectors that are stored in the columns of "cm". Each row of the array "comb" represents a set-point of the plant. The total number of states of the plant is also returned in the variable "nComb"

Author

Andrea Facci

Definition at line 102 of file `graphTools.f90`.

References `abortexecution()`, and `myarithmetic::rnan()`.

Referenced by `main()`.

5.17.2.2 subroutine `graphtools::grapharcs` ()

Generates the graph arcs. Note that only vertex relative to consecutive time-steps are connected and that arcs are oriented in the direction of increasing time. Arcs are stored in the form of a predecessor list "predList", that associates to each node all its predecessors. A weight is associated to each element of the predecessor list, equal to the weight of the predecessor vertex plus a cost connected to the variation of state between the actual and predecessor state.

$$arcCost(i, j) = pointCost(c(i)) + fireCost(i, j)$$

The number of predecessors for each vertex is asle stored in the "nPre(i)" array.

Definition at line 308 of file `graphTools.f90`.

References `economy::firecost()`.

Referenced by `main()`.

5.17.2.3 subroutine `graphtools::graphpoints` ()

Generates the graph veteces, starting from the array of the set-point combinations. For each time-step determines which plant state respects the energy (staitc) constraints, and associates them to a graph vertex. A weight, that accounts for the costs/revenues of operating the power plant from time-step t to t+1 at the vertex state, is also calculated for each vertex. The time-step relative to each vertex and the number of veteces for each time step are associated to "pointTime" and "nt" vectors respectively. Vertex 0 will be the starting point of the graph and vertex nPoint + 1 the arriving point

Author

Andrea Facci.

Definition at line 206 of file `graphTools.f90`.

References `abortexecution()`, `euristics::chredundant()`, `mathtools::locaterow()`, `objfunction()`, and `euristics::thredundant()`.

Referenced by `main()`.

5.17.2.4 subroutine `graphtools::minpathtopobw` (integer, dimension(0:ntime+1,nm), intent(out) *ottLoad*, real(kind = prec), intent(out) *minCost*, real(kind = prec), dimension(0:ntime+1,2*nm), intent(out) *upTime*, integer, dimension(0:ntime+1) *minPath*)

This function determines the minimum path that connects the start point (0) to the arrival point of the graph (nPoint + 1), using backward dynamic programming. Specifically the optimization the algorithm is tailored to sort acyclic graphs with topological ordering. Constraints on the duration of operative intervals (on and off states) are considered.

Author

Andrea Facci.

Definition at line 472 of file `graphTools.f90`.

References `mathtools::locaterow()`, `timeconstr()`, and `uptimecalc()`.

Referenced by `main()`.

5.17.2.5 subroutine `graphtools::minpathtopofw` (integer, dimension(0:ntime+1,nm), intent(out) *ottLoad*, real(kind = prec), intent(out) *minCost*)

This function determines the minimum path that connects the start point (0) to the arrival point of the graph (nPoint + 1), using dynamic programming. Specifically the optimization the algorithm is tailored to sort acyclic graphs with topological ordering.

Author

Andrea Facci.

Definition at line 413 of file `graphTools.f90`.

Referenced by `main()`.

5.17.2.6 real(kind = prec) function, dimension(2*nm) `graphtools::uptimecalc` (real(kind = prec), dimension(2*nm) *upT*, integer, dimension(nm), intent(in) *c*, integer, intent(in) *t*)

Definition at line 549 of file `graphTools.f90`.

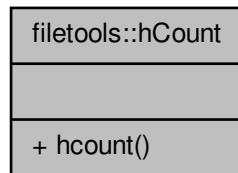
Referenced by `minpathtopobw()`.

The documentation for this module was generated from the following file:

- </home/Codici/Blink/PowerManager/src/graphTools.f90>

5.18 filetools::hCount Interface Reference

Collaboration diagram for filetools::hCount:



Public Member Functions

- integer function [hcount](#) (value_, first_, last_)

5.18.1 Detailed Description

Definition at line 62 of file fileTools.f90.

5.18.2 Member Function/Subroutine Documentation

5.18.2.1 integer function filetools::hCount::hcount (character(len=100), intent(in) *value_*, character(len=1), optional *first_*, character(len=1), optional *last_*)

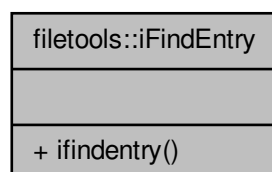
Definition at line 62 of file fileTools.f90.

The documentation for this interface was generated from the following file:

- [/home/Codici/Blink/PowerManager/src/fileTools.f90](#)

5.19 filetools::iFindEntry Interface Reference

Collaboration diagram for filetools::iFindEntry:



Public Member Functions

- subroutine [ifindentry](#) (entry, n, theUnit, rew, valore, isPresent, nRow)

5.19.1 Detailed Description

Definition at line 111 of file fileTools.f90.

5.19.2 Member Function/Subroutine Documentation

- 5.19.2.1 subroutine filetools::iFindEntry::ifindentry (character(len=*), intent(in) *entry*, integer, intent(in) *n*, integer, intent(in) *theUnit*, logical, intent(in), optional *rew*, integer, dimension(n), intent(out), optional *valore*, logical, intent(out), optional *isPresent*, integer, intent(out), optional *nRow*)

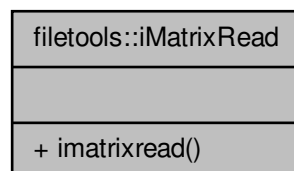
Definition at line 111 of file fileTools.f90.

The documentation for this interface was generated from the following file:

- /home/Codici/Blink/PowerManager/src/[fileTools.f90](#)

5.20 filetools::iMatrixRead Interface Reference

Collaboration diagram for filetools::iMatrixRead:



Public Member Functions

- real(kind=prec) function,
dimension(nline, ncol) [imatrixread](#) (theUnit, nline, ncol, first_, last_)

5.20.1 Detailed Description

Definition at line 81 of file fileTools.f90.

5.20.2 Member Function/Subroutine Documentation

- 5.20.2.1 real(kind = prec) function, dimension(nline,ncol) filetools::iMatrixRead::imatrixread (integer, intent(in) *theUnit*, integer, intent(in) *nline*, integer, intent(in) *ncol*, character(len=1), optional *first_*, character(len=1), optional *last_*)

Definition at line 81 of file fileTools.f90.

The documentation for this interface was generated from the following file:

- [/home/Codici/Blink/PowerManager/src/fileTools.f90](#)

5.21 inputvar Module Reference

Input variables collection.

Collaboration diagram for inputvar:



Public Attributes

- character(len=20) [gridconnection](#)
- integer [ntimes](#)
- logical [ideg](#)
- real(kind=prec), dimension(:), allocatable [startpoint](#)
- real(kind=prec), dimension(:), allocatable [uptime0](#)
- real(kind=prec), dimension(:), allocatable [downtime0](#)
- real(kind=prec) [dt1](#)
- character(len=100) [obj](#)
- character(len=100) [method](#)
- logical [writepower](#) = .false.
- logical [writeenergy](#) = .false.
- logical [writeefficiency](#) = .false.
- logical [writeelectricrev](#) = .false.
- logical [writethermalrev](#) = .false.
- logical [writechillingrev](#) = .false.

- logical `writedemand` = .false.
- logical `writeinput` = .false.
- logical `writecosts` = .false.
- logical `writetrig` = .false.
- logical `writeboi` = .false.
- logical `writetechi` = .false.
- logical `global` = .false.
- logical `useheuristics` = .false.
- integer `ntrig`
- integer, dimension(:), allocatable `nspt`
- integer, dimension(:), allocatable `netael`
- integer, dimension(:), allocatable `netatht`
- integer, dimension(:), allocatable `netacht`
- integer, dimension(:), allocatable `ntct`
- integer, dimension(:), allocatable `npct`
- integer, dimension(:), allocatable `nact`
- real(kind=prec), dimension(:), allocatable `pmaxt`
- real(kind=prec), dimension(:), allocatable `fuelcostt`
- real(kind=prec), dimension(:), allocatable `fuelhvt`
- real(kind=prec), dimension(:), allocatable `lifet`
- real(kind=prec), dimension(:), allocatable `firecostt`
- real(kind=prec), dimension(:), allocatable `maintcostt`
- real(kind=prec), dimension(:), allocatable `minuptimet`
- real(kind=prec), dimension(:), allocatable `mindowntimet`
- real(kind=prec), dimension(:,:), allocatable `spt`
- real(kind=prec), dimension(:,:), allocatable `etaelt`
- real(kind=prec), dimension(:,:), allocatable `etatht`
- real(kind=prec), dimension(:,:), allocatable `etacht`
- real(kind=prec), dimension(:,:), allocatable `tempcorrt`
- real(kind=prec), dimension(:,:), allocatable `prescorrt`
- real(kind=prec), dimension(:,:), allocatable `altcorrt`
- character(len=50), dimension(:), allocatable `tect`
- integer `nboi`
- integer, dimension(:), allocatable `nsptb`
- integer, dimension(:), allocatable `netab`
- integer, dimension(:), allocatable `ntcb`
- integer, dimension(:), allocatable `npctb`
- integer, dimension(:), allocatable `nacbt`
- real(kind=prec), dimension(:), allocatable `pmaxb`

- real(kind=prec), dimension(:), allocatable [fuelcostb](#)
- real(kind=prec), dimension(:), allocatable [fuellhvb](#)
- real(kind=prec), dimension(:), allocatable [lifeb](#)
- real(kind=prec), dimension(:), allocatable [firecostb](#)
- real(kind=prec), dimension(:), allocatable [maintcostb](#)
- real(kind=prec), dimension(:), allocatable [minuptimeb](#)
- real(kind=prec), dimension(:), allocatable [mindowntimeb](#)
- real(kind=prec), dimension(:,:), allocatable [spb](#)
- real(kind=prec), dimension(:,:,:), allocatable [etab](#)
- real(kind=prec), dimension(:,:,:), allocatable [tempcorrb](#)
- real(kind=prec), dimension(:,:,:), allocatable [prescorrb](#)
- real(kind=prec), dimension(:,:,:), allocatable [altcorrb](#)
- character(len=50), dimension(:), allocatable [tecb](#)
- integer [nchi](#)
- integer, dimension(:), allocatable [nspc](#)
- integer, dimension(:), allocatable [nsizec](#)
- integer, dimension(:), allocatable [netac](#)
- integer, dimension(:), allocatable [ntcc](#)
- integer, dimension(:), allocatable [npcc](#)
- integer, dimension(:), allocatable [nacc](#)
- real(kind=prec), dimension(:), allocatable [pmaxc](#)
- real(kind=prec), dimension(:), allocatable [firecostc](#)
- real(kind=prec), dimension(:), allocatable [maintcostc](#)
- real(kind=prec), dimension(:), allocatable [minuptimec](#)
- real(kind=prec), dimension(:), allocatable [mindowntimec](#)
- real(kind=prec), dimension(:,:), allocatable [spc](#)
- real(kind=prec), dimension(:,:,:), allocatable [etac](#)
- real(kind=prec), dimension(:,:,:), allocatable [tempcorrc](#)
- real(kind=prec), dimension(:,:,:), allocatable [prescorrc](#)
- real(kind=prec), dimension(:,:,:), allocatable [altcorrc](#)
- character(len=50), dimension(:), allocatable [tecc](#)
- integer [ntime](#)

- integer [nload](#)
- integer [itime](#)
- integer [iel](#)
- integer [ith](#)
- integer [ich](#)
- integer [ielp](#)
- integer [ithp](#)
- integer [ichp](#)
- real(kind=prec), dimension(:,:), allocatable [uel](#)
- real(kind=prec), dimension(:,:), allocatable [uth](#)
- real(kind=prec), dimension(:,:), allocatable [uch](#)
- real(kind=prec), dimension(:,:), allocatable [cel](#)
- real(kind=prec), dimension(:,:), allocatable [cth](#)
- real(kind=prec), dimension(:,:), allocatable [cch](#)
- real(kind=prec), dimension(:), allocatable [time](#)
- real(kind=prec), dimension(:), allocatable [gridbuycost](#)
- real(kind=prec), dimension(:), allocatable [gridsellcost](#)
- integer, dimension(:), allocatable [nld](#)
- integer, dimension(:), allocatable [nlp](#)
- real(kind=prec), dimension(:), allocatable [pamb](#)
- real(kind=prec), dimension(:), allocatable [tamb](#)
- real(kind=prec), dimension(1) [altitude](#)

5.21.1 Detailed Description

This module collects all the variables read from input files. Include this module To use these variable anywhere in the code.

Author

Andrea Facci.

Definition at line 40 of file inputVar.f90.

5.21.2 Member Data Documentation

5.21.2.1 `real(kind= prec), dimension(:,:), allocatable inputvar::altcorrb`

Definition at line 81 of file inputVar.f90.

5.21.2.2 `real(kind= prec), dimension(:,:), allocatable inputvar::altcorrc`

Definition at line 89 of file inputVar.f90.

5.21.2.3 `real(kind= prec), dimension(:,,:), allocatable inputvar::altcorr`

Definition at line 72 of file `inputVar.f90`.

5.21.2.4 `real(kind= prec), dimension(1) inputvar::altitude`

Definition at line 100 of file `inputVar.f90`.

5.21.2.5 `real(kind= prec), dimension(:,,:), allocatable inputvar::cch`

Definition at line 94 of file `inputVar.f90`.

5.21.2.6 `real(kind= prec), dimension(:,,:), allocatable inputvar::cel`

Definition at line 94 of file `inputVar.f90`.

5.21.2.7 `real(kind= prec), dimension(:,,:), allocatable inputvar::cth`

Definition at line 94 of file `inputVar.f90`.

5.21.2.8 `real(kind = prec), dimension(:), allocatable inputvar::downtime0`

Definition at line 48 of file `inputVar.f90`.

5.21.2.9 `real(kind = prec) inputvar::dt1`

Definition at line 49 of file `inputVar.f90`.

5.21.2.10 `real(kind= prec), dimension(:,,:), allocatable inputvar::etab`

Definition at line 81 of file `inputVar.f90`.

5.21.2.11 `real(kind= prec), dimension(:,,:), allocatable inputvar::etac`

Definition at line 89 of file `inputVar.f90`.

5.21.2.12 `real(kind= prec), dimension(:,,:), allocatable inputvar::etacht`

Definition at line 72 of file `inputVar.f90`.

5.21.2.13 `real(kind= prec), dimension(:,,:), allocatable inputvar::etaelt`

Definition at line 72 of file `inputVar.f90`.

5.21.2.14 `real(kind= prec), dimension(:,,:), allocatable inputvar::etatht`

Definition at line 72 of file `inputVar.f90`.

5.21.2.15 `real(kind= prec), dimension(:), allocatable inputvar::firecostb`

Definition at line 78 of file inputVar.f90.

5.21.2.16 `real(kind= prec), dimension(:), allocatable inputvar::firecostc`

Definition at line 87 of file inputVar.f90.

5.21.2.17 `real(kind= prec), dimension(:), allocatable inputvar::firecostt`

Definition at line 69 of file inputVar.f90.

5.21.2.18 `real(kind= prec), dimension(:), allocatable inputvar::fuelcostb`

Definition at line 78 of file inputVar.f90.

5.21.2.19 `real(kind= prec), dimension(:), allocatable inputvar::fuelcostt`

Definition at line 69 of file inputVar.f90.

5.21.2.20 `real(kind= prec), dimension(:), allocatable inputvar::fuellhvb`

Definition at line 78 of file inputVar.f90.

5.21.2.21 `real(kind= prec), dimension(:), allocatable inputvar::fuellhvt`

Definition at line 69 of file inputVar.f90.

5.21.2.22 `logical inputvar::global = .false.`

Definition at line 63 of file inputVar.f90.

5.21.2.23 `real(kind= prec), dimension(:), allocatable inputvar::gridbuycost`

Definition at line 95 of file inputVar.f90.

5.21.2.24 `character(len=20) inputvar::gridconnection`

Definition at line 45 of file inputVar.f90.

5.21.2.25 `real(kind= prec), dimension(:), allocatable inputvar::gridsellcost`

Definition at line 95 of file inputVar.f90.

5.21.2.26 `integer inputvar::ich`

Definition at line 93 of file inputVar.f90.

5.21.2.27 integer inputvar::ichp

Definition at line 93 of file inputVar.f90.

5.21.2.28 logical inputvar::ideg

Definition at line 47 of file inputVar.f90.

5.21.2.29 integer inputvar::iel

Definition at line 93 of file inputVar.f90.

5.21.2.30 integer inputvar::ielp

Definition at line 93 of file inputVar.f90.

5.21.2.31 integer inputvar::ith

Definition at line 93 of file inputVar.f90.

5.21.2.32 integer inputvar::ithp

Definition at line 93 of file inputVar.f90.

5.21.2.33 integer inputvar::itime

Definition at line 93 of file inputVar.f90.

5.21.2.34 real(kind= prec), dimension(:), allocatable inputvar::lifeb

Definition at line 78 of file inputVar.f90.

5.21.2.35 real(kind= prec), dimension(:), allocatable inputvar::lifet

Definition at line 69 of file inputVar.f90.

5.21.2.36 real(kind= prec), dimension(:), allocatable inputvar::maintcostb

Definition at line 78 of file inputVar.f90.

5.21.2.37 real(kind= prec), dimension(:), allocatable inputvar::maintcostc

Definition at line 87 of file inputVar.f90.

5.21.2.38 real(kind= prec), dimension(:), allocatable inputvar::maintcostt

Definition at line 69 of file inputVar.f90.

5.21.2.39 character(len=100) inputvar::method

Definition at line 50 of file inputVar.f90.

5.21.2.40 real(kind= prec), dimension(:), allocatable inputvar::mindowntimeb

Definition at line 78 of file inputVar.f90.

5.21.2.41 real(kind= prec), dimension(:), allocatable inputvar::mindowntimec

Definition at line 87 of file inputVar.f90.

5.21.2.42 real(kind= prec), dimension(:), allocatable inputvar::mindowntimet

Definition at line 69 of file inputVar.f90.

5.21.2.43 real(kind= prec), dimension(:), allocatable inputvar::minuptimeb

Definition at line 78 of file inputVar.f90.

5.21.2.44 real(kind= prec), dimension(:), allocatable inputvar::minuptimec

Definition at line 87 of file inputVar.f90.

5.21.2.45 real(kind= prec), dimension(:), allocatable inputvar::minuptimet

Definition at line 69 of file inputVar.f90.

5.21.2.46 integer, dimension(:), allocatable inputvar::nach

Definition at line 77 of file inputVar.f90.

5.21.2.47 integer, dimension(:), allocatable inputvar::nacc

Definition at line 86 of file inputVar.f90.

5.21.2.48 integer, dimension(:), allocatable inputvar::nact

Definition at line 68 of file inputVar.f90.

5.21.2.49 integer inputvar::nboi

Definition at line 76 of file inputVar.f90.

5.21.2.50 integer inputvar::nchi

Definition at line 85 of file inputVar.f90.

5.21.2.51 integer, dimension(:), allocatable inputvar::netab

Definition at line 77 of file inputVar.f90.

5.21.2.52 integer, dimension(:), allocatable inputvar::netac

Definition at line 86 of file inputVar.f90.

5.21.2.53 integer, dimension(:), allocatable inputvar::netacht

Definition at line 68 of file inputVar.f90.

5.21.2.54 integer, dimension(:), allocatable inputvar::netaelt

Definition at line 68 of file inputVar.f90.

5.21.2.55 integer, dimension(:), allocatable inputvar::netatht

Definition at line 68 of file inputVar.f90.

5.21.2.56 integer, dimension(:), allocatable inputvar::nld

Definition at line 96 of file inputVar.f90.

5.21.2.57 integer inputvar::nload

Definition at line 93 of file inputVar.f90.

5.21.2.58 integer, dimension(:), allocatable inputvar::nlp

Definition at line 96 of file inputVar.f90.

5.21.2.59 integer, dimension(:), allocatable inputvar::npcb

Definition at line 77 of file inputVar.f90.

5.21.2.60 integer, dimension(:), allocatable inputvar::npcc

Definition at line 86 of file inputVar.f90.

5.21.2.61 integer, dimension(:), allocatable inputvar::npct

Definition at line 68 of file inputVar.f90.

5.21.2.62 integer, dimension(:), allocatable inputvar::nsizec

Definition at line 86 of file inputVar.f90.

5.21.2.63 integer, dimension(:), allocatable inputvar::nspb

Definition at line 77 of file inputVar.f90.

5.21.2.64 integer, dimension(:), allocatable inputvar::nspc

Definition at line 86 of file inputVar.f90.

5.21.2.65 integer, dimension(:), allocatable inputvar::nspt

Definition at line 68 of file inputVar.f90.

5.21.2.66 integer, dimension(:), allocatable inputvar::ntcb

Definition at line 77 of file inputVar.f90.

5.21.2.67 integer, dimension(:), allocatable inputvar::ntcc

Definition at line 86 of file inputVar.f90.

5.21.2.68 integer, dimension(:), allocatable inputvar::ntct

Definition at line 68 of file inputVar.f90.

5.21.2.69 integer inputvar::ntime

Definition at line 93 of file inputVar.f90.

5.21.2.70 integer inputvar::ntimes

Definition at line 46 of file inputVar.f90.

5.21.2.71 integer inputvar::ntrig

Definition at line 67 of file inputVar.f90.

5.21.2.72 character(len=100) inputvar::obj

Definition at line 50 of file inputVar.f90.

5.21.2.73 real(kind= prec), dimension(:), allocatable inputvar::pamb

Definition at line 99 of file inputVar.f90.

5.21.2.74 real(kind= prec), dimension(:), allocatable inputvar::pmaxb

Definition at line 78 of file inputVar.f90.

5.21.2.75 `real(kind= prec), dimension(:), allocatable inputvar::pmaxc`

Definition at line 87 of file inputVar.f90.

5.21.2.76 `real(kind= prec), dimension(:), allocatable inputvar::pmaxt`

Definition at line 69 of file inputVar.f90.

5.21.2.77 `real(kind= prec), dimension(:, :, :), allocatable inputvar::prescorrb`

Definition at line 81 of file inputVar.f90.

5.21.2.78 `real(kind= prec), dimension(:, :, :), allocatable inputvar::prescorrc`

Definition at line 89 of file inputVar.f90.

5.21.2.79 `real(kind= prec), dimension(:, :, :), allocatable inputvar::prescorrt`

Definition at line 72 of file inputVar.f90.

5.21.2.80 `real(kind= prec), dimension(:, :), allocatable inputvar::spb`

Definition at line 80 of file inputVar.f90.

5.21.2.81 `real(kind= prec), dimension(:, :), allocatable inputvar::spc`

Definition at line 88 of file inputVar.f90.

5.21.2.82 `real(kind= prec), dimension(:, :), allocatable inputvar::spt`

Definition at line 71 of file inputVar.f90.

5.21.2.83 `real(kind = prec), dimension(:), allocatable inputvar::startpoint`

Definition at line 48 of file inputVar.f90.

5.21.2.84 `real(kind= prec), dimension(:), allocatable inputvar::tamb`

Definition at line 99 of file inputVar.f90.

5.21.2.85 `character(len=50), dimension(:), allocatable inputvar::tecb`

Definition at line 82 of file inputVar.f90.

5.21.2.86 `character(len=50), dimension(:), allocatable inputvar::tecc`

Definition at line 90 of file inputVar.f90.

5.21.2.87 `character(len=50), dimension(:), allocatable inputvar::tect`

Definition at line 73 of file inputVar.f90.

5.21.2.88 `real(kind= prec), dimension(:, :, :), allocatable inputvar::tempcorrb`

Definition at line 81 of file inputVar.f90.

5.21.2.89 `real(kind= prec), dimension(:, :, :), allocatable inputvar::tempcorrc`

Definition at line 89 of file inputVar.f90.

5.21.2.90 `real(kind= prec), dimension(:, :, :), allocatable inputvar::tempcorrt`

Definition at line 72 of file inputVar.f90.

5.21.2.91 `real(kind= prec), dimension(:), allocatable inputvar::time`

Definition at line 95 of file inputVar.f90.

5.21.2.92 `real(kind= prec), dimension(:, :), allocatable inputvar::uch`

Definition at line 94 of file inputVar.f90.

5.21.2.93 `real(kind= prec), dimension(:, :), allocatable inputvar::uel`

Definition at line 94 of file inputVar.f90.

5.21.2.94 `real(kind = prec), dimension(:), allocatable inputvar::uptime0`

Definition at line 48 of file inputVar.f90.

5.21.2.95 `logical inputvar::useheuristics = .false.`

Definition at line 64 of file inputVar.f90.

5.21.2.96 `real(kind= prec), dimension(:, :), allocatable inputvar::uth`

Definition at line 94 of file inputVar.f90.

5.21.2.97 `logical inputvar::writeboi = .false.`

Definition at line 61 of file inputVar.f90.

5.21.2.98 `logical inputvar::writechi = .false.`

Definition at line 62 of file inputVar.f90.

5.21.2.99 logical inputvar::writechillingrev = .false.

Definition at line 56 of file inputVar.f90.

5.21.2.100 logical inputvar::writecosts = .false.

Definition at line 59 of file inputVar.f90.

5.21.2.101 logical inputvar::writedemand = .false.

Definition at line 57 of file inputVar.f90.

5.21.2.102 logical inputvar::writeefficiency = .false.

Definition at line 53 of file inputVar.f90.

5.21.2.103 logical inputvar::writeelectricrev = .false.

Definition at line 54 of file inputVar.f90.

5.21.2.104 logical inputvar::writeenergy = .false.

Definition at line 52 of file inputVar.f90.

5.21.2.105 logical inputvar::writeinput = .false.

Definition at line 58 of file inputVar.f90.

5.21.2.106 logical inputvar::writepower = .false.

Definition at line 51 of file inputVar.f90.

5.21.2.107 logical inputvar::writethermalrev = .false.

Definition at line 55 of file inputVar.f90.

5.21.2.108 logical inputvar::writetrig = .false.

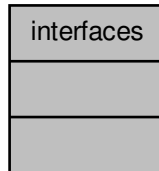
Definition at line 60 of file inputVar.f90.

The documentation for this module was generated from the following file:

- [/home/Codici/Blink/PowerManager/src/inputVar.f90](#)

5.22 interfaces Module Reference

Collaboration diagram for interfaces:



Data Types

- interface [abortExecution](#)
- interface [performances](#)
- interface [warning](#)

5.22.1 Detailed Description

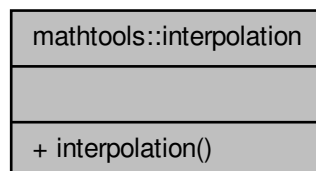
Definition at line 35 of file [interfaces.f90](#).

The documentation for this module was generated from the following file:

- [/home/Codici/Blink/PowerManager/src/interfaces.f90](#)

5.23 mathtools::interpolation Interface Reference

Collaboration diagram for mathtools::interpolation:



Public Member Functions

- `real(kind=prec) function,`
`dimension(m) interpolation (xIn, yIn, n, xOut, m, warn)`

5.23.1 Detailed Description

Definition at line 41 of file mathTools.f90.

5.23.2 Constructor & Destructor Documentation

5.23.2.1 `real(kind = prec) function, dimension(m) mathtools::interpolation::interpolation (real(kind = prec), dimension(n), intent(in) xIn, real(kind = prec), dimension(n), intent(in) yIn, integer, intent(in) n, real(kind = prec), dimension(m), intent(in) xOut, integer, intent(in) m, integer, dimension(2), intent(in), optional warn)`

Definition at line 41 of file mathTools.f90.

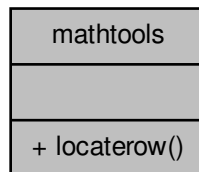
The documentation for this interface was generated from the following file:

- [/home/Codici/Blink/PowerManager/src/mathTools.f90](#)

5.24 mathtools Module Reference

Collection of interfaces for basic mathematical tools.

Collaboration diagram for mathtools:



Data Types

- interface [interpolation](#)

Public Member Functions

- integer function [locaterow](#) (row, mat, n, m, error)

5.24.1 Detailed Description

Collection of interfaces for basic mathematical tools.

Author

Andrea Facci.

Definition at line 36 of file mathTools.f90.

5.24.2 Member Function/Subroutine Documentation

5.24.2.1 integer function `mathtools::locaterow` (`real(kind = prec)`, `dimension(n)`, `intent(in) row`, `real(kind = prec)`, `dimension(m,n)`, `intent(in) mat`, `integer`, `intent(in) n`, `integer`, `intent(in) m`, `logical`, `intent(out)`, `optional error`)

Definition at line 54 of file `mathTools.f90`.

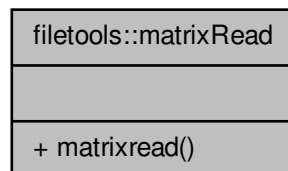
Referenced by `graphtools::graphpoints()`, and `graphtools::minpathtopobw()`.

The documentation for this module was generated from the following file:

- `/home/Codici/Blink/PowerManager/src/mathTools.f90`

5.25 filetools::matrixRead Interface Reference

Collaboration diagram for `filetools::matrixRead`:



Public Member Functions

- `character(len=100)` function, `dimension(nline)` `matrixread` (`theUnit`, `nline`, `first_`, `last_`)

5.25.1 Detailed Description

Definition at line 101 of file `fileTools.f90`.

5.25.2 Member Function/Subroutine Documentation

5.25.2.1 `character(len=100)` function, `dimension(nline)` `filetools::matrixRead::matrixread` (`integer`, `intent(in) theUnit`, `integer`, `intent(in) nline`, `character(len=1)`, `optional first_`, `character(len=1)`, `optional last_`)

Definition at line 101 of file `fileTools.f90`.

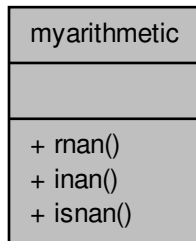
The documentation for this interface was generated from the following file:

- `/home/Codici/Blink/PowerManager/src/fileTools.f90`

5.26 myarithmetic Module Reference

Creates and detects NaN.

Collaboration diagram for myarithmetic:



Public Member Functions

- real(kind=prec) function [rnan](#) (x)
Creates NaN.
- integer function [inan](#) (x)
Creates NaN.
- logical function [isnan](#) (x)
Detects NaNs.

5.26.1 Detailed Description

This module collects some subroutine useful to create and detect NaNs. It tries to imitate the IEEE_ARITHMETIC module that unfortunately is still not available for the gfortran compiler.

Author

Definition at line 36 of file myArithmetic.f90.

5.26.2 Member Function/Subroutine Documentation

5.26.2.1 integer function myarithmetic::inan (integer, intent(in) x)

Creates a NaN to be associated to an integer, variable.

Parameters

<code>in</code>	<code>x</code>	random integer number.
-----------------	----------------	------------------------

Author

Andrea Facci.

Definition at line 61 of file myArithmetic.f90.

5.26.2.2 logical function `myarithmetric::isnan (x)`

This subroutine determine if a value is NaN.

Parameters

<code>in</code>	<code>x</code>	the value to be tested.
-----------------	----------------	-------------------------

Author

Andrea Facci

Definition at line 72 of file `myArithmetic.f90`.

5.26.2.3 `real(kind = prec)` function `myarithmetric::rnan (real(kind = prec), intent(in) x)`

Creates a NaN to be associated to a real, double precision variable.

Parameters

<code>in</code>	<code>x</code>	random double precision number.
-----------------	----------------	---------------------------------

Author

Andrea Facci.

Definition at line 47 of file `myArithmetic.f90`.

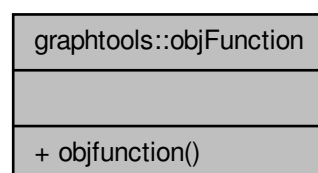
Referenced by `graphtools::allcombin()`, `buildplant()`, `readboiler()`, `readchillers()`, and `readtrigen()`.

The documentation for this module was generated from the following file:

- [/home/Codici/Blink/PowerManager/src/myArithmetic.f90](#)

5.27 graphtools::objFunction Interface Reference

Collaboration diagram for `graphtools::objFunction`:



Public Member Functions

- `real(kind=prec)` function [objfunction](#) (`c`, `t`, `obj`)

5.27.1 Detailed Description

Definition at line 59 of file graphTools.f90.

5.27.2 Member Function/Subroutine Documentation

5.27.2.1 `real(kind = prec) function graphtools::objFunction::objfunction (integer, dimension(nm), intent(in) c, integer, intent(in) t, character(len=100), intent(in) obj)`

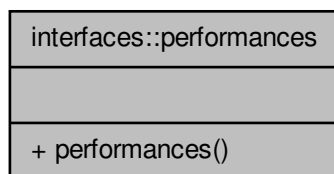
Definition at line 59 of file graphTools.f90.

The documentation for this interface was generated from the following file:

- [/home/Codici/Blink/PowerManager/src/graphTools.f90](#)

5.28 interfaces::performances Interface Reference

Collaboration diagram for interfaces::performances:



Public Member Functions

- subroutine [performances](#) (*c*, *cOld*, *equip*, *num*, *t*, *pEI*, *pTh*, *pCh*, *eln*, *mf*, *cfu*, *cm*, *cOn*)

5.28.1 Detailed Description

Definition at line 61 of file interfaces.f90.

5.28.2 Constructor & Destructor Documentation

5.28.2.1 `subroutine interfaces::performances::performances (integer, intent(in) c, integer, intent(in), optional cOld, character(len=*), intent(in) equip, integer, intent(in) num, integer, intent(in) t, real(kind = prec), intent(out), optional pEI, real(kind = prec), intent(out), optional pTh, real(kind = prec), intent(out), optional pCh, real(kind = prec), intent(out), optional eln, real(kind = prec), intent(out), optional mf, real(kind = prec), intent(out), optional cfu, real(kind = prec), intent(out), optional cm, real(kind = prec), intent(out), optional cOn)`

Definition at line 61 of file interfaces.f90.

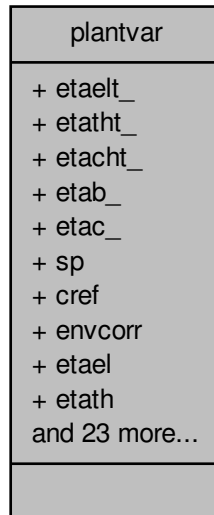
The documentation for this interface was generated from the following file:

- [/home/Codici/Blink/PowerManager/src/interfaces.f90](#)

5.29 plantvar Module Reference

collection of variables relative to the power plant structure.

Collaboration diagram for plantvar:



Public Attributes

- `real(kind=prec), dimension(:,:), allocatable etaelt_`
- `real(kind=prec), dimension(:,:), allocatable etatht_`
- `real(kind=prec), dimension(:,:), allocatable etacht_`
- `real(kind=prec), dimension(:,:), allocatable etab_`
- `real(kind=prec), dimension(:,:), allocatable etac_`
- `real(kind=prec), dimension(:,:), allocatable sp`
- `real(kind=prec), dimension(:,:), allocatable cref`
- `real(kind=prec), dimension(:,:,:), allocatable envcorr`
- `real(kind=prec), dimension(:,:), allocatable etael`
- `real(kind=prec), dimension(:,:), allocatable etath`
- `real(kind=prec), dimension(:,:), allocatable etach`
- `real(kind=prec), dimension(:,:), allocatable timevinc`

- integer, dimension(:,:), allocatable [cr](#)
- real(kind=prec), dimension(:), allocatable [pmax](#)
- real(kind=prec), dimension(:), allocatable [dt](#)
- real(kind=prec), dimension(:), allocatable [cf](#)
- real(kind=prec), dimension(:), allocatable [lhv](#)
- real(kind=prec), dimension(:), allocatable [onoffcost](#)
- real(kind=prec), dimension(:), allocatable [oemcost](#)
- real(kind=prec), dimension(:), allocatable [minuptime](#)
- real(kind=prec), dimension(:), allocatable [mindowntime](#)
- character(len=4), dimension(:), allocatable [pes](#)
- integer [nsptot](#)
- integer [nm](#)
- integer [it](#)
- integer [ib](#)
- integer [ic](#)
- integer, dimension(3) [is](#)
- integer, dimension(3) [ie](#)
- integer, dimension(:), allocatable [nsp](#)
- integer, dimension(:), allocatable [ntv](#)
- integer, dimension(:), allocatable [esource](#)
- character(len=50), dimension(:), allocatable [tec](#)

5.29.1 Detailed Description

collection of variables relative to the power plant structure.

Author

Definition at line 35 of file plantVar.f90.

5.29.2 Member Data Documentation

5.29.2.1 real(kind = prec), dimension(:), allocatable plantvar::cf

Definition at line 44 of file plantVar.f90.

5.29.2.2 integer, dimension(:,:), allocatable plantvar::cr

Definition at line 43 of file plantVar.f90.

5.29.2.3 `real(kind = prec), dimension(:,,:), allocatable plantvar::cref`

Definition at line 39 of file plantVar.f90.

5.29.2.4 `real(kind = prec), dimension(:,), allocatable plantvar::dt`

Definition at line 44 of file plantVar.f90.

5.29.2.5 `real(kind = prec), dimension(:,,:), allocatable plantvar::envcorr`

Definition at line 41 of file plantVar.f90.

5.29.2.6 `integer, dimension(:,), allocatable plantvar::esource`

Definition at line 48 of file plantVar.f90.

5.29.2.7 `real(kind = prec), dimension(:,,:), allocatable plantvar::etab_`

Definition at line 39 of file plantVar.f90.

5.29.2.8 `real(kind = prec), dimension(:,,:), allocatable plantvar::etac_`

Definition at line 39 of file plantVar.f90.

5.29.2.9 `real(kind = prec), dimension(:,,:), allocatable plantvar::etach`

Definition at line 42 of file plantVar.f90.

5.29.2.10 `real(kind = prec), dimension(:,,:), allocatable plantvar::etacht_`

Definition at line 39 of file plantVar.f90.

5.29.2.11 `real(kind = prec), dimension(:,,:), allocatable plantvar::etael`

Definition at line 42 of file plantVar.f90.

5.29.2.12 `real(kind = prec), dimension(:,,:), allocatable plantvar::etaelt_`

Definition at line 39 of file plantVar.f90.

5.29.2.13 `real(kind = prec), dimension(:,,:), allocatable plantvar::etath`

Definition at line 42 of file plantVar.f90.

5.29.2.14 `real(kind = prec), dimension(:,,:), allocatable plantvar::etatht_`

Definition at line 39 of file plantVar.f90.

5.29.2.15 integer plantvar::ib

Definition at line 46 of file plantVar.f90.

5.29.2.16 integer plantvar::ic

Definition at line 46 of file plantVar.f90.

5.29.2.17 integer, dimension(3) plantvar::ie

Definition at line 47 of file plantVar.f90.

5.29.2.18 integer, dimension(3) plantvar::is

Definition at line 47 of file plantVar.f90.

5.29.2.19 integer plantvar::it

Definition at line 46 of file plantVar.f90.

5.29.2.20 real(kind = prec), dimension(:), allocatable plantvar::lhw

Definition at line 44 of file plantVar.f90.

5.29.2.21 real(kind = prec), dimension(:), allocatable plantvar::mindowntime

Definition at line 44 of file plantVar.f90.

5.29.2.22 real(kind = prec), dimension(:), allocatable plantvar::minuptime

Definition at line 44 of file plantVar.f90.

5.29.2.23 integer plantvar::nm

Definition at line 46 of file plantVar.f90.

5.29.2.24 integer, dimension(:), allocatable plantvar::nsp

Definition at line 48 of file plantVar.f90.

5.29.2.25 integer plantvar::nsptot

Definition at line 46 of file plantVar.f90.

5.29.2.26 integer, dimension(:), allocatable plantvar::ntv

Definition at line 48 of file plantVar.f90.

5.29.2.27 `real(kind = prec), dimension(:), allocatable plantvar::oemcost`

Definition at line 44 of file plantVar.f90.

5.29.2.28 `real(kind = prec), dimension(:), allocatable plantvar::onoffcost`

Definition at line 44 of file plantVar.f90.

5.29.2.29 `character(len=4), dimension(:), allocatable plantvar::pes`

Definition at line 45 of file plantVar.f90.

5.29.2.30 `real(kind = prec), dimension(:), allocatable plantvar::pmax`

Definition at line 44 of file plantVar.f90.

5.29.2.31 `real(kind = prec), dimension(:, :), allocatable plantvar::sp`

Definition at line 39 of file plantVar.f90.

5.29.2.32 `character(len=50), dimension(:), allocatable plantvar::tec`

Definition at line 49 of file plantVar.f90.

5.29.2.33 `real(kind = prec), dimension(:, :), allocatable plantvar::timevinc`

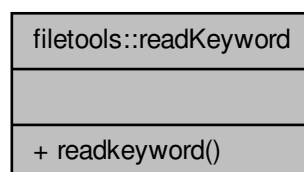
Definition at line 42 of file plantVar.f90.

The documentation for this module was generated from the following file:

- [/home/Codici/Blink/PowerManager/src/plantVar.f90](#)

5.30 filetools::readKeyword Interface Reference

Collaboration diagram for filetools::readKeyword:



Public Member Functions

- subroutine [readkeyword](#) (theUnit, rew, keyword, value, error, nRow)

5.30.1 Detailed Description

Definition at line 166 of file fileTools.f90.

5.30.2 Member Function/Subroutine Documentation

- 5.30.2.1 subroutine filetools::readKeyword::readkeyword (integer, intent(in) *theUnit*, logical, intent(in), optional *rew*, character(len=100), intent(out) *keyword*, character(len=100), intent(out) *value*, integer, intent(out), optional *error*, integer, intent(out), optional *nRow*)

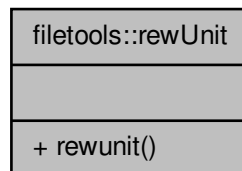
Definition at line 166 of file fileTools.f90.

The documentation for this interface was generated from the following file:

- /home/Codici/Blink/PowerManager/src/[fileTools.f90](#)

5.31 filetools::rewUnit Interface Reference

Collaboration diagram for filetools::rewUnit:



Public Member Functions

- subroutine [rewunit](#) (theUnit, n)

5.31.1 Detailed Description

Definition at line 44 of file fileTools.f90.

5.31.2 Member Function/Subroutine Documentation

- 5.31.2.1 subroutine filetools::rewUnit::rewunit (integer, intent(in) *theUnit*, integer, intent(in) *n*)

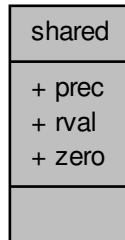
Definition at line 44 of file fileTools.f90.

The documentation for this interface was generated from the following file:

- [/home/Codici/Blink/PowerManager/src/fileTools.f90](#)

5.32 shared Module Reference

Collaboration diagram for shared:



Public Attributes

- integer, parameter [prec](#) = kind(1.0)
- real(kind=[prec](#)), parameter [rval](#) = 1.0
- real(kind=[prec](#)), parameter [zero](#) = 0.0

5.32.1 Detailed Description

Definition at line 32 of file [shared.f90](#).

5.32.2 Member Data Documentation

5.32.2.1 integer, parameter [shared::prec](#) = kind(1.0)

Definition at line 38 of file [shared.f90](#).

5.32.2.2 real(kind = [prec](#)), parameter [shared::rval](#) = 1.0

Definition at line 39 of file [shared.f90](#).

5.32.2.3 real(kind = [prec](#)), parameter [shared::zero](#) = 0.0

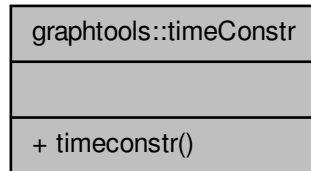
Definition at line 40 of file [shared.f90](#).

The documentation for this module was generated from the following file:

- [/home/Codici/Blink/PowerManager/src/shared.f90](#)

5.33 graphtools::timeConstr Interface Reference

Collaboration diagram for graphtools::timeConstr:



Public Member Functions

- logical function [timeconstr](#) (cindex, tState)

5.33.1 Detailed Description

Definition at line 82 of file graphTools.f90.

5.33.2 Member Function/Subroutine Documentation

- 5.33.2.1 logical function `graphtools::timeConstr::timeconstr` (integer, dimension(nm), intent(in) *cindex*, real(kind = prec), dimension(nm), intent(in) *tState*)

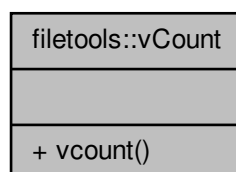
Definition at line 82 of file graphTools.f90.

The documentation for this interface was generated from the following file:

- [/home/Codici/Blink/PowerManager/src/graphTools.f90](#)

5.34 filetools::vCount Interface Reference

Collaboration diagram for filetools::vCount:



Public Member Functions

- integer function [vcount](#) (theUnit, rew_, first_, last_)

5.34.1 Detailed Description

Definition at line 52 of file fileTools.f90.

5.34.2 Member Function/Subroutine Documentation

5.34.2.1 integer function filetools::vCount::vcount (integer, intent(in) *theUnit*, logical, optional *rew_*, character(len=1), intent(in), optional *first_*, character(len=1), intent(in), optional *last_*)

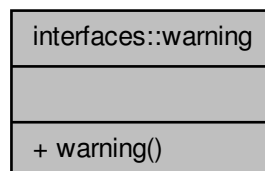
Definition at line 52 of file fileTools.f90.

The documentation for this interface was generated from the following file:

- /home/Codici/Blink/PowerManager/src/[fileTools.f90](#)

5.35 interfaces::warning Interface Reference

Collaboration diagram for interfaces::warning:



Public Member Functions

- subroutine [warning](#) (i, j, k, line, word, r1, r2)

5.35.1 Detailed Description

Definition at line 50 of file interfaces.f90.

5.35.2 Constructor & Destructor Documentation

5.35.2.1 subroutine interfaces::warning::warning (integer, intent(in), optional *i*, integer, intent(in), optional *j*, integer, intent(in), optional *k*, integer, intent(in), optional *line*, character(len=*), intent(in), optional *word*, real(kind = prec), intent(in), optional *r1*, real(kind = prec), intent(in), optional *r2*)

Definition at line 50 of file interfaces.f90.

The documentation for this interface was generated from the following file:

- [/home/Codici/Blink/PowerManager/src/interfaces.f90](#)

Chapter 6

File Documentation

6.1 /home/Codici/Blink/PowerManager/src/abortExecution.f90 File Reference

terminates the execution in case of error.

Functions/Subroutines

- subroutine [abortexecution](#) (*i*, *j*, *line*, *word*, *r1*, *r2*, *iVec*)
abort the program execution.

6.1.1 Detailed Description

aborts the program execution in case of error and print the error message according to the error code given as input.

Author

Andrea Facci.

Definition in file [abortExecution.f90](#).

6.1.2 Function/Subroutine Documentation

6.1.2.1 subroutine `abortexecution` (`integer, intent(in), optional i`, `integer, intent(in), optional j`, `integer, intent(in), optional line`, `character(len=*)`, `intent(in), optional word`, `real(kind = prec)`, `intent(in), optional r1`, `real(kind = prec)`, `intent(in), optional r2`, `real(kind = prec)`, `dimension(*)`, `intent(in), optional iVec`)

aborts the program execution in case of error and print the error message according to the error code given as input.

Parameters

<code>in</code>	<code>i,j</code>	integers that identify the error.
<code>in</code>	<code>line</code>	integer that identifies the line affected by the mistake in an input file
<code>in</code>	<code>word</code>	character input to report the misspelled or unexpected sentence
<code>in</code>	<code>r1,r2</code>	double precision reals. Useful to report incoherence between plant parameters

Definition at line 42 of file `abortExecution.f90`.

Referenced by `graphtools::allcombin()`, `checkplant()`, `graphtools::graphpoints()`, `readboiler()`, `readchillers()`, `readenv()`, `readgeneral()`, `readloads()`, and `readtrigen()`.

6.2 /home/Codici/Blink/PowerManager/src/aiuto.f90 File Reference

prints a very short help.

Functions/Subroutines

- subroutine [aiuto](#)
prints a very short help.

6.2.1 Detailed Description

prints a very short help.

Author

Andrea Facci.

Definition in file [aiuto.f90](#).

6.2.2 Function/Subroutine Documentation

6.2.2.1 subroutine `aiuto` ()

prints a very short help. This is called if `-help` option is given from the command line.

Author

Andrea Facci.

Definition at line 39 of file `aiuto.f90`.

References `endexecution()`.

Referenced by `commandline()`.

6.3 /home/Codici/Blink/PowerManager/src/allocateVar.f90 File Reference

variable allocation.

Functions/Subroutines

- subroutine [allocatevar](#) (what, num)
variable allocation

6.3.1 Detailed Description

Author

Definition in file [allocateVar.f90](#).

6.3.2 Function/Subroutine Documentation

6.3.2.1 subroutine `allocatevar` (`integer`, `intent(in)` *what*, `integer`, `intent(in)`, optional *num*)

Allocates the variables according to the inputs. All the allocation of "global" variables should be done here.

Parameters

<code>in</code>	<i>what</i>	defines the variables to be allocated
<code>in</code>	<i>num</i>	defines the dimension of the array

Author

Andrea Facci

Definition at line 36 of file `allocateVar.f90`.

Referenced by `buildplant()`, `readboiler()`, `readchillers()`, `readenv()`, `readgeneral()`, `readloads()`, and `readtrigen()`.

6.4 /home/Codici/Blink/PowerManager/src/buildPlant.f90 File Reference

Collects all the informations relative to the power plant.

Functions/Subroutines

- subroutine [buildplant](#)
Collects all the informations relative to the power plant.

6.4.1 Detailed Description

Author

Definition in file [buildPlant.f90](#).

6.4.2 Function/Subroutine Documentation

6.4.2.1 subroutine `buildplant` ()

Collects all the informations relative to the power plant starting from the Inputs of the single machinery. Calculates the efficiencies for each given set-point and store them in a single array. Perform units of measure conversions when necessary.

Author

Definition at line 40 of file `buildPlant.f90`.

References `allocatevar()`, `checkplant()`, `deallocatevar()`, and `myarithmetic::rnan()`.

Referenced by `main()`.

6.5 /home/Codici/Blink/PowerManager/src/checkPlant.f90 File Reference

Check the power plant coherence with the energy demand.

Functions/Subroutines

- subroutine [checkplant](#)
Check the power plant coherence with the energy demand.

6.5.1 Detailed Description

Author

Definition in file [checkPlant.f90](#).

6.5.2 Function/Subroutine Documentation

6.5.2.1 subroutine checkplant ()

Check the power plant coherence with the energy demand. Specifically the maximum thermal and chilling power must be higher than the relative energy demand. Thermal load includes also thermal energy needed by absorption chillers. If the plant is not grid connected also rated electrical power must be greater than maximum electrical demand.

Author

Definition at line 41 of file checkPlant.f90.

References [abortexecution\(\)](#).

Referenced by [buildplant\(\)](#).

6.6 /home/Codici/Blink/PowerManager/src/cmdVar.f90 File Reference

Collects the variable read from command line.

Data Types

- module [cmdvar](#)
Collects the variable read from command line.

6.6.1 Detailed Description

Collects the variable read from command line.

Author

Definition in file [cmdVar.f90](#).

6.7 /home/Codici/Blink/PowerManager/src/cmdVarWr.f90 File Reference

Data Types

- module [cmdvarwr](#)
Collects the variable read from command line.

6.8 /home/Codici/Blink/PowerManager/src/command.f90 File Reference

Reads the command line.

Data Types

- module [command](#)

6.8.1 Detailed Description

subroutine to read the execution options from command line.

Author

Andrea Facci

Definition in file [command.f90](#).

6.9 /home/Codici/Blink/PowerManager/src/commandline.f90 File Reference

Reads the command line.

Functions/Subroutines

- subroutine [commandline](#)

6.9.1 Detailed Description

subroutine to read the execution options from command line.

Author

Andrea Facci

Definition in file [commandline.f90](#).

6.9.2 Function/Subroutine Documentation

6.9.2.1 subroutine [commandline](#) ()

Definition at line 35 of file [commandline.f90](#).

References [aiuto\(\)](#).

Referenced by [main\(\)](#).

6.10 /home/Codici/Blink/PowerManager/src/constraints.f90 File Reference

static constraints.

Functions/Subroutines

- logical function [constraints](#) (c, t)
static constraints
- logical function [timeconstr](#) (cindex, tState)

6.10.1 Detailed Description

This file contains a subroutine that checks if the plant respects the energy constraints for a given set-point and time-step.

Definition in file [constraints.f90](#).

6.10.2 Function/Subroutine Documentation

6.10.2.1 logical function constraints (integer, dimension(nm), intent(in) c, integer, intent(in) t)

This function checks if the plant respects the energy constraints for a given set-point and time-step. The constraints are:

- Thermal Power: $\sum U_{th} + U_{th}^{self} \leq \sum P_{th}$
- Chilling Power: $\sum U_{ch} \leq \sum P_{ch}$
- Electrical Power : $\sum U_{el} + U_{el}^{self} \leq \sum P_{el}$ only if the pant is bnot grid connected

Parameters

in	c	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_{-}(i))$
in	t	time step index. Note t=x meas the x'th time step from the

Author

Andrea Facci

Definition at line 46 of file constraints.f90.

References [energy::chprod\(\)](#), [energy::thprod\(\)](#), and [energy::thselfcons\(\)](#).

6.10.2.2 logical function timeconstr (integer, dimension(nm), intent(in) cindex, real(kind = prec), dimension(2*nm), intent(in) tState)

Definition at line 94 of file constraints.f90.

Referenced by [graphtools::minpathtopobw\(\)](#).

6.11 /home/Codici/Blink/PowerManager/src/deallocateVar.f90 File Reference

Functions/Subroutines

- subroutine [deallocatevar](#) (what, num)

variable allocation

6.11.1 Function/Subroutine Documentation

6.11.1.1 subroutine deallocatevar (integer, intent(in) *what*, integer, intent(in), optional *num*)

Allocates the variables according to the inputs. All the allocation of "global" variables should be done here.

Parameters

<i>in</i>	<i>what</i>	defines the variables to be allocated
<i>in</i>	<i>num</i>	defines the dimension of the array

Author

Andrea Facci

Definition at line 36 of file deallocateVar.f90.

Referenced by buildplant().

6.12 /home/Codici/Blink/PowerManager/src/economy.f90 File Reference

costs and revenues calculation procedures.

Data Types

- module [economy](#)

6.12.1 Detailed Description

This module contains the definition of all the procedures that perform economic calculations for a given set-point and time-step, that are, fuel costs, O&M costs, and the revenues from thermal, electric, and chilling, energy selling.

Definition in file [economy.f90](#).

6.13 /home/Codici/Blink/PowerManager/src/endExecution.f90 File Reference

normal termination of the execution.

Functions/Subroutines

- subroutine [endexecution](#)
normal termination of the execution.

6.13.1 Detailed Description

Author

Definition in file [endExecution.f90](#).

6.13.2 Function/Subroutine Documentation

6.13.2.1 subroutine endexecution ()

This procedure is called for the normal termination of the program execution.

Author

Definition at line 35 of file endExecution.f90.

Referenced by aiuto(), and main().

6.14 /home/Codici/Blink/PowerManager/src/energy.f90 File Reference

Collection of function for energy flow calculation.

Data Types

- module [energy](#)
module for energy calculations.

6.14.1 Detailed Description

This file contains a module that collects all the procedure for energy calculations

Author

Andrea Facci.

Definition in file [energy.f90](#).

6.15 /home/Codici/Blink/PowerManager/src/euristics.f90 File Reference

Module that contains the function to apply an euristics to the graph.

Data Types

- module [euristics](#)
Module that contains the function to apply an euristics to the graph.
- interface [euristics::constraints](#)

6.15.1 Detailed Description

Module that contains the function to apply an euristics to the graph. in order to reduce the number of points and arcs of the graph.

Author

Andrea Facci.

Definition in file [euristics.f90](#).

6.16 /home/Codici/Blink/PowerManager/src/fileTools.f90 File Reference

collection of procedure interfaces useful to read from files.

Data Types

- module [filetools](#)
Interfaces of procedures to read from file.
- interface [filetools::rewUnit](#)
- interface [filetools::vCount](#)
- interface [filetools::hCount](#)
- interface [filetools::dMatrixRead](#)
- interface [filetools::iMatrixRead](#)
- interface [filetools::cMatrixRead](#)
- interface [filetools::matrixRead](#)
- interface [filetools::iFindEntry](#)
- interface [filetools::dFindEntry](#)
- interface [filetools::cFindEntry](#)
- interface [filetools::findEntry](#)
- interface [filetools::readKeyword](#)

6.16.1 Detailed Description

Author

Definition in file [fileTools.f90](#).

6.17 /home/Codici/Blink/PowerManager/src/findEntry2.f90 File Reference

Collection of procedures to find a specific entry inside a file.

Functions/Subroutines

- subroutine [ifindentry](#) (entry, n, theUnit, rew, valore, isPresent, nRow)
Finds an entry with integer value.
- subroutine [dfindentry](#) (entry, n, theUnit, rew, valore, isPresent, nRow)
Finds an entry with double precision value.
- subroutine [cfindentry](#) (entry, n, theUnit, rew, valore, isPresent, nRow)
Finds an entry with character value.
- subroutine [findentry](#) (entry, theUnit, rew, valore, isPresent, nRow)
Finds an entry and returns the value as it is.

6.17.1 Detailed Description

Author

Definition in file [findEntry2.f90](#).

6.17.2 Function/Subroutine Documentation

6.17.2.1 `subroutine cfindentry (character(len=*), intent(in) entry, integer, intent(in) n, integer, intent(in) theUnit, logical, intent(in), optional rew, character(len=100), dimension(n), intent(out), optional valore, logical, intent(out), optional isPresent, integer, intent(out), optional nRow)`

Use this subroutine to find a specific entry in a file, when an character value is expected. in the value field, that is between the two "|". The string in the value field is associated to a character array of dimension "n" Optionally it returns also the number of lines from the current cursor position to locate the required entry, and a logical for the precence of the required field. Optionally it is also possible to rewind the file. Note that the maximum length of each element of the "value" vector is 100.

Parameters

in	<i>entry</i>	the keyword to be located
in	<i>n</i>	the number of elements expected for the value field
in	<i>theUnit</i>	the unit corresponding to the file to be searched
in	<i>rew</i>	wether the unit is to be rewinded or not
out	<i>valore</i>	the vaule correspondind to keyword that is searched for
out	<i>isPresent</i>	logical that tells if the entry is present or not
out	<i>nRow</i>	number of rows needed to located the entry.

Author

Andrea Facci

Definition at line 187 of file findEntry2.f90.

References findentry().

Referenced by readchillers().

6.17.2.2 `subroutine dfindentry (character(len=*), intent(in) entry, integer, intent(in) n, integer, intent(in) theUnit, logical, intent(in), optional rew, real(kind = prec), dimension(n), intent(out), optional valore, logical, intent(out), optional isPresent, integer, intent(out), optional nRow)`

Use this subroutine to find a specific entry in a file, when a double precision value is expected. in the value field, that is between the two "|". The string in the value field is associated to double precision array of dimension "n" Optionally it returns also the number of lines from the current cursor position to locate the required entry, and a logical for the precence of the required field. Optionally it is also possible to rewind the file.

Parameters

in	<i>entry</i>	the keyword to be located
in	<i>n</i>	the number of elements expected for the value field
in	<i>theUnit</i>	the unit corresponding to the file to be searched
in	<i>rew</i>	wether the unit is to be rewinded or not
out	<i>valore</i>	the vaule correspondind to keyword that is searched for
out	<i>isPresent</i>	logical that tells if the entry is present or not
out	<i>nRow</i>	number of rows needed to located the entry.

Author

Andrea Facci

Definition at line 117 of file findEntry2.f90.

References findentry().

6.17.2.3 subroutine findentry (character(len=*) intent(in) entry, integer, intent(in) theUnit, logical, intent(in), optional rew, character(len=100), intent(out), optional valore, logical, intent(out), optional isPresent, integer, intent(out), optional nRow)

Use this subroutine to find a specific entry in a file. If given as parameter returns the "value" field, that is the value comprised between the two "|" in the input file. This subroutine does not associate the value field to any array but returns a single string exactly as it is in the input file. Optionally it returns also the number of lines from the current cursor position to locate the required entry, and a logical for the presence of the required field. Optionally it is also possible to rewind the file.

Parameters

in	entry	the keyword to be located
in	n	the number of elements expected for the value field
in	theUnit	the unit corresponding to the file to be searched
in	rew	whether the unit is to be rewinded or not
out	valore	the value corresponding to keyword that is searched for
out	isPresent	logical that tells if the entry is present or not
out	nRow	number of rows needed to locate the entry.

Author

Andrea Facci

Definition at line 256 of file findEntry2.f90.

References rewunit().

Referenced by cfindexentry(), dfindexentry(), ifindexentry(), and readloads().

6.17.2.4 subroutine ifindexentry (character(len=*) intent(in) entry, integer, intent(in) n, integer, intent(in) theUnit, logical, intent(in), optional rew, integer, dimension(n), intent(out), optional valore, logical, intent(out), optional isPresent, integer, intent(out), optional nRow)

Use this subroutine to find a specific entry in a file, when an integer value is expected in the value field, that is between the two "|". The string in the value field is associated to an integer array of dimension "n". Optionally it returns also the number of lines from the current cursor position to locate the required entry, and a logical for the presence of the required field. Optionally it is also possible to rewind the file.

Parameters

in	entry	the keyword to be located
in	n	the number of elements expected for the value field
in	theUnit	the unit corresponding to the file to be searched
in	rew	whether the unit is to be rewinded or not
out	valore	the value corresponding to keyword that is searched for
out	isPresent	logical that tells if the entry is present or not
out	nRow	number of rows needed to locate the entry.

Author

Andrea Facci

Definition at line 47 of file findEntry2.f90.

References findentry().

Referenced by readboiler(), readchillers(), and readtrigen().

6.18 /home/Codici/Blink/PowerManager/src/globalResults.f90 File Reference

Data Types

- module [globalresults](#)

6.19 /home/Codici/Blink/PowerManager/src/graphTools.f90 File Reference

graph construction and mininum path.

Data Types

- module [graphtools](#)
- interface [graphtools::objFunction](#)
- interface [graphtools::constraints](#)
- interface [graphtools::timeConstr](#)

6.19.1 Detailed Description

this file implements a module (graphTools) that contains all the routines necessary to build the graph representing the problem and find the mininum path across the graph. The graph is acyclic (no closed paths) and represented in topological ordering using a predecessor list.

Author

Andrea Facci.

Definition in file [graphTools.f90](#).

6.20 /home/Codici/Blink/PowerManager/src/hCount.f90 File Reference

Count the number of elements of a vector in a text file.

Functions/Subroutines

- integer function [hcount](#) (value_, first_, last_)
Count the number of elements of a vector in a text file.

6.20.1 Detailed Description

Author

Andrea Facci.

Definition in file [hCount.f90](#).

6.20.2 Function/Subroutine Documentation

6.20.2.1 integer function hcount (character(len=100), intent(in) *value_*, character(len=1), optional *first_*, character(len=1), optional *last_*)

Use This subroutine to determine the length of a vector, that is a series of values enclosed between two delimiters, when reading from text file. The vector must be specified as a single string input (max length=100), while delimiters may be specified as single character input or left to default that is "(" for opening and ")" for closing.

Parameters

in	<i>value_</i>	the string containing the vector whose length is to be determined
in	<i>first_</i> , <i>last_</i>	single character delimiters of the vector (open and close respectively). If not provided "(" and ")" will be assumed as defaults

Author

Andrea Facci.

Definition at line 43 of file hCount.f90.

Referenced by readboiler(), readchillers(), readgeneral(), readloads(), and readtrigen().

6.21 /home/Codici/Blink/PowerManager/src/inputVar.f90 File Reference

Input variables collection.

Data Types

- module [inputvar](#)
Input variables collection.

6.21.1 Detailed Description

Author

Definition in file [inputVar.f90](#).

6.22 /home/Codici/Blink/PowerManager/src/interfaces.f90 File Reference

Collection of general purpose interfaces.

Data Types

- module [interfaces](#)
- interface [interfaces::abortExecution](#)
- interface [interfaces::warning](#)
- interface [interfaces::performances](#)

6.22.1 Detailed Description

Author

Definition in file [interfaces.f90](#).

6.23 /home/Codici/Blink/PowerManager/src/interpolation.f90 File Reference

Remaps a discrete scalar field on a given 1d grid.

Functions/Subroutines

- `real(kind=prec) function,`
`dimension(m) interpolation (xIn, yIn, n, xOut, m, warn)`
Remaps a discrete scalar field on a given 1d grid.

6.23.1 Detailed Description

Author

Andrea Facci.

Definition in file [interpolation.f90](#).

6.23.2 Function/Subroutine Documentation

6.23.2.1 `real(kind = prec) function, dimension(m) interpolation (real(kind = prec), dimension(n), intent(in) xIn, real(kind = prec), dimension(n), intent(in) yIn, integer, intent(in) n, real(kind = prec), dimension(m), intent(in) xOut, integer, intent(in) m, integer, dimension(2), intent(out), optional warn)`

This subroutine takes a discrete scalar field defined over a 1d grid and remaps it on another 1d mesh given as input. If any of the values in the new mesh is outside the range defined by the original grid, values are extrapolated and an optional warning code is returned.

Parameters

in	<i>xIn,yIn</i>	grid and values of the 1d field to be mapped
in	<i>n</i>	number of elements of the discrete field (size(xIn))
in	<i>xOut</i>	1d grid where the field is sampled
in	<i>m</i>	number of elements of the interpolation grid (xOut)
out	<i>warn</i>	warning code.

Author

Andrea Facci.

Definition at line 46 of file [interpolation.f90](#).

6.24 /home/Codici/Blink/PowerManager/src/main.f90 File Reference

this is the main driver.

Functions/Subroutines

- program [main](#)

this is the main driver.

6.24.1 Detailed Description

main driver

Author

Andrea Facci.

Definition in file [main.f90](#).

6.24.2 Function/Subroutine Documentation

6.24.2.1 program main ()

main driver

Author

Andrea Facci.

Definition at line 37 of file main.f90.

References [graphtools::allcombin\(\)](#), [buildplant\(\)](#), [commandline\(\)](#), [endexecution\(\)](#), [graphtools::grapharcs\(\)](#), [graphtools::graphpoints\(\)](#), [graphtools::minpathtopobw\(\)](#), [graphtools::minpathtopofw\(\)](#), [output\(\)](#), [readboiler\(\)](#), [readchillers\(\)](#), [readenv\(\)](#), [readgeneral\(\)](#), [readloads\(\)](#), and [readtrigen\(\)](#).

6.25 /home/Codici/Blink/PowerManager/src/manual.f90 File Reference

6.26 /home/Codici/Blink/PowerManager/src/mathTools.f90 File Reference

Collection of interfaces for basic mathematical tools.

Data Types

- module [mathtools](#)

Collection of interfaces for basic mathematical tools.

- interface [mathtools::interpolation](#)

6.26.1 Detailed Description

Author

Andrea Facci.

Definition in file [mathTools.f90](#).

6.27 /home/Codici/Blink/PowerManager/src/matrixRead2.f90 File Reference

reads 2D array of values from a file

Functions/Subroutines

- `real(kind=prec) function,`
`dimension(nline, ncol) dmatrixread (theUnit, nline, ncol, first_, last_)`
reads 2D array of double precision values from a file.
- `integer function, dimension(nline,`
`ncol) imatrixread (theUnit, nline, ncol, first_, last_)`
reads 2D array of integer values from a file.
- `character(len=100) function,`
`dimension(nline, ncol) cmatrixread (theUnit, nline, ncol, first_, last_)`
reads 2D array of character values from a file.
- `character(len=100) function,`
`dimension(nline) matrixread (theUnit, nline, first_, last_)`
reads 2D array from a file.

6.27.1 Detailed Description

Collection of procedures to read 2D arrays from files. Each function associates the values to a different data type (integer, double, character). the first letter of the function name indicates the data type.

Author

Andrea Facci.

Definition in file [matrixRead2.f90](#).

6.27.2 Function/Subroutine Documentation

6.27.2.1 `character(len=100) function, dimension(nline,ncol) cmatrixread (integer, intent(in) theUnit, integer, intent(in) nline, integer, intent(in) ncol, character(len=1), optional first_, character(len=1), optional last_)`

This subroutine reads a 2D array of character values directly from a specified unit. The unit must be already opened and associated to the desired file. All the values to read must be written in "value" field of the text file that is between the two "|" (see the following box for an example of input text) and delimited by appropriate opening and closure delimiters.

```
A Lot of useless stuff because only text between begin and end is read.
begin
  !Commented line
  KeywordField | ValueField | !Comment two
end
```

Moreover the cursor needs to be before the first line of the array to be read. The procedure automatically skips all the lines until the opening character. Opening and closing character may be specified as single character input, or left to the default values that are "(" and ")", respectively. Avoid blank or commented lines inside the text array.

The correct file syntax to read the 2x2 array "exArr":

$$exArr = \begin{bmatrix} Scrudge & DonaldDuck \\ Goofy & MickeyMouse \end{bmatrix}$$

is:

```

begin
  !Comment one
  exArr | (Scrudge DonadDuck | !Comment two
          | Goofy MikeyMouse) |
end

```

or equivalently:

```

begin
  !Comment one
  exArr |          | !Comment two
          | (Scrudge DonaldDuck |
          | Goofy MickeyMouse) |
end

```

so that each line in the text represnts a row of the vector output and colons are space separated.

Parameters

in	<i>TheUnit</i>	the unit to be read.
in	<i>nline</i>	The number of rows of the 2D array
in	<i>ncol</i>	The number of columns of the 2D array
in	<i>first_</i>	the opening delimiter of the array. Default is "("
in	<i>last_</i>	the closing delimiter of the array. Default is ")"

Author

Andrea Facci.

Definition at line 289 of file matrixRead2.f90.

References matrixread().

Referenced by readenv(), and readloads().

6.27.2.2 `real(kind = prec) function, dimension(nline,ncol) dmatrixread (integer, intent(in) theUnit, integer, intent(in) nline, integer, intent(in) ncol, character(len=1), optional first_, character(len=1), optional last_)`

This subroutine reads a 2D array of double precision values directly from a specified unit. The unit must be already opened and associated to the desired file. All the values to read must be written in "value" field of the text file that is between the two "|" (see the following box for an example of input text) and delimited by appropriate opening and closure delimiters.

```

A Lot of usless stuff because only text between begin and end is read.
begin
  !Commented line
  KeywordField | ValueField | !Comment two
end

```

Moreover the cursor needs to be before the first line of the array to be read. The procedure atomatically skeeps all the lines until the opening character. Opening and closing character may be specified as single character input, or left to the defult values that are "(" and ")", respectively. Avoid blank or commented lines inside the text array.

The correct file syntax to read the 2x2 array "exArr":

$$exArr = \begin{bmatrix} 1.1 & 1.2 \\ 2.1 & 2.2 \end{bmatrix}$$

is:

```

begin
  !Comment one
  exArr | (1.1 1.2 | !Comment two
          | 2.1 2.2) |
end

```

or equivalently:

```
begin
  !Comment one
  exArr |      | !Comment two
        | (1.1 1.2 |
        | 2.1 2.2) |
end
```

so that each line in the text represents a row of the vector output and colons are space separated.

Parameters

in	<i>TheUnit</i>	the unit to be read.
in	<i>nline</i>	The number of rows of the 2D array
in	<i>ncol</i>	The number of columns of the 2D array
in	<i>first_</i>	the opening delimiter of the array. Default is "("
in	<i>last_</i>	the closing delimiter of the array. Default is ")"

Author

Andrea Facci.

Definition at line 86 of file matrixRead2.f90.

References matrixread().

Referenced by readboiler(), readchillers(), readenv(), readloads(), and readtrigen().

6.27.2.3 integer function, dimension(nline,ncol) imatrixread (integer, intent(in) *theUnit*, integer, intent(in) *nline*, integer, intent(in) *ncol*, character(len=1), optional *first_*, character(len=1), optional *last_*)

This subroutine reads a 2D array of integer values directly from a specified unit. The unit must be already opened and associated to the desired file. All the values to read must be written in "value" field of the text file that is between the two "|" (see the following box for an example of input text) and delimited by appropriate opening and closure delimiters.

```
A Lot of useless stuff because only text between begin and end is read.
begin
  !Commented line
  KeywordField | ValueField | !Comment two
end
```

Moreover the cursor needs to be before the first line of the array to be read. The procedure automatically skips all the lines until the opening character. Opening and closing character may be specified as single character input, or left to the default values that are "(" and ")", respectively. Avoid blank or commented lines inside the text array.

The correct file syntax to read the 2x2 array "exArr":

$$exArr = \begin{bmatrix} 11 & 12 \\ 21 & 22 \end{bmatrix}$$

is:

```
begin
  !Comment one
  exArr | (11 12 | !Comment two
        | 21 22) |
end
```

or equivalently:

```

begin
  !Comment one
  exArr |      | !Comment two
        | (11 12 |
        |  21 22) |
end

```

so that each line in the text represents a row of the vector output and columns are space separated.

Parameters

in	<i>TheUnit</i>	the unit to be read.
in	<i>nline</i>	The number of rows of the 2D array
in	<i>ncol</i>	The number of columns of the 2D array
in	<i>first_</i>	the opening delimiter of the array. Default is "("
in	<i>last_</i>	the closing delimiter of the array. Default is ")"

Author

Andrea Facci.

Definition at line 188 of file matrixRead2.f90.

References matrixread().

6.27.2.4 `character(len=100) function, dimension(nline) matrixread (integer, intent(in) theUnit, integer, intent(in) nline, character(len=1), optional first_, character(len=1), optional last_)`

This subroutine reads a 2D array of directly from a specified unit. This procedure does not associate the elements of the text array to any data type. On the contrary each line in the text array is associated to a row of a character type row vector, as it is. All the values to read must be written in "value" field of the text file that is between the two "|" (see the following box for an example of input text) and delimited by appropriate opening and closure delimiters.

```

A Lot of useless stuff because only text between begin and end is read.
begin
  !Commented line
  KeywordField | ValueField | !Comment two
end

```

The unit must be already opened and associated to the desired file. Moreover the cursor needs to be before the first line of the array to be read. The procedure automatically skips all the lines until the opening character. Opening and closing character may be specified as single character input, or left to the default values that are "(" and ")", respectively. Avoid blank or commented lines inside the text array.

Parameters

in	<i>TheUnit</i>	the unit to be read.
in	<i>nline</i>	The number of rows of the 2D array
in	<i>first_</i>	the opening delimiter of the array. Default is "("
in	<i>last_</i>	the closing delimiter of the array. Default is ")"

Author

Andrea Facci.

Definition at line 364 of file matrixRead2.f90.

References rewunit().

Referenced by cmatrixread(), dmatrixread(), imatrixread(), and readloads().

6.28 /home/Codici/Blink/PowerManager/src/myArithmetic.f90 File Reference

creates and detects NaN

Data Types

- module [myarithmetic](#)
Creates and detects NaN.

6.28.1 Detailed Description

Author

Definition in file [myArithmetic.f90](#).

6.29 /home/Codici/Blink/PowerManager/src/objFunction.f90 File Reference

Objective function.

Functions/Subroutines

- real(kind=prec) function [objfunction](#) (c, t, obj)
Objective function.

6.29.1 Detailed Description

Author

Definition in file [objFunction.f90](#).

6.29.2 Function/Subroutine Documentation

6.29.2.1 real(kind = prec) function objfunction (integer, dimension(nm), intent(in) c, integer, intent(in) t, character(len=100), intent(in) obj)

Returns the value of the objective function for a given plant state, time step and optimization criterion. This procedure accounts only for functions that are local in time, that is, that are function only of the plant state at time t and not at time > t, neither at time < t.

Parameters

in	obj	the optimization criterion.
in	c	index of the given set-point to be given as input. Defines the state of the plant $sp(i) = sp(c_ (i))$
in	t	time step index. Note t=x meas the x'th time step from the

Definition at line 39 of file objFunction.f90.

References [economy::currcost\(\)](#).

Referenced by `graphtools::graphpoints()`.

6.30 /home/Codici/Blink/PowerManager/src/openUnit.f90 File Reference

checks file presence and opens the unit.

Functions/Subroutines

- subroutine `openunit` (`fl`, `unt`, `prsnt`)
checks file presence and opens the unit.

6.30.1 Detailed Description

Author

Andrea Facci.

Definition in file `openUnit.f90`.

6.30.2 Function/Subroutine Documentation

6.30.2.1 subroutine `openunit` (`character(len=*)`, `intent(in)` `fl`, `integer`, `intent(in)` `unt`, `logical`, `intent(out)` `prsnt`)

checks file presence and opens the unit.

Parameters

<code>in</code>	<code>fl</code>	The file name
<code>in</code>	<code>unt</code>	The unit number
<code>out</code>	<code>prsnt</code>	Logical for file presence.

Author

Andrea Facci.

Definition at line 37 of file `openUnit.f90`.

Referenced by `readenv()`, and `readloads()`.

6.31 /home/Codici/Blink/PowerManager/src/output.f90 File Reference

Print the output.

Functions/Subroutines

- subroutine `output` (`setPoint`, `postProcessing`, `path`)
- subroutine `preparefile` (`u`, `nome`, `folder`)

6.31.1 Detailed Description

Print the output. Andrea Facci.

Definition in file `output.f90`.

6.31.2 Function/Subroutine Documentation

6.31.2.1 subroutine `output` (integer, dimension(0:ntime+1,nm), intent(in) *setPoint*, logical, intent(in) *postProcessing*, character(len=*), intent(in) *path*)

Definition at line 31 of file `output.f90`.

References `energy::chprod()`, `energy::elprod()`, `energy::energyinput()`, `economy::firecost()`, `energy::fuelcons()`, `economy::fuelcost()`, `globalresults::globchrev()`, `globalresults::globcost()`, `globalresults::globelrev()`, `globalresults::globfirecost()`, `globalresults::globfuelcost()`, `globalresults::globmaintcost()`, `globalresults::globprofit()`, `globalresults::globrevenues()`, `globalresults::globthrev()`, `economy::grideconomy()`, `economy::maintenancecost()`, `preparefile()`, and `energy::thprod()`.

Referenced by `main()`, and `warning()`.

6.31.2.2 subroutine `preparefile` (integer, intent(in) *u*, character(len=*), intent(in) *nome*, character(len=*), intent(in) *folder*)

Definition at line 491 of file `output.f90`.

Referenced by `output()`.

6.32 /home/Codici/Blink/PowerManager/src/performances.f90 File Reference

calculates the performances of a single equipment.

Functions/Subroutines

- subroutine `performances` (c, cOld, equip, num, t, pEl, pTh, pCh, eln, mf, cfu, cm, cOn)

6.32.1 Detailed Description

calculates the performances of a single equipment. .

Author

Andrea Facci.

Definition in file `performances.f90`.

6.32.2 Function/Subroutine Documentation

6.32.2.1 subroutine `performances` (integer, intent(in) *c*, integer, intent(in), optional *cOld*, character(len=*), intent(in) *equip*, integer, intent(in) *num*, integer, intent(in) *t*, real(kind = prec), intent(out), optional *pEl*, real(kind = prec), intent(out), optional *pTh*, real(kind = prec), intent(out), optional *pCh*, real(kind = prec), intent(out), optional *eln*, real(kind = prec), intent(out), optional *mf*, real(kind = prec), intent(out), optional *cfu*, real(kind = prec), intent(out), optional *cm*, real(kind = prec), intent(out), optional *cOn*)

Definition at line 31 of file `performances.f90`.

6.33 /home/Codici/Blink/PowerManager/src/plantVar.f90 File Reference

collection of variables relative to the power plant structure.

Data Types

- module [plantvar](#)
collection of variables relative to the power plant structure.

6.33.1 Detailed Description

Author

Definition in file [plantVar.f90](#).

6.34 /home/Codici/Blink/PowerManager/src/prototipo.f90 File Reference

File prototype.

Functions/Subroutines

- subroutine [implicit](#) none

6.34.1 Detailed Description

this is the prototype for all the files of the PowerManger project. Copy, rename, and modify this this file to create a new procedure or module. Andrea Facci.

Definition in file [prototipo.f90](#).

6.34.2 Function/Subroutine Documentation

6.34.2.1 subroutine implicit (none)

Definition at line 32 of file prototipo.f90.

6.35 /home/Codici/Blink/PowerManager/src/readBoiler.f90 File Reference

Reads Boiler.inp file.

Functions/Subroutines

- subroutine [readboiler](#)
Reads Boiler.inp file.

6.35.1 Detailed Description

Author

Andrea Facci.

Definition in file [readBoiler.f90](#).

6.35.2 Function/Subroutine Documentation

6.35.2.1 subroutine readboiler ()

This subroutine reads the file "Boilers.inp". The procedure looks for each specific entry in the "keyword field", and associates the value in the "value" field, to the corresponding variable. If the desired entry is not present returns an error message and aborts the execution. The structure of the input file is clarified in the following example along with the meaning of "keyword" and "value" field.

```
A Lot of useless stuff because only text between begin and end is read.
begin
  !Commented line
  KeywordField | ValueField | !Comment two

  KeywordField | ValueField |
  ScalarValue  | 1          |
  Vector       | 1 2 n          |
  VectorSeries |(a b c ) (d e f)|
  Matrix       |(11 12 13 |
              | 21 22 23 |
              | 31 32 33) |
end
A lof really useless text
```

Note that only the text between "begin" and "end" is read. Blank lines are automatically discarded, while any unrecognized entry is discarded returning a warning code. Line beginning with "!" are considered comments and discarded.

Author

Andrea Facci.

Definition at line 61 of file readBoiler.f90.

References `abortexecution()`, `allocatevar()`, `dmatrixread()`, `hcount()`, `ifindentry()`, `readkeyword()`, `rewunit()`, `myarithmetic::rnan()`, and `vcount()`.

Referenced by `main()`.

6.36 /home/Codici/Blink/PowerManager/src/readChiller.f90 File Reference

Reads Chiller.inp file.

Functions/Subroutines

- subroutine [readchillers](#)

Reads Chiller.inp file.

6.36.1 Detailed Description

Author

Andrea Facci.

Definition in file [readChiller.f90](#).

6.36.2 Function/Subroutine Documentation

6.36.2.1 subroutine readchillers ()

This subroutine reads the file "Chiller.inp". The procedure looks for each specific entry in the "keyword field", and associates the value in the "value" field, to the corresponding variable. If the desired entry is not present returns an error message and aborts the execution. The structure of the input file is clarified in the following example along with the meaning of "keyword" and "value" field.

```
A Lot of useless stuff because only text between begin and end is read.
begin
  !Commented line
  KeywordField | ValueField | !Comment two

  KeywordField | ValueField |
  ScalarValue  | 1         |
  Vector       | 1 2 n       |
  VectorSeries |(a b c ) (d e f)|
  Matrix       |(11 12 13   |
              | 21 22 23   |
              | 31 32 33)  |
end
A lot really useless text
```

Note that only the text between "begin" and "end" is read. Blank lines are automatically discarded, while any unrecognized entry is discarded returning a warning code. Line beginning with "!" are considered comments and discarded.

Author

Andrea Facci.

Definition at line 61 of file readChiller.f90.

References `abortexecution()`, `allocatevar()`, `cfindentry()`, `dmatrixread()`, `hcount()`, `ifindentry()`, `readkeyword()`, `rewunit()`, `myarithmetic::rnan()`, and `vcount()`.

Referenced by `main()`.

6.37 /home/Codici/Blink/PowerManager/src/readEnv.f90 File Reference

Functions/Subroutines

- subroutine [readenv](#)
Reads Load.inp file.

6.37.1 Function/Subroutine Documentation

6.37.1.1 subroutine readenv ()

This subroutine reads the file "Load.inp". The procedure looks for each specific entry in the "keyword field", and associates the value in the "value" field, to the corresponding variable. If the desired entry is not present returns an error message and aborts the execution. The structure of the input file is clarified in the following example along with the meaning of "keyword" and "value" field.

```
A Lot of useless stuff because only text between begin and end is read.
begin
  !Commented line
  KeywordField | ValueField | !Comment two
```

```

KeywordField | ValueField |
ScalarValue  | 1          |
Vector       | 1 2 n      |
VectorSeries | (a b c ) (d e f) |
Matrix       | (11 12 13 |
              | 21 22 23 |
              | 31 32 33) |

end
A lot really useless text

```

Note that only the text between "begin" and "end" is read. Blank lines are automatically discarded, while any unrecognized entry is discarded returning a warning code. Line beginning with "!" are considered comments and discarded.

Author

Andrea Facci.

Definition at line 61 of file readEnv.f90.

References abortexecution(), allocatevar(), cmatrixread(), dmatrixread(), openunit(), readkeyword(), rewunit(), and vcount().

Referenced by main().

6.38 /home/Codici/Blink/PowerManager/src/readGeneral.f90 File Reference

Reads General.inp file.

Functions/Subroutines

- subroutine [readgeneral](#)
Reads General.inp file.

6.38.1 Detailed Description

Author

Andrea Facci.

Definition in file [readGeneral.f90](#).

6.38.2 Function/Subroutine Documentation

6.38.2.1 subroutine readgeneral ()

This subroutine reads the file "General.inp". The procedure looks for each specific entry in the "keyword field", and associates the value in the "value" field, to the corresponding variable. If the desired entry is not present returns an error message and aborts the execution. The structure of the input file is clarified in the following example along with the meaning of "keyword" and "value" field.

```

A Lot of useless stuff because only text between begin and end is read.
begin
  !Commented line
  KeywordField | ValueField | !Comment two

  KeywordField | ValueField |
  ScalarValue  | 1          |

```

```

Vector      | 1 2 n      |
VectorSeries | (a b c ) (d e f) |
Matrix      | (11 12 13 |
            | 21 22 23 |
            | 31 32 33) |
end
A lof really useless text

```

Note that only the text between "begin" and "end" is read. Blank lines are automatically discarded, while any unrecognized entry is discarded returning a warning code. Line beginning with "!" are considered comments and discarded.

Author

Andrea Facci.

Definition at line 60 of file readGeneral.f90.

References abortexecution(), allocatevar(), hcount(), and readkeyword().

Referenced by main().

6.39 /home/Codici/Blink/PowerManager/src/readKeyword.f90 File Reference

Reads a line of a text file.

Functions/Subroutines

- subroutine [readkeyword](#) (theUnit, rew, keyword, value, error, nRow)
Reads a line of a text file.

6.39.1 Detailed Description

Author

Andrea Facci.

Definition in file [readKeyword.f90](#).

6.39.2 Function/Subroutine Documentation

6.39.2.1 subroutine [readkeyword](#) (integer, intent(in) *theUnit*, logical, intent(in), optional *rew*, character(len=*), intent(out) *keyword*, character(len=*), intent(out) *value*, integer, intent(out), optional *error*, integer, intent(out), optional *nRow*)

This procedure reads a line of a unit and returns the "keyword" and "value" fields as single value characters of maximum length = 100. The Unit needs to be already opened and associated to the desired file. This subroutine reads the line corresponding to the actual position of the cursor. The structure expected for the input text file is clarified in the following example along with the meaning of "keyword" and "value" field.

```

A Lot of useless stuff because only text between begin and end is read.
begin
  !Commented line
  KeywordField | ValueField | !Comment two

  KeywordField | ValueField |
  ScalarValue  | 1          |
  Vector       | 1 2 n      |
  VectorSeries | (a b c ) (d e f) |

```

```

Matrix      | (11 12 13 |
            | 21 22 23 |
            | 31 32 33) |
end
A lof really useless text

```

Note that only the text between "begin" and "end" is read. Blank lines are automatically discarded, as well as lines beginning with "!" that are considered comments. If only one delimiter "|" is present the line is considered misspelled, no value are is associated to keyword nor to value, and an optional error code is returned. The number of lines that were read before the first valid line can be returned with the optional argument "nRow". If the optional argument "rew" is set to true the unit is rewinded exactly of nRow lines at the end of the procedure.

Parameters

out	<i>Keyword,value</i>	keyword and value fields
out	<i>error</i>	Error code = 1 in case of misspelled lines
out	<i>nRow</i>	The number of lines that were read before the first valid line
in	<i>theUnit</i>	Unit number associated to the desired file.
in	<i>rew</i>	Wether to rewind or not the unit. Default is false.

Author

Andrea Facci.

Definition at line 71 of file readKeyword.f90.

References rewunit().

Referenced by readboiler(), readchillers(), readenv(), readgeneral(), and readtrigen().

6.40 /home/Codici/Blink/PowerManager/src/readLoad.f90 File Reference

Reads Load.inp file.

Functions/Subroutines

- subroutine [readloads](#)
Reads Load.inp file.

6.40.1 Detailed Description

Author

Andrea Facci.

Definition in file [readLoad.f90](#).

6.40.2 Function/Subroutine Documentation

6.40.2.1 subroutine readloads ()

This subroutine reads the file "Load.inp". The procedure looks for each specific entry in the "keyword field", and associates the value in the "value" field, to the corresponding variable. If the desired entry is not present returns an error message and aborts the execution. The structure of the input file is clarified in the following example along with the meaning of "keyword" and "value" field.


```

A Lot of useless stuff because only text between begin and end is read.
begin
  !Commented line
  KeywordField | ValueField | !Comment two

  KeywordField | ValueField |
  ScalarValue  | 1          |
  Vector        | 1 2 n      |
  VectorSeries  |(a b c ) (d e f)|
  Matrix        |(11 12 13 |
                | 21 22 23 |
                | 31 32 33) |

end
A lof really useless text

```

Note that only the text between "begin" and "end" is read. Blank lines are automatically discarded, while any unrecognized entry is discarded returning a warning code. Line beginning with "!" are considered comments and discarded.

Author

Andrea Facci.

Definition at line 61 of file readLoad.f90.

References abortexecution(), allocatevar(), cmatrixread(), dmatrixread(), findentry(), hcount(), matrixread(), openunit(), rewunit(), and vcount().

Referenced by main().

6.41 /home/Codici/Blink/PowerManager/src/readSteamPlant.f90 File Reference

Functions/Subroutines

- subroutine [readtrigen](#)
Reads Trigenation.inp file.

6.41.1 Function/Subroutine Documentation

6.41.1.1 subroutine readtrigen ()

This subroutine reads the file "Trigenation.inp". The procedure looks for each specific entry in the "keyword field", and associates the value in the "value" field, to the corresponding variable. If the desired entry is not present returns an error message and aborts the execution. The structure of the input file is clarified in the following example along with the meaning of "keyword" and "value" field.

```

A Lot of useless stuff because only text between begin and end is read.
begin
  !Commented line
  KeywordField | ValueField | !Comment two

  KeywordField | ValueField |
  ScalarValue  | 1          |
  Vector        | 1 2 n      |
  VectorSeries  |(a b c ) (d e f)|
  Matrix        |(11 12 13 |
                | 21 22 23 |
                | 31 32 33) |

end
A lof really useless text

```

Note that only the text between "begin" and "end" is read. Blank lines are automatically discarded, while any unrecognized entry is discarded returning a warning code. Line beginning with "!" are considered comments and discarded.

Author

Andrea Facci.

Definition at line 62 of file readSteamPlant.f90.

References abortexecution(), allocatevar(), dmatrixread(), hcount(), ifindentry(), readkeyword(), rewunit(), and vcount().

Referenced by main().

6.42 /home/Codici/Blink/PowerManager/src/readTrigen.f90 File Reference

Reads Trigenation.inp file.

Functions/Subroutines

- subroutine [readtrigen](#)
Reads Trigenation.inp file.

6.42.1 Detailed Description

Author

Andrea Facci.

Definition in file [readTrigen.f90](#).

6.42.2 Function/Subroutine Documentation

6.42.2.1 subroutine readtrigen ()

This subroutine reads the file "Trigenation.inp". The procedure looks for each specific entry in the "keyword field", and associates the value in the "value" field, to the corresponding variable. If the desired entry is not present returns an error message and aborts the execution. The structure of the input file is clarified in the following example along with the meaning of "keyword" and "value" field.

```
A Lot of useless stuff because only text between begin and end is read.
begin
  !Commented line
  KeywordField | ValueField | !Comment two

  KeywordField | ValueField |
  ScalarValue  | 1         |
  Vector       | 1 2 n       |
  VectorSeries |(a b c ) (d e f)|
  Matrix       |(11 12 13 |
              | 21 22 23 |
              | 31 32 33) |
end
A lot really useless text
```

Note that only the text between "begin" and "end" is read. Blank lines are automatically discarded, while any unrecognized entry is discarded returning a warning code. Line beginning with "!" are considered comments and discarded.

Author

Andrea Facci.

Definition at line 62 of file readTrigen.f90.

References abortexecution(), allocatevar(), dmatrixread(), hcount(), ifindentry(), readkeyword(), rewunit(), myarithmetic::rnan(), and vcount().

6.43 /home/Codici/Blink/PowerManager/src/rewUnit.f90 File Reference

rewind a unit.

Functions/Subroutines

- subroutine [rewunit](#) (theUnit, n)
rewind a unit.

6.43.1 Detailed Description**Author**

Definition in file [rewUnit.f90](#).

6.43.2 Function/Subroutine Documentation

6.43.2.1 subroutine rewunit (integer, intent(in) *theUnit*, integer, intent(in) *n*)

Rewinds a unit for a specified number of lines. Note that the unit needs to be opened.

Parameters

<i>in</i>	<i>theUnit</i>	The Unit to be rewinded.
<i>in</i>	<i>n</i>	The number of lines to rewind.

Author

Definition at line 38 of file rewUnit.f90.

Referenced by findentry(), matrixread(), readboiler(), readchillers(), readenv(), readkeyword(), readloads(), readtrigen(), and vcount().

6.44 /home/Codici/Blink/PowerManager/src/shared.f90 File Reference

File prototype.

Data Types

- module [shared](#)

6.44.1 Detailed Description

this is the prototype for all the files of the PowerManger project. Copy, rename, and modify this this file to create a new procedure or module. Andrea Facci.

Definition in file [shared.f90](#).

6.45 /home/Codici/Blink/PowerManager/src/vCount.f90 File Reference

Counts the lines of a text matrix.

Functions/Subroutines

- integer function [vcount](#) (theUnit, rew_, first_, last_)
Counts the lines of a text matrix.

6.45.1 Detailed Description

Author

Definition in file [vCount.f90](#).

6.45.2 Function/Subroutine Documentation

6.45.2.1 integer function vcount (integer, intent(in) *theUnit*, logical, optional *rew_*, character(len=1), intent(in), optional *first_*, character(len=1), intent(in), optional *last_*)

This subroutine determines the number of lines between the two array delimiters. The unit must be already opened and associated to the desired file. All the values to read must be written in "value" field of the text file that is between the two "|" (see the following box for an example of input text) and delimited by appropriate opening and closure delimiters.

```
A Lot of useless stuff because only text between begin and end is read.
begin
  !Commented line
  KeywordField | ValueField | !Comment two
end
```

Moreover the cursor needs to be before the first line of the array to be read. The procedure automatically skips all the lines until the opening character. Opening and closing character may be specified as single character input, or left to the default values that are "(" and ")", respectively. Avoid blank or commented lines inside the text array.

The correct file syntax to read the 2x2 array "exArr":

$$exArr = \begin{bmatrix} 1.1 & 1.2 \\ 2.1 & 2.2 \end{bmatrix}$$

is:

```
begin
  !Comment one
  exArr | (1.1 1.2 | !Comment two
        | 2.1 2.2)|
end
```

or equivalently:

```

begin
  !Comment one
  exArr |      | !Comment two
        | (1.1 1.2 |
        |  2.1 2.2) |
end

```

so that each line in the text represents a row of the vector output and columns are space separated.

Parameters

<code>in</code>	<i>theUnit</i>	The unit associated to the file to be read
<code>in</code>	<i>rew_</i>	Weather to rewind or not the unit at the end. Default is false.
<code>in</code>	<i>first_,last_</i>	Opening and closing characters of the array.

Author

Definition at line 80 of file vCount.f90.

References `rewunit()`.

Referenced by `readboiler()`, `readchillers()`, `readenv()`, `readloads()`, and `readtrigen()`.

6.46 /home/Codici/Blink/PowerManager/src/warning.f90 File Reference

prints warnings to standard output

Functions/Subroutines

- subroutine [warning](#) (*i*, *j*, *k*, *line*, *word*, *r1*, *r2*)
print warnings to standard output

6.46.1 Detailed Description

Author

Andrea Facci.

Definition in file [warning.f90](#).

6.46.2 Function/Subroutine Documentation

6.46.2.1 subroutine `warning` (`integer`, intent(in), optional *i*, `integer`, intent(in), optional *j*, `integer`, intent(in), optional *k*, `integer`, intent(in), optional *line*, `character(len=*)`, intent(in), optional *word*, `real(kind = prec)`, intent(in), optional *r1*, `real(kind = prec)`, intent(in), optional *r2*)

print warnings to standard output

Author

Andrea Facci

Parameters

in	<i>i,j,k</i>	error codes
in	<i>line</i>	line to locate the error position
in	<i>word</i>	mispelled or unrecognized word
in	<i>r1,r</i>	real numbers for unexpected values.

Definition at line 38 of file warning.f90.

References output().

Index

/home/Codici/Blink/PowerManager/src/abortExecution.-
f90, 69
/home/Codici/Blink/PowerManager/src/aiuto.f90, 70
/home/Codici/Blink/PowerManager/src/allocateVar.f90,
70
/home/Codici/Blink/PowerManager/src/buildPlant.f90,
71
/home/Codici/Blink/PowerManager/src/checkPlant.f90,
72
/home/Codici/Blink/PowerManager/src/cmdVar.f90, 72
/home/Codici/Blink/PowerManager/src/cmdVarWr.f90,
73
/home/Codici/Blink/PowerManager/src/command.f90,
73
/home/Codici/Blink/PowerManager/src/commandline.-
f90, 73
/home/Codici/Blink/PowerManager/src/constraints.f90,
74
/home/Codici/Blink/PowerManager/src/deallocateVar.-
f90, 74
/home/Codici/Blink/PowerManager/src/economy.f90, 75
/home/Codici/Blink/PowerManager/src/endExecution.-
f90, 75
/home/Codici/Blink/PowerManager/src/energy.f90, 76
/home/Codici/Blink/PowerManager/src/euristics.f90, 76
/home/Codici/Blink/PowerManager/src/fileTools.f90, 77
/home/Codici/Blink/PowerManager/src/findEntry2.f90,
77
/home/Codici/Blink/PowerManager/src/globalResults.-
f90, 80
/home/Codici/Blink/PowerManager/src/graphTools.f90,
80
/home/Codici/Blink/PowerManager/src/hCount.f90, 80
/home/Codici/Blink/PowerManager/src/inputVar.f90, 81
/home/Codici/Blink/PowerManager/src/interfaces.f90, 81
/home/Codici/Blink/PowerManager/src/interpolation.f90,
82
/home/Codici/Blink/PowerManager/src/main.f90, 82
/home/Codici/Blink/PowerManager/src/manual.f90, 83
/home/Codici/Blink/PowerManager/src/mathTools.f90,
83
/home/Codici/Blink/PowerManager/src/matrixRead2.-
f90, 84
/home/Codici/Blink/PowerManager/src/myArithmetic.-
f90, 88
/home/Codici/Blink/PowerManager/src/objFunction.f90,
88
/home/Codici/Blink/PowerManager/src/openUnit.f90, 89
/home/Codici/Blink/PowerManager/src/output.f90, 89
/home/Codici/Blink/PowerManager/src/performances.-
f90, 90
/home/Codici/Blink/PowerManager/src/plantVar.f90, 90
/home/Codici/Blink/PowerManager/src/prototipo.f90, 91
/home/Codici/Blink/PowerManager/src/readBoiler.f90,
91
/home/Codici/Blink/PowerManager/src/readChiller.f90,
92
/home/Codici/Blink/PowerManager/src/readEnv.f90, 93
/home/Codici/Blink/PowerManager/src/readGeneral.f90,
94
/home/Codici/Blink/PowerManager/src/readKeyword.-
f90, 95
/home/Codici/Blink/PowerManager/src/readLoad.f90, 96
/home/Codici/Blink/PowerManager/src/readSteam-
Plant.f90, 97
/home/Codici/Blink/PowerManager/src/readTrigen.f90,
98
/home/Codici/Blink/PowerManager/src/rewUnit.f90, 99
/home/Codici/Blink/PowerManager/src/shared.f90, 99
/home/Codici/Blink/PowerManager/src/vCount.f90, 100
/home/Codici/Blink/PowerManager/src/warning.f90, 101
abortExecution.f90
 abortexecution, 69
abortexecution
 abortExecution.f90, 69
 interfaces::abortExecution, 9
aiuto
 aiuto.f90, 70
aiuto.f90
 aiuto, 70
allcombin
 graphtools, 36
allocateVar.f90
 allocatevar, 71
allocatevar
 allocateVar.f90, 71
altcorrb
 inputvar, 43
altcorrc
 inputvar, 43
altcorrt
 inputvar, 43
altitude
 inputvar, 44
buildPlant.f90
 buildplant, 71
buildplant

- buildPlant.f90, 71
- cartella
 - command, 14
- cch
 - inputvar, 44
- cel
 - inputvar, 44
- cf
 - plantvar, 60
- cfindentry
 - filetools::cFindEntry, 10
 - findEntry2.f90, 78
- checkPlant.f90
 - checkplant, 72
- checkplant
 - checkPlant.f90, 72
- chprod
 - energy, 23
- chredundant
 - heuristics, 27
- chrev
 - economy, 18
- cmatrixread
 - filetools::cMatrixRead, 11
 - matrixRead2.f90, 84
- cmdvar, 11
 - debug, 12
 - out, 12
 - silent, 12
 - verb, 12
 - vsilent, 12
- cmdvarwr, 12
 - folder, 13
- cmdvarwrite
 - command, 14
- cogthprod
 - energy, 23
- command, 13
 - cartella, 14
 - cmdvarwrite, 14
 - protected, 14
- commandline
 - commandline.f90, 73
- commandline.f90
 - commandline, 73
- constraints
 - constraints.f90, 74
 - heuristics::constraints, 15
 - graphtools::constraints, 15
- constraints.f90
 - constraints, 74
 - timeconstr, 74
- cr
 - plantvar, 60
- cref
 - plantvar, 60
- cth
 - inputvar, 44
- currcost
 - economy, 18
- deallocateVar.f90
 - deallocatevar, 75
- deallocatevar
 - deallocateVar.f90, 75
- debug
 - cmdvar, 12
- dfindentry
 - filetools::dFindEntry, 16
 - findEntry2.f90, 78
- dmatrixread
 - filetools::dMatrixRead, 17
 - matrixRead2.f90, 85
- downtime0
 - inputvar, 44
- dt
 - plantvar, 61
- dt1
 - inputvar, 44
- economy, 17
 - chrev, 18
 - currcost, 18
 - elrev, 19
 - firecost, 20
 - fuelcost, 20
 - grideconomy, 21
 - maintenancecost, 21
 - threv, 21
- elprod
 - energy, 24
- elrev
 - economy, 19
- elseifcons
 - energy, 24
- endExecution.f90
 - endexecution, 76
- endexecution
 - endExecution.f90, 76
- energy, 22
 - chprod, 23
 - cogthprod, 23
 - elprod, 24
 - elseifcons, 24
 - energyexhaust, 24
 - energyinput, 25
 - fuelcons, 25
 - limitrecovery, 25
 - thprod, 26
 - thselfcons, 26
- energyexhaust
 - energy, 24
- energyinput
 - energy, 25
- envcorr
 - plantvar, 61
- esource

- plantvar, 61
- etab
 - inputvar, 44
- etab_
 - plantvar, 61
- etac
 - inputvar, 44
- etac_
 - plantvar, 61
- etach
 - plantvar, 61
- etacht
 - inputvar, 44
- etacht_
 - plantvar, 61
- etael
 - plantvar, 61
- etaelt
 - inputvar, 44
- etaelt_
 - plantvar, 61
- etath
 - plantvar, 61
- etatht
 - inputvar, 44
- etatht_
 - plantvar, 61
- heuristics, 27
 - chredundant, 27
 - thredundant, 28
- heuristics::constraints, 15
 - constraints, 15
- filetools, 28
- filetools::cFindEntry, 10
 - cfindentry, 10
- filetools::cMatrixRead, 10
 - cmatrixread, 11
- filetools::dFindEntry, 16
 - dfindentry, 16
- filetools::dMatrixRead, 16
 - dmatrixread, 17
- filetools::findEntry, 29
 - findentry, 30
- filetools::hCount, 38
 - hcount, 38
- filetools::iFindEntry, 38
 - ifindentry, 39
- filetools::iMatrixRead, 39
 - imatrixread, 39
- filetools::matrixRead, 55
 - matrixread, 55
- filetools::readKeyword, 63
 - readkeyword, 64
- filetools::rewUnit, 64
 - rewunit, 64
- filetools::vCount, 66
 - vcount, 67
- findEntry2.f90
 - cfindentry, 78
 - dfindentry, 78
 - findentry, 78
 - ifindentry, 79
- findentry
 - filetools::findEntry, 30
 - findEntry2.f90, 78
- firecost
 - economy, 20
- firecostb
 - inputvar, 44
- firecostc
 - inputvar, 45
- firecostt
 - inputvar, 45
- folder
 - cmdvarwr, 13
- fuelcons
 - energy, 25
- fuelcost
 - economy, 20
- fuelcostb
 - inputvar, 45
- fuelcostt
 - inputvar, 45
- fuellhvb
 - inputvar, 45
- fuellhvt
 - inputvar, 45
- global
 - inputvar, 45
- globalresults, 30
 - globchrev, 31
 - globcost, 31
 - globelrev, 31
 - globfirecost, 32
 - globfuelcost, 33
 - globmaintcost, 33
 - globprofit, 34
 - globrevenues, 34
 - globthrev, 34
- globchrev
 - globalresults, 31
- globcost
 - globalresults, 31
- globelrev
 - globalresults, 31
- globfirecost
 - globalresults, 32
- globfuelcost
 - globalresults, 33
- globmaintcost
 - globalresults, 33
- globprofit
 - globalresults, 34
- globrevenues
 - globalresults, 34
- globthrev

- globalresults, 34
- grapharcs
 - graphtools, 36
- graphpoints
 - graphtools, 36
- graphtools, 35
 - allcombin, 36
 - grapharcs, 36
 - graphpoints, 36
 - minpathtopobw, 36
 - minpathtopofw, 37
 - uptimecalc, 37
- graphtools::constraints, 14
 - constraints, 15
- graphtools::objFunction, 57
 - objfunction, 58
- graphtools::timeConstr, 66
 - timeconstr, 66
- gridbuycost
 - inputvar, 45
- gridconnection
 - inputvar, 45
- grideconomy
 - economy, 21
- gridsellcost
 - inputvar, 45
- hCount.f90
 - hcount, 81
- hcount
 - filetools::hCount, 38
 - hCount.f90, 81
- ib
 - plantvar, 61
- ic
 - plantvar, 62
- ich
 - inputvar, 45
- ichp
 - inputvar, 45
- ideg
 - inputvar, 46
- ie
 - plantvar, 62
- iel
 - inputvar, 46
- ielp
 - inputvar, 46
- ifindentry
 - filetools::iFindEntry, 39
 - findEntry2.f90, 79
- imatrixread
 - filetools::iMatrixRead, 39
 - matrixRead2.f90, 86
- implicit
 - prototipo.f90, 91
- inan
 - myarithmetic, 56
- inputvar, 40
 - altcorrb, 43
 - altcorr, 43
 - altcorr, 43
 - altitude, 44
 - cch, 44
 - cel, 44
 - cth, 44
 - downtime0, 44
 - dt1, 44
 - etab, 44
 - etac, 44
 - etacht, 44
 - etaelt, 44
 - etatht, 44
 - firecostb, 44
 - firecostc, 45
 - firecostt, 45
 - fuelcostb, 45
 - fuelcostt, 45
 - fuellhvb, 45
 - fuellhvt, 45
 - global, 45
 - gridbuycost, 45
 - gridconnection, 45
 - gridsellcost, 45
 - ich, 45
 - ichp, 45
 - ideg, 46
 - iel, 46
 - ielp, 46
 - ith, 46
 - ithp, 46
 - itime, 46
 - lifeb, 46
 - lifet, 46
 - maintcostb, 46
 - maintcostc, 46
 - maintcostt, 46
 - method, 46
 - mindowntimeb, 47
 - mindowntimec, 47
 - mindowntimet, 47
 - minuptimeb, 47
 - minuptimec, 47
 - minuptimet, 47
 - nacb, 47
 - nacc, 47
 - nact, 47
 - nboi, 47
 - nchi, 47
 - netab, 47
 - netac, 48
 - netacht, 48
 - netaelt, 48
 - netatht, 48
 - nld, 48
 - nload, 48

- nlp, 48
- npcb, 48
- npcc, 48
- npct, 48
- nsizec, 48
- nspb, 48
- nspc, 49
- nspt, 49
- ntcb, 49
- ntcc, 49
- ntct, 49
- ntime, 49
- ntimes, 49
- ntrig, 49
- obj, 49
- pamb, 49
- pmaxb, 49
- pmaxc, 49
- pmaxt, 50
- prescorrb, 50
- prescorrc, 50
- prescorrt, 50
- spb, 50
- spc, 50
- spt, 50
- startpoint, 50
- tamb, 50
- tecb, 50
- tecc, 50
- tect, 50
- tempcorrb, 51
- tempcorrc, 51
- tempcorrt, 51
- time, 51
- uch, 51
- uel, 51
- uptime0, 51
- useheuristics, 51
- uth, 51
- writeboi, 51
- writetechi, 51
- writetechillingrev, 51
- writecosts, 52
- writedemand, 52
- writeefficiency, 52
- writeelectricrev, 52
- writeenergy, 52
- writeinput, 52
- writepower, 52
- writethermalrev, 52
- writetrig, 52
- interfaces, 53
- interfaces::abortExecution, 9
- interfaces::performances, 58
- interfaces::performances, 58
- interfaces::warning, 67
- interfaces::warning, 67
- interpolation
 - interpolation.f90, 82
 - mathtools::interpolation, 54
- interpolation.f90
 - interpolation, 82
- is
 - plantvar, 62
- isnan
 - myarithmetic, 56
- it
 - plantvar, 62
- ith
 - inputvar, 46
- ithp
 - inputvar, 46
- itime
 - inputvar, 46
- lhv
 - plantvar, 62
- lifeb
 - inputvar, 46
- lifet
 - inputvar, 46
- limitrecovery
 - energy, 25
- locaterow
 - mathtools, 55
- main
 - main.f90, 83
- main.f90
 - main, 83
- maintcostb
 - inputvar, 46
- maintcostc
 - inputvar, 46
- maintcostt
 - inputvar, 46
- maintenancecost
 - economy, 21
- mathtools, 54
 - locaterow, 55
- mathtools::interpolation, 53
 - interpolation, 54
- matrixRead2.f90
 - cmatrixread, 84
 - dmatrixread, 85
 - imatrixread, 86
 - matrixread, 87
- matrixread
 - filetools::matrixRead, 55
 - matrixRead2.f90, 87
- method
 - inputvar, 46
- mindowntime
 - plantvar, 62
- mindowntimeb
 - inputvar, 47

- mindowntimec
 - inputvar, [47](#)
- mindowntimet
 - inputvar, [47](#)
- minpathtopobw
 - graphtools, [36](#)
- minpathtopofw
 - graphtools, [37](#)
- minuptime
 - plantvar, [62](#)
- minuptimeb
 - inputvar, [47](#)
- minuptimec
 - inputvar, [47](#)
- minuptimet
 - inputvar, [47](#)
- myarithmetic, [55](#)
 - inan, [56](#)
 - isnan, [56](#)
 - rnan, [57](#)
- nacb
 - inputvar, [47](#)
- nacc
 - inputvar, [47](#)
- nact
 - inputvar, [47](#)
- nboi
 - inputvar, [47](#)
- nchi
 - inputvar, [47](#)
- netab
 - inputvar, [47](#)
- netac
 - inputvar, [48](#)
- netacht
 - inputvar, [48](#)
- netaelt
 - inputvar, [48](#)
- netatht
 - inputvar, [48](#)
- nld
 - inputvar, [48](#)
- nload
 - inputvar, [48](#)
- nlp
 - inputvar, [48](#)
- nm
 - plantvar, [62](#)
- npcb
 - inputvar, [48](#)
- npcc
 - inputvar, [48](#)
- npct
 - inputvar, [48](#)
- nsizec
 - inputvar, [48](#)
- nsp
 - plantvar, [62](#)
- nspb
 - inputvar, [48](#)
- nspc
 - inputvar, [49](#)
- nspt
 - inputvar, [49](#)
- nsptot
 - plantvar, [62](#)
- ntcb
 - inputvar, [49](#)
- ntcc
 - inputvar, [49](#)
- ntct
 - inputvar, [49](#)
- ntime
 - inputvar, [49](#)
- ntimes
 - inputvar, [49](#)
- ntrig
 - inputvar, [49](#)
- ntv
 - plantvar, [62](#)
- obj
 - inputvar, [49](#)
- objFunction.f90
 - objfunction, [88](#)
- objfunction
 - graphtools::objFunction, [58](#)
 - objFunction.f90, [88](#)
- oemcost
 - plantvar, [62](#)
- onoffcost
 - plantvar, [63](#)
- openUnit.f90
 - openunit, [89](#)
- openunit
 - openUnit.f90, [89](#)
- out
 - cmdvar, [12](#)
- output
 - output.f90, [90](#)
- output.f90
 - output, [90](#)
 - preparefile, [90](#)
- pamb
 - inputvar, [49](#)
- performances
 - interfaces::performances, [58](#)
 - performances.f90, [90](#)
- performances.f90
 - performances, [90](#)
- pes
 - plantvar, [63](#)
- plantvar, [59](#)
 - cf, [60](#)
 - cr, [60](#)
 - cref, [60](#)

- dt, [61](#)
- envcorr, [61](#)
- esource, [61](#)
- etab_, [61](#)
- etac_, [61](#)
- etach, [61](#)
- etacht_, [61](#)
- etael, [61](#)
- etaelt_, [61](#)
- etath, [61](#)
- etatht_, [61](#)
- ib, [61](#)
- ic, [62](#)
- ie, [62](#)
- is, [62](#)
- it, [62](#)
- lhv, [62](#)
- mindowntime, [62](#)
- minuptime, [62](#)
- nm, [62](#)
- nsp, [62](#)
- nsptot, [62](#)
- ntv, [62](#)
- oemcost, [62](#)
- onoffcost, [63](#)
- pes, [63](#)
- pmax, [63](#)
- sp, [63](#)
- tec, [63](#)
- timevinc, [63](#)
- pmax
 - plantvar, [63](#)
- pmaxb
 - inputvar, [49](#)
- pmaxc
 - inputvar, [49](#)
- pmaxt
 - inputvar, [50](#)
- prec
 - shared, [65](#)
- preparefile
 - output.f90, [90](#)
- prescorrb
 - inputvar, [50](#)
- prescorrc
 - inputvar, [50](#)
- prescorrt
 - inputvar, [50](#)
- protected
 - command, [14](#)
- prototipo.f90
 - implicit, [91](#)
- readBoiler.f90
 - readboiler, [92](#)
- readChiller.f90
 - readchillers, [93](#)
- readEnv.f90
 - readenv, [93](#)
- readGeneral.f90
 - readgeneral, [94](#)
- readKeyword.f90
 - readkeyword, [95](#)
- readLoad.f90
 - readloads, [96](#)
- readSteamPlant.f90
 - readtrigen, [97](#)
- readTrigen.f90
 - readtrigen, [98](#)
- readboiler
 - readBoiler.f90, [92](#)
- readchillers
 - readChiller.f90, [93](#)
- readenv
 - readEnv.f90, [93](#)
- readgeneral
 - readGeneral.f90, [94](#)
- readkeyword
 - filetools::readKeyword, [64](#)
 - readKeyword.f90, [95](#)
- readloads
 - readLoad.f90, [96](#)
- readtrigen
 - readSteamPlant.f90, [97](#)
 - readTrigen.f90, [98](#)
- rewUnit.f90
 - rewunit, [99](#)
- rewunit
 - filetools::rewUnit, [64](#)
 - rewUnit.f90, [99](#)
- rnan
 - myarithmetic, [57](#)
- rval
 - shared, [65](#)
- shared, [65](#)
 - prec, [65](#)
 - rval, [65](#)
 - zero, [65](#)
- silent
 - cmdvar, [12](#)
- sp
 - plantvar, [63](#)
- spb
 - inputvar, [50](#)
- spc
 - inputvar, [50](#)
- spt
 - inputvar, [50](#)
- startpoint
 - inputvar, [50](#)
- tamb
 - inputvar, [50](#)
- tec
 - plantvar, [63](#)
- tecb
 - inputvar, [50](#)

- tecc
 - inputvar, [50](#)
- tect
 - inputvar, [50](#)
- tempcorrb
 - inputvar, [51](#)
- tempcorr
 - inputvar, [51](#)
- tempcorrt
 - inputvar, [51](#)
- thprod
 - energy, [26](#)
- thredundant
 - heuristics, [28](#)
- threv
 - economy, [21](#)
- thselfcons
 - energy, [26](#)
- time
 - inputvar, [51](#)
- timeconstr
 - constraints.f90, [74](#)
 - graphtools::timeConstr, [66](#)
- timevinc
 - plantvar, [63](#)
- uch
 - inputvar, [51](#)
- uel
 - inputvar, [51](#)
- uptime0
 - inputvar, [51](#)
- uptimecalc
 - graphtools, [37](#)
- useheuristics
 - inputvar, [51](#)
- uth
 - inputvar, [51](#)
- vCount.f90
 - vcount, [100](#)
- vcount
 - filetools::vCount, [67](#)
 - vCount.f90, [100](#)
- verb
 - cmdvar, [12](#)
- vsilent
 - cmdvar, [12](#)
- warning
 - interfaces::warning, [67](#)
 - warning.f90, [101](#)
- warning.f90
 - warning, [101](#)
- writeboi
 - inputvar, [51](#)
- writchi
 - inputvar, [51](#)
- writchillingrev
 - inputvar, [51](#)
- writcosts
 - inputvar, [52](#)
- writdemand
 - inputvar, [52](#)
- writeefficiency
 - inputvar, [52](#)
- writeelectricrev
 - inputvar, [52](#)
- writeenergy
 - inputvar, [52](#)
- writeinput
 - inputvar, [52](#)
- writpower
 - inputvar, [52](#)
- writethermalrev
 - inputvar, [52](#)
- writetrig
 - inputvar, [52](#)
- zero
 - shared, [65](#)