

Reference Manual

Generated by Doxygen 1.7.1

Wed Nov 24 2010 18:55:00

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	GALAXY Struct Reference	5
3.1.1	Detailed Description	7
3.1.2	Field Documentation	7
3.1.2.1	BlackHoleMass	7
3.1.2.2	BulgeMass	7
3.1.2.3	BulgeSize	7
3.1.2.4	CentralGal	7
3.1.2.5	CentralMvir	7
3.1.2.6	ColdGas	7
3.1.2.7	CoolingGas	8
3.1.2.8	CoolingRadius	8
3.1.2.9	DescendantGal	8
3.1.2.10	DisruptOn	8
3.1.2.11	dObsLum	8
3.1.2.12	dObsLumBulge	8
3.1.2.13	dObsLumDust	8
3.1.2.14	dObsYLum	8
3.1.2.15	dObsYLumBulge	9
3.1.2.16	Done	9
3.1.2.17	EjectedMass	9
3.1.2.18	FirstProgGal	9
3.1.2.19	GalID	9

3.1.2.20	GasDiskRadius	9
3.1.2.21	GasSpin	9
3.1.2.22	HaloNr	9
3.1.2.23	HaloSpin	10
3.1.2.24	HotFrac	10
3.1.2.25	HotGas	10
3.1.2.26	HotRadius	10
3.1.2.27	ICLLum	10
3.1.2.28	ICM	10
3.1.2.29	Inclination	10
3.1.2.30	InfallSnap	11
3.1.2.31	InfallVmax	11
3.1.2.32	LastProgGal	11
3.1.2.33	Len	11
3.1.2.34	Lum	11
3.1.2.35	LumBulge	11
3.1.2.36	LumDust	11
3.1.2.37	MainLeaf	11
3.1.2.38	MassWeightAge	12
3.1.2.39	MergCentralPos	12
3.1.2.40	MergeOn	12
3.1.2.41	MergeSat	12
3.1.2.42	MergTime	12
3.1.2.43	MetalsBulgeMass	12
3.1.2.44	MetalsColdGas	12
3.1.2.45	MetalsEjectedMass	12
3.1.2.46	MetalsHotGas	13
3.1.2.47	MetalsICM	13
3.1.2.48	MetalsStellarMass	13
3.1.2.49	MostBoundID	13
3.1.2.50	Mvir	13
3.1.2.51	NextProgGal	13
3.1.2.52	ObsICL	13
3.1.2.53	ObsLum	14
3.1.2.54	ObsLumBulge	14
3.1.2.55	ObsLumDust	14

3.1.2.56	ObsYLum	14
3.1.2.57	ObsYLumBulge	14
3.1.2.58	OriMergTime	14
3.1.2.59	Pos	14
3.1.2.60	Rvir	15
3.1.2.61	Sfr	15
3.1.2.62	SfrBulge	15
3.1.2.63	SnapNum	15
3.1.2.64	StarMerge	15
3.1.2.65	StellarDiskRadius	15
3.1.2.66	StellarMass	15
3.1.2.67	StellarSpin	16
3.1.2.68	TreeRoot	16
3.1.2.69	Type	16
3.1.2.70	Vel	16
3.1.2.71	Vmax	16
3.1.2.72	Vvir	16
3.1.2.73	XrayLum	16
3.1.2.74	YLum	17
3.1.2.75	YLumBulge	17
3.2	GALAXY_OUTPUT Struct Reference	17
3.2.1	Detailed Description	18
3.2.2	Field Documentation	19
3.2.2.1	BlackHoleMass	19
3.2.2.2	BulgeMass	19
3.2.2.3	BulgeSize	19
3.2.2.4	CentralMvir	19
3.2.2.5	ColdGas	19
3.2.2.6	CoolingRadius	19
3.2.2.7	DescendantGal	19
3.2.2.8	DisruptOn	19
3.2.2.9	dObsMag	20
3.2.2.10	dObsMagBulge	20
3.2.2.11	dObsMagDust	20
3.2.2.12	dObsMagICL	20
3.2.2.13	EjectedMass	20

3.2.2.14	FileTreeNr	20
3.2.2.15	FirstProgGal	20
3.2.2.16	FOFCentralGal	20
3.2.2.17	GalID	20
3.2.2.18	GasDiskRadius	21
3.2.2.19	GasSpin	21
3.2.2.20	HaloID	21
3.2.2.21	HaloIndex	21
3.2.2.22	HotGas	21
3.2.2.23	HotRadius	21
3.2.2.24	ICM	21
3.2.2.25	InfallSnap	21
3.2.2.26	InfallVmax	21
3.2.2.27	LastProgGal	22
3.2.2.28	Len	22
3.2.2.29	Mag	22
3.2.2.30	MagBulge	22
3.2.2.31	MagDust	22
3.2.2.32	MagICL	22
3.2.2.33	MainLeafId	22
3.2.2.34	MassWeightAge	22
3.2.2.35	MergeOn	22
3.2.2.36	MergTime	23
3.2.2.37	MetalsBulgeMass	23
3.2.2.38	MetalsColdGas	23
3.2.2.39	MetalsEjectedMass	23
3.2.2.40	MetalsHotGas	23
3.2.2.41	MetalsICM	23
3.2.2.42	MetalsStellarMass	23
3.2.2.43	MMSubID	23
3.2.2.44	Mvir	23
3.2.2.45	NextProgGal	24
3.2.2.46	ObsMag	24
3.2.2.47	ObsMagBulge	24
3.2.2.48	ObsMagDust	24
3.2.2.49	ObsMagICL	24

3.2.2.50	OriMergTime	24
3.2.2.51	PeanoKey	24
3.2.2.52	Pos	24
3.2.2.53	Redshift	24
3.2.2.54	Rvir	25
3.2.2.55	Sfr	25
3.2.2.56	SfrBulge	25
3.2.2.57	SnapNum	25
3.2.2.58	StellarDiskRadius	25
3.2.2.59	StellarMass	25
3.2.2.60	StellarSpin	25
3.2.2.61	SubID	25
3.2.2.62	TreeRootId	25
3.2.2.63	Type	26
3.2.2.64	Vel	26
3.2.2.65	Vmax	26
3.2.2.66	Vvir	26
3.2.2.67	XrayLum	26
3.3	halo_aux_data Struct Reference	26
3.3.1	Detailed Description	26
3.3.2	Field Documentation	27
3.3.2.1	DoneFlag	27
3.3.2.2	FirstGalaxy	27
3.3.2.3	HaloFlag	27
3.3.2.4	NGalaxies	27
3.4	halo_data Struct Reference	27
3.4.1	Detailed Description	28
3.4.2	Field Documentation	28
3.4.2.1	Descendant	28
3.4.2.2	FileNr	28
3.4.2.3	FirstHaloInFOFgroup	28
3.4.2.4	FirstProgenitor	28
3.4.2.5	Len	28
3.4.2.6	M_Crit200	28
3.4.2.7	M_Mean200	28
3.4.2.8	M_TopHat	29

3.4.2.9	MostBoundID	29
3.4.2.10	NextHaloInFOFgroup	29
3.4.2.11	NextProgenitor	29
3.4.2.12	Pos	29
3.4.2.13	SnapNum	29
3.4.2.14	Spin	29
3.4.2.15	SubHalfMass	29
3.4.2.16	SubhaloIndex	29
3.4.2.17	Vel	30
3.4.2.18	VelDisp	30
3.4.2.19	Vmax	30
3.5	halo_ids_data Struct Reference	30
3.5.1	Detailed Description	30
3.5.2	Field Documentation	30
3.5.2.1	Descendant	30
3.5.2.2	dummy	31
3.5.2.3	FileTreeNr	31
3.5.2.4	FirstHaloInFOFgroup	31
3.5.2.5	FirstProgenitor	31
3.5.2.6	HaloID	31
3.5.2.7	LastProgenitor	31
3.5.2.8	MainLeafID	31
3.5.2.9	NextHaloInFOFgroup	31
3.5.2.10	NextProgenitor	31
3.5.2.11	PeanoKey	31
3.5.2.12	Redshift	31
3.6	MOMAF_INPUTS Struct Reference	32
3.6.1	Detailed Description	32
3.6.2	Field Documentation	32
3.6.2.1	dObsMagBulge	32
3.6.2.2	dObsMagDust	32
3.6.2.3	GalID	32
3.6.2.4	HaloID	32
3.6.2.5	ObsMagBulge	33
3.6.2.6	ObsMagDust	33
3.6.2.7	Pos	33

3.6.2.8	SnapNum	33
3.6.2.9	Vel	33
4	File Documentation	35
4.1	code/age.c File Reference	35
4.1.1	Detailed Description	35
4.1.2	Function Documentation	35
4.1.2.1	integrand_time_to_present	35
4.1.2.2	time_to_present	35
4.2	code/age.c	36
4.3	code/allvars.c File Reference	36
4.3.1	Variable Documentation	39
4.3.1.1	a0	39
4.3.1.2	AA	40
4.3.1.3	Age	40
4.3.1.4	AgeTab	40
4.3.1.5	AgnEfficiency	40
4.3.1.6	AGNrecipeOn	40
4.3.1.7	ar	40
4.3.1.8	BaryonFrac	40
4.3.1.9	BlackHoleGrowthRate	40
4.3.1.10	BoxSize	41
4.3.1.11	BulgeFormationInMinorMergersOn	41
4.3.1.12	CoolFunctionsDir	41
4.3.1.13	CoolingCutoff	41
4.3.1.14	CoolingVelocityCutOff	41
4.3.1.15	Corrections	41
4.3.1.16	CountIDs_halo	41
4.3.1.17	CountIDs_snaptree	41
4.3.1.18	DeltaM	41
4.3.1.19	DiskRadiusMethod	42
4.3.1.20	EjectionOn	42
4.3.1.21	EjectionRecipe	42
4.3.1.22	EjectPreVelocity	42
4.3.1.23	EjectSlope	42
4.3.1.24	EnergySN	42
4.3.1.25	EnergySNcode	42

4.3.1.26	EtaSN	42
4.3.1.27	EtaSNcode	43
4.3.1.28	FeedbackEjectionEfficiency	43
4.3.1.29	FeedbackEpsilon	43
4.3.1.30	FeedbackRecipe	43
4.3.1.31	FeedbackReheatingEpsilon	43
4.3.1.32	FileNameGalaxies	43
4.3.1.33	FileNrDir	43
4.3.1.34	FilesPerSnapshot	43
4.3.1.35	FileWithOutputSamps	44
4.3.1.36	FileWithSnapList	44
4.3.1.37	filled_galaxydata	44
4.3.1.38	filled_galsnapdata	44
4.3.1.39	filled_momafdata	44
4.3.1.40	FirstFile	44
4.3.1.41	FirstHaloInSnap	44
4.3.1.42	FoF_MaxGals	44
4.3.1.43	Frac	44
4.3.1.44	FracZtoHot	45
4.3.1.45	G	45
4.3.1.46	Gal	45
4.3.1.47	GalaxiesInOrder	45
4.3.1.48	GalCount	45
4.3.1.49	H2	45
4.3.1.50	Halo	45
4.3.1.51	HaloAux	46
4.3.1.52	HaloGal	46
4.3.1.53	HaloIDs	46
4.3.1.54	Hashbits	46
4.3.1.55	Hubble	46
4.3.1.56	Hubble_h	46
4.3.1.57	IdList	46
4.3.1.58	LastFile	46
4.3.1.59	LastSnapShotNr	47
4.3.1.60	ListInputFilrNr	47
4.3.1.61	ListOutputSamps	47

4.3.1.62	MagTableZz	47
4.3.1.63	MaxGals	47
4.3.1.64	maxstorage_galaxydata	47
4.3.1.65	maxstorage_galsnapdata	47
4.3.1.66	maxstorage_momafdata	47
4.3.1.67	metalcold	48
4.3.1.68	MetallicityOption	48
4.3.1.69	mu_seed	48
4.3.1.70	Nids	48
4.3.1.71	NTask	48
4.3.1.72	NtotHalos	48
4.3.1.73	Ntrees	48
4.3.1.74	NumGals	48
4.3.1.75	NumMergers	49
4.3.1.76	offset_auxdata	49
4.3.1.77	offset_dbids	49
4.3.1.78	offset_galaxydata	49
4.3.1.79	offset_galsnapdata	49
4.3.1.80	offset_momafdata	49
4.3.1.81	offset_treedata	49
4.3.1.82	OffsetIDs	49
4.3.1.83	OffsetIDs_halo	50
4.3.1.84	OffsetIDs_snaptree	50
4.3.1.85	Omega	50
4.3.1.86	OmegaLambda	50
4.3.1.87	output_file	50
4.3.1.88	OutputDir	50
4.3.1.89	PartMass	50
4.3.1.90	PhotDir	50
4.3.1.91	PhotPrefix	51
4.3.1.92	PosList	51
4.3.1.93	ptr_auxdata	51
4.3.1.94	ptr_dbids	51
4.3.1.95	ptr_galaxydata	51
4.3.1.96	ptr_galsnapdata	51
4.3.1.97	ptr_momafdata	51

4.3.1.98	ptr_treedata	51
4.3.1.99	random_generator	51
4.3.1.100	RecGas	52
4.3.1.101	RecycleFraction	52
4.3.1.102	RedshiftTab	52
4.3.1.103	ReheatPreVelocity	52
4.3.1.104	ReheatSlope	52
4.3.1.105	ReIncorporationFactor	52
4.3.1.106	ReIncorporationRecipe	52
4.3.1.107	Reion_Mc	52
4.3.1.108	Reion_z	52
4.3.1.109	Reionization_z0	53
4.3.1.110	Reionization_zr	53
4.3.1.111	ReionizationOn	53
4.3.1.112	Rho	53
4.3.1.113	RhoCrit	53
4.3.1.114	SatelliteRecipe	53
4.3.1.115	SfrAlpha	53
4.3.1.116	SfrEfficiency	53
4.3.1.117	SfrLawPivotVelocity	53
4.3.1.118	SfrLawSlope	54
4.3.1.119	SimulationDir	54
4.3.1.120	Snaplistlen	54
4.3.1.121	SNinReheat	54
4.3.1.122	StarBurstRecipe	54
4.3.1.123	StarBurstsInMajorMergersOn	54
4.3.1.124	StarFormationRecipe	54
4.3.1.125	ThisTask	54
4.3.1.126	ThreshMajorMerger	55
4.3.1.127	TotGalaxies	55
4.3.1.128	TotGalCount	55
4.3.1.129	TotHalos	55
4.3.1.130	TotIds	55
4.3.1.131	TotSnaps	55
4.3.1.132	TrackDiskInstability	55
4.3.1.133	tree_file	55

4.3.1.134 treeaux_file	55
4.3.1.135 treedbids_file	56
4.3.1.136 TreeFirstHalo	56
4.3.1.137 TreeNgals	56
4.3.1.138 TreeNHalos	56
4.3.1.139 UnitCoolingRate_in_cgs	56
4.3.1.140 UnitDensity_in_cgs	56
4.3.1.141 UnitEnergy_in_cgs	56
4.3.1.142 UnitLength_in_cm	56
4.3.1.143 UnitMass_in_g	57
4.3.1.144 UnitPressure_in_cgs	57
4.3.1.145 UnitTime_in_Megayears	57
4.3.1.146 UnitTime_in_s	57
4.3.1.147 UnitVelocity_in_cm_per_s	57
4.3.1.148 VelList	57
4.3.1.149 Yield	57
4.3.1.150 ZZ	57
4.4 code/allvars.c	58
4.5 code/allvars.h File Reference	61
4.5.1 Detailed Description	64
4.5.2 Variable Documentation	64
4.5.2.1 a0	64
4.5.2.2 AA	65
4.5.2.3 Age	65
4.5.2.4 AgeTab	65
4.5.2.5 AgnEfficiency	65
4.5.2.6 AGNrecipeOn	65
4.5.2.7 ar	65
4.5.2.8 BaryonFrac	65
4.5.2.9 BlackHoleGrowthRate	65
4.5.2.10 BoxSize	66
4.5.2.11 BulgeFormationInMinorMergersOn	66
4.5.2.12 CoolFunctionsDir	66
4.5.2.13 CoolingCutoff	66
4.5.2.14 CoolingVelocityCutOff	66
4.5.2.15 Corrections	66

4.5.2.16	CountIDs_halo	66
4.5.2.17	CountIDs_snaptree	66
4.5.2.18	DeltaM	66
4.5.2.19	DiskRadiusMethod	67
4.5.2.20	EjectionOn	67
4.5.2.21	EjectionRecipe	67
4.5.2.22	EjectPreVelocity	67
4.5.2.23	EjectSlope	67
4.5.2.24	EnergySN	67
4.5.2.25	EnergySNcode	67
4.5.2.26	EtaSN	67
4.5.2.27	EtaSNcode	68
4.5.2.28	FeedbackEjectionEfficiency	68
4.5.2.29	FeedbackEpsilon	68
4.5.2.30	FeedbackRecipe	68
4.5.2.31	FeedbackReheatingEpsilon	68
4.5.2.32	FileNameGalaxies	68
4.5.2.33	FileNrDir	68
4.5.2.34	FilesPerSnapshot	68
4.5.2.35	FileWithOutputSamps	69
4.5.2.36	FileWithSnapList	69
4.5.2.37	filled_galaxydata	69
4.5.2.38	filled_galsnapdata	69
4.5.2.39	filled_momafdata	69
4.5.2.40	FirstFile	69
4.5.2.41	FirstHaloInSnap	69
4.5.2.42	FoF_MaxGals	69
4.5.2.43	Frac	69
4.5.2.44	FracZtoHot	70
4.5.2.45	G	70
4.5.2.46	Gal	70
4.5.2.47	GalaxiesInOrder	70
4.5.2.48	GalCount	70
4.5.2.49	H2	70
4.5.2.50	Halo	70
4.5.2.51	HaloAux	70

4.5.2.52	HaloGal	70
4.5.2.53	HaloIDs	70
4.5.2.54	Hashbits	70
4.5.2.55	Hubble	70
4.5.2.56	Hubble_h	71
4.5.2.57	IdList	71
4.5.2.58	IdxTable	71
4.5.2.59	IdxType	71
4.5.2.60	Idxw5	71
4.5.2.61	Idxw6	71
4.5.2.62	LastFile	71
4.5.2.63	LastSnapShotNr	71
4.5.2.64	ListInputFilrNr	71
4.5.2.65	ListOutputSamps	71
4.5.2.66	MagTableZz	71
4.5.2.67	MaxGals	72
4.5.2.68	maxstorage_galaxydata	72
4.5.2.69	maxstorage_galsnapdata	72
4.5.2.70	maxstorage_momafdata	72
4.5.2.71	metalcold	72
4.5.2.72	MetallicityOption	72
4.5.2.73	mu_seed	72
4.5.2.74	Nids	72
4.5.2.75	NTask	73
4.5.2.76	NtotHalos	73
4.5.2.77	Ntrees	73
4.5.2.78	NumGals	73
4.5.2.79	NumMergers	73
4.5.2.80	offset_auxdata	73
4.5.2.81	offset_dbids	73
4.5.2.82	offset_galaxydata	73
4.5.2.83	offset_galsnapdata	74
4.5.2.84	offset_momafdata	74
4.5.2.85	offset_treedata	74
4.5.2.86	OffsetIDs	74
4.5.2.87	OffsetIDs_halo	74

4.5.2.88	OffsetIDs_snaptree	74
4.5.2.89	Omega	74
4.5.2.90	OmegaLambda	74
4.5.2.91	output_file	75
4.5.2.92	OutputDir	75
4.5.2.93	PartMass	75
4.5.2.94	PhotDir	75
4.5.2.95	PhotPrefix	75
4.5.2.96	PosList	75
4.5.2.97	ptr_auxdata	75
4.5.2.98	ptr_dbids	75
4.5.2.99	ptr_galaxydata	76
4.5.2.100	ptr_galsnapdata	76
4.5.2.101	ptr_momafdata	76
4.5.2.102	ptr_treedata	76
4.5.2.103	random_generator	76
4.5.2.104	RecGas	76
4.5.2.105	RecycleFraction	76
4.5.2.106	RedshiftTab	76
4.5.2.107	ReheatPreVelocity	76
4.5.2.108	ReheatSlope	77
4.5.2.109	ReIncorporationFactor	77
4.5.2.110	ReIncorporationRecipe	77
4.5.2.111	Reion_Mc	77
4.5.2.112	Reion_z	77
4.5.2.113	Reionization_z0	77
4.5.2.114	Reionization_zr	77
4.5.2.115	ReionizationOn	77
4.5.2.116	Rho	77
4.5.2.117	RhoCrit	78
4.5.2.118	SatelliteRecipe	78
4.5.2.119	SfrAlpha	78
4.5.2.120	SfrEfficiency	78
4.5.2.121	SfrLawPivotVelocity	78
4.5.2.122	SfrLawSlope	78
4.5.2.123	SimulationDir	78

4.5.2.124	Snaplistlen	78
4.5.2.125	SNinReheat	78
4.5.2.126	StarBurstRecipe	79
4.5.2.127	StarBurstsInMajorMergersOn	79
4.5.2.128	StarFormationRecipe	79
4.5.2.129	StelliteRecipe	79
4.5.2.130	ThisTask	79
4.5.2.131	ThreshMajorMerger	79
4.5.2.132	TotGalaxies	79
4.5.2.133	TotGalCount	79
4.5.2.134	TotHalos	79
4.5.2.135	TotIds	80
4.5.2.136	TotSnaps	80
4.5.2.137	TrackDiskInstability	80
4.5.2.138	tree_file	80
4.5.2.139	treeaux_file	80
4.5.2.140	treedbids_file	80
4.5.2.141	TreeFirstHalo	80
4.5.2.142	TreeNgals	80
4.5.2.143	TreeNHalos	80
4.5.2.144	UnitCoolingRate_in_cgs	81
4.5.2.145	UnitDensity_in_cgs	81
4.5.2.146	UnitEnergy_in_cgs	81
4.5.2.147	UnitLength_in_cm	81
4.5.2.148	UnitMass_in_g	81
4.5.2.149	UnitPressure_in_cgs	81
4.5.2.150	UnitTime_in_Megayears	81
4.5.2.151	UnitTime_in_s	81
4.5.2.152	UnitVelocity_in_cm_per_s	82
4.5.2.153	VelList	82
4.5.2.154	Yield	82
4.5.2.155	ZZ	82
4.6	code/allvars.h	82
4.7	code/cool_func.c File Reference	92
4.7.1	Function Documentation	92
4.7.1.1	dmin	92

4.7.1.2	get_metaldependent_cooling_rate	92
4.7.1.3	get_rate	92
4.7.1.4	read_cooling_functions	92
4.7.1.5	test	93
4.7.2	Variable Documentation	93
4.7.2.1	CoolRate	93
4.7.2.2	metallicities	93
4.7.2.3	name	93
4.8	code/cool_func.c	93
4.9	code/init.c File Reference	96
4.9.1	Detailed Description	97
4.9.2	Function Documentation	98
4.9.2.1	find_index	98
4.9.2.2	find_interpolate_h2	98
4.9.2.3	find_interpolate_reionization	98
4.9.2.4	find_interpolated_lum	99
4.9.2.5	find_interpolated_obs_lum	99
4.9.2.6	find_interpolation_point	99
4.9.2.7	get_jump_index	99
4.9.2.8	init	99
4.9.2.9	init_jump_index	99
4.9.2.10	read_file_nrs	100
4.9.2.11	read_output_snaps	100
4.9.2.12	read_recgas	100
4.9.2.13	read_reionization	100
4.9.2.14	read_sfrz	100
4.9.2.15	read_snap_list	100
4.9.2.16	read_tsud_tables	100
4.9.2.17	set_units	101
4.9.2.18	setup_spectrophotometric_model	102
4.9.3	Variable Documentation	102
4.9.3.1	jumpfac	102
4.9.3.2	jumptab	103
4.10	code/init.c	103
4.11	code/io_tree.c File Reference	120
4.11.1	Detailed Description	121

4.11.2 Function Documentation	122
4.11.2.1 closeallfiles	122
4.11.2.2 free_galaxies_and_tree	122
4.11.2.3 free_tree_table	122
4.11.2.4 load_all_auxdata	122
4.11.2.5 load_all_dbids	122
4.11.2.6 load_all_treedata	123
4.11.2.7 load_tree	123
4.11.2.8 load_tree_table	123
4.11.2.9 myfread	124
4.11.2.10 myfseek	124
4.11.2.11 myfwrite	124
4.11.2.12 open_outputtree_file	125
4.11.2.13 open_tree_file	125
4.11.2.14 open_treeaux_file	125
4.11.2.15 open_treedbids_file	125
4.11.2.16 openallfiles	125
4.11.2.17 write_all_galaxy_data	125
4.11.2.18 write_galaxy_data_snap	125
4.11.2.19 write_galaxy_for_momaf	126
4.12 code/io_tree.c	126
4.13 code/main.c File Reference	141
4.13.1 Detailed Description	141
4.13.2 Function Documentation	142
4.13.2.1 main	142
4.14 code/main.c	143
4.15 code/mymalloc.c File Reference	154
4.15.1 Function Documentation	155
4.15.1.1 myfree	155
4.15.1.2 mymalloc	155
4.15.1.3 print_allocated	155
4.15.2 Variable Documentation	155
4.15.2.1 HighMarkMem	155
4.15.2.2 Nblocks	155
4.15.2.3 OldPrintedHighMark	155
4.15.2.4 SizeTable	155

4.15.2.5	Table	156
4.15.2.6	TotMem	156
4.16	code/mymalloc.c	156
4.17	code/peano.c File Reference	157
4.17.1	Function Documentation	157
4.17.1.1	peano_hilbert_key	157
4.17.2	Variable Documentation	158
4.17.2.1	quadrants	158
4.17.2.2	rotx_table	158
4.17.2.3	rotxmap_table	158
4.17.2.4	roty_table	158
4.17.2.5	rotymap_table	158
4.17.2.6	sense_table	158
4.18	code/peano.c	159
4.19	code/proto.h File Reference	160
4.19.1	Function Documentation	166
4.19.1.1	add_galaxies_together	166
4.19.1.2	add_infall_to_hot	167
4.19.1.3	add_to_luminosities	167
4.19.1.4	bulge_from_disk	167
4.19.1.5	bulgemass_r	167
4.19.1.6	bulgesize_from_merger	168
4.19.1.7	cal_gas_recycle	168
4.19.1.8	cal_H2	168
4.19.1.9	check_disk_instability	168
4.19.1.10	check_options	169
4.19.1.11	checkbulgesize_main	169
4.19.1.12	closeallfiles	169
4.19.1.13	collisional_starburst_recipe	169
4.19.1.14	construct_galaxies	169
4.19.1.15	cool_gas_onto_galaxy	169
4.19.1.16	cooling_recipe	169
4.19.1.17	deal_with_galaxy_merger	170
4.19.1.18	diskmass_r	170
4.19.1.19	disrupt	170
4.19.1.20	dmax	170

4.19.1.21	dmin	171
4.19.1.22	do_AGN_heating	171
4.19.1.23	do_major_merger_starburst	171
4.19.1.24	do_reionization	171
4.19.1.25	Energy_in_Reheat	171
4.19.1.26	estimate_merging_time	172
4.19.1.27	evolve_galaxies	172
4.19.1.28	finalize_galaxy_file	172
4.19.1.29	finalize_momaf_file	172
4.19.1.30	find_interpolate_h2	172
4.19.1.31	find_interpolated_lum	173
4.19.1.32	find_interpolated_obs_lum	173
4.19.1.33	find_interpolation_point	173
4.19.1.34	fix_units_for_ouput	173
4.19.1.35	free_galaxies_and_tree	173
4.19.1.36	free_tree_table	173
4.19.1.37	func_size	174
4.19.1.38	get_coordinates	174
4.19.1.39	get_disk_radius	174
4.19.1.40	get_gas_disk_radius	174
4.19.1.41	get_initial_disk_radius	174
4.19.1.42	get_jump_index	175
4.19.1.43	get_metaldependent_cooling_rate	175
4.19.1.44	get_metallicity	175
4.19.1.45	get_rate	175
4.19.1.46	get_stellar_disk_radius	175
4.19.1.47	get_virial_mass	176
4.19.1.48	get_virial_radius	176
4.19.1.49	get_virial_velocity	176
4.19.1.50	grow_black_hole	176
4.19.1.51	hot_retain_sat	177
4.19.1.52	infall_recipe	177
4.19.1.53	init	177
4.19.1.54	init_galaxy	178
4.19.1.55	init_jump_index	178
4.19.1.56	integrand_time_to_present	178

4.19.1.57 join_galaxies_of_progenitors	178
4.19.1.58 join_galaxies_of_progenitors	178
4.19.1.59 load_all_auxdata	178
4.19.1.60 load_all_dbids	179
4.19.1.61 load_all_treedata	179
4.19.1.62 load_tree	179
4.19.1.63 load_tree_table	179
4.19.1.64 lum_to_mag	180
4.19.1.65 make_bulge_from_burst	180
4.19.1.66 momaf_fwrite	180
4.19.1.67 myfread	180
4.19.1.68 myfree	181
4.19.1.69 myfseek	181
4.19.1.70 myfwrite	181
4.19.1.71 mymalloc	181
4.19.1.72 NumToTime	182
4.19.1.73 open_outputtree_file	182
4.19.1.74 open_tree_file	182
4.19.1.75 open_treeaux_file	182
4.19.1.76 open_treedbids_file	182
4.19.1.77 openallfiles	182
4.19.1.78 peano_hilbert_key	182
4.19.1.79 peri_radius	183
4.19.1.80 prepare_galaxy_for_momaf	183
4.19.1.81 prepare_galaxy_for_output	183
4.19.1.82 print_allocated	184
4.19.1.83 read_cooling_functions	184
4.19.1.84 read_file_nrs	184
4.19.1.85 read_output_snaps	184
4.19.1.86 read_parameter_file	184
4.19.1.87 read_recgas	185
4.19.1.88 read_sfrz	185
4.19.1.89 read_snap_list	185
4.19.1.90 read_tsud_tables	185
4.19.1.91 RedshiftObs	186
4.19.1.92 reincorporate_gas	186

4.19.1.93 sat_radius	186
4.19.1.94 save_galaxies	186
4.19.1.95 save_galaxy_tree	187
4.19.1.96 set_merger_center	187
4.19.1.97 set_units	187
4.19.1.98 setup_spectrophotometric_model	188
4.19.1.99 slab_model	189
4.19.1.100starformation_and_feedback	189
4.19.1.101time_to_present	189
4.19.1.102study	189
4.19.1.103update_bulge_from_disk	189
4.19.1.104update_centralgal	190
4.19.1.105update_from_feedback	190
4.19.1.106update_from_recycle	190
4.19.1.107update_from_star_formation	190
4.19.1.108update_hot_frac	190
4.19.1.109update_hotgas	190
4.19.1.110update_ICL	191
4.19.1.111update_type_1	191
4.19.1.112update_type_2	191
4.19.1.113walk	191
4.19.1.114write_all_galaxy_data	192
4.19.1.115write_galaxy_data_snap	192
4.19.1.116write_galaxy_for_momaf	192
4.20 code/proto.h	192
4.21 code/read_parameters.c File Reference	195
4.21.1 Detailed Description	195
4.21.2 Function Documentation	196
4.21.2.1 read_parameter_file	196
4.22 code/read_parameters.c	196
4.23 code/recipe_cooling.c File Reference	202
4.23.1 Detailed Description	202
4.23.2 Function Documentation	203
4.23.2.1 cool_gas_onto_galaxy	203
4.23.2.2 cooling_recipe	203
4.23.2.3 do_AGN_heating	203

4.24 code/recipe_cooling.c	203
4.25 code/recipe_disrupt.c File Reference	209
4.25.1 Detailed Description	209
4.25.2 Function Documentation	209
4.25.2.1 bulgemass_r	209
4.25.2.2 diskmass_r	210
4.25.2.3 disrupt	210
4.25.2.4 peri_radius	210
4.25.2.5 sat_radius	210
4.26 code/recipe_disrupt.c	210
4.27 code/recipe_dust.c File Reference	215
4.27.1 Detailed Description	215
4.27.2 Function Documentation	215
4.27.2.1 gasdev	215
4.27.2.2 ran1	216
4.27.2.3 tstudy	216
4.28 code/recipe_dust.c	216
4.29 code/recipe_infall.c File Reference	220
4.29.1 Detailed Description	221
4.29.2 Function Documentation	221
4.29.2.1 add_infall_to_hot	221
4.29.2.2 do_reionization	221
4.29.2.3 hot_retain_sat	222
4.29.2.4 infall_recipe	222
4.29.2.5 update_hot_frac	222
4.29.2.6 update_ICL	222
4.30 code/recipe_infall.c	223
4.31 code/recipe_mergers.c File Reference	237
4.31.1 Detailed Description	238
4.31.2 Function Documentation	239
4.31.2.1 add_galaxies_together	239
4.31.2.2 bulgesize_from_merger	239
4.31.2.3 collisional_starburst_recipe	239
4.31.2.4 deal_with_galaxy_merger	240
4.31.2.5 do_major_merger_starburst	240
4.31.2.6 estimate_merging_time	241

4.31.2.7	grow_black_hole	241
4.31.2.8	make_bulge_from_burst	241
4.31.2.9	set_merger_center	241
4.32	code/recipe_mergers.c	242
4.33	code/recipe_misc.c File Reference	259
4.33.1	Detailed Description	261
4.33.2	Function Documentation	261
4.33.2.1	add_to_luminosities	261
4.33.2.2	cal_H2	261
4.33.2.3	dmax	262
4.33.2.4	get_disk_radius	262
4.33.2.5	get_gas_disk_radius	262
4.33.2.6	get_initial_disk_radius	262
4.33.2.7	get_metallicity	263
4.33.2.8	get_stellar_disk_radius	263
4.33.2.9	get_virial_mass	263
4.33.2.10	get_virial_radius	263
4.33.2.11	get_virial_velocity	263
4.33.2.12	init_galaxy	264
4.33.2.13	lum_to_mag	264
4.33.2.14	NumToTime	264
4.33.2.15	RedshiftObs	264
4.33.2.16	update_centralgal	265
4.33.2.17	update_hotgas	265
4.33.2.18	update_type_1	265
4.33.2.19	update_type_2	265
4.34	code/recipe_misc.c	266
4.35	code/recipe_reincorporation.c File Reference	280
4.35.1	Detailed Description	281
4.35.2	Function Documentation	281
4.35.2.1	reincorporate_gas	281
4.36	code/recipe_reincorporation.c	281
4.37	code/recipe_starformation_and_feedback.c File Reference	283
4.37.1	Detailed Description	283
4.37.2	Function Documentation	284
4.37.2.1	cal_gas_recycle	284

4.37.2.2	check_disk_instability	284
4.37.2.3	starformation_and_feedback	284
4.37.2.4	update_from_feedback	285
4.37.2.5	update_from_star_formation	285
4.38	code/recipe_starformation_and_feedback.c	285
4.39	code/save.c File Reference	301
4.39.1	Detailed Description	302
4.39.2	Function Documentation	302
4.39.2.1	finalize_galaxy_file	302
4.39.2.2	finalize_momaf_file	302
4.39.2.3	fix_units_for_ouput	303
4.39.2.4	get_coordinates	303
4.39.2.5	prepare_galaxy_for_momaf	303
4.39.2.6	prepare_galaxy_for_output	303
4.39.2.7	save_galaxies	304
4.39.2.8	save_galaxy_tree	305
4.39.2.9	walk	305
4.40	code/save.c	305

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

GALAXY	5
GALAXY_OUTPUT (Galaxy structure for output)	17
halo_aux_data	26
halo_data	27
halo_ids_data	30
MOMAF_INPUTS	32

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

code/age.c (Returns time to present for a given redshift)	35
code/allvars.c	36
code/allvars.h (TODO add description for all variables that do not have one yet)	61
code/cool_func.c	92
code/init.c (Sets up some unit conversion variables; converts SN and AGN feedback variables into internal units; and reads in input tables, including the desired output snapshots, the photometric and dust tables, the cooling functions and reionization tables)	96
code/io_tree.c (Reads in the data from the dark matter simulation merger trees, creates output files and after calculations frees the allocated memory)	120
code/main.c (The file containing the main() function for L-Galaxies)	141
code/mymalloc.c	154
code/peano.c	157
code/proto.h	160
code/read_parameters.c (Reads all the parameters in input.par into global variables that can be used by the code)	195
code/recipe_cooling.c (Recipe_cooling.c calculates the amount of mass that cools from the hot to the cold phase at each timestep (this fraction is then reduced by AGN heating) and updates the hot and cold gas fractions accordingly)	202
code/recipe_disrupt.c (Recipe_disrupt.c checks if a type 2 satellite galaxy should or not be disrupted due to tidal forces)	209
code/recipe_dust.c (Recipe_dust.c is used to compute dust extinction as described in Delucia2007 + redshift dependence as Kitzbichler & White 2007)	215
code/recipe_infall.c (Recipe_infall.c calculates the amount of gas that infalls into the galaxy hot gas component at each time step)	220
code/recipe_mergers.c (Calculates the merging time, the central galaxy (for type 1's), adds galaxies together, calculates SF from bursts and grows black holes)	237
code/recipe_misc.c (Recipe_misc.c contains a mix of recipes used to: calculate disk sizes, initiate a galaxy structure, get the metallicity, add luminosities, convert snap to age, convert snap to z, calculate max of two numbers, get virial mass, get virial velocity, get virial radius, luminosity to mass, H2 conversion, update central galaxy, update type 1 and type2)	259
code/recipe_reincorporation.c (Recipe_reincorporation.c calculates the fraction of ejected gas that gets reincorporated into the hot fraction per timestep)	280

code/ recipe_starformation_and_feedback.c (Recipe_starformation_and_feedback.c computes the amount of stars formed from the cold gas, the amount of gas reheated from cold to hot and the amount of gas ejected from hot to external)	283
code/ save.c (Copies the relevant properties in Galaxy structure into Galaxy_Output structure and saves them into the output files (SA_z**_**) - redshift/filenr)	301

Chapter 3

Data Structure Documentation

3.1 GALAXY Struct Reference

```
#include <allvars.h>
```

Data Fields

- int **GalID**
- int **FirstProgGal**
- int **NextProgGal**
- int **LastProgGal**
- int **DescendantGal**
- int **MainLeaf**
- int **TreeRoot**
- int **Done**
- int **Type**
- int **HaloNr**
- long long **MostBoundID**
- int **SnapNum**
- int **CentralGal**
- float **CentralMvir**
- float **Pos** [3]
- float **MergCentralPos** [3]
- float **Vel** [3]
- float **HaloSpin** [3]
- float **GasSpin** [3]
- float **StellarSpin** [3]
- int **Len**
- float **Mvir**
- float **Rvir**
- float **Vvir**
- float **Vmax**
- float **InfallVmax**
- float **InfallSnap**
- float **CoolingGas**

- float **HotFrac**
- float **HotRadius**
- float **ColdGas**
- float **StellarMass**
- float **BulgeMass**
- float **HotGas**
- float **EjectedMass**
- float **BlackHoleMass**
- float **MetalsColdGas**
- float **MetalsStellarMass**
- float **MetalsBulgeMass**
- float **MetalsHotGas**
- float **MetalsEjectedMass**
- float **Sfr** [MAXSNAPS]
- float **SfrBulge** [MAXSNAPS]
- float **StarMerge**
- float **XrayLum**
- float **BulgeSize**
- float **StellarDiskRadius**
- float **GasDiskRadius**
- int **DisruptOn**
- float **Inclination**
- float **OriMergTime**
- float **MergeSat**
- float **MergTime**
- float **MergeOn**
- float **CoolingRadius**
- float **ICLLum** [NMAG][NOUT]
- float **Lum** [NMAG][NOUT]
- float **YLum** [NMAG][NOUT]
- float **LumBulge** [NMAG][NOUT]
- float **YLumBulge** [NMAG][NOUT]
- float **LumDust** [NMAG][NOUT]
- float **MassWeightAge** [NOUT]
- float **ICM**
- float **MetalsICM**
- float **ObsLum** [NMAG][NOUT]
- float **ObsYLum** [NMAG][NOUT]
- float **ObsLumBulge** [NMAG][NOUT]
- float **ObsYLumBulge** [NMAG][NOUT]
- float **ObsLumDust** [NMAG][NOUT]
- float **ObsICL** [NMAG][NOUT]
- float **dObsLum** [NMAG][NOUT]
- float **dObsYLum** [NMAG][NOUT]
- float **dObsLumBulge** [NMAG][NOUT]
- float **dObsYLumBulge** [NMAG][NOUT]
- float **dObsLumDust** [NMAG][NOUT]

3.1.1 Detailed Description

Definition at line 203 of file [allvars.h](#).

3.1.2 Field Documentation

3.1.2.1 float GALAXY::BlackHoleMass

Definition at line 245 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [do_AGN_heating\(\)](#), [grow_black_hole\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.1.2.2 float GALAXY::BulgeMass

Definition at line 242 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [bulgesize_from_merger\(\)](#), [check_disk_instability\(\)](#), [deal_with_galaxy_merger\(\)](#), [init_galaxy\(\)](#), [make_bulge_from_burst\(\)](#), [prepare_galaxy_for_output\(\)](#), [sat_radius\(\)](#), and [update_from_star_formation\(\)](#).

3.1.2.3 float GALAXY::BulgeSize

Definition at line 268 of file [allvars.h](#).

Referenced by [bulgesize_from_merger\(\)](#), [check_disk_instability\(\)](#), [deal_with_galaxy_merger\(\)](#), [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), and [sat_radius\(\)](#).

3.1.2.4 int GALAXY::CentralGal

Definition at line 219 of file [allvars.h](#).

Referenced by [disrupt\(\)](#), [do_major_merger_stellar\(\)](#), [infall_recipe\(\)](#), [main\(\)](#), [starformation_and_feedback\(\)](#), [update_from_feedback\(\)](#), and [update_hotgas\(\)](#).

3.1.2.5 float GALAXY::CentralMvir

Definition at line 220 of file [allvars.h](#).

3.1.2.6 float GALAXY::ColdGas

Definition at line 240 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [bulgesize_from_merger\(\)](#), [cal_H2\(\)](#), [collisional_starburst_recipe\(\)](#), [cool_gas_onto_galaxy\(\)](#), [deal_with_galaxy_merger\(\)](#), [disrupt\(\)](#), [do_major_merger_stellar\(\)](#), [grow_black_hole\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), [sat_radius\(\)](#), [starformation_and_feedback\(\)](#), [tsudy\(\)](#), [update_from_feedback\(\)](#), [update_from_star_formation\(\)](#), and [update_type_1\(\)](#).

3.1.2.7 float GALAXY::CoolingGas

Definition at line 236 of file [allvars.h](#).

Referenced by [cooling_recipe\(\)](#), and [main\(\)](#).

3.1.2.8 float GALAXY::CoolingRadius

Definition at line 281 of file [allvars.h](#).

Referenced by [cooling_recipe\(\)](#), [init_galaxy\(\)](#), [main\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.1.2.9 int GALAXY::DescendantGal

Definition at line 210 of file [allvars.h](#).

Referenced by [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), and [save_galaxy_tree\(\)](#).

3.1.2.10 int GALAXY::DisruptOn

Definition at line 272 of file [allvars.h](#).

Referenced by [disrupt\(\)](#), [init_galaxy\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.1.2.11 float GALAXY::dObsLum[NMAG][NOUT]

Definition at line 309 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [add_to_luminosities\(\)](#), [check_disk_instability\(\)](#), [init_galaxy\(\)](#), [make_bulge_from_burst\(\)](#), [prepare_galaxy_for_output\(\)](#), and [tsudy\(\)](#).

3.1.2.12 float GALAXY::dObsLumBulge[NMAG][NOUT]

Definition at line 311 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [check_disk_instability\(\)](#), [init_galaxy\(\)](#), [make_bulge_from_burst\(\)](#), [prepare_galaxy_for_momaf\(\)](#), [prepare_galaxy_for_output\(\)](#), and [tsudy\(\)](#).

3.1.2.13 float GALAXY::dObsLumDust[NMAG][NOUT]

Definition at line 313 of file [allvars.h](#).

Referenced by [init_galaxy\(\)](#), [prepare_galaxy_for_momaf\(\)](#), [prepare_galaxy_for_output\(\)](#), and [tsudy\(\)](#).

3.1.2.14 float GALAXY::dObsYLum[NMAG][NOUT]

Definition at line 310 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [add_to_luminosities\(\)](#), [check_disk_instability\(\)](#), [init_galaxy\(\)](#), [make_bulge_from_burst\(\)](#), and [tsudy\(\)](#).

3.1.2.15 float GALAXY::dObsYLumBulge[NMAG][NOUT]

Definition at line 312 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [check_disk_instability\(\)](#), [init_galaxy\(\)](#), [make_bulge_from_burst\(\)](#), and [tsudy\(\)](#).

3.1.2.16 int GALAXY::Done

Definition at line 213 of file [allvars.h](#).

Referenced by [save_galaxy_tree\(\)](#), and [walk\(\)](#).

3.1.2.17 float GALAXY::EjectedMass

Definition at line 244 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), [reincorporate_gas\(\)](#), and [update_from_feedback\(\)](#).

3.1.2.18 int GALAXY::FirstProgGal

Definition at line 207 of file [allvars.h](#).

Referenced by [deal_with_galaxy_merger\(\)](#), [disrupt\(\)](#), [init_galaxy\(\)](#), [main\(\)](#), [prepare_galaxy_for_output\(\)](#), [save_galaxy_tree\(\)](#), and [walk\(\)](#).

3.1.2.19 int GALAXY::GalID

Definition at line 206 of file [allvars.h](#).

Referenced by [prepare_galaxy_for_momaf\(\)](#), [prepare_galaxy_for_output\(\)](#), [save_galaxy_tree\(\)](#), and [walk\(\)](#).

3.1.2.20 float GALAXY::GasDiskRadius

Definition at line 270 of file [allvars.h](#).

Referenced by [bulgesize_from_merger\(\)](#), [cal_H2\(\)](#), [deal_with_galaxy_merger\(\)](#), [get_gas_disk_radius\(\)](#), [init_galaxy\(\)](#), [main\(\)](#), [prepare_galaxy_for_output\(\)](#), [sat_radius\(\)](#), [starformation_and_feedback\(\)](#), and [tsudy\(\)](#).

3.1.2.21 float GALAXY::GasSpin[3]

Definition at line 226 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [get_gas_disk_radius\(\)](#), [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), and [update_from_star_formation\(\)](#).

3.1.2.22 int GALAXY::HaloNr

Definition at line 216 of file [allvars.h](#).

Referenced by [bulgesize_from_merger\(\)](#), [collisional_starburst_recipe\(\)](#), [do_major_merger_starburst\(\)](#), [hot_retain_sat\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), [main\(\)](#), [peri_radius\(\)](#), [prepare_galaxy_for_momaf\(\)](#), [prepare_galaxy_for_output\(\)](#), [reincorporate_gas\(\)](#), [starformation_and_feedback\(\)](#), [update_from_feedback\(\)](#), and [update_hotgas\(\)](#).

3.1.2.23 float GALAXY::HaloSpin[3]

Definition at line [225](#) of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [init_galaxy\(\)](#), and [update_centralgal\(\)](#).

3.1.2.24 float GALAXY::HotFrac

Definition at line [237](#) of file [allvars.h](#).

Referenced by [hot_retain_sat\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), [update_hot_frac\(\)](#), [update_hotgas\(\)](#), [update_type_1\(\)](#), and [update_type_2\(\)](#).

3.1.2.25 float GALAXY::HotGas

Definition at line [243](#) of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [add_infall_to_hot\(\)](#), [cool_gas_onto_galaxy\(\)](#), [cooling_recipe\(\)](#), [disrupt\(\)](#), [do_AGN_heating\(\)](#), [hot_retain_sat\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), [reincorporate_gas\(\)](#), [update_from_feedback\(\)](#), [update_hotgas\(\)](#), and [update_type_1\(\)](#).

3.1.2.26 float GALAXY::HotRadius

Definition at line [238](#) of file [allvars.h](#).

Referenced by [cooling_recipe\(\)](#), [do_AGN_heating\(\)](#), [hot_retain_sat\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), [update_centralgal\(\)](#), [update_from_feedback\(\)](#), [update_hot_frac\(\)](#), and [update_type_2\(\)](#).

3.1.2.27 float GALAXY::ICLLum[NMAG][NOUT]

Definition at line [285](#) of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [disrupt\(\)](#), [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), and [update_ICL\(\)](#).

3.1.2.28 float GALAXY::ICM

Definition at line [295](#) of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [disrupt\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.1.2.29 float GALAXY::Inclination

Definition at line [276](#) of file [allvars.h](#).

Referenced by [init_galaxy\(\)](#), and [tsudy\(\)](#).

3.1.2.30 float GALAXY::InfallSnap

Definition at line 235 of file [allvars.h](#).

Referenced by [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), and [update_centralgal\(\)](#).

3.1.2.31 float GALAXY::InfallVmax

Definition at line 233 of file [allvars.h](#).

Referenced by [check_disk_instability\(\)](#), [collisional_starburst_recipe\(\)](#), [get_gas_disk_radius\(\)](#), [get_stellar_disk_radius\(\)](#), [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), [starformation_and_feedback\(\)](#), and [update_centralgal\(\)](#).

3.1.2.32 int GALAXY::LastProgGal

Definition at line 209 of file [allvars.h](#).

Referenced by [prepare_galaxy_for_output\(\)](#), [save_galaxy_tree\(\)](#), and [walk\(\)](#).

3.1.2.33 int GALAXY::Len

Definition at line 228 of file [allvars.h](#).

Referenced by [estimate_merging_time\(\)](#), [hot_retain_sat\(\)](#), [init_galaxy\(\)](#), [main\(\)](#), [prepare_galaxy_for_output\(\)](#), [set_merger_center\(\)](#), [update_hot_frac\(\)](#), [update_hotgas\(\)](#), and [update_type_1\(\)](#).

3.1.2.34 float GALAXY::Lum[NMAG][NOUT]

Definition at line 287 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [add_to_luminosities\(\)](#), [check_disk_instability\(\)](#), [disrupt\(\)](#), [init_galaxy\(\)](#), [make_bulge_from_burst\(\)](#), [prepare_galaxy_for_output\(\)](#), and [tsudy\(\)](#).

3.1.2.35 float GALAXY::LumBulge[NMAG][NOUT]

Definition at line 289 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [check_disk_instability\(\)](#), [init_galaxy\(\)](#), [make_bulge_from_burst\(\)](#), [prepare_galaxy_for_output\(\)](#), and [tsudy\(\)](#).

3.1.2.36 float GALAXY::LumDust[NMAG][NOUT]

Definition at line 291 of file [allvars.h](#).

Referenced by [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), and [tsudy\(\)](#).

3.1.2.37 int GALAXY::MainLeaf

Definition at line 211 of file [allvars.h](#).

Referenced by [prepare_galaxy_for_output\(\)](#), [save_galaxy_tree\(\)](#), and [walk\(\)](#).

3.1.2.38 float GALAXY::MassWeightAge[NOUT]

Definition at line 292 of file `allvars.h`.

Referenced by `add_galaxies_together()`, `add_to_luminosities()`, `init_galaxy()`, and `prepare_galaxy_for_output()`.

3.1.2.39 float GALAXY::MergCentralPos[3]

Definition at line 223 of file `allvars.h`.

Referenced by `init_galaxy()`, `prepare_galaxy_for_momaf()`, and `prepare_galaxy_for_output()`.

3.1.2.40 float GALAXY::MergeOn

Definition at line 280 of file `allvars.h`.

Referenced by `deal_with_galaxy_merger()`, `prepare_galaxy_for_output()`, `update_centralgal()`, `update_type_1()`, and `update_type_2()`.

3.1.2.41 float GALAXY::MergeSat

Definition at line 278 of file `allvars.h`.

Referenced by `add_galaxies_together()`, and `init_galaxy()`.

3.1.2.42 float GALAXY::MergTime

Definition at line 279 of file `allvars.h`.

Referenced by `init_galaxy()`, `main()`, `prepare_galaxy_for_momaf()`, `prepare_galaxy_for_output()`, `update_type_1()`, and `update_type_2()`.

3.1.2.43 float GALAXY::MetalsBulgeMass

Definition at line 249 of file `allvars.h`.

Referenced by `add_galaxies_together()`, `check_disk_instability()`, `init_galaxy()`, `make_bulge_from_burst()`, and `prepare_galaxy_for_output()`.

3.1.2.44 float GALAXY::MetalsColdGas

Definition at line 247 of file `allvars.h`.

Referenced by `add_galaxies_together()`, `cal_H2()`, `collisional_starburst_recipe()`, `cool_gas_onto_galaxy()`, `disrupt()`, `do_major_merger_starburst()`, `grow_black_hole()`, `init_galaxy()`, `prepare_galaxy_for_output()`, `starformation_and_feedback()`, `tsudy()`, `update_from_feedback()`, and `update_from_starFormation()`.

3.1.2.45 float GALAXY::MetalsEjectedMass

Definition at line 251 of file `allvars.h`.

Referenced by [add_galaxies_together\(\)](#), [do_major_merger_starburst\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), [reincorporate_gas\(\)](#), and [update_from_feedback\(\)](#).

3.1.2.46 float GALAXY::MetalsHotGas

Definition at line [250](#) of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [add_infall_to_hot\(\)](#), [collisional_starburst_recipe\(\)](#), [cool_gas_onto_galaxy\(\)](#), [cooling_recipe\(\)](#), [disrupt\(\)](#), [do_AGN_heating\(\)](#), [do_major_merger_starburst\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), [reincorporate_gas\(\)](#), [starformation_and_feedback\(\)](#), and [update_from_feedback\(\)](#).

3.1.2.47 float GALAXY::MetalsICM

Definition at line [296](#) of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [disrupt\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.1.2.48 float GALAXY::MetalsStellarMass

Definition at line [248](#) of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [check_disk_instability\(\)](#), [disrupt\(\)](#), [do_major_merger_starburst\(\)](#), [init_galaxy\(\)](#), [make_bulge_from_burst\(\)](#), [prepare_galaxy_for_output\(\)](#), and [update_from_starFormation\(\)](#).

3.1.2.49 long long GALAXY::MostBoundID

Definition at line [217](#) of file [allvars.h](#).

Referenced by [init_galaxy\(\)](#), and [main\(\)](#).

3.1.2.50 float GALAXY::Mvir

Definition at line [229](#) of file [allvars.h](#).

Referenced by [disrupt\(\)](#), [do_AGN_heating\(\)](#), [do_reionization\(\)](#), [hot_retain_sat\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), [update_centralgal\(\)](#), [update_hot_frac\(\)](#), and [update_type_10\(\)](#).

3.1.2.51 int GALAXY::NextProgGal

Definition at line [208](#) of file [allvars.h](#).

Referenced by [deal_with_galaxy_merger\(\)](#), [disrupt\(\)](#), [init_galaxy\(\)](#), [main\(\)](#), [prepare_galaxy_for_output\(\)](#), [save_galaxy_tree\(\)](#), and [walk\(\)](#).

3.1.2.52 float GALAXY::ObsICL[NMAG][NOUT]

Definition at line [304](#) of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [disrupt\(\)](#), [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), and [update_ICL\(\)](#).

3.1.2.53 float GALAXY::ObsLum[NMAG][NOUT]

Definition at line 298 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [add_to_luminosities\(\)](#), [check_disk_instability\(\)](#), [disrupt\(\)](#), [init_galaxy\(\)](#), [make_bulge_from_burst\(\)](#), [prepare_galaxy_for_output\(\)](#), and [tsudy\(\)](#).

3.1.2.54 float GALAXY::ObsLumBulge[NMAG][NOUT]

Definition at line 300 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [check_disk_instability\(\)](#), [init_galaxy\(\)](#), [make_bulge_from_burst\(\)](#), [prepare_galaxy_for_momaf\(\)](#), [prepare_galaxy_for_output\(\)](#), and [tsudy\(\)](#).

3.1.2.55 float GALAXY::ObsLumDust[NMAG][NOUT]

Definition at line 302 of file [allvars.h](#).

Referenced by [init_galaxy\(\)](#), [prepare_galaxy_for_momaf\(\)](#), [prepare_galaxy_for_output\(\)](#), and [tsudy\(\)](#).

3.1.2.56 float GALAXY::ObsYLum[NMAG][NOUT]

Definition at line 299 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [add_to_luminosities\(\)](#), [check_disk_instability\(\)](#), [init_galaxy\(\)](#), [make_bulge_from_burst\(\)](#), and [tsudy\(\)](#).

3.1.2.57 float GALAXY::ObsYLumBulge[NMAG][NOUT]

Definition at line 301 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [check_disk_instability\(\)](#), [init_galaxy\(\)](#), [make_bulge_from_burst\(\)](#), and [tsudy\(\)](#).

3.1.2.58 float GALAXY::OriMergTime

Definition at line 277 of file [allvars.h](#).

Referenced by [init_galaxy\(\)](#), [prepare_galaxy_for_momaf\(\)](#), [prepare_galaxy_for_output\(\)](#), [update_type_1\(\)](#), and [update_type_2\(\)](#).

3.1.2.59 float GALAXY::Pos[3]

Definition at line 222 of file [allvars.h](#).

Referenced by [collisional_starburst_recipe\(\)](#), [do_major_merger_starburst\(\)](#), [estimate_merging_time\(\)](#), [hot_retain_sat\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), [main\(\)](#), [peri_radius\(\)](#), [prepare_galaxy_for_momaf\(\)](#), [prepare_galaxy_for_output\(\)](#), [reincorporate_gas\(\)](#), [starformation_and_feedback\(\)](#), [update_from_feedback\(\)](#), and [update_hotgas\(\)](#).

3.1.2.60 float GALAXY::Rvir

Definition at line 230 of file [allvars.h](#).

Referenced by [collisional_starburst_recipe\(\)](#), [cooling_recipe\(\)](#), [disrupt\(\)](#), [do_AGN_heating\(\)](#), [do_major_merger_starburst\(\)](#), [get_disk_radius\(\)](#), [get_initial_disk_radius\(\)](#), [hot_retain_sat\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), [reincorporate_gas\(\)](#), [starformation_and_feedback\(\)](#), [update_centralgal\(\)](#), [update_from_feedback\(\)](#), [update_hot_frac\(\)](#), and [update_hotgas\(\)](#).

3.1.2.61 float GALAXY::Sfr

Definition at line 255 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [collisional_starburst_recipe\(\)](#), [do_major_merger_starburst\(\)](#), [init_galaxy\(\)](#), [main\(\)](#), [make_bulge_from_burst\(\)](#), [prepare_galaxy_for_output\(\)](#), and [starformation_and_feedback\(\)](#).

3.1.2.62 float GALAXY::SfrBulge

Definition at line 256 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [init_galaxy\(\)](#), [main\(\)](#), [make_bulge_from_burst\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.1.2.63 int GALAXY::SnapNum

Definition at line 218 of file [allvars.h](#).

Referenced by [collisional_starburst_recipe\(\)](#), [do_major_merger_starburst\(\)](#), [init_galaxy\(\)](#), [main\(\)](#), [peri_radius\(\)](#), [prepare_galaxy_for_momaf\(\)](#), [prepare_galaxy_for_output\(\)](#), [starformation_and_feedback\(\)](#), [tsudy\(\)](#), [update_type_1\(\)](#), and [update_type_2\(\)](#).

3.1.2.64 float GALAXY::StarMerge

Definition at line 266 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [collisional_starburst_recipe\(\)](#), and [init_galaxy\(\)](#).

3.1.2.65 float GALAXY::StellarDiskRadius

Definition at line 269 of file [allvars.h](#).

Referenced by [bulgesize_from_merger\(\)](#), [check_disk_instability\(\)](#), [deal_with_galaxy_merger\(\)](#), [get_stellar_disk_radius\(\)](#), [init_galaxy\(\)](#), [main\(\)](#), [prepare_galaxy_for_output\(\)](#), and [sat_radius\(\)](#).

3.1.2.66 float GALAXY::StellarMass

Definition at line 241 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [bulgesize_from_merger\(\)](#), [check_disk_instability\(\)](#), [deal_with_galaxy_merger\(\)](#), [disrupt\(\)](#), [do_major_merger_starburst\(\)](#), [estimate_merging_time\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), [make_bulge_from_burst\(\)](#), [prepare_galaxy_for_output\(\)](#), [sat_radius\(\)](#), [starformation_and_feedback\(\)](#), [update_from_star_formation\(\)](#), and [update_type_1\(\)](#).

3.1.2.67 float GALAXY::StellarSpin[3]

Definition at line 227 of file [allvars.h](#).

Referenced by [get_stellar_disk_radius\(\)](#), [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), and [update_from_star_formation\(\)](#).

3.1.2.68 int GALAXY::TreeRoot

Definition at line 212 of file [allvars.h](#).

Referenced by [prepare_galaxy_for_output\(\)](#), [save_galaxy_tree\(\)](#), and [walk\(\)](#).

3.1.2.69 int GALAXY::Type

Definition at line 215 of file [allvars.h](#).

Referenced by [check_disk_instability\(\)](#), [collisional_starburst_recipe\(\)](#), [deal_with_galaxy_merger\(\)](#), [disrupt\(\)](#), [estimate_merging_time\(\)](#), [get_gas_disk_radius\(\)](#), [get_stellar_disk_radius\(\)](#), [hot_retain_sat\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), [main\(\)](#), [prepare_galaxy_for_momaf\(\)](#), [prepare_galaxy_for_output\(\)](#), [reincorporate_gas\(\)](#), [set_merger_center\(\)](#), [starformation_and_feedback\(\)](#), [update_centralgal\(\)](#), [update_from_feedback\(\)](#), [update_hotgas\(\)](#), [update_type_1\(\)](#), and [update_type_2\(\)](#).

3.1.2.70 float GALAXY::Vel[3]

Definition at line 224 of file [allvars.h](#).

Referenced by [init_galaxy\(\)](#), [main\(\)](#), [peri_radius\(\)](#), [prepare_galaxy_for_momaf\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.1.2.71 float GALAXY::Vmax

Definition at line 232 of file [allvars.h](#).

Referenced by [check_disk_instability\(\)](#), [collisional_starburst_recipe\(\)](#), [get_gas_disk_radius\(\)](#), [get_stellar_disk_radius\(\)](#), [init_galaxy\(\)](#), [main\(\)](#), [prepare_galaxy_for_output\(\)](#), and [starformation_and_feedback\(\)](#).

3.1.2.72 float GALAXY::Vvir

Definition at line 231 of file [allvars.h](#).

Referenced by [collisional_starburst_recipe\(\)](#), [cooling_recipe\(\)](#), [do_AGN_heating\(\)](#), [get_disk_radius\(\)](#), [get_initial_disk_radius\(\)](#), [grow_black_hole\(\)](#), [init_galaxy\(\)](#), [peri_radius\(\)](#), [prepare_galaxy_for_output\(\)](#), [reincorporate_gas\(\)](#), [starformation_and_feedback\(\)](#), and [update_centralgal\(\)](#).

3.1.2.73 float GALAXY::XrayLum

Definition at line 267 of file [allvars.h](#).

Referenced by [cooling_recipe\(\)](#), [init_galaxy\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.1.2.74 float GALAXY::YLum[NMAG][NOUT]

Definition at line 288 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [add_to_luminosities\(\)](#), [check_disk_instability\(\)](#), [init_galaxy\(\)](#), [make_bulge_from_burst\(\)](#), and [tsudy\(\)](#).

3.1.2.75 float GALAXY::YLumBulge[NMAG][NOUT]

Definition at line 290 of file [allvars.h](#).

Referenced by [add_galaxies_together\(\)](#), [check_disk_instability\(\)](#), [init_galaxy\(\)](#), [make_bulge_from_burst\(\)](#), and [tsudy\(\)](#).

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

- [code/allvars.h](#)

3.2 GALAXY_OUTPUT Struct Reference

Galaxy structure for output.

```
#include <allvars.h>
```

Data Fields

- long long [GalID](#)
- long long [HaloID](#)

ID of galaxy, unique within simulation and SAM run.
- long long [FirstProgGal](#)
- long long [NextProgGal](#)
- long long [LastProgGal](#)
- long long [FOFCentralGal](#)
- long long [FileTreeNr](#)
- long long [DescendantGal](#)
- long long [MainLeafId](#)
- long long [TreeRootId](#)
- long long [SubID](#)
- long long [MMSubID](#)
- int [PeanoKey](#)
- float [Redshift](#)
- int [Type](#)
- int [HaloIndex](#)
- int [SnapNum](#)
- float [CentralMvir](#)
- float [Pos](#) [3]
- float [Vel](#) [3]
- int [Len](#)
- float [Mvir](#)
- float [Rvir](#)

- float [Vvir](#)
- float [Vmax](#)
- float [GasSpin](#) [3]
- float [StellarSpin](#) [3]
- float [InfallVmax](#)
- int [InfallSnap](#)
- float [HotRadius](#)
- float [OriMergTime](#)
- float [MergTime](#)
- float [ColdGas](#)
- float [StellarMass](#)
- float [BulgeMass](#)
- float [HotGas](#)
- float [EjectedMass](#)
- float [BlackHoleMass](#)
- float [ICM](#)
- float [MetalsColdGas](#)
- float [MetalsStellarMass](#)
- float [MetalsBulgeMass](#)
- float [MetalsHotGas](#)
- float [MetalsEjectedMass](#)
- float [MetalsICM](#)
- float [Sfr](#) [MAXSNAPS]
- float [SfrBulge](#) [MAXSNAPS]
- float [XrayLum](#)
- float [BulgeSize](#)
- float [StellarDiskRadius](#)
- float [GasDiskRadius](#)
- int [DisruptOn](#)
- int [MergeOn](#)
- float [CoolingRadius](#)
- float [Mag](#) [NMAG]
- float [MagBulge](#) [NMAG]
- float [MagDust](#) [NMAG]
- float [MassWeightAge](#)
- float [MagICL](#) [NMAG]
- float [ObsMag](#) [NMAG]
- float [ObsMagBulge](#) [NMAG]
- float [ObsMagDust](#) [NMAG]
- float [dObsMag](#) [NMAG]
- float [dObsMagBulge](#) [NMAG]
- float [dObsMagDust](#) [NMAG]
- float [ObsMagICL](#) [NMAG]
- float [dObsMagICL](#) [NMAG]

3.2.1 Detailed Description

Definition at line 80 of file [allvars.h](#).

3.2.2 Field Documentation

3.2.2.1 float GALAXY_OUTPUT::BlackHoleMass

Definition at line 127 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.2 float GALAXY_OUTPUT::BulgeMass

Definition at line 124 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.3 float GALAXY_OUTPUT::BulgeSize

Definition at line 145 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.4 float GALAXY_OUTPUT::CentralMvir

Definition at line 104 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.5 float GALAXY_OUTPUT::ColdGas

Definition at line 122 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.6 float GALAXY_OUTPUT::CoolingRadius

Definition at line 152 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.7 long long GALAXY_OUTPUT::DescendantGal

Definition at line 91 of file [allvars.h](#).

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

3.2.2.8 int GALAXY_OUTPUT::DisruptOn

Definition at line 148 of file [allvars.h](#).

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

3.2.2.9 float GALAXY_OUTPUT::dObsMag[NMAG]

Definition at line 173 of file [allvars.h](#).

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

3.2.2.10 float GALAXY_OUTPUT::dObsMagBulge[NMAG]

Definition at line 174 of file [allvars.h](#).

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

3.2.2.11 float GALAXY_OUTPUT::dObsMagDust[NMAG]

Definition at line 175 of file [allvars.h](#).

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

3.2.2.12 float GALAXY_OUTPUT::dObsMagICL[NMAG]

Definition at line 180 of file [allvars.h](#).

3.2.2.13 float GALAXY_OUTPUT::EjectedMass

Definition at line 126 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.14 long long GALAXY_OUTPUT::FileTreeNr

Definition at line 90 of file [allvars.h](#).

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

3.2.2.15 long long GALAXY_OUTPUT::FirstProgGal

Definition at line 86 of file [allvars.h](#).

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

3.2.2.16 long long GALAXY_OUTPUT::FOFCentralGal

Definition at line 89 of file [allvars.h](#).

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

3.2.2.17 long long GALAXY_OUTPUT::GalID

Definition at line 84 of file [allvars.h](#).

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

3.2.2.18 float GALAXY_OUTPUT::GasDiskRadius

Definition at line 147 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.19 float GALAXY_OUTPUT::GasSpin[3]

Definition at line 113 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.20 long long GALAXY_OUTPUT::HaloID

Definition at line 85 of file [allvars.h](#).

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

3.2.2.21 int GALAXY_OUTPUT::HaloIndex

Definition at line 101 of file [allvars.h](#).

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

3.2.2.22 float GALAXY_OUTPUT::HotGas

Definition at line 125 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.23 float GALAXY_OUTPUT::HotRadius

Definition at line 117 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.24 float GALAXY_OUTPUT::ICM

Definition at line 129 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.25 int GALAXY_OUTPUT::InfallSnap

Definition at line 116 of file [allvars.h](#).

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

3.2.2.26 float GALAXY_OUTPUT::InfallVmax

Definition at line 115 of file [allvars.h](#).

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

3.2.2.27 long long GALAXY_OUTPUT::LastProgGal

Definition at line 88 of file [allvars.h](#).

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

3.2.2.28 int GALAXY_OUTPUT::Len

Definition at line 108 of file [allvars.h](#).

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

3.2.2.29 float GALAXY_OUTPUT::Mag[NMAG]

Definition at line 157 of file [allvars.h](#).

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

3.2.2.30 float GALAXY_OUTPUT::MagBulge[NMAG]

Definition at line 158 of file [allvars.h](#).

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

3.2.2.31 float GALAXY_OUTPUT::MagDust[NMAG]

Definition at line 159 of file [allvars.h](#).

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

3.2.2.32 float GALAXY_OUTPUT::MagICL[NMAG]

Definition at line 162 of file [allvars.h](#).

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

3.2.2.33 long long GALAXY_OUTPUT::MainLeafId

Definition at line 92 of file [allvars.h](#).

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

3.2.2.34 float GALAXY_OUTPUT::MassWeightAge

Definition at line 160 of file [allvars.h](#).

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

3.2.2.35 int GALAXY_OUTPUT::MergeOn

Definition at line 150 of file [allvars.h](#).

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

3.2.2.36 float GALAXY_OUTPUT::MergTime

Definition at line 120 of file [allvars.h](#).

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

3.2.2.37 float GALAXY_OUTPUT::MetalsBulgeMass

Definition at line 132 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.38 float GALAXY_OUTPUT::MetalsColdGas

Definition at line 130 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.39 float GALAXY_OUTPUT::MetalsEjectedMass

Definition at line 134 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.40 float GALAXY_OUTPUT::MetalsHotGas

Definition at line 133 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.41 float GALAXY_OUTPUT::MetalsICM

Definition at line 135 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.42 float GALAXY_OUTPUT::MetalsStellarMass

Definition at line 131 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.43 long long GALAXY_OUTPUT::MMSubID

Definition at line 95 of file [allvars.h](#).

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

3.2.2.44 float GALAXY_OUTPUT::Mvir

Definition at line 109 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.45 long long GALAXY_OUTPUT::NextProgGal

Definition at line 87 of file [allvars.h](#).

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

3.2.2.46 float GALAXY_OUTPUT::ObsMag[NMAG]

Definition at line 167 of file [allvars.h](#).

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

3.2.2.47 float GALAXY_OUTPUT::ObsMagBulge[NMAG]

Definition at line 168 of file [allvars.h](#).

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

3.2.2.48 float GALAXY_OUTPUT::ObsMagDust[NMAG]

Definition at line 169 of file [allvars.h](#).

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

3.2.2.49 float GALAXY_OUTPUT::ObsMagICL[NMAG]

Definition at line 178 of file [allvars.h](#).

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

3.2.2.50 float GALAXY_OUTPUT::OriMergTime

Definition at line 119 of file [allvars.h](#).

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

3.2.2.51 int GALAXY_OUTPUT::PeanoKey

Definition at line 96 of file [allvars.h](#).

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

3.2.2.52 float GALAXY_OUTPUT::Pos[3]

Definition at line 106 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.53 float GALAXY_OUTPUT::Redshift

Definition at line 97 of file [allvars.h](#).

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

3.2.2.54 float GALAXY_OUTPUT::Rvir

Definition at line 110 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.55 float GALAXY_OUTPUT::Sfr

Definition at line 138 of file [allvars.h](#).

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

3.2.2.56 float GALAXY_OUTPUT::SfrBulge

Definition at line 139 of file [allvars.h](#).

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

3.2.2.57 int GALAXY_OUTPUT::SnapNum

Definition at line 103 of file [allvars.h](#).

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

3.2.2.58 float GALAXY_OUTPUT::StellarDiskRadius

Definition at line 146 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.59 float GALAXY_OUTPUT::StellarMass

Definition at line 123 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.60 float GALAXY_OUTPUT::StellarSpin[3]

Definition at line 114 of file [allvars.h](#).

Referenced by [fix_units_for_output\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.2.2.61 long long GALAXY_OUTPUT::SubID

Definition at line 94 of file [allvars.h](#).

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

3.2.2.62 long long GALAXY_OUTPUT::TreeRootId

Definition at line 93 of file [allvars.h](#).

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

3.2.2.63 int GALAXY_OUTPUT::Type

Definition at line 99 of file [allvars.h](#).

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

3.2.2.64 float GALAXY_OUTPUT::Vel[3]

Definition at line 107 of file [allvars.h](#).

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

3.2.2.65 float GALAXY_OUTPUT::Vmax

Definition at line 112 of file [allvars.h](#).

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

3.2.2.66 float GALAXY_OUTPUT::Vvir

Definition at line 111 of file [allvars.h](#).

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

3.2.2.67 float GALAXY_OUTPUT::XrayLum

Definition at line 144 of file [allvars.h](#).

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

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

- code/allvars.h

3.3 halo_aux_data Struct Reference

```
#include <allvars.h>
```

Data Fields

- int DoneFlag
- int HaloFlag
- int NGalaxies
- int FirstGalaxy

3.3.1 Detailed Description

Definition at line 371 of file [allvars.h](#).

3.3.2 Field Documentation

3.3.2.1 int halo_aux_data::DoneFlag

Definition at line 373 of file [allvars.h](#).

Referenced by [load_tree\(\)](#), and [main\(\)](#).

3.3.2.2 int halo_aux_data::FirstGalaxy

Definition at line 376 of file [allvars.h](#).

Referenced by [main\(\)](#), [prepare_galaxy_for_output\(\)](#), and [set_merger_center\(\)](#).

3.3.2.3 int halo_aux_data::HaloFlag

Definition at line 374 of file [allvars.h](#).

Referenced by [load_tree\(\)](#), and [main\(\)](#).

3.3.2.4 int halo_aux_data::NGalaxies

Definition at line 375 of file [allvars.h](#).

Referenced by [load_tree\(\)](#), [main\(\)](#), [prepare_galaxy_for_output\(\)](#), and [set_merger_center\(\)](#).

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

- code/[allvars.h](#)

3.4 halo_data Struct Reference

```
#include <allvars.h>
```

Data Fields

- int Descendant
- int FirstProgenitor
- int NextProgenitor
- int FirstHaloInFOFgroup
- int NextHaloInFOFgroup
- int Len
- float [M_Mean200](#)
- float [M_Crit200](#)
- float [M_TopHat](#)
- float [Pos](#) [3]
- float [Vel](#) [3]
- float [VelDisp](#)
- float [Vmax](#)
- float [Spin](#) [3]
- long long [MostBoundID](#)

- int `SnapNum`
- int `FileNr`
- int `SubhaloIndex`
- float `SubHalfMass`

3.4.1 Detailed Description

Definition at line 321 of file `allvars.h`.

3.4.2 Field Documentation

3.4.2.1 int halo_data::Descendant

Definition at line 324 of file `allvars.h`.

Referenced by `estimate_merging_time()`, and `update_type_1()`.

3.4.2.2 int halo_data::FileNr

Definition at line 342 of file `allvars.h`.

Referenced by `prepare_galaxy_for_output()`.

3.4.2.3 int halo_data::FirstHaloInFOFgroup

Definition at line 327 of file `allvars.h`.

Referenced by `main()`, `prepare_galaxy_for_output()`, and `update_type_1()`.

3.4.2.4 int halo_data::FirstProgenitor

Definition at line 325 of file `allvars.h`.

Referenced by `estimate_merging_time()`, `main()`, and `set_merger_center()`.

3.4.2.5 int halo_data::Len

Definition at line 331 of file `allvars.h`.

Referenced by `get_virial_mass()`, `init_galaxy()`, `main()`, `prepare_galaxy_for_output()`, and `set_merger_center()`.

3.4.2.6 float halo_data::M_Crit200

Definition at line 332 of file `allvars.h`.

Referenced by `get_virial_mass()`.

3.4.2.7 float halo_data::M_Mean200

Definition at line 332 of file `allvars.h`.

3.4.2.8 float halo_data::M_TopHat

Definition at line 332 of file [allvars.h](#).

3.4.2.9 long long halo_data::MostBoundID

Definition at line 338 of file [allvars.h](#).

Referenced by [init_galaxy\(\)](#), and [main\(\)](#).

3.4.2.10 int halo_data::NextHaloInFOFgroup

Definition at line 328 of file [allvars.h](#).

Referenced by [main\(\)](#), [prepare_galaxy_for_output\(\)](#), and [set_merger_center\(\)](#).

3.4.2.11 int halo_data::NextProgenitor

Definition at line 326 of file [allvars.h](#).

Referenced by [main\(\)](#), and [set_merger_center\(\)](#).

3.4.2.12 float halo_data::Pos[3]

Definition at line 333 of file [allvars.h](#).

Referenced by [init_galaxy\(\)](#), and [main\(\)](#).

3.4.2.13 int halo_data::SnapNum

Definition at line 341 of file [allvars.h](#).

Referenced by [collisional_starburst_recipe\(\)](#), [do_major_merger_starburst\(\)](#), [estimate_merging_time\(\)](#), [get_virial_radius\(\)](#), [hot_retain_sat\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), [main\(\)](#), [reincorporate_gas\(\)](#), [starformation_and_feedback\(\)](#), [update_centralgal\(\)](#), [update_from_feedback\(\)](#), and [update_hotgas\(\)](#).

3.4.2.14 float halo_data::Spin[3]

Definition at line 337 of file [allvars.h](#).

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

3.4.2.15 float halo_data::SubHalfMass

Definition at line 344 of file [allvars.h](#).

3.4.2.16 int halo_data::SubhaloIndex

Definition at line 343 of file [allvars.h](#).

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

3.4.2.17 float halo_data::Vel[3]

Definition at line 334 of file [allvars.h](#).

Referenced by [init_galaxy\(\)](#), and [main\(\)](#).

3.4.2.18 float halo_data::VelDisp

Definition at line 335 of file [allvars.h](#).

3.4.2.19 float halo_data::Vmax

Definition at line 336 of file [allvars.h](#).

Referenced by [init_galaxy\(\)](#), [main\(\)](#), and [update_centralgal\(\)](#).

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

- code/[allvars.h](#)

3.5 halo_ids_data Struct Reference

```
#include <allvars.h>
```

Data Fields

- long long [HaloID](#)
- long long [FileTreeNr](#)
- long long [FirstProgenitor](#)
- long long [LastProgenitor](#)
- long long [NextProgenitor](#)
- long long [Descendant](#)
- long long [FirstHaloInFOFgroup](#)
- long long [NextHaloInFOFgroup](#)
- long long [MainLeafID](#)
- double [Redshift](#)
- int [PeanoKey](#)
- int [dummy](#)

3.5.1 Detailed Description

Definition at line 350 of file [allvars.h](#).

3.5.2 Field Documentation

3.5.2.1 long long halo_ids_data::Descendant

Definition at line 357 of file [allvars.h](#).

3.5.2.2 int halo_ids_data::dummy

Definition at line 365 of file [allvars.h](#).

3.5.2.3 long long halo_ids_data::FileTreeNr

Definition at line 353 of file [allvars.h](#).

3.5.2.4 long long halo_ids_data::FirstHaloInFOFgroup

Definition at line 358 of file [allvars.h](#).

3.5.2.5 long long halo_ids_data::FirstProgenitor

Definition at line 354 of file [allvars.h](#).

3.5.2.6 long long halo_ids_data::HaloID

Definition at line 352 of file [allvars.h](#).

Referenced by [prepare_galaxy_for_momaf\(\)](#), and [prepare_galaxy_for_output\(\)](#).

3.5.2.7 long long halo_ids_data::LastProgenitor

Definition at line 355 of file [allvars.h](#).

3.5.2.8 long long halo_ids_data::MainLeafID

Definition at line 361 of file [allvars.h](#).

3.5.2.9 long long halo_ids_data::NextHaloInFOFgroup

Definition at line 359 of file [allvars.h](#).

3.5.2.10 long long halo_ids_data::NextProgenitor

Definition at line 356 of file [allvars.h](#).

3.5.2.11 int halo_ids_data::PeanoKey

Definition at line 364 of file [allvars.h](#).

3.5.2.12 double halo_ids_data::Redshift

Definition at line 363 of file [allvars.h](#).

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

- code/allvars.h

3.6 MOMAF_INPUTS Struct Reference

```
#include <allvars.h>
```

Data Fields

- long long **GalID**
- long long **HaloID**
- int **SnapNum**
- float **Pos** [3]
- float **Vel** [3]
- float **ObsMagBulge** [NMAG]
- float **ObsMagDust** [NMAG]
- float **dObsMagBulge** [NMAG]
- float **dObsMagDust** [NMAG]

3.6.1 Detailed Description

Definition at line 188 of file [allvars.h](#).

3.6.2 Field Documentation

3.6.2.1 float MOMAF_INPUTS::dObsMagBulge[NMAG]

Definition at line 197 of file [allvars.h](#).

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

3.6.2.2 float MOMAF_INPUTS::dObsMagDust[NMAG]

Definition at line 198 of file [allvars.h](#).

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

3.6.2.3 long long MOMAF_INPUTS::GalID

Definition at line 190 of file [allvars.h](#).

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

3.6.2.4 long long MOMAF_INPUTS::HaloID

Definition at line 191 of file [allvars.h](#).

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

3.6.2.5 float MOMAF_INPUTS::ObsMagBulge[NMAG]

Definition at line 195 of file [allvars.h](#).

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

3.6.2.6 float MOMAF_INPUTS::ObsMagDust[NMAG]

Definition at line 196 of file [allvars.h](#).

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

3.6.2.7 float MOMAF_INPUTS::Pos[3]

Definition at line 193 of file [allvars.h](#).

Referenced by [prepare_galaxy_for_momaf\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

3.6.2.8 int MOMAF_INPUTS::SnapNum

Definition at line 192 of file [allvars.h](#).

Referenced by [prepare_galaxy_for_momaf\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

3.6.2.9 float MOMAF_INPUTS::Vel[3]

Definition at line 194 of file [allvars.h](#).

Referenced by [prepare_galaxy_for_momaf\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

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

- [code/allvars.h](#)

Chapter 4

File Documentation

4.1 code/age.c File Reference

Returns time to present for a given redshift.

Functions

- double `time_to_present` (double z)
- double `integrand_time_to_present` (double a, void *param)

4.1.1 Detailed Description

For a given redshift, returns the time from that redshift until redshift zero: $H_0 t_0 = \int_0^z \frac{dz}{(1+z)\sqrt{(1+z)^2(1+z\Omega_m)-z(2+x)\Omega_\Lambda}}$ Returns Age in Myr

Definition in file `age.c`.

4.1.2 Function Documentation

4.1.2.1 double `integrand_time_to_present` (double a, void * param)

Definition at line 42 of file `age.c`.

References `Omega`, and `OmegaLambda`.

4.1.2.2 double `time_to_present` (double z)

Definition at line 19 of file `age.c`.

References `Hubble`.

Referenced by `init()`.

4.2 code/age.c

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <string.h>
00004 #include <math.h>
00005 #include <gsl/gsl_math.h>
00006 #include <gsl/gsl_integration.h>
00007
00008 #include "allvars.h"
00009 #include "proto.h"
00010
00011 /**@file age.c Returns time to present for a given redshift.
00012 *
00013 * @brief For a given redshift, returns the time from that redshift until
00014 * redshift zero:
00015 * \f$H_0t_0=\int_0^z\frac{dz}{(1+z)\sqrt{(1+z)^2(1+\Omega_m)-z(2+\Lambda)bda}}\f$
00016 *
00017 * Returns Age in Myr*/
00018
00019 double time_to_present(double z)
00020 {
00021 #define WORKSIZE 1000
00022     gsl_function F;
00023     gsl_integration_workspace *workspace;
00024     double time, result, abserr;
00025
00026     workspace = gsl_integration_workspace_alloc(WORKSIZE);
00027     F.function = &integrand_time_to_present;
00028
00029     gsl_integration_qag(&F, 1.0 / (z + 1), 1.0, 1.0 / Hubble,
00030                         1.0e-8, WORKSIZE, GSL_INTEG_GAUSS21, workspace, &result, &a
00031     bserr);
00032     time = 1 / Hubble * result;
00033
00034     gsl_integration_workspace_free(workspace);
00035
00036
00037
00038     return time;
00039 }
00040
00041
00042 double integrand_time_to_present(double a, void *param)
00043 {
00044     return 1 / sqrt(Omega / a + (1 - Omega - OmegaLambda) + OmegaLambda * a * a);
00045 }
```

4.3 code/allvars.c File Reference

Variables

- struct **GALAXY** * Gal
- struct **GALAXY** * HaloGal
- struct **halo_data** * Halo
- struct **halo_aux_data** * HaloAux
- struct **halo_ids_data** * HaloIDs
- int FirstFile
- int LastFile

- int MaxGals
- int FoF_MaxGals
- int Ntrees
- int NumGals
- char PhotDir [512]
- char PhotPrefix [50]
- char CoolFunctionsDir [512]
- char OutputDir [512]
- char FileNameGalaxies [512]
- char SimulationDir [512]
- char FileWithOutputSnaps [512]
- char FileWithSnapList [512]
- char FileNrDir [512]
- int ListInputFilrNr [111]
- int TotHalos
- int TotGalaxies [NOUT]
- int * TreeNgals [NOUT]
- int FilesPerSnapshot
- int LastSnapShotNr
- int * FirstHaloInSnap
- int * TreeNHalos
- int * TreeFirstHalo
- int ThisTask
- int NTask
- int GalCount
- int TotGalCount
- double Omega
- double OmegaLambda
- double Hubble_h
- double PartMass
- double EnergySNcode
- double EnergySN
- double EtaSNcode
- double EtaSN
- int NumMergers
- int StarBurstsInMajorMergersOn
- int BulgeFormationInMinorMergersOn
- int MetallicityOption
- int EjectionOn
- int CoolingCutoff
- int ReionizationOn
- int StarFormationRecipe
- int SatelliteRecipe
- int StarBurstRecipe
- int FeedbackRecipe
- int EjectionRecipe
- int DiskRadiusMethod
- int TrackDiskInstability
- int AGNrecipeOn
- int ReIncorporationRecipe

- double `SfrLawPivotVelocity`
- double `SfrLawSlope`
- double `SfrAlpha`
- double `ReheatPreVelocity`
- double `ReheatSlope`
- double `EjectPreVelocity`
- double `EjectSlope`
- double `SNinReheat`
- double `FeedbackEpsilon`
- double `RecycleFraction`
- double `Yield`
- double `FracZtoHot`
- double `ReIncorporationFactor`
- double `CoolingVelocityCutOff`
- double `ThreshMajorMerger`
- double `BaryonFrac`
- double `SfrEfficiency`
- double `FeedbackReheatingEpsilon`
- double `FeedbackEjectionEfficiency`
- double `AgnEfficiency`
- double `BlackHoleGrowthRate`
- double `Reionization_z0`
- double `Reionization_zr`
- double `UnitLength_in_cm`
- double `UnitTime_in_s`
- double `UnitVelocity_in_cm_per_s`
- double `UnitMass_in_g`
- double `RhoCrit`
- double `UnitPressure_in_cgs`
- double `UnitDensity_in_cgs`
- double `UnitCoolingRate_in_cgs`
- double `UnitEnergy_in_cgs`
- double `UnitTime_in_Megayears`
- double `G`
- double `Hubble`
- double `a0`
- double `ar`
- int `ListOutputSamps` [NOUT]
- double `ZZ` [MAXSNAPS]
- double `AA` [MAXSNAPS]
- double `Age` [MAXSNAPS]
- int `Snaplistlen`
- `gsl_rng * random_generator`
- float `AgeTab` [ZL_LEN]
- float `RedshiftTab` [IZ]
- float `MagTableZz` [NMAG][IZZ][IZ][ZL_LEN]
- float `DeltaM` [NMAG+1][NZZ_EXT][IZ]
- float `Corrections` [NMAG][IZ]
- long `mu_seed`
- int `NtotHalos`

- int TotIds
- int Nids
- int TotSnaps
- int OffsetIDs
- int * CountIDs_halo
- int * OffsetIDs_halo
- int * CountIDs_snaptree
- int * OffsetIDs_snaptree
- long long * IdList
- float * PosList
- float * VelList
- int Hashbits
- double BoxSize
- char * ptr_auxdata
- char * ptr_treedata
- char * ptr_dbids
- char * ptr_galaxydata
- char * ptr_galsnapdata [NOUT]
- size_t offset_auxdata
- size_t offset_treedata
- size_t offset_dbids
- size_t offset_galaxydata
- size_t maxstorage_galaxydata
- size_t filled_galaxydata
- size_t offset_galsnapdata [NOUT]
- size_t maxstorage_galsnapdata [NOUT]
- size_t filled_galsnapdata [NOUT]
- char * ptr_momafdata [NOUT]
- size_t offset_momafdata [NOUT]
- size_t maxstorage_momafdata [NOUT]
- size_t filled_momafdata [NOUT]
- float RecGas
- float Frac [ZL_LEN]
- float metalcold
- float Rho [RHO_LEN]
- float H2 [RHO_LEN][Z_LEN]
- float Reion_z [46]
- float Reion_Mc [46]
- FILE * tree_file
- FILE * treeaux_file
- FILE * treedbids_file
- FILE * output_file
- int * GalaxiesInOrder

4.3.1 Variable Documentation

4.3.1.1 double a0

Definition at line 118 of file [allvars.c](#).

Referenced by [do_reionization\(\)](#), and [init\(\)](#).

4.3.1.2 double AA[MAXSNAPS]

Definition at line 130 of file [allvars.c](#).

Referenced by [get_coordinates\(\)](#), [init\(\)](#), and [read_snap_list\(\)](#).

4.3.1.3 double Age[MAXSNAPS]

Definition at line 131 of file [allvars.c](#).

Referenced by [init\(\)](#), and [NumToTime\(\)](#).

4.3.1.4 float AgeTab[ZL_LEN]

Definition at line 144 of file [allvars.c](#).

Referenced by [find_interpolated_lum\(\)](#), [find_interpolated_obs_lum\(\)](#), [find_interpolation_point\(\)](#), [get_jump_index\(\)](#), [init_jump_index\(\)](#), and [setup_spectrophotometric_model\(\)](#).

4.3.1.5 double AgnEfficiency

Definition at line 112 of file [allvars.c](#).

Referenced by [do_AGN_heating\(\)](#), [init\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.6 int AGNrecipeOn

Definition at line 89 of file [allvars.c](#).

Referenced by [cooling_recipe\(\)](#), [deal_with_galaxy_merger\(\)](#), [do_AGN_heating\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.7 double ar

Definition at line 118 of file [allvars.c](#).

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

4.3.1.8 double BaryonFrac

Definition at line 108 of file [allvars.c](#).

Referenced by [infall_recipe\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.9 double BlackHoleGrowthRate

Definition at line 113 of file [allvars.c](#).

Referenced by [grow_black_hole\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.10 double BoxSize

Definition at line 163 of file [allvars.c](#).

Referenced by [prepare_galaxy_for_output\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.11 int BulgeFormationInMinorMergersOn

Definition at line 77 of file [allvars.c](#).

Referenced by [add_galaxies_together\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.12 char CoolFunctionsDir[512]

Definition at line 30 of file [allvars.c](#).

Referenced by [read_cooling_functions\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.13 int CoolingCutoff

Definition at line 80 of file [allvars.c](#).

Referenced by [cooling_recipe\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.14 double CoolingVelocityCutOff

Definition at line 106 of file [allvars.c](#).

Referenced by [cooling_recipe\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.15 float Corrections[NMAG][IZ]

Definition at line 150 of file [allvars.c](#).

Referenced by [read_tsud_tables\(\)](#), and [tsudy\(\)](#).

4.3.1.16 int* CountIDs_halo

Definition at line 156 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [get_coordinates\(\)](#), and [load_tree_table\(\)](#).

4.3.1.17 int * CountIDs_snaptree

Definition at line 156 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [load_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.18 float DeltaM[NMAG+1][NZZ_EXT][IZ]

Definition at line 149 of file [allvars.c](#).

Referenced by [read_tsud_tables\(\)](#), and [tsudy\(\)](#).

4.3.1.19 int DiskRadiusMethod

Definition at line 87 of file [allvars.c](#).

Referenced by [cool_gas_onto_galaxy\(\)](#), [deal_with_galaxy_merger\(\)](#), [get_disk_radius\(\)](#), [get_initial_disk_radius\(\)](#), [main\(\)](#), [read_parameter_file\(\)](#), [starformation_and_feedback\(\)](#), and [update_from_star_formation\(\)](#).

4.3.1.20 int EjectionOn

Definition at line 79 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [do_major_merger_starburst\(\)](#), [read_parameter_file\(\)](#), [starformation_and_feedback\(\)](#), and [update_from_feedback\(\)](#).

4.3.1.21 int EjectionRecipe

Definition at line 86 of file [allvars.c](#).

Referenced by [infall_recipe\(\)](#), [read_parameter_file\(\)](#), and [update_from_feedback\(\)](#).

4.3.1.22 double EjectPreVelocity

Definition at line 98 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.3.1.23 double EjectSlope

Definition at line 99 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.3.1.24 double EnergySN

Definition at line 70 of file [allvars.c](#).

Referenced by [init\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.25 double EnergySNcode

Definition at line 70 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [init\(\)](#), and [starformation_and_feedback\(\)](#).

4.3.1.26 double EtaSN

Definition at line 71 of file [allvars.c](#).

Referenced by [init\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.27 double EtaSNcode

Definition at line 71 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [init\(\)](#), and [starformation_and_feedback\(\)](#).

4.3.1.28 double FeedbackEjectionEfficiency

Definition at line 111 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.3.1.29 double FeedbackEpsilon

Definition at line 101 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.3.1.30 int FeedbackRecipe

Definition at line 85 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [do_major_merger_starburst\(\)](#), [read_parameter_file\(\)](#), [starformation_and_feedback\(\)](#), and [update_from_feedback\(\)](#).

4.3.1.31 double FeedbackReheatingEpsilon

Definition at line 110 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.3.1.32 char FileNameGalaxies[512]

Definition at line 32 of file [allvars.c](#).

Referenced by [finalize_galaxy_file\(\)](#), [finalize_momaf_file\(\)](#), [load_tree_table\(\)](#), [main\(\)](#), [open_outputtree_file\(\)](#), [read_parameter_file\(\)](#), [save_galaxies\(\)](#), [save_galaxy_tree\(\)](#), [write_all_galaxy_data\(\)](#), [write_galaxy_data_snap\(\)](#), and [write_galaxy_for_momaf\(\)](#).

4.3.1.33 char FileNrDir[512]

Definition at line 38 of file [allvars.c](#).

Referenced by [read_file_nrs\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.34 int FilesPerSnapshot

Definition at line 46 of file [allvars.c](#).

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

4.3.1.35 char FileWithOutputSnaps[512]

Definition at line 34 of file [allvars.c](#).

Referenced by [read_output_snaps\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.36 char FileWithSnapList[512]

Definition at line 35 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), and [read_snap_list\(\)](#).

4.3.1.37 size_t filled_galaxydata

Definition at line 169 of file [allvars.c](#).

Referenced by [load_all_auxdata\(\)](#), [load_all_treedata\(\)](#), [myfwrite\(\)](#), and [write_all_galaxy_data\(\)](#).

4.3.1.38 size_t filled_galsnapdata[NOUT]

Definition at line 170 of file [allvars.c](#).

Referenced by [load_all_treedata\(\)](#), [myfwrite\(\)](#), and [write_galaxy_data_snap\(\)](#).

4.3.1.39 size_t filled_momafdata[NOUT]

Definition at line 173 of file [allvars.c](#).

Referenced by [load_all_treedata\(\)](#), [myfwrite\(\)](#), and [write_galaxy_for_momaf\(\)](#).

4.3.1.40 int FirstFile

Definition at line 20 of file [allvars.c](#).

Referenced by [main\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.41 int* FirstHaloInSnap

Definition at line 49 of file [allvars.c](#).

4.3.1.42 int FoF_MaxGals

Definition at line 24 of file [allvars.c](#).

Referenced by [load_tree\(\)](#), and [main\(\)](#).

4.3.1.43 float Frac[ZL_LEN]

Definition at line 182 of file [allvars.c](#).

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

4.3.1.44 double FracZtoHot

Definition at line 104 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.3.1.45 double G

Definition at line 118 of file [allvars.c](#).

Referenced by [check_disk_instability\(\)](#), [do_AGN_heating\(\)](#), [estimate_merging_time\(\)](#), [get_virial_radius\(\)](#), [get_virial_velocity\(\)](#), and [set_units\(\)](#).

4.3.1.46 struct GALAXY* Gal

Definition at line 5 of file [allvars.c](#).

Referenced by [add_galaxies_together\(\)](#), [add_infall_to_hot\(\)](#), [add_to_luminosities\(\)](#), [bulgesize_from_merger\(\)](#), [cal_H2\(\)](#), [check_disk_instability\(\)](#), [collisional_starburst_recipe\(\)](#), [cool_gas_onto_galaxy\(\)](#), [cooling_recipe\(\)](#), [deal_with_galaxy_merger\(\)](#), [disrupt\(\)](#), [do_AGN_heating\(\)](#), [do_major_merger_starburst\(\)](#), [do_reionization\(\)](#), [estimate_merging_time\(\)](#), [free_galaxies_and_tree\(\)](#), [get_disk_radius\(\)](#), [get_gas_disk_radius\(\)](#), [get_initial_disk_radius\(\)](#), [get_stellar_disk_radius\(\)](#), [grow_black_hole\(\)](#), [hot_retain_sat\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), [load_tree\(\)](#), [main\(\)](#), [make_bulge_from_burst\(\)](#), [peri_radius\(\)](#), [reincorporate_gas\(\)](#), [sat_radius\(\)](#), [starformation_and_feedback\(\)](#), [tsudy\(\)](#), [update_centralgal\(\)](#), [update_from_feedback\(\)](#), [update_from_starFormation\(\)](#), [update_hot_frac\(\)](#), [update_hotgas\(\)](#), [update_ICL\(\)](#), [update_type_1\(\)](#), and [update_type_2\(\)](#).

4.3.1.47 int* GalaxiesInOrder

Definition at line 205 of file [allvars.c](#).

Referenced by [save_galaxy_tree\(\)](#), and [walk\(\)](#).

4.3.1.48 int GalCount

Definition at line 59 of file [allvars.c](#).

Referenced by [main\(\)](#), and [walk\(\)](#).

4.3.1.49 float H2[RHO_LEN][Z_LEN]

Definition at line 189 of file [allvars.c](#).

Referenced by [cal_H2\(\)](#), and [read_sfrz\(\)](#).

4.3.1.50 struct halo_data* Halo

Definition at line 9 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [do_major_merger_starburst\(\)](#), [estimate_merging_time\(\)](#), [free_galaxies_and_tree\(\)](#), [get_disk_radius\(\)](#), [get_initial_disk_radius\(\)](#), [get_virial_mass\(\)](#), [get_virial_radius\(\)](#), [hot_retain_sat\(\)](#), [infall_recipe\(\)](#), [init_galaxy\(\)](#), [load_tree\(\)](#), [main\(\)](#), [peri_radius\(\)](#), [prepare_-](#)

`galaxy_for_output()`, `reincorporate_gas()`, `set_merger_center()`, `starformation_and_feedback()`, `tsudy()`, `update_centralgal()`, `update_from_feedback()`, `update_hotgas()`, `update_type_1()`, and `update_type_2()`.

4.3.1.51 struct halo_aux_data* HaloAux

Definition at line 12 of file `allvars.c`.

Referenced by `free_galaxies_and_tree()`, `load_tree()`, `main()`, `prepare_galaxy_for_output()`, and `set_merger_center()`.

4.3.1.52 struct GALAXY * HaloGal

Definition at line 5 of file `allvars.c`.

Referenced by `deal_with_galaxy_merger()`, `disrupt()`, `free_galaxies_and_tree()`, `load_tree()`, `main()`, `prepare_galaxy_for_output()`, `save_galaxies()`, `save_galaxy_tree()`, `set_merger_center()`, and `walk()`.

4.3.1.53 struct halo_ids_data* HaloIDs

Definition at line 15 of file `allvars.c`.

Referenced by `free_galaxies_and_tree()`, `load_tree()`, `prepare_galaxy_for_momaf()`, and `prepare_galaxy_for_output()`.

4.3.1.54 int Hashbits

Definition at line 162 of file `allvars.c`.

Referenced by `prepare_galaxy_for_output()`, and `read_parameter_file()`.

4.3.1.55 double Hubble

Definition at line 118 of file `allvars.c`.

Referenced by `get_virial_radius()`, `set_units()`, and `time_to_present()`.

4.3.1.56 double Hubble_h

Definition at line 68 of file `allvars.c`.

Referenced by `add_to_luminosities()`, `fix_units_for_ouput()`, `init()`, `prepare_galaxy_for_output()`, `read_parameter_file()`, and `setup_spectrophotometric_model()`.

4.3.1.57 long long* IdList

Definition at line 157 of file `allvars.c`.

Referenced by `get_coordinates()`, `save_galaxies()`, and `save_galaxy_tree()`.

4.3.1.58 int LastFile

Definition at line 21 of file `allvars.c`.

Referenced by [main\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.59 int LastSnapShotNr

Definition at line [47](#) of file [allvars.c](#).

Referenced by [load_all_auxdata\(\)](#), [load_all_dbids\(\)](#), [load_all_treedata\(\)](#), [load_tree\(\)](#), [load_tree_table\(\)](#), [open_tree_file\(\)](#), [open_treeaux_file\(\)](#), [open_treedbids_file\(\)](#), [read_parameter_file\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.60 int ListInputFilrNr[111]

Definition at line [39](#) of file [allvars.c](#).

Referenced by [main\(\)](#), and [read_file_nrs\(\)](#).

4.3.1.61 int ListOutputSnaps[NOUT]

Definition at line [126](#) of file [allvars.c](#).

Referenced by [add_to_luminosities\(\)](#), [collisional_starburst_recipe\(\)](#), [do_major_merger_starburst\(\)](#), [finalize_galaxy_file\(\)](#), [finalize_momaf_file\(\)](#), [load_tree_table\(\)](#), [main\(\)](#), [read_output_snaps\(\)](#), [save_galaxies\(\)](#), [save_galaxy_tree\(\)](#), [starformation_and_feedback\(\)](#), [tsudy\(\)](#), [write_galaxy_data_snap\(\)](#), and [write_galaxy_for_momaf\(\)](#).

4.3.1.62 float MagTableZz[NMAG][IZZ][IZ][ZL_LEN]

Definition at line [146](#) of file [allvars.c](#).

Referenced by [add_to_luminosities\(\)](#), and [setup_spectrophotometric_model\(\)](#).

4.3.1.63 int MaxGals

Definition at line [23](#) of file [allvars.c](#).

Referenced by [load_tree\(\)](#), and [main\(\)](#).

4.3.1.64 size_t maxstorage_galaxydata

Definition at line [169](#) of file [allvars.c](#).

Referenced by [load_all_treedata\(\)](#), and [myfwrite\(\)](#).

4.3.1.65 size_t maxstorage_galsnapdata[NOUT]

Definition at line [170](#) of file [allvars.c](#).

Referenced by [load_all_treedata\(\)](#), and [myfwrite\(\)](#).

4.3.1.66 size_t maxstorage_momafdata[NOUT]

Definition at line [173](#) of file [allvars.c](#).

Referenced by [load_all_treedata\(\)](#), and [myfwrite\(\)](#).

4.3.1.67 float metalcold

Definition at line [183](#) of file [allvars.c](#).

4.3.1.68 int MetallicityOption

Definition at line [78](#) of file [allvars.c](#).

Referenced by [add_to_luminosities\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.69 long mu_seed

Definition at line [151](#) of file [allvars.c](#).

Referenced by [main\(\)](#), and [tsudy\(\)](#).

4.3.1.70 int Nids

Definition at line [155](#) of file [allvars.c](#).

Referenced by [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.71 int NTask

Definition at line [54](#) of file [allvars.c](#).

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

4.3.1.72 int NtotHalos

Definition at line [155](#) of file [allvars.c](#).

Referenced by [load_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.73 int Ntrees

Definition at line [25](#) of file [allvars.c](#).

Referenced by [finalize_galaxy_file\(\)](#), [load_tree\(\)](#), [load_tree_table\(\)](#), [main\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.74 int NumGals

Definition at line [26](#) of file [allvars.c](#).

Referenced by [main\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.75 int NumMergers

Definition at line 72 of file [allvars.c](#).

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

4.3.1.76 size_t offset_auxdata

Definition at line 168 of file [allvars.c](#).

Referenced by [load_all_auxdata\(\)](#), [load_tree_table\(\)](#), [myfread\(\)](#), [myfseek\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.77 size_t offset_dbids

Definition at line 168 of file [allvars.c](#).

Referenced by [load_all_dbids\(\)](#), [load_tree\(\)](#), [myfread\(\)](#), and [myfseek\(\)](#).

4.3.1.78 size_t offset_galaxydata

Definition at line 169 of file [allvars.c](#).

Referenced by [finalize_galaxy_file\(\)](#), [load_all_treedata\(\)](#), [myfseek\(\)](#), [myfwrite\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.79 size_t offset_galsnapdata[NOUT]

Definition at line 170 of file [allvars.c](#).

Referenced by [finalize_galaxy_file\(\)](#), [load_all_treedata\(\)](#), [myfseek\(\)](#), [myfwrite\(\)](#), and [save_galaxies\(\)](#).

4.3.1.80 size_t offset_momafdata[NOUT]

Definition at line 173 of file [allvars.c](#).

Referenced by [finalize_momaf_file\(\)](#), [load_all_treedata\(\)](#), [myfseek\(\)](#), [myfwrite\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.81 size_t offset_treedata

Definition at line 168 of file [allvars.c](#).

Referenced by [load_all_treedata\(\)](#), [load_tree\(\)](#), [load_tree_table\(\)](#), [myfread\(\)](#), and [myfseek\(\)](#).

4.3.1.82 int OffsetIDs

Definition at line 155 of file [allvars.c](#).

Referenced by [get_coordinates\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.83 int * OffsetIDs_halo

Definition at line 156 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [get_coordinates\(\)](#), and [load_tree_table\(\)](#).

4.3.1.84 int * OffsetIDs_snaptree

Definition at line 156 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [load_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.85 double Omega

Definition at line 66 of file [allvars.c](#).

Referenced by [do_reionization\(\)](#), [get_virial_radius\(\)](#), [integrand_time_to_present\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.86 double OmegaLambda

Definition at line 67 of file [allvars.c](#).

Referenced by [get_virial_radius\(\)](#), [integrand_time_to_present\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.87 FILE* output_file

Definition at line 200 of file [allvars.c](#).

Referenced by [closeallfiles\(\)](#), [finalize_galaxy_file\(\)](#), [openallfiles\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.88 char OutputDir[512]

Definition at line 31 of file [allvars.c](#).

Referenced by [finalize_galaxy_file\(\)](#), [finalize_momaf_file\(\)](#), [load_tree_table\(\)](#), [main\(\)](#), [open_outputtree_file\(\)](#), [read_parameter_file\(\)](#), [save_galaxies\(\)](#), [save_galaxy_tree\(\)](#), [write_all_galaxy_data\(\)](#), [write_galaxy_data_snap\(\)](#), and [write_galaxy_for_momaf\(\)](#).

4.3.1.89 double PartMass

Definition at line 69 of file [allvars.c](#).

Referenced by [get_virial_mass\(\)](#), [hot_retain_sat\(\)](#), [read_parameter_file\(\)](#), [update_hot_frac\(\)](#), [update_hotgas\(\)](#), and [update_type_1\(\)](#).

4.3.1.90 char PhotDir[512]

Definition at line 28 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), [read_tsud_tables\(\)](#), and [setup_spectrophotometric_model\(\)](#).

4.3.1.91 char PhotPrefix[50]

Definition at line 29 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), [read_tsud_tables\(\)](#), and [setup_spectrophotometric_model\(\)](#).

4.3.1.92 float* PosList

Definition at line 158 of file [allvars.c](#).

Referenced by [get_coordinates\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.93 char* ptr_auxdata

Definition at line 167 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [load_all_auxdata\(\)](#), and [myfread\(\)](#).

4.3.1.94 char * ptr_dbids

Definition at line 167 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [load_all_dbids\(\)](#), and [myfread\(\)](#).

4.3.1.95 char * ptr_galaxydata

Definition at line 167 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [load_all_treedata\(\)](#), [myfwrite\(\)](#), and [write_all_galaxy_data\(\)](#).

4.3.1.96 char * ptr_galsnapdata[NOUT]

Definition at line 167 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [load_all_treedata\(\)](#), [myfwrite\(\)](#), and [write_galaxy_data_snap\(\)](#).

4.3.1.97 char* ptr_momafdata[NOUT]

Definition at line 172 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [load_all_treedata\(\)](#), [myfwrite\(\)](#), and [write_galaxy_for_momaf\(\)](#).

4.3.1.98 char * ptr_treedata

Definition at line 167 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [load_all_treedata\(\)](#), and [myfread\(\)](#).

4.3.1.99 gsl_rng* random_generator

Definition at line 135 of file [allvars.c](#).

Referenced by [init\(\)](#), [init_galaxy\(\)](#), and [main\(\)](#).

4.3.1.100 float RecGas

Definition at line 181 of file [allvars.c](#).

4.3.1.101 double RecycleFraction

Definition at line 102 of file [allvars.c](#).

Referenced by [add_to_luminosities\(\)](#), [do_major_merger_starburst\(\)](#), [read_parameter_file\(\)](#), and [update_from_star_formation\(\)](#).

4.3.1.102 float RedshiftTab[IZ]

Definition at line 145 of file [allvars.c](#).

Referenced by [find_interpolated_obsลม\(\)](#), and [setup_spectrophotometric_model\(\)](#).

4.3.1.103 double ReheatPreVelocity

Definition at line 96 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.3.1.104 double ReheatSlope

Definition at line 97 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.3.1.105 double ReIncorporationFactor

Definition at line 105 of file [allvars.c](#).

Referenced by [main\(\)](#), [read_parameter_file\(\)](#), and [reincorporate_gas\(\)](#).

4.3.1.106 int ReIncorporationRecipe

Definition at line 91 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), and [reincorporate_gas\(\)](#).

4.3.1.107 float Reion_Mc[46]

Definition at line 192 of file [allvars.c](#).

Referenced by [do_reionization\(\)](#), and [read_reionization\(\)](#).

4.3.1.108 float Reion_z[46]

Definition at line 192 of file [allvars.c](#).

Referenced by [find_interpolate_reionization\(\)](#), and [read_reionization\(\)](#).

4.3.1.109 double Reionization_z0

Definition at line 114 of file [allvars.c](#).

Referenced by [init\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.110 double Reionization_zr

Definition at line 115 of file [allvars.c](#).

Referenced by [init\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.111 int ReionizationOn

Definition at line 81 of file [allvars.c](#).

Referenced by [cooling_recipe\(\)](#), [infall_recipe\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.112 float Rho[RHO_LEN]

Definition at line 188 of file [allvars.c](#).

Referenced by [find_interpolate_h2\(\)](#), and [read_sfrz\(\)](#).

4.3.1.113 double RhoCrit

Definition at line 118 of file [allvars.c](#).

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

4.3.1.114 int SatelliteRecipe

Definition at line 83 of file [allvars.c](#).

Referenced by [infall_recipe\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.115 double SfrAlpha

Definition at line 95 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.3.1.116 double SfrEfficiency

Definition at line 109 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.3.1.117 double SfrLawPivotVelocity

Definition at line 93 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.3.1.118 double SfrLawSlope

Definition at line 94 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.3.1.119 char SimulationDir[512]

Definition at line 33 of file [allvars.c](#).

Referenced by [load_all_auxdata\(\)](#), [load_all_dbids\(\)](#), [load_all_treedata\(\)](#), [load_tree\(\)](#), [load_tree_table\(\)](#), [open_tree_file\(\)](#), [open_treeaux_file\(\)](#), [open_treedbids_file\(\)](#), [read_parameter_file\(\)](#), and [save_galaxies\(\)](#).

4.3.1.120 int Snaplistlen

Definition at line 133 of file [allvars.c](#).

Referenced by [init\(\)](#), and [read_snap_list\(\)](#).

4.3.1.121 double SNinReheat

Definition at line 100 of file [allvars.c](#).

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

4.3.1.122 int StarBurstRecipe

Definition at line 84 of file [allvars.c](#).

Referenced by [deal_with_galaxy_merger\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.123 int StarBurstsInMajorMergersOn

Definition at line 76 of file [allvars.c](#).

Referenced by [do_major_merger_starburst\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.124 int StarFormationRecipe

Definition at line 82 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.3.1.125 int ThisTask

Definition at line 54 of file [allvars.c](#).

Referenced by [init\(\)](#), [main\(\)](#), [read_cooling_functions\(\)](#), [read_parameter_file\(\)](#), [read_snap_list\(\)](#), and [setup_spectrophotometric_model\(\)](#).

4.3.1.126 double ThreshMajorMerger

Definition at line 107 of file [allvars.c](#).

Referenced by [bulgesize_from_merger\(\)](#), [collisional_starburst_recipe\(\)](#), [deal_with_galaxy_merger\(\)](#), and [read_parameter_file\(\)](#).

4.3.1.127 int TotGalaxies[NOUT]

Definition at line 43 of file [allvars.c](#).

Referenced by [finalize_galaxy_file\(\)](#), [finalize_momaf_file\(\)](#), [load_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.128 int TotGalCount

Definition at line 60 of file [allvars.c](#).

Referenced by [finalize_galaxy_file\(\)](#), [load_tree_table\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.129 int TotHalos

Definition at line 42 of file [allvars.c](#).

4.3.1.130 int TotIds

Definition at line 155 of file [allvars.c](#).

Referenced by [load_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.131 int TotSnaps

Definition at line 155 of file [allvars.c](#).

Referenced by [load_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.132 int TrackDiskInstability

Definition at line 88 of file [allvars.c](#).

Referenced by [deal_with_galaxy_merger\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.3.1.133 FILE* tree_file

Definition at line 197 of file [allvars.c](#).

Referenced by [closeallfiles\(\)](#), [load_tree\(\)](#), [load_tree_table\(\)](#), and [openallfiles\(\)](#).

4.3.1.134 FILE* treeaux_file

Definition at line 198 of file [allvars.c](#).

Referenced by [closeallfiles\(\)](#), [load_tree_table\(\)](#), [openallfiles\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.135 FILE* treedbids_file

Definition at line [199](#) of file [allvars.c](#).

Referenced by [closeallfiles\(\)](#), [load_tree\(\)](#), and [openallfiles\(\)](#).

4.3.1.136 int* TreeFirstHalo

Definition at line [51](#) of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [get_coordinates\(\)](#), [load_tree\(\)](#), and [load_tree_table\(\)](#).

4.3.1.137 int* TreeNgals[NOUT]

Definition at line [44](#) of file [allvars.c](#).

Referenced by [finalize_galaxy_file\(\)](#), [free_tree_table\(\)](#), [load_tree_table\(\)](#), and [save_galaxies\(\)](#).

4.3.1.138 int* TreeNHalos

Definition at line [50](#) of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [load_tree\(\)](#), [load_tree_table\(\)](#), and [main\(\)](#).

4.3.1.139 double UnitCoolingRate_in_cgs

Definition at line [118](#) of file [allvars.c](#).

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

4.3.1.140 double UnitDensity_in_cgs

Definition at line [118](#) of file [allvars.c](#).

Referenced by [cooling_recipe\(\)](#), and [set_units\(\)](#).

4.3.1.141 double UnitEnergy_in_cgs

Definition at line [118](#) of file [allvars.c](#).

Referenced by [do_AGN_heating\(\)](#), [init\(\)](#), and [set_units\(\)](#).

4.3.1.142 double UnitLength_in_cm

Definition at line [118](#) of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), and [set_units\(\)](#).

4.3.1.143 double UnitMass_in_g

Definition at line 118 of file [allvars.c](#).

Referenced by [do_AGN_heating\(\)](#), [init\(\)](#), [prepare_galaxy_for_output\(\)](#), [read_parameter_file\(\)](#), and [set_units\(\)](#).

4.3.1.144 double UnitPressure_in_cgs

Definition at line 118 of file [allvars.c](#).

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

4.3.1.145 double UnitTime_in_Megayears

Definition at line 118 of file [allvars.c](#).

Referenced by [add_to_luminosities\(\)](#), [prepare_galaxy_for_output\(\)](#), [set_units\(\)](#), and [setup_spectrophotometric_model\(\)](#).

4.3.1.146 double UnitTime_in_s

Definition at line 118 of file [allvars.c](#).

Referenced by [cooling_recipe\(\)](#), [do_AGN_heating\(\)](#), [init\(\)](#), [prepare_galaxy_for_output\(\)](#), and [set_units\(\)](#).

4.3.1.147 double UnitVelocity_in_cm_per_s

Definition at line 118 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), and [set_units\(\)](#).

4.3.1.148 float * VelList

Definition at line 158 of file [allvars.c](#).

Referenced by [get_coordinates\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.3.1.149 double Yield

Definition at line 103 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [do_major_merger_starburst\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.3.1.150 double ZZ[MAXSNAPS]

Definition at line 129 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [do_major_merger_starburst\(\)](#), [estimate_merging_time\(\)](#), [finalize_galaxy_file\(\)](#), [get_virial_radius\(\)](#), [hot_retain_sat\(\)](#), [infall_recipe\(\)](#), [init\(\)](#), [load_tree_table\(\)](#), [main\(\)](#), [peri_radius\(\)](#), [prepare_galaxy_for_momaf\(\)](#), [prepare_galaxy_for_output\(\)](#), [RedshiftObs\(\)](#),

`reincorporate_gas()`, `save_galaxies()`, `starformation_and_feedback()`, `tsudy()`, `update_from_feedback()`, `update_hotgas()`, and `write_galaxy_data_snap()`.

4.4 code/allvars.c

```

00001 // TODO add description for all variables that do not have one yet
00002 #include "allvars.h"
00003
00004
00005 struct GALAXY           /* Galaxy data */
00006 *Gal, *HaloGal;
00007
00008
00009 struct halo_data *Halo;
00010
00011
00012 struct halo_aux_data    /* auxiliary halo data */
00013 *HaloAux;
00014
00015 struct halo_ids_data *HaloIDs;
00016
00017
00018 /* misc */
00019
00020 int FirstFile;          /* first and last file for processing */
00021 int LastFile;
00022
00023 int MaxGals;
00024 int FoF_MaxGals;
00025 int Ntrees;             /* number of trees in current file */
00026 int NumGals;            /* Total number of galaxies stored for current tr
     ee */
00027
00028 char PhotDir[512];
00029 char PhotPrefix[50];
00030 char CoolFunctionsDir[512];
00031 char OutputDir[512];
00032 char FileNameGalaxies[512];
00033 char SimulationDir[512];
00034 char FileWithOutputSnaps[512];
00035 char FileWithSnapList[512];
00036
00037 #ifdef SPECIFYFILENR
00038 char FileNrDir[512];
00039 int ListInputFilrNr[111];
00040 #endif
00041
00042 int TotHalos;
00043 int TotGalaxies[NOUT];
00044 int *TreeNgals[NOUT];
00045
00046 int FilesPerSnapshot;
00047 int LastSnapShotNr;
00048
00049 int *FirstHaloInSnap;
00050 int *TreeNHalos;
00051 int *TreeFirstHalo;
00052
00053 #ifdef PARALLEL
00054 int ThisTask, NTask;
00055 #endif
00056
00057
00058 #ifdef GALAXYTREE

```

```
00059 int GalCount;
00060 int TotGalCount;
00061 #endif
00062
00063
00064
00065 /* more misc */
00066 double Omega;
00067 double OmegaLambda;
00068 double Hubble_h;
00069 double PartMass;
00070 double EnergySNcode, EnergySN;
00071 double EtaSNcode, EtaSN;
00072 int NumMergers;
00073
00074
00075 /* recipe flags */
00076 int StarBurstsInMajorMergersOn;
00077 int BulgeFormationInMinorMergersOn;
00078 int MetallicityOption;
00079 int EjectionOn;
00080 int CoolingCutoff;
00081 int ReionizationOn;
00082 int StarFormationRecipe;
00083 int SatelliteRecipe;
00084 int StarBurstRecipe;
00085 int FeedbackRecipe;
00086 int EjectionRecipe;
00087 int DiskRadiusMethod;
00088 int TrackDiskInstability;
00089 int AGNrecipeOn;
00090 int SatelliteRecipe;
00091 int ReIncorporationRecipe;
00092 /* recipe parameters */
00093 double SfrLawPivotVelocity;
00094 double SfrLawSlope;
00095 double SfrAlpha;
00096 double ReheatPreVelocity;
00097 double ReheatSlope;
00098 double EjectPreVelocity;
00099 double EjectSlope;
00100 double SNinReheat;
00101 double FeedbackEpsilon;
00102 double RecycleFraction;
00103 double Yield;
00104 double FracZtoHot;
00105 double ReIncorporationFactor;
00106 double CoolingVelocityCutOff;
00107 double ThreshMajorMerger;
00108 double BaryonFrac;
00109 double SfrEfficiency;
00110 double FeedbackReheatingEpsilon;
00111 double FeedbackEjectionEfficiency;
00112 double AgnEfficiency;
00113 double BlackHoleGrowthRate;
00114 double Reionization_z0;
00115 double Reionization_zr;
00116
00117
00118 double UnitLength_in_cm,
00119     UnitTime_in_s,
00120     UnitVelocity_in_cm_per_s,
00121     UnitMass_in_g,
00122     RhoCrit,
00123     UnitPressure_in_cgs,
00124     UnitDensity_in_cgs, UnitCoolingRate_in_cgs, UnitEnergy_in_cgs,
     UnitTime_in_Megayears, G, Hubble, a0, ar;
```

```

00125
00126 int ListOutputSnaps[NOUT];
00127
00128
00129 double ZZ[MAXSNAPS];
00130 double AA[MAXSNAPS];
00131 double Age[MAXSNAPS];
00132
00133 int Snaplistlen;
00134
00135 gsl_rng *random_generator;
00136
00137
00138 /* tabulated stuff */
00139
00140 /* fixed-metallicity spectrophotometric model */
00141 /* tables hold magnitudes of starburst population as a function of age */
00142
00143
00144 float AgeTab[ZL_LEN];
00145 float RedshiftTab[IZ];
00146 float MagTableZZ[NMAG][IZZ][IZ][ZL_LEN];
00147
00148 // tsud
00149 float DeltaM[NMAG+1][NZZ_EXT][IZ];
00150 float Corrections[NMAG][IZ];
00151 long mu_seed;
00152
00153
00154 #ifdef UPDATETYPETWO
00155 int NtotHalos, TotIds, Nids, TotSnaps, OffsetIDs,
00156 int *CountIDs_halo, *OffsetIDs_halo, *CountIDs_snaptree, *OffsetIDs_snaptree;
00157 long long *IdList;
00158 float *PosList, *VelList;
00159#endif
00160
00161 #ifdef GALAXYTREE
00162 int Hashbits;
00163 double BoxSize;
00164#endif
00165
00166 #ifdef USE_MEMORY_TO_MINIMIZE_IO
00167 char *ptr_auxdata, *ptr_treedata, *ptr_dbids, *ptr_galaxydata, *ptr_galsnapdata[N
    OUT];
00168 size_t offset_auxdata, offset_treedata, offset_dbids;
00169 size_t offset_galaxydata, maxstorage_galaxydata, filled_galaxydata;
00170 size_t offset_galsnapdata[NOUT], maxstorage_galsnapdata[NOUT],
    filled_galsnapdata[NOUT];
00171 #ifdef OUTPUT_MOMAF_INPUTS
00172 char *ptr_momafdata[NOUT];
00173 size_t offset_momafdata[NOUT], maxstorage_momafdata[NOUT], filled_momafdata[NOUT];
00174#endif
00175#endif
00176
00177
00178 #ifdef GASRECYCLE
00179
00180 /* gas recycle for SF */
00181 float RecGas;
00182 float Frac[ZL_LEN];
00183 float metalcold;
00184#endif
00185
00186 /*H2 formation table */
00187
00188 float Rho[RHO_LEN];
00189 float H2[RHO_LEN][Z_LEN];

```

```
00190
00191 /* reionization Okamoto et al. 2008*/
00192 float Reion_z[46],Reion_Mc[46];
00193
00194
00195
00196 #ifdef NEW_IO
00197 FILE *tree_file;
00198 FILE *treeaux_file;
00199 FILE *treedbids_file;
00200 FILE *output_file;
00201 /*extern FILE* momaf_file;
00202 #endif
00203
00204
00205 int* GalaxiesInOrder;
00206
```

4.5 code/allvars.h File Reference

TODO add description for all variables that do not have one yet.

Data Structures

- struct [GALAXY_OUTPUT](#)
Galaxy structure for output.
- struct [MOMAF_INPUTS](#)
- struct [GALAXY](#)
- struct [halo_data](#)
- struct [halo_ids_data](#)
- struct [halo_aux_data](#)

Variables

- struct [GALAXY](#) * Gal
- struct [GALAXY](#) * HaloGal
- struct [halo_data](#) * Halo
- struct [halo_ids_data](#) * HaloIDs
- struct [halo_aux_data](#) * HaloAux
- int FirstFile
- int LastFile
- int Ntrees
- int NumGals
- int MaxGals
- int FoF_MaxGals
- int FilesPerSnapshot
- int LastSnapShotNr
- char [PhotDir](#) [512]
- char [PhotPrefix](#) [50]
- char [CoolFunctionsDir](#) [512]
- char [OutputDir](#) [512]

- char `FileNameGalaxies` [512]
- char `SimulationDir` [512]
- char `FileWithOutputSnaps` [512]
- char `FileWithSnapList` [512]
- char `FileNrDir` [512]
- int `ListInputFilrNr` [111]
- int `TotHalos`
- int `TotGalaxies` [NOUT]
- int * `TreeNgals` [NOUT]
- int * `FirstHaloInSnap`
- int * `TreeNHalos`
- int * `TreeFirstHalo`
- int `ThisTask`
- int `NTask`
- int `GalCount`
- int `TotGalCount`
- double `Omega`
- double `OmegaLambda`
- double `PartMass`
- double `Hubble_h`
- double `EnergySNcode`
- double `EnergySN`
- double `EtaSNcode`
- double `EtaSN`
- int `NumMergers`
- int `StarBurstsInMajorMergersOn`
- int `BulgeFormationInMinorMergersOn`
- int `MetallicityOption`
- int `EjectionOn`
- int `CoolingCutoff`
- int `ReionizationOn`
- int `StarFormationRecipe`
- int `StelliteRecipe`
- int `StarBurstRecipe`
- int `FeedbackRecipe`
- int `EjectionRecipe`
- int `DiskRadiusMethod`
- int `TrackDiskInstability`
- int `AGNrecipeOn`
- int `SatelliteRecipe`
- int `ReIncorporationRecipe`
- double `SfrLawPivotVelocity`
- double `SfrLawSlope`
- double `SfrAlpha`
- double `ReheatPreVelocity`
- double `ReheatSlope`
- double `SNinReheat`
- double `FeedbackEpsilon`
- double `EjectPreVelocity`
- double `EjectSlope`

- double RecycleFraction
- double Yield
- double FracZtoHot
- double ReIncorporationFactor
- double CoolingVelocityCutOff
- double ThreshMajorMerger
- double BaryonFrac
- double SfrEfficiency
- double FeedbackReheatingEpsilon
- double FeedbackEjectionEfficiency
- double AgnEfficiency
- double BlackHoleGrowthRate
- double Reionization_z0
- double Reionization_zr
- double UnitLength_in_cm
- double UnitTime_in_s
- double UnitVelocity_in_cm_per_s
- double UnitMass_in_g
- double RhoCrit
- double UnitPressure_in_cgs
- double UnitDensity_in_cgs
- double UnitCoolingRate_in_cgs
- double UnitEnergy_in_cgs
- double UnitTime_in_Megayears
- double G
- double Hubble
- double a0
- double ar
- int ListOutputSamps [NOUT]
- double ZZ [MAXSNAPS]
- double AA [MAXSNAPS]
- double Age [MAXSNAPS]
- int Snaplistlen
- gsl_rng * random_generator
- float AgeTab [ZL_LEN]
- float RedshiftTab [IZ]
- float MagTableZz [NMAG][IZZ][IZ][ZL_LEN]
- float IdxTable [2 *NIDX][IZZ][ZL_LEN]
- int IdxType [NIDX]
- float Idxw5 [NIDX]
- float Idxw6 [NIDX]
- float DeltaM [NMAG+1][NZZ_EXT][IZ]
- float Corrections [NMAG][IZ]
- long mu_seed
- int NtotHalos
- int TotIds
- int Nids
- int TotSamps
- int OffsetIDs
- int * CountIDs_halo

- int * [OffsetIDs_halo](#)
- int * [CountIDs_snaptree](#)
- int * [OffsetIDs_snaptree](#)
- long long * [IdList](#)
- float * [PosList](#)
- float * [VelList](#)
- int [Hashbits](#)
- double [BoxSize](#)
- char * [ptr_auxdata](#)
- char * [ptr_treedata](#)
- char * [ptr_dbids](#)
- char * [ptr_galaxydata](#)
- char * [ptr_galsnapdata](#) [NOUT]
- size_t [offset_auxdata](#)
- size_t [offset_treedata](#)
- size_t [offset_dbids](#)
- size_t [offset_galaxydata](#)
- size_t [maxstorage_galaxydata](#)
- size_t [filled_galaxydata](#)
- size_t [offset_galsnapdata](#) [NOUT]
- size_t [maxstorage_galsnapdata](#) [NOUT]
- size_t [filled_galsnapdata](#) [NOUT]
- char * [ptr_momafdata](#) [NOUT]
- size_t [offset_momafdata](#) [NOUT]
- size_t [maxstorage_momafdata](#) [NOUT]
- size_t [filled_momafdata](#) [NOUT]
- float [RecGas](#)
- float [Frac](#) [ZL_LEN]
- float [metalcold](#)
- float [Rho](#) [RHO_LEN]
- float [H2](#) [RHO_LEN][Z_LEN]
- float [Reion_z](#) [46]
- float [Reion_Mc](#) [46]
- FILE * [tree_file](#)
- FILE * [treeaux_file](#)
- FILE * [treedbids_file](#)
- FILE * [output_file](#)
- int * [GalaxiesInOrder](#)

4.5.1 Detailed Description

bla bla bla.

Definition in file [allvars.h](#).

4.5.2 Variable Documentation

4.5.2.1 double a0

Definition at line 118 of file [allvars.c](#).

Referenced by [do_reionization\(\)](#), and [init\(\)](#).

4.5.2.2 double AA[MAXSNAPS]

Definition at line 130 of file [allvars.c](#).

Referenced by [get_coordinates\(\)](#), [init\(\)](#), and [read_snap_list\(\)](#).

4.5.2.3 double Age[MAXSNAPS]

Definition at line 131 of file [allvars.c](#).

Referenced by [init\(\)](#), and [NumToTime\(\)](#).

4.5.2.4 float AgeTab[ZL_LEN]

Definition at line 144 of file [allvars.c](#).

Referenced by [find_interpolated_lum\(\)](#), [find_interpolated_obs_lum\(\)](#), [find_interpolation_point\(\)](#), [get_jump_index\(\)](#), [init_jump_index\(\)](#), and [setup_spectrophotometric_model\(\)](#).

4.5.2.5 double AgnEfficiency

Definition at line 112 of file [allvars.c](#).

Referenced by [do_AGN_heating\(\)](#), [init\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.6 int AGNrecipeOn

Definition at line 89 of file [allvars.c](#).

Referenced by [cooling_recipe\(\)](#), [deal_with_galaxy_merger\(\)](#), [do_AGN_heating\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.7 double ar

Definition at line 118 of file [allvars.c](#).

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

4.5.2.8 double BaryonFrac

Definition at line 108 of file [allvars.c](#).

Referenced by [infall_recipe\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.9 double BlackHoleGrowthRate

Definition at line 113 of file [allvars.c](#).

Referenced by [grow_black_hole\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.10 double BoxSize

Definition at line 163 of file [allvars.c](#).

Referenced by [prepare_galaxy_for_output\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.11 int BulgeFormationInMinorMergersOn

Definition at line 77 of file [allvars.c](#).

Referenced by [add_galaxies_together\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.12 char CoolFunctionsDir[512]

Definition at line 30 of file [allvars.c](#).

Referenced by [read_cooling_functions\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.13 int CoolingCutoff

Definition at line 80 of file [allvars.c](#).

Referenced by [cooling_recipe\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.14 double CoolingVelocityCutOff

Definition at line 106 of file [allvars.c](#).

Referenced by [cooling_recipe\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.15 float Corrections[NMAG][IZ]

Definition at line 150 of file [allvars.c](#).

Referenced by [read_tsud_tables\(\)](#), and [tsudy\(\)](#).

4.5.2.16 int* CountIDs_halo

Definition at line 156 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [get_coordinates\(\)](#), and [load_tree_table\(\)](#).

4.5.2.17 int * CountIDs_snaptree

Definition at line 156 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [load_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.18 float DeltaM[NMAG+1][NZZ_EXT][IZ]

Definition at line 149 of file [allvars.c](#).

Referenced by [read_tsud_tables\(\)](#), and [tsudy\(\)](#).

4.5.2.19 int DiskRadiusMethod

Definition at line 87 of file [allvars.c](#).

Referenced by [cool_gas_onto_galaxy\(\)](#), [deal_with_galaxy_merger\(\)](#), [get_disk_radius\(\)](#), [get_initial_disk_radius\(\)](#), [main\(\)](#), [read_parameter_file\(\)](#), [starformation_and_feedback\(\)](#), and [update_from_starFormation\(\)](#).

4.5.2.20 int EjectionOn

Definition at line 79 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [do_major_merger_starburst\(\)](#), [read_parameter_file\(\)](#), [starformation_and_feedback\(\)](#), and [update_from_feedback\(\)](#).

4.5.2.21 int EjectionRecipe

Definition at line 86 of file [allvars.c](#).

Referenced by [infall_recipe\(\)](#), [read_parameter_file\(\)](#), and [update_from_feedback\(\)](#).

4.5.2.22 double EjectPreVelocity

Definition at line 98 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.5.2.23 double EjectSlope

Definition at line 99 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.5.2.24 double EnergySN

Definition at line 70 of file [allvars.c](#).

Referenced by [init\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.25 double EnergySNcode

Definition at line 70 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [init\(\)](#), and [starformation_and_feedback\(\)](#).

4.5.2.26 double EtaSN

Definition at line 71 of file [allvars.c](#).

Referenced by [init\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.27 double EtaSNcode

Definition at line 71 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [init\(\)](#), and [starformation_and_feedback\(\)](#).

4.5.2.28 double FeedbackEjectionEfficiency

Definition at line 111 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.5.2.29 double FeedbackEpsilon

Definition at line 101 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.5.2.30 int FeedbackRecipe

Definition at line 85 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [do_major_merger_starburst\(\)](#), [read_parameter_file\(\)](#), [starformation_and_feedback\(\)](#), and [update_from_feedback\(\)](#).

4.5.2.31 double FeedbackReheatingEpsilon

Definition at line 110 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.5.2.32 char FileNameGalaxies[512]

Definition at line 32 of file [allvars.c](#).

Referenced by [finalize_galaxy_file\(\)](#), [finalize_momaf_file\(\)](#), [load_tree_table\(\)](#), [main\(\)](#), [open_outputtree_file\(\)](#), [read_parameter_file\(\)](#), [save_galaxies\(\)](#), [save_galaxy_tree\(\)](#), [write_all_galaxy_data\(\)](#), [write_galaxy_data_snap\(\)](#), and [write_galaxy_for_momaf\(\)](#).

4.5.2.33 char FileNrDir[512]

Definition at line 38 of file [allvars.c](#).

Referenced by [read_file_nrs\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.34 int FilesPerSnapshot

Definition at line 46 of file [allvars.c](#).

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

4.5.2.35 char FileWithOutputSnaps[512]

Definition at line 34 of file [allvars.c](#).

Referenced by [read_output_snaps\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.36 char FileWithSnapList[512]

Definition at line 35 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), and [read_snap_list\(\)](#).

4.5.2.37 size_t filled_galaxydata

Definition at line 169 of file [allvars.c](#).

Referenced by [load_all_auxdata\(\)](#), [load_all_treedata\(\)](#), [myfwrite\(\)](#), and [write_all_galaxy_data\(\)](#).

4.5.2.38 size_t filled_galsnapdata[NOUT]

Definition at line 170 of file [allvars.c](#).

Referenced by [load_all_treedata\(\)](#), [myfwrite\(\)](#), and [write_galaxy_data_snap\(\)](#).

4.5.2.39 size_t filled_momafdata[NOUT]

Definition at line 173 of file [allvars.c](#).

Referenced by [load_all_treedata\(\)](#), [myfwrite\(\)](#), and [write_galaxy_for_momaf\(\)](#).

4.5.2.40 int FirstFile

Definition at line 20 of file [allvars.c](#).

Referenced by [main\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.41 int* FirstHaloInSnap

Definition at line 49 of file [allvars.c](#).

4.5.2.42 int FoF_MaxGals

Definition at line 24 of file [allvars.c](#).

Referenced by [load_tree\(\)](#), and [main\(\)](#).

4.5.2.43 float Frac[ZL_LEN]

Definition at line 182 of file [allvars.c](#).

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

4.5.2.44 double FracZtoHot

Definition at line 104 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.5.2.45 double G

Definition at line 118 of file [allvars.c](#).

Referenced by [check_disk_instability\(\)](#), [do_AGN_heating\(\)](#), [estimate_merging_time\(\)](#), [get_virial_radius\(\)](#), [get_virial_velocity\(\)](#), and [set_units\(\)](#).

4.5.2.46 struct GALAXY * Gal**4.5.2.47 int* GalaxiesInOrder**

Definition at line 205 of file [allvars.c](#).

Referenced by [save_galaxy_tree\(\)](#), and [walk\(\)](#).

4.5.2.48 int GalCount

Definition at line 59 of file [allvars.c](#).

Referenced by [main\(\)](#), and [walk\(\)](#).

4.5.2.49 float H2[RHO_LEN][Z_LEN]

Definition at line 189 of file [allvars.c](#).

Referenced by [cal_H2\(\)](#), and [read_sfrz\(\)](#).

4.5.2.50 struct halo_data * Halo**4.5.2.51 struct halo_aux_data * HaloAux****4.5.2.52 struct GALAXY * HaloGal****4.5.2.53 struct halo_ids_data * HaloIDs****4.5.2.54 int Hashbits**

Definition at line 162 of file [allvars.c](#).

Referenced by [prepare_galaxy_for_output\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.55 double Hubble

Definition at line 118 of file [allvars.c](#).

Referenced by [get_virial_radius\(\)](#), [set_units\(\)](#), and [time_to_present\(\)](#).

4.5.2.56 double Hubble_h

Definition at line 68 of file [allvars.c](#).

Referenced by [add_to_luminosities\(\)](#), [fix_units_for_ouput\(\)](#), [init\(\)](#), [prepare_galaxy_for_output\(\)](#), [read_parameter_file\(\)](#), and [setup_spectrophotometric_model\(\)](#).

4.5.2.57 long long* IdList

Definition at line 157 of file [allvars.c](#).

Referenced by [get_coordinates\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.58 float IdxTable[2 *NIDX][IZZ][ZL_LEN]**4.5.2.59 int IdxType[NIDX]****4.5.2.60 float Idxw5[NIDX]****4.5.2.61 float Idxw6[NIDX]****4.5.2.62 int LastFile**

Definition at line 21 of file [allvars.c](#).

Referenced by [main\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.63 int LastSnapShotNr

Definition at line 47 of file [allvars.c](#).

Referenced by [load_all_auxdata\(\)](#), [load_all_dbids\(\)](#), [load_all_treedata\(\)](#), [load_tree\(\)](#), [load_tree_table\(\)](#), [open_tree_file\(\)](#), [open_treeaux_file\(\)](#), [open_treedbids_file\(\)](#), [read_parameter_file\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.64 int ListInputFilrNr[111]

Definition at line 39 of file [allvars.c](#).

Referenced by [main\(\)](#), and [read_file_nrs\(\)](#).

4.5.2.65 int ListOutputSnaps[NOUT]

Definition at line 126 of file [allvars.c](#).

Referenced by [add_to_luminosities\(\)](#), [collisional_starburst_recipe\(\)](#), [do_major_merger_starburst\(\)](#), [finalize_galaxy_file\(\)](#), [finalize_momaf_file\(\)](#), [load_tree_table\(\)](#), [main\(\)](#), [read_output_snaps\(\)](#), [save_galaxies\(\)](#), [save_galaxy_tree\(\)](#), [starformation_and_feedback\(\)](#), [tsudy\(\)](#), [write_galaxy_data_snap\(\)](#), and [write_galaxy_for_momaf\(\)](#).

4.5.2.66 float MagTableZz[NMAG][IZZ][IZ][ZL_LEN]

Definition at line 146 of file [allvars.c](#).

Referenced by [add_to_luminosities\(\)](#), and [setup_spectrophotometric_model\(\)](#).

4.5.2.67 int MaxGals

Definition at line [23](#) of file [allvars.c](#).

Referenced by [load_tree\(\)](#), and [main\(\)](#).

4.5.2.68 size_t maxstorage_galaxydata

Definition at line [169](#) of file [allvars.c](#).

Referenced by [load_all_treedata\(\)](#), and [myfwrite\(\)](#).

4.5.2.69 size_t maxstorage_galsnapdata[NOUT]

Definition at line [170](#) of file [allvars.c](#).

Referenced by [load_all_treedata\(\)](#), and [myfwrite\(\)](#).

4.5.2.70 size_t maxstorage_momafdata[NOUT]

Definition at line [173](#) of file [allvars.c](#).

Referenced by [load_all_treedata\(\)](#), and [myfwrite\(\)](#).

4.5.2.71 float metalcold

Definition at line [183](#) of file [allvars.c](#).

4.5.2.72 int MetallicityOption

Definition at line [78](#) of file [allvars.c](#).

Referenced by [add_to_luminosities\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.73 long mu_seed

Definition at line [151](#) of file [allvars.c](#).

Referenced by [main\(\)](#), and [tsudy\(\)](#).

4.5.2.74 int Nids

Definition at line [155](#) of file [allvars.c](#).

Referenced by [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.75 int NTask

Definition at line 54 of file [allvars.c](#).

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

4.5.2.76 int NtotHalos

Definition at line 155 of file [allvars.c](#).

Referenced by [load_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.77 int Ntrees

Definition at line 25 of file [allvars.c](#).

Referenced by [finalize_galaxy_file\(\)](#), [load_tree\(\)](#), [load_tree_table\(\)](#), [main\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.78 int NumGals

Definition at line 26 of file [allvars.c](#).

Referenced by [main\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.79 int NumMergers

Definition at line 72 of file [allvars.c](#).

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

4.5.2.80 size_t offset_auxdata

Definition at line 168 of file [allvars.c](#).

Referenced by [load_all_auxdata\(\)](#), [load_tree_table\(\)](#), [myfread\(\)](#), [myfseek\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.81 size_t offset_dbids

Definition at line 168 of file [allvars.c](#).

Referenced by [load_all_dbids\(\)](#), [load_tree\(\)](#), [myfread\(\)](#), and [myfseek\(\)](#).

4.5.2.82 size_t offset_galaxydata

Definition at line 169 of file [allvars.c](#).

Referenced by [finalize_galaxy_file\(\)](#), [load_all_treedata\(\)](#), [myfseek\(\)](#), [myfwrite\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.83 size_t offset_galsnapdata[NOUT]

Definition at line 170 of file [allvars.c](#).

Referenced by [finalize_galaxy_file\(\)](#), [load_all_treedata\(\)](#), [myfseek\(\)](#), [myfwrite\(\)](#), and [save_galaxies\(\)](#).

4.5.2.84 size_t offset_momafdata[NOUT]

Definition at line 173 of file [allvars.c](#).

Referenced by [finalize_momaf_file\(\)](#), [load_all_treedata\(\)](#), [myfseek\(\)](#), [myfwrite\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.85 size_t offset_treedata

Definition at line 168 of file [allvars.c](#).

Referenced by [load_all_treedata\(\)](#), [load_tree\(\)](#), [load_tree_table\(\)](#), [myfread\(\)](#), and [myfseek\(\)](#).

4.5.2.86 int OffsetIDs

Definition at line 155 of file [allvars.c](#).

Referenced by [get_coordinates\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.87 int * OffsetIDs_halo

Definition at line 156 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [get_coordinates\(\)](#), and [load_tree_table\(\)](#).

4.5.2.88 int * OffsetIDs_snaptree

Definition at line 156 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [load_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.89 double Omega

Definition at line 66 of file [allvars.c](#).

Referenced by [do_reionization\(\)](#), [get_virial_radius\(\)](#), [integrand_time_to_present\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.90 double OmegaLambda

Definition at line 67 of file [allvars.c](#).

Referenced by [get_virial_radius\(\)](#), [integrand_time_to_present\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.91 FILE* output_file

Definition at line 200 of file [allvars.c](#).

Referenced by [closeallfiles\(\)](#), [finalize_galaxy_file\(\)](#), [openallfiles\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.92 char OutputDir[512]

Definition at line 31 of file [allvars.c](#).

Referenced by [finalize_galaxy_file\(\)](#), [finalize_momaf_file\(\)](#), [load_tree_table\(\)](#), [main\(\)](#), [open_outputtree_file\(\)](#), [read_parameter_file\(\)](#), [save_galaxies\(\)](#), [save_galaxy_tree\(\)](#), [write_all_galaxy_data\(\)](#), [write_galaxy_data_snap\(\)](#), and [write_galaxy_for_momaf\(\)](#).

4.5.2.93 double PartMass

Definition at line 69 of file [allvars.c](#).

Referenced by [get_virial_mass\(\)](#), [hot_retain_sat\(\)](#), [read_parameter_file\(\)](#), [update_hot_frac\(\)](#), [update_hotgas\(\)](#), and [update_type_1\(\)](#).

4.5.2.94 char PhotDir[512]

Definition at line 28 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), [read_tsud_tables\(\)](#), and [setup_spectrophotometric_model\(\)](#).

4.5.2.95 char PhotPrefix[50]

Definition at line 29 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), [read_tsud_tables\(\)](#), and [setup_spectrophotometric_model\(\)](#).

4.5.2.96 float* PosList

Definition at line 158 of file [allvars.c](#).

Referenced by [get_coordinates\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.97 char* ptr_auxdata

Definition at line 167 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [load_all_auxdata\(\)](#), and [myfread\(\)](#).

4.5.2.98 char * ptr_dbids

Definition at line 167 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [load_all_dbids\(\)](#), and [myfread\(\)](#).

4.5.2.99 char * ptr_galaxydata

Definition at line 167 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [load_all_treedata\(\)](#), [myfwrite\(\)](#), and [write_all_galaxy_data\(\)](#).

4.5.2.100 char * ptr_galsnapdata[NOUT]

Definition at line 167 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [load_all_treedata\(\)](#), [myfwrite\(\)](#), and [write_galaxy_data_snap\(\)](#).

4.5.2.101 char* ptr_momafdata[NOUT]

Definition at line 172 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [load_all_treedata\(\)](#), [myfwrite\(\)](#), and [write_galaxy_for_momaf\(\)](#).

4.5.2.102 char * ptr_treedata

Definition at line 167 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [load_all_treedata\(\)](#), and [myfread\(\)](#).

4.5.2.103 gsl_rng* random_generator

Definition at line 135 of file [allvars.c](#).

Referenced by [init\(\)](#), [init_galaxy\(\)](#), and [main\(\)](#).

4.5.2.104 float RecGas

Definition at line 181 of file [allvars.c](#).

4.5.2.105 double RecycleFraction

Definition at line 102 of file [allvars.c](#).

Referenced by [add_to_luminosities\(\)](#), [do_major_merger_starburst\(\)](#), [read_parameter_file\(\)](#), and [update_from_star_formation\(\)](#).

4.5.2.106 float RedshiftTab[IZ]

Definition at line 145 of file [allvars.c](#).

Referenced by [find_interpolated_obsลม\(\)](#), and [setup_spectrophotometric_model\(\)](#).

4.5.2.107 double ReheatPreVelocity

Definition at line 96 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.5.2.108 double ReheatSlope

Definition at line 97 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.5.2.109 double ReIncorporationFactor

Definition at line 105 of file [allvars.c](#).

Referenced by [main\(\)](#), [read_parameter_file\(\)](#), and [reincorporate_gas\(\)](#).

4.5.2.110 int ReIncorporationRecipe

Definition at line 91 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), and [reincorporate_gas\(\)](#).

4.5.2.111 float Reion_Mc[46]

Definition at line 192 of file [allvars.c](#).

Referenced by [do_reionization\(\)](#), and [read_reionization\(\)](#).

4.5.2.112 float Reion_z[46]

Definition at line 192 of file [allvars.c](#).

Referenced by [find_interpolate_reionization\(\)](#), and [read_reionization\(\)](#).

4.5.2.113 double Reionization_z0

Definition at line 114 of file [allvars.c](#).

Referenced by [init\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.114 double Reionization_zr

Definition at line 115 of file [allvars.c](#).

Referenced by [init\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.115 int ReionizationOn

Definition at line 81 of file [allvars.c](#).

Referenced by [cooling_recipe\(\)](#), [infall_recipe\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.116 float Rho[RHO_LEN]

Definition at line 188 of file [allvars.c](#).

Referenced by [find_interpolate_h2\(\)](#), and [read_sfrz\(\)](#).

4.5.2.117 double RhoCrit

Definition at line 118 of file [allvars.c](#).

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

4.5.2.118 int SatelliteRecipe

Definition at line 83 of file [allvars.c](#).

Referenced by [infall_recipe\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.119 double SfrAlpha

Definition at line 95 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.5.2.120 double SfrEfficiency

Definition at line 109 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.5.2.121 double SfrLawPivotVelocity

Definition at line 93 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.5.2.122 double SfrLawSlope

Definition at line 94 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.5.2.123 char SimulationDir[512]

Definition at line 33 of file [allvars.c](#).

Referenced by [load_all_auxdata\(\)](#), [load_all_dbids\(\)](#), [load_all_treedata\(\)](#), [load_tree\(\)](#), [load_tree_table\(\)](#), [open_tree_file\(\)](#), [open_treeaux_file\(\)](#), [open_treedbids_file\(\)](#), [read_parameter_file\(\)](#), and [save_galaxies\(\)](#).

4.5.2.124 int Snaplistlen

Definition at line 133 of file [allvars.c](#).

Referenced by [init\(\)](#), and [read_snap_list\(\)](#).

4.5.2.125 double SNinReheat

Definition at line 100 of file [allvars.c](#).

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

4.5.2.126 int StarBurstRecipe

Definition at line [84](#) of file [allvars.c](#).

Referenced by [deal_with_galaxy_merger\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.127 int StarBurstsInMajorMergersOn

Definition at line [76](#) of file [allvars.c](#).

Referenced by [do_major_merger_starburst\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.128 int StarFormationRecipe

Definition at line [82](#) of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.5.2.129 int StelliteRecipe

4.5.2.130 int ThisTask

Definition at line [54](#) of file [allvars.c](#).

Referenced by [init\(\)](#), [main\(\)](#), [read_cooling_functions\(\)](#), [read_parameter_file\(\)](#), [read_snap_list\(\)](#), and [setup_spectrophotometric_model\(\)](#).

4.5.2.131 double ThreshMajorMerger

Definition at line [107](#) of file [allvars.c](#).

Referenced by [bulgesize_from_merger\(\)](#), [collisional_starburst_recipe\(\)](#), [deal_with_galaxy_merger\(\)](#), and [read_parameter_file\(\)](#).

4.5.2.132 int TotGalaxies[NOUT]

Definition at line [43](#) of file [allvars.c](#).

Referenced by [finalize_galaxy_file\(\)](#), [finalize_momaf_file\(\)](#), [load_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.133 int TotGalCount

Definition at line [60](#) of file [allvars.c](#).

Referenced by [finalize_galaxy_file\(\)](#), [load_tree_table\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.134 int TotHalos

Definition at line [42](#) of file [allvars.c](#).

4.5.2.135 int TotIds

Definition at line 155 of file [allvars.c](#).

Referenced by [load_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.136 int TotSnaps

Definition at line 155 of file [allvars.c](#).

Referenced by [load_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.137 int TrackDiskInstability

Definition at line 88 of file [allvars.c](#).

Referenced by [deal_with_galaxy_merger\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.5.2.138 FILE* tree_file

Definition at line 197 of file [allvars.c](#).

Referenced by [closeallfiles\(\)](#), [load_tree\(\)](#), [load_tree_table\(\)](#), and [openallfiles\(\)](#).

4.5.2.139 FILE* treeaux_file

Definition at line 198 of file [allvars.c](#).

Referenced by [closeallfiles\(\)](#), [load_tree_table\(\)](#), [openallfiles\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.140 FILE* treedbids_file

Definition at line 199 of file [allvars.c](#).

Referenced by [closeallfiles\(\)](#), [load_tree\(\)](#), and [openallfiles\(\)](#).

4.5.2.141 int* TreeFirstHalo

Definition at line 51 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [get_coordinates\(\)](#), [load_tree\(\)](#), and [load_tree_table\(\)](#).

4.5.2.142 int* TreeNgals[NOUT]

Definition at line 44 of file [allvars.c](#).

Referenced by [finalize_galaxy_file\(\)](#), [free_tree_table\(\)](#), [load_tree_table\(\)](#), and [save_galaxies\(\)](#).

4.5.2.143 int* TreeNHalos

Definition at line 50 of file [allvars.c](#).

Referenced by [free_tree_table\(\)](#), [load_tree\(\)](#), [load_tree_table\(\)](#), and [main\(\)](#).

4.5.2.144 double UnitCoolingRate_in_cgs

Definition at line 118 of file [allvars.c](#).

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

4.5.2.145 double UnitDensity_in_cgs

Definition at line 118 of file [allvars.c](#).

Referenced by [cooling_recipe\(\)](#), and [set_units\(\)](#).

4.5.2.146 double UnitEnergy_in_cgs

Definition at line 118 of file [allvars.c](#).

Referenced by [do_AGN_heating\(\)](#), [init\(\)](#), and [set_units\(\)](#).

4.5.2.147 double UnitLength_in_cm

Definition at line 118 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), and [set_units\(\)](#).

4.5.2.148 double UnitMass_in_g

Definition at line 118 of file [allvars.c](#).

Referenced by [do_AGN_heating\(\)](#), [init\(\)](#), [prepare_galaxy_for_output\(\)](#), [read_parameter_file\(\)](#), and [set_units\(\)](#).

4.5.2.149 double UnitPressure_in_cgs

Definition at line 118 of file [allvars.c](#).

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

4.5.2.150 double UnitTime_in_Megayears

Definition at line 118 of file [allvars.c](#).

Referenced by [add_to_luminosities\(\)](#), [prepare_galaxy_for_output\(\)](#), [set_units\(\)](#), and [setup_spectrophotometric_model\(\)](#).

4.5.2.151 double UnitTime_in_s

Definition at line 118 of file [allvars.c](#).

Referenced by [cooling_recipe\(\)](#), [do_AGN_heating\(\)](#), [init\(\)](#), [prepare_galaxy_for_output\(\)](#), and [set_units\(\)](#).

4.5.2.152 double UnitVelocity_in_cm_per_s

Definition at line 118 of file [allvars.c](#).

Referenced by [read_parameter_file\(\)](#), and [set_units\(\)](#).

4.5.2.153 float * VelList

Definition at line 158 of file [allvars.c](#).

Referenced by [get_coordinates\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.5.2.154 double Yield

Definition at line 103 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [do_major_merger_starburst\(\)](#), [read_parameter_file\(\)](#), and [starformation_and_feedback\(\)](#).

4.5.2.155 double ZZ[MAXSNAPS]

Definition at line 129 of file [allvars.c](#).

Referenced by [collisional_starburst_recipe\(\)](#), [do_major_merger_starburst\(\)](#), [estimate_merging_time\(\)](#), [finalize_galaxy_file\(\)](#), [get_virial_radius\(\)](#), [hot_retain_sat\(\)](#), [infall_recipe\(\)](#), [init\(\)](#), [load_tree_table\(\)](#), [main\(\)](#), [peri_radius\(\)](#), [prepare_galaxy_for_momaf\(\)](#), [prepare_galaxy_for_output\(\)](#), [RedshiftObs\(\)](#), [reincorporate_gas\(\)](#), [save_galaxies\(\)](#), [starformation_and_feedback\(\)](#), [tsudy\(\)](#), [update_from_feedback\(\)](#), [update_hotgas\(\)](#), and [write_galaxy_data_snap\(\)](#).

4.6 code/allvars.h

```

00001 /**
00002  * @brief TODO add description for all variables that do not have one yet.
00003  * bla bla bla.
00004 */
00005 #ifndef ALLVARS_H
00006 #define ALLVARS_H
00007
00008
00009
00010 #include <stdio.h>
00011 #include <gsl/gsl_rng.h>
00012
00013 #define min(x,y) ((x)<(y) ?(x):(y)) // TODO dmin in cool_func.c does the same
00014
00015 #define NMAG 5           /* Number of filters */
00016 #define NIDX 10
00017
00018 #ifdef MRII
00019 #define MAXSNAPS 68      /* Number of snapshots in the dark matter simulation */
00020
00021 */                      /* SIMULATION DEPENDENT!!!  INSERT APPROPRIATE NUMBER */
00022 #else
00023 #define MAXSNAPS 64
00024 #endif
00025 #define MAXGALFAC 2.3     /* previous value 2.3 - maximum fraction of satellite w

```

```

        ithout a halo (for memory allocation) */
00025 #define STEPS 20           /* Number of integration intervals between two snapshot
      s */
00026
00027
00028 #define ALLOCPARAMETER 50. /* new definition !!! THIS HAS TO BE 50 !!! DONT EVE
      R EVER EVER CHANGE !!! */
00029
00030 //To understand the units in the code read through set_units in init.c!!!
00031 #define GRAVITY    6.672e-8
00032 #define SOLAR_MASS 1.989e33
00033 #define SOLAR_LUM  3.826e33
00034 #define RAD_CONST 7.565e-15
00035 #define AVOGADRO  6.0222e23
00036 #define BOLTZMANN 1.3806e-16
00037 #define GAS_CONST 8.31425e7
00038 #define C         2.9979e10
00039 #define PLANCK    6.6262e-27
00040 #define CM_PER_MPC 3.085678e24 // TODO this is read in from input.par
00041 #define PROTONMASS 1.6726e-24
00042 #define HUBBLE    3.2407789e-18 /* in h/sec */
00043
00044 //To understand the units in the code read through set_units in init.c!!!
00045 #define SEC_PER_MEGAYEAR 3.155e13 //Normally referred as a million years....
00046 #define SEC_PER_YEAR   3.155e7
00047
00048
00049 /* Added for dust implementation (Gabriella)      */
00050
00051 #define tauBstar 0.8
00052 #define beta 0.5
00053 #define Lbstar 3.7e8 /*In solar luminosity is 5.75e10 */
00054 #define mu_dust (1/3.)
00055
00056 /* extinction curve of Cardelli et al. (1989) */
00057 //
00058 //##define Ab_Av 1.337
00059 //##define Ar_Av 0.751
00060 //##define Ai_Av 0.479
00061 //##define Ak_Av 0.114
00062 //
00063 /* and for SDSS bands */
00064 //
00065 //##define Au_Av_SDSS 1.577
00066 //##define Ag_Av_SDSS 1.224
00067 //##define Ar_Av_SDSS 0.877
00068 //##define Ai_Av_SDSS 0.672
00069 //##define Az_Av_SDSS 0.491
00070
00071
00072 #ifdef GALAXYTREE
00073 #undef NOUT
00074 #define NOUT MAXSNAPS
00075 #endif
00076
00077 /**
00078 * Galaxy structure for output
00079 */
00080 struct GALAXY_OUTPUT
00081 {
00082 #ifndef NO_PROPS_OUTPUTS
00083 #ifdef GALAXYTREE
00084     long long GalID; /** ID of galaxy, unique within simulation and SAM run.*/
00085     long long HaloID; // Unique ID of MPA halo containing this galaxy
00086     long long FirstProgGal; // Main progenitor of this galaxy. Also the first proge
          nitor in a linked list representation of the merger tree.
00087     long long NextProgGal; // Next progenitor of this galaxy in linked list represe

```

```

ntation of merger tree
00088 long long LastProgGal; // Galaxies with id between this galaxyId and this lastP
rogenitorId form the merger tree rooted in this galaxy.
00089 long long FOFCentralGal; // TODO has been coded, but that code must be tested.
00090 long long FileTreeNr;
00091 long long DescendantGal; // Pointer to the descendant of this galaxy in its mer
ger tree; -1 if there is no descendant
00092 long long MainLeafId;
00093 long long TreeRootId;
00094 long long SubID; //
00095 long long MMSubID; // fofId, the subhaloid of the subhalo at the center of the
fof group
00096 int PeanoKey; // Peano-Hilbert key, (bits=8), for position in 500/h Mpc box
00097 float Redshift; // redshift of the snapshot where this galaxy resides
00098 #endif
00099 int Type; // Galaxy type: 0 for central galaxies of a main halo, 1 for centra
l galaxies in sub-halos, 2 for satellites without halo.
00100 #ifndef GALAXYTREE
00101 int HaloIndex;
00102 #endif
00103 int SnapNum; // The snapshot number where this galaxy was identified.
00104 float CentralMvir; // 10^10/h Msun virial mass of background (FOF) halo contain
ing this galaxy
00105 /* properties of subhalo at the last time this galaxy was a central galaxay */
00106 float Pos[3]; // 1/h Mpc - Galaxy Positions
00107 float Vel[3]; // km/s - Galaxy Velocities
00108 int Len;
00109 float Mvir; // 10^10/h Msun - Virial mass of the subhalo the galaxy is/was the
center of.
00110 float Rvir; // Mpc/h - Virial radius of the subhalo the galaxy is/was the cente
r of.
00111 float Vvir; // km/s - Virial velocity of the subhalo the galaxy is/was the cent
er of.
00112 float Vmax; // km/s - Maximum rotational velocity of the subhalo, or the last v
alue for type 2's galaxies.
00113 float GasSpin[3]; // Gas Spin
00114 float StellarSpin[3]; // Stellar Spin
00115 float InfallVmax; // km/s - Vmax at infall
00116 int InfallSnap; // Snapnum at infall
00117 float HotRadius; // Mpc/h - Radius of the hot gas
00118 /*dynamical friction merger time*/
00119 float OriMergTime;
00120 float MergTime;
00121 /* baryonic reservoirs */
00122 float ColdGas; // 10^10/h Msun - Mass in cold gas.
00123 float StellarMass; // 10^10/h Msun - Mass in stars
00124 float BulgeMass; // 10^10/h Msun - Mass in the bulge
00125 float HotGas; // 10^10/h Msun - Mass in hot gas
00126 float EjectedMass; // 10^10/h Msun - Mass in ejected gas
00127 float BlackHoleMass; // 10^10/h Msun - Mass in black hole
00128 /* ICL magnitude and mass*/
00129 float ICM; // mass in intra-cluster stars, for type 0,1
00130 float MetalsColdGas; // 10^10/h Msun - Mass in metals in cold gas.
00131 float MetalsStellarMass; // 10^10/h Msun - Mass in metals in stars
00132 float MetalsBulgeMass; // 10^10/h Msun - Mass in metals in the bulge
00133 float MetalsHotGas; // 10^10/h Msun - Mass in metals in the hot gas
00134 float MetalsEjectedMass; // 10^10/h Msun - Mass in metals in the ejected gas

00135 float MetalsICM; // total mass in metals in intra-cluster stars, for type 0,1
00136 /* misc */
00137 #ifdef UseFullSfr // Note: do not use this when running for database
00138 float Sfr[MAXSNAPS];
00139 float SfrBulge[MAXSNAPS];
00140 #else
00141 float Sfr;
00142 float SfrBulge;
00143 #endif

```

```

00144     float XrayLum;
00145     float BulgeSize;
00146     float StellarDiskRadius;
00147     float GasDiskRadius;
00148     int DisruptOn; // 0: galaxy merged onto merger center; 1: galaxy was disrupted
00149     before merging onto its descendant, matter went into ICM of merger center
00150     #ifdef MERGE01
00151     int MergeOn; // 0: standard delucia-like merger behaviour for type 1 galaxy
00152     ; 1: galaxy mass > halo mass, separate dynamical friction time calculated ....
00153     #endif
00154     float CoolingRadius; // Q: store this ? (was stored in Delucia20006a)
00155     #endif // NO_PROPS_OUTPUTS
00156     /* magnitudes in various bands */
00157     #ifdef OUTPUT_REST_MAGS
00158         float Mag[NMAG]; // rest-frame absolute mags
00159         float MagBulge[NMAG]; // rest-frame absolute mags for the bulge
00160         float MagDust[NMAG]; // dust corrected, rest-frame absolute mags
00161         float MassWeightAge; // Q: store? does this belong inside #ifdef OUTPUT_REST_M
00162         AGS? If not, check rest of code it is set outside of this.
00163     #endif
00164     #endif
00165
00166     #ifdef OUTPUT_OBS_MAGS
00167         float ObsMag[NMAG]; // obs-frame absolute mags
00168         float ObsMagBulge[NMAG]; // obs-frame absolute mags for the bulge
00169         float ObsMagDust[NMAG]; // dust-corrected, obs-frame absolute mags
00170
00171     // one would expect these to be important only for MOMAF outputs, but then need to
00172     // change code elsewhere as well.
00173     #ifdef OUTPUT_MOMAF_INPUTS
00174         float dObsMag[NMAG];
00175         float dObsMagBulge[NMAG];
00176     #endif
00177     #ifdef ICL
00178         float ObsMagICL[NMAG]; // observer-frame absolute mags for intra-cluster light
00179     #ifdef OUTPUT_MOMAF_INPUTS
00180         float dObsMagICL[NMAG]; // -frame absolute mags
00181     #endif
00182     #endif
00183     #endif
00184 };
00185
00186 // TODO add documentation, also for all fields
00187 #ifdef OUTPUT_MOMAF_INPUTS
00188     struct MOMAF_INPUTS
00189     {
00190         long long GalID;
00191         long long HaloID;
00192         int SnapNum;
00193         float Pos[3];
00194         float Vel[3];
00195         float ObsMagBulge[NMAG];
00196         float ObsMagDust[NMAG];
00197         float dObsMagBulge[NMAG];
00198         float dObsMagDust[NMAG];
00199     };
00200 #endif
00201
00202 /*Structure with all the data associated with galaxies (this is not the same as the output!)*/
00203     struct GALAXY /* Galaxy data */
00204 {

```

```
00205 #ifdef GALAXYTREE
00206     int   GalID;
00207     int   FirstProgGal;
00208     int   NextProgGal;
00209     int   LastProgGal;
00210     int   DescendantGal;
00211     int   MainLeaf;
00212     int   TreeRoot;
00213     int   Done;
00214 #endif
00215     int   Type;
00216     int   HaloNr;
00217     long long MostBoundID;
00218     int   SnapNum;
00219     int   CentralGal;
00220     float CentralMvir;
00221     /* properties of subhalo at the last time this galaxy was a central galaxy */
00222     float Pos[3];
00223     float MergCentralPos[3];
00224     float Vel[3];
00225     float HaloSpin[3];
00226     float GasSpin[3];
00227     float StellarSpin[3];
00228     int   Len;
00229     float Mvir;
00230     float Rvir;
00231     float Vvir;
00232     float Vmax;
00233     float InfallVmax;
00234     /*ram pressure*/
00235     float InfallSnap;
00236     float CoolingGas;
00237     float HotFrac;
00238     float HotRadius;
00239     /* baryonic reservoirs */
00240     float ColdGas;
00241     float StellarMass;
00242     float BulgeMass;
00243     float HotGas;
00244     float EjectedMass;
00245     float BlackHoleMass;
00246     /* metals (a la Gabriella) */
00247     float MetalsColdGas;
00248     float MetalsStellarMass;
00249     float MetalsBulgeMass;
00250     float MetalsHotGas;
00251     float MetalsEjectedMass;
00252
00253     /* misc */
00254 #ifdef UseFullSfr
00255     float Sfr[MAXSNAPS];
00256     float SfrBulge[MAXSNAPS];
00257 #else
00258 #ifdef SAVE_MEMORY
00259     float Sfr;
00260     float SfrBulge;
00261 #else
00262     float Sfr[NOUT];
00263     float SfrBulge[NOUT];
00264 #endif
00265 #endif
00266     float StarMerge;
00267     float XrayLum;
00268     float BulgeSize;
00269     float StellarDiskRadius;
00270     float GasDiskRadius;
00271 #ifdef GALAXYTREE
```

```

00272     int DisruptOn;
00273 #endif
00274     // float halfradius;
00275     //float periradius;
00276     float Inclination;
00277     float OriMergTime;
00278     float MergeSat;
00279     float MergTime;
00280     float MergeOn;
00281     float CoolingRadius;
00282     /* luminosities in various bands */
00283 #ifdef OUTPUT_REST_MAGS
00284 #ifdef ICL
00285     float ICLLum[NMAG][NOUT];
00286 #endif
00287     float Lum[NMAG][NOUT];
00288     float YLum[NMAG][NOUT];
00289     float LumBulge[NMAG][NOUT];
00290     float YLumBulge[NMAG][NOUT];
00291     float LumDust[NMAG][NOUT];
00292     float MassWeightAge[NOUT];
00293
00294 #endif
00295     float ICM;
00296     float MetalsICM;
00297 #ifdef COMPUTE_OBS_MAGS
00298     float ObsLum[NMAG][NOUT];
00299     float ObsYLum[NMAG][NOUT];
00300     float ObsLumBulge[NMAG][NOUT];
00301     float ObsYLumBulge[NMAG][NOUT];
00302     float ObsLumDust[NMAG][NOUT];
00303 #ifdef ICL
00304     float ObsICL[NMAG][NOUT];
00305 #endif
00306     /* NB: it would seem we can put this here only ifdef OUTPUT_MOMAF_INPUTS,
00307      * but this requires code elsewhere to be updated as well, eg recipe_dust etc.
00308 */
00309 #ifdef OUTPUT_MOMAF_INPUTS
00310     float dObsLum[NMAG][NOUT];
00311     float dObsYLum[NMAG][NOUT];
00312     float dObsLumBulge[NMAG][NOUT];
00313     float dObsYLumBulge[NMAG][NOUT];
00314     float dObsLumDust[NMAG][NOUT];
00315 #endif
00316 }
00317 *Gal, *HaloGal;
00318
00319
00320 // TODO add documentation, also for all fields
00321 struct halo_data
00322 {
00323     /* merger tree pointers */
00324     int Descendant;
00325     int FirstProgenitor;
00326     int NextProgenitor;
00327     int FirstHaloInFOFgroup;
00328     int NextHaloInFOFgroup;
00329
00330     /* properties of halo */
00331     int Len;
00332     float M_Mean200, M_Crit200, M_TopHat;
00333     float Pos[3];
00334     float Vel[3];
00335     float VelDisp;
00336     float Vmax;
00337     float Spin[3];

```

```

00338     long long MostBoundID;
00339
00340     /* original position in subfind output */
00341     int SnapNum;
00342     int FileNr;
00343     int SubhaloIndex;
00344     float SubHalfMass;
00345 }
00346 *Halo;
00347
00348
00349 // TODO add documentation, also for all fields
00350 extern struct halo_ids_data
00351 {
00352     long long HaloID;
00353     long long FileTreeNr;
00354     long long FirstProgenitor;
00355     long long LastProgenitor;
00356     long long NextProgenitor;
00357     long long Descendant;
00358     long long FirstHaloInFOFgroup;
00359     long long NextHaloInFOFgroup;
00360 #ifdef MRII
00361     long long MainLeafID;
00362 #endif
00363     double Redshift;
00364     int PeanoKey;
00365     int dummy;      /* need to use this padding for 64bit alignment */
00366 } *HaloIDs;
00367
00368
00369
00370 // TODO add documentation, also for all fields
00371 struct halo_aux_data /* auxiliary halo data */
00372 {
00373     int DoneFlag;
00374     int HaloFlag;
00375     int NGalaxies;
00376     int FirstGalaxy;
00377 }
00378 *HaloAux;
00379
00380
00381 extern int FirstFile; /* first and last file for processing */
00382 extern int LastFile;
00383
00384 extern int Ntrees;    /* number of trees in current file */
00385 extern int NumGals;   /* Total number of galaxies stored for current tree */
00386 extern int MaxGals;   /* Maximum number of galaxies allowed for current tree
   */
00387 extern int FoF_MaxGals;
00388
00389 extern int FilesPerSnapshot;
00390 extern int LastSnapShotNr;
00391
00392 extern char PhotDir[512];
00393 extern char PhotPrefix[50];
00394 extern char CoolFunctionsDir[512];
00395 extern char OutputDir[512];
00396 extern char FileNameGalaxies[512];
00397 extern char SimulationDir[512];
00398 extern char FileWithOutputSamps[512];
00399 extern char FileWithSnapList[512];
00400
00401 #ifdef SPECIFYFILENR
00402 extern char FileNrDir[512];
00403 extern int ListInputFilrNr[111];

```

```
00404 #endif
00405
00406 extern int TotHalos;
00407 extern int TotGalaxies[NOUT];
00408 extern int *TreeNgals[NOUT];
00409
00410 extern int *FirstHaloInSnap;
00411
00412 extern int *TreeNHalos;
00413 extern int *TreeFirstHalo;
00414
00415 #ifdef PARALLEL
00416 extern int ThisTask, NTask;
00417#endif
00418
00419 #ifdef GALAXYTREE
00420 extern int GalCount;
00421 extern int TotGalCount;
00422#endif
00423
00424 /* more misc */
00425 extern double Omega;
00426 extern double OmegaLambda;
00427 extern double PartMass;
00428 extern double Hubble_h;
00429 extern double EnergySNcode, EnergySN;
00430 extern double EtaSNcode, EtaSN;
00431 extern int NumMergers;
00432
00433
00434 /* recipe flags */
00435 extern int StarBurstsInMajorMergersOn;
00436 extern int BulgeFormationInMinorMergersOn;
00437 extern int MetallicityOption;
00438 extern int EjectionOn;
00439 extern int CoolingCutoff;
00440 extern int ReionizationOn;
00441 extern int StarFormationRecipe;
00442 extern int StelliteRecipe;
00443 extern int StarBurstRecipe;
00444 extern int FeedbackRecipe;
00445 extern int EjectionRecipe;
00446 extern int DiskRadiusMethod;
00447 extern int TrackDiskInstability;
00448 extern int AGNrecipeOn;
00449 extern int SatelliteRecipe;
00450 extern int ReIncorporationRecipe;
00451 /* recipe parameters */
00452 extern double SfrLawPivotVelocity;
00453 extern double SfrLawSlope;
00454 extern double SfrAlpha;
00455 extern double ReheatPreVelocity;
00456 extern double ReheatSlope;
00457 extern double SNinReheat;
00458 extern double FeedbackEpsilon;
00459 extern double EjectPreVelocity;
00460 extern double EjectSlope;
00461 extern double RecycleFraction;
00462 extern double Yield;
00463 extern double FracZtoHot;
00464 extern double ReIncorporationFactor;
00465 extern double CoolingVelocityCutOff;
00466 extern double ThreshMajorMerger;
00467 extern double BaryonFrac;
00468 extern double SfrEfficiency;
00469 extern double FeedbackReheatingEpsilon;
00470 extern double FeedbackEjectionEfficiency;
```

```

00471 extern double AgnEfficiency;
00472 extern double BlackHoleGrowthRate;
00473 extern double Reionization_z0;
00474 extern double Reionization_zr;
00475
00476
00477 extern double UnitLength_in_cm,
00478     UnitTime_in_s,
00479     UnitVelocity_in_cm_per_s,
00480     UnitMass_in_g,
00481     RhoCrit,
00482     UnitPressure_in_cgs,
00483     UnitDensity_in_cgs,
00484     UnitCoolingRate_in_cgs,
00485     UnitEnergy_in_cgs,
00486     UnitTime_in_Megayears,
00487     G,
00488     Hubble,
00489     a0, ar;
00490
00491 extern int ListOutputSamps[NOUT];
00492
00493 extern double ZZ[MAXSNAPS];
00494 extern double AA[MAXSNAPS];
00495 extern double Age[MAXSNAPS];
00496
00497 extern int Snaplistlen;
00498
00499 extern gsl_rng *random_generator;
00500
00501
00502
00503
00504 /* tabulated stuff */
00505
00506 /* fixed-metallicity spectrophotometric model */
00507 /* tables hold magnitudes of starburst population as a function of age */
00508
00509 #define ZL_LEN 221
00510 #define IZZ 6
00511 #define IZ 64 /* Number of redshifts used in the tables */
00512
00513
00514
00515 #define RHO_LEN 101
00516 #define Z_LEN 13
00517
00518 extern float AgeTab[ZL_LEN];
00519
00520
00521 /* Variables to hold metallicity and redshift dependent tables (Gabriella) */
00522 extern float RedshiftTab[IZ];
00523 extern float MagTableZz[NMAG][IZZ][IZ][ZL_LEN];
00524 extern float IdxTable[2*NIDX][IZZ][ZL_LEN];
00525 extern int IdxType[NIDX];
00526 extern float Idxw5[NIDX];
00527 extern float Idxw6[NIDX];
00528
00529
00530 // tsud
00531 /* Tabulated values for new dust correction */
00532 #define NZZ_EXT 100 // number of (gas-)metallcity bins for tabulated extinction

00533 #define lzzmin -3 // (log of) minimun metallicity of gas for extinction (in solar unit)
00534 #define lzzmax 1 // .... maximum .....
00535 extern float DeltaM[NMAG+1][NZZ_EXT][IZ]; // + 1 for V-band always tabulated ...

```

```
00536 extern float Corrections[NMAG][IZ];
00537
00538 #define ExpTauBCBulge 0.5 // constant extinction for young stars in bulges.
00539 #define MUWIDTH 0.2
00540 #define MUCENTER 0.3
00541 extern long mu_seed;
00542
00543
00544 #ifdef UPDATETYPE TWO
00545 extern int NtotHalos, TotIds, Nids, TotSamps, OffsetIDs;
00546 extern int *CountIDs_halo, *OffsetIDs_halo, *CountIDs_snaptree, *
    OffsetIDs_snaptree;
00547 extern long long *IdList;
00548 extern float *PosList, *VelList;
00549#endif
00550
00551 #ifdef GALAXYTREE
00552 #define DOUBLE_to_HASHBITS(y) ((int)((*((long long *) &y)) & 0xFFFFFFFFFFFFFFllu)
    >> (52 - Hashbits))
00553 extern int Hashbits;
00554 extern double BoxSize;
00555#endif
00556
00557
00558 #ifdef USE_MEMORY_TO_MINIMIZE_IO
00559 extern char *ptr_auxdata, *ptr_treedata, *ptr_dbids, *ptr_galaxydata, *
    ptr_galsnapdata[NOUT];
00560 extern size_t offset_auxdata, offset_treedata, offset_dbids;
00561 extern size_t offset_galaxydata, maxstorage_galaxydata, filled_galaxydata;
00562 extern size_t offset_galsnapdata[NOUT], maxstorage_galsnapdata[NOUT],
    filled_galsnapdata[NOUT];
00563 #ifdef OUTPUT_MOMAF_INPUTS
00564 extern char *ptr_momafdata[NOUT];
00565 extern size_t offset_momafdata[NOUT], maxstorage_momafdata[NOUT], filled_momafdata[
    NOUT];
00566#endif
00567#endif
00568
00569#endif
00570
00571 #ifdef GASRECYCLE
00572 extern float RecGas;
00573 extern float Frac[ZL_LEN];
00574 extern float metalcold;
00575#endif
00576
00577 extern float Rho[RHO_LEN];
00578 extern float H2[RHO_LEN][Z_LEN];
00579
00580 extern float Reion_z[46], Reion_Mc[46];
00581
00582
00583 // ~ variables for option NEW_IO ~
00584 // define global file pointers and keep these open during running.
00585 // NB does not work for MOMAF outputs (yet)
00586 #ifdef NEW_IO
00587 extern FILE *tree_file;
00588 extern FILE *treeaux_file;
00589 extern FILE *treedbids_file;
00590 extern FILE *output_file;
00591 //extern FILE* momaf_file;
00592#endif
00593
00594
00595 extern int* GalaxiesInOrder;
00596
```

00597

4.7 code/cool_func.c File Reference

Functions

- void [read_cooling_functions](#) (void)
- double [get_metaldependent_cooling_rate](#) (double logTemp, double logZ)
- double [get_rate](#) (int tab, double logTemp)
- double [dmin](#) (double x, double y)
Returns the minimum of two numbers.
- void [test](#) (void)

Variables

- static char * [name](#) []
- static double [metallicities](#) [8]
- static double [CoolRate](#) [8][TABSIZE]

4.7.1 Function Documentation

4.7.1.1 double dmin (double x, double y)

Definition at line 143 of file [cool_func.c](#).

4.7.1.2 double get_metaldependent_cooling_rate (double logTemp, double logZ)

Definition at line 89 of file [cool_func.c](#).

References [get_rate\(\)](#), and [metallicities](#).

Referenced by [cooling_recipe\(\)](#), and [test\(\)](#).

4.7.1.3 double get_rate (int tab, double logTemp)

Definition at line 119 of file [cool_func.c](#).

References [CoolRate](#).

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

4.7.1.4 void read_cooling_functions (void)

Definition at line 41 of file [cool_func.c](#).

References [CoolFunctionsDir](#), [CoolRate](#), [metallicities](#), [name](#), and [ThisTask](#).

Referenced by [init\(\)](#), and [test\(\)](#).

4.7.1.5 void test (void)

Definition at line 153 of file [cool_func.c](#).

References [get_metaldependent_cooling_rate\(\)](#), and [read_cooling_functions\(\)](#).

4.7.2 Variable Documentation

4.7.2.1 double CoolRate[8][TABSIZE] [static]

Definition at line 38 of file [cool_func.c](#).

Referenced by [get_rate\(\)](#), and [read_cooling_functions\(\)](#).

4.7.2.2 double metallicities[8] [static]

Initial value:

```
{  
-5.0,  
-3.0,  
-2.0,  
-1.5,  
-1.0,  
-0.5,  
+0.0,  
+0.5  
}
```

Definition at line 27 of file [cool_func.c](#).

Referenced by [get_metaldependent_cooling_rate\(\)](#), and [read_cooling_functions\(\)](#).

4.7.2.3 char* name[] [static]

Initial value:

```
{  
"stripped_mzero.cie",  
"stripped_m-30.cie",  
"stripped_m-20.cie",  
"stripped_m-15.cie",  
"stripped_m-10.cie",  
"stripped_m-05.cie",  
"stripped_m-00.cie",  
"stripped_m+05.cie"  
}
```

Definition at line 13 of file [cool_func.c](#).

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

4.8 code/cool_func.c

```
00001 #include <stdio.h>
```



```
00069         CoolRate[i][n] = sd_logLnorm;
00070     }
00071
00072     fclose(fd);
00073 }
00074
00075 #ifdef PARALLEL
00076     if(ThisTask == 0)
00077         printf("cooling functions read.\n\n");
00078 #else
00079     printf("cooling functions read.\n\n");
00080 #endif
00081
00082 }
00083
00084
00085
00086
00087 /* pass: log10(temperatuue/Kelvin), log10(metallicity) */
00088
00089 double get_metaldependent_cooling_rate(double logTemp, double logZ)
00090 {
00091     int i;
00092     double get_rate(int tab, double logTemp);
00093     double rate1, rate2, rate;
00094
00095
00096     if(logZ < metallicities[0])
00097         logZ = metallicities[0];
00098
00099     if(logZ > metallicities[7])
00100         logZ = metallicities[7];
00101
00102     i = 0;
00103     while(logZ > metallicities[i + 1])
00104     {
00105         i++;
00106     }
00107     /* look up at i and i+1 */
00108
00109
00110     rate1 = get_rate(i, logTemp);
00111     rate2 = get_rate(i + 1, logTemp);
00112
00113     rate = rate1 + (rate2 - rate1) / (metallicities[i + 1] - metallicities[i]) * (logZ - metallicities[i]);
00114
00115     return pow(10, rate);
00116 }
00117
00118
00119 double get_rate(int tab, double logTemp)
00120 {
00121     int index;
00122     double rate1, rate2, rate, logTindex;
00123
00124
00125     if(logTemp < 4.0)
00126         logTemp = 4.0;
00127
00128     index = (logTemp - 4.0) / 0.05;
00129     if(index >= 90)
00130         index = 89;
00131
00132     logTindex = 4.0 + 0.05 * index;
00133
00134     rate1 = CoolRate[tab][index];
```

```

00135     rate2 = CoolRate[tab][index + 1];
00136
00137     rate = rate1 + (rate2 - rate1) / (0.05) * (logTemp - logTindex);
00138
00139     return rate;
00140 }
00141
00142 /**@brief Returns the minimum of two numbers*/
00143 double dmin(double x, double y)
00144 {
00145     if(x < y)
00146         return x;
00147     else
00148         return y;
00149 }
00150
00151
00152 //TODO remove or rename to something meaningful
00153 void test(void)
00154 {
00155     double z;
00156
00157     read_cooling_functions();
00158
00159     for(z = -8.0; z < 2.0; z += 0.5)
00160         printf("z=%g\t rate= %g\n", z, log10(get_metaldependent_cooling_rate(1.0, z))
00161 }

```

4.9 code/init.c File Reference

Sets up some unit conversion variables; converts SN and AGN feedback variables into internal units; and reads in input tables, including the desired output snapshots, the photometric and dust tables, the cooling functions and reionization tables.

Functions

- void [read_reionization](#) (void)
- void [init](#) (void)

controlling recipe for init.c, calls functions to read in tables and defines some SN and AGN feedback parameters.
- void [read_output_snaps](#) (void)

Reads in the list of output snapshots from file /input/desired_output_snaps.txt.
- void [read_snap_list](#) (void)
- void [read_tsud_tables](#) (void)

Reads in the dust extinction tables.
- void [setup_spectrophotometric_model](#) (void)

Reads in the look up tables from Stellar Population Synthesis Models.
- void [init_jump_index](#) (void)
- int [get_jump_index](#) (double age)
- void [find_interpolation_point](#) (double timenow, double timetarget, int *index, double *f1, double *f2)

Used by update_from_recycle() (not supported) to interpolate into the age table from the SSP look up tables.

- void [find_interpolated_lum](#) (double timenow, double timetarget, double metallicity, int *metindex, int *tabindex, double *f1, double *f2, double *fmet1, double *fmet2)

Used by add_to_luminosities() to interpolates into the age and metallicity in the SSP tables.

- void [find_interpolated_obs_lum](#) (double timenow, double timetarget, double metallicity, double redshift, int *metindex, int *tabindex, int *redindex, double *f1, double *f2, double *fmet1, double *fmet2, double *fred1, double *fred2)

Same as [find_interpolated_lum\(\)](#) but also interpolates in redshift; Not needed if the redshifts for the "inverted" kcorrections in the SSP tables correspond to the simulation's redshifts.

- int [find_index](#) (float *xx, int n, float x)

NEVER USED.

- void [set_units](#) (void)

Sets up some variables used to convert from physical to internal units (as UnitDensity_in_cgs); These are obtained from the three defined in input.par: UnitLength_in_cm (cm to Mpc), UnitMass_in_g (g to 1e10Msun) and UnitVelocity_in_cm_per_s (cm/s to km/s).

- void [read_recgas](#) (void)
- void [read_file_nrs](#) (void)
- void [read_sfrz](#) (void)
- void [find_interpolate_h2](#) (double metalicity, double rho, int *tabindex, int *metindex, double *f1, double *f2, double *fmet1, double *fmet2)
- void [find_interpolate_reionization](#) (double zcurr, int *tabindex, double *f1, double *f2)

Variables

- int [jumptab](#) [NJUMPTAB]
- double [jumpfac](#)

4.9.1 Detailed Description

set_units() - THIS IS FUNDAMENTAL TO UNDERSTAND THE UNITS IN THE CODE! sets ups some variables used to convert from physical to internal units (as UnitDensity_in_cgs). These are obtained from the three defined in input.par: UnitLength_in_cm, UnitMass_in_g and UnitVelocity_in_cm_per_s.

read_output_snaps() - reads in the list of output snapshots from file ./input/desired_output_snaps.txt

read_snap_list() - reads in 1/(z+1) from FileWithSnapList defined in ./input/input.par for the list of output snapshots. This will then be use to create a table with output redshift ZZ[] and ages Age[].

read_recgas() - Done if GASRECYCLE OFF - option not supported. Reads in Frac[] from gas_m62.dat.

read_file_nrs() - Done if SPECIFYFILENR OFF - the dark matter files to read can be defined in a file, instead of being read sequentially. These are defined in FileNrDir, in input.par, and read into ListInputFirNr[].

read_sfrz() - Done if H2FORMATION OFF - option not supported. Reads in Rho[] and H2[][].

read_reionization() - Reads in Reion_z[] and Reion_Mc[] from ./input/Mc.txt. These are used if UPDATEREIONIZATION OFF to get Okamoto(2008) fitting parameters (Mc) for the Hoeft(2006) reionization formalism (Guo2010) instead of Gnedin+Kravtsov (Croton2005).

setup_spectrophotometric_model() - Reads in the look up tables from a given Stellar Population Synthesis Model (**_Set_Phot_table_m**.dat). Default - Bruzual & Charlot 2003. Detailed description in [add_to_luminosities\(\)](#) in [recipe_misc.c](#). There are different files, each one corresponding to a different metallicity. On each file the tables have the Magnitudes for single bursts of $1 M_{\odot}$ for a range of ages and "inversely" k-corrected in case the code needs to compute observed frame magnitudes.

read_tsud_tables() - Reads in the dust extinction for the same bands read from the spectrophotometric tables both for the inter-galactic medium (**_Set_Ext_table.dat) and young birth clouds (**_Set_YSExt_table.dat). Detailed description at [recipe_dust.c](#)

read_cooling_functions() - calls the functions that read in the cooling functions defined in [cool_func.c](#)

In [init.c](#), but called from other files, are the function to interpolate properties into different tables. They are all the same but interpolate into different properties (have different inputs):

find_interpolation_point() - Used by [update_from_recycle\(\)](#) (not supported) to interpolate into the age table from the SSP look up tables.

find_interpolated_lum() - Used by [add_to_luminosities\(\)](#) to interpolates into the age and metallicity in the SSP tables.

find_interpolated_obs_lum() - Same as [find_interpolated_lum\(\)](#) but also interpolates in redshift. Not needed if the redshifts for the "inverted" kcorrections in the SSP tables correspond to the simulations redshifts.

find_interpolate_reionization() - Called from [recipe_infall](#) interpolates into the Reion_z[], Reion_Mc[] table, giving a value of Mc for a given redshift.

SuperNovae and AGN feedback parameters are converted into internal units:

$$AgnEfficiency = \frac{UnitMass_g}{1.58e^{-26}UnitTime_s}$$

$$EnergySNcode = \frac{EnergySN}{UnitEnergy_{cgs}}h; EtaSNcode = EtaSN \frac{UnitMass_g}{M_{\odot}h}.$$

Definition in file [init.c](#).

4.9.2 Function Documentation

4.9.2.1 int find_index (float * xx, int n, float x)

Definition at line 719 of file [init.c](#).

4.9.2.2 void find_interpolate_h2 (double metalicity, double rho, int * tabindex, int * metindex, double * f1, double * f2, double * fmet1, double * fmet2)

Definition at line 945 of file [init.c](#).

References [Rho](#).

Referenced by [cal_H2\(\)](#).

4.9.2.3 void find_interpolate_reionization (double zcurr, int * tabindex, double * f1, double * f2)

Definition at line 1032 of file [init.c](#).

References [Reion_z](#).

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

4.9.2.4 void find_interpolated_lum (double timenow, double timetarget, double metallicity, int * metindex, int * tabindex, double * f1, double * f2, double * fmet1, double * fmet2)

Definition at line 508 of file [init.c](#).

References [AgeTab](#), and [get_jump_index\(\)](#).

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

4.9.2.5 void find_interpolated_obsLum (double timenow, double timetarget, double metallicity, double redshift, int * metindex, int * tabindex, int * redindex, double * f1, double * f2, double * fmet1, double * fmet2, double * fred1, double * fred2)

Definition at line 600 of file [init.c](#).

References [AgeTab](#), [get_jump_index\(\)](#), and [RedshiftTab](#).

4.9.2.6 void find_interpolation_point (double timenow, double timetarget, int * index, double * f1, double * f2)

Definition at line 460 of file [init.c](#).

References [AgeTab](#), and [get_jump_index\(\)](#).

4.9.2.7 int get_jump_index (double age)

Definition at line 453 of file [init.c](#).

References [AgeTab](#), [jumpfac](#), and [jumptab](#).

Referenced by [find_interpolated_lum\(\)](#), [find_interpolated_obsLum\(\)](#), and [find_interpolation_point\(\)](#).

4.9.2.8 void init (void)

Calls [set_units\(\)](#), [read_output_snaps\(\)](#), [read_snap_list\(\)](#), [read_recgas\(\)](#), [read_file_nrs\(\)](#), [read_sfrz\(\)](#), [read_reionization\(\)](#), [setup_spectrophotometric_model\(\)](#), [read_tsud_tables\(\)](#) and [read_cooling_functions\(\)](#).

Converts EnergySN (->EnergySNcode), EtaSN (->EtaSNcode) and AgnEfficiency into internal units.

Definition at line 108 of file [init.c](#).

References [a0](#), [AA](#), [Age](#), [AgnEfficiency](#), [ar](#), [EnergySN](#), [EnergySNcode](#), [EtaSN](#), [EtaSNcode](#), [Hubble_h](#), [random_generator](#), [read_cooling_functions\(\)](#), [read_file_nrs\(\)](#), [read_output_snaps\(\)](#), [read_recgas\(\)](#), [read_reionization\(\)](#), [read_sfrz\(\)](#), [read_snap_list\(\)](#), [read_tsud_tables\(\)](#), [Reionization_z0](#), [Reionization_zr](#), [set_units\(\)](#), [setup_spectrophotometric_model\(\)](#), [SnpListlen](#), [ThisTask](#), [time_to_present\(\)](#), [UnitEnergy_in_cgs](#), [UnitMass_in_g](#), [UnitTime_in_s](#), and [ZZ](#).

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

4.9.2.9 void init_jump_index (void)

Definition at line 435 of file [init.c](#).

References [AgeTab](#), [jumpfac](#), and [jumptab](#).

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

4.9.2.10 void read_file_nrs (void)

Definition at line 891 of file [init.c](#).

References [FileNrDir](#), and [ListInputFilrNr](#).

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

4.9.2.11 void read_output_snaps (void)

Definition at line 187 of file [init.c](#).

References [FileWithOutputSnaps](#), and [ListOutputSnaps](#).

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

4.9.2.12 void read_recgas (void)

Definition at line 865 of file [init.c](#).

References [Frac](#).

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

4.9.2.13 void read_reionization (void)

Definition at line 1003 of file [init.c](#).

References [Reion_Mc](#), and [Reion_z](#).

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

4.9.2.14 void read_sfrz (void)

Definition at line 920 of file [init.c](#).

References [H2](#), and [Rho](#).

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

4.9.2.15 void read_snap_list (void)

Definition at line 222 of file [init.c](#).

References [AA](#), [FileWithSnapList](#), [Snaplistlen](#), and [ThisTask](#).

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

4.9.2.16 void read_tsud_tables (void)

The extinction tables are read for the same bands read for the spectrophotometric tables both for the intergalactic medium (*_Set_Ext_table.dat) and young birth clouds (**_Set_YSExt_table.dat). Detailed description at [recipe_dust.c](#). The prefix corresponds to different magnitude sets: DataRelease or SDSS corresponding respectively to BVRIK or ugriz bands.

Extinction curves for the ISM: $\left(\frac{A_\lambda}{A_v}\right)_{Z_\odot} \left(\frac{Z_{\text{gas}}}{Z_\odot}\right)^s$ are read into DeltaM[number of magnitudes+1][gas metallicity][redshift] - one extra column in the end that corresponds to V_band extinction used for the YS component.

The optical depth for YS (τ_λ^{BC}) is calibrated from the ISM optical depth in the V-band:

$$\tau_\lambda^{BC} = \tau_v^{\text{ISM}} \left(\frac{1}{\mu} - 1\right) \left(\frac{\lambda}{5500}\right)^{-0.7},$$

where $\left(\frac{\lambda}{5500}\right)^{-0.7}$ is read into CORRECTIONS[number of magnitudes][redshift] from file (_YSExt_table.dat)

All the values are inversely k-corrected as the magnitudes in order to compute dust corrected observed frame magnitudes.

For each snapshot there are 6 times 100 numbers corresponding to 100 different metallicities for each band. The first tree numbers on the file indicate the number of grid points (100) and the min and max point of the grid (in terms of metallicities).

Definition at line 292 of file [init.c](#).

References [Corrections](#), [DeltaM](#), [PhotDir](#), and [PhotPrefix](#).

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

4.9.2.17 void set_units (void)

As defined in input.par, $\text{UnitLength}_{\text{cm}} = 3.08568 \times 10^{24} \text{cm}$, converts from cm into Mpc and $\text{UnitVelocity}_{\text{cm/s}} = 10000 \text{cm/s}$, converts from cm/s to Km/s (cm to km). In [set_units\(\)](#) an interesting coincidence is used to derived UnitTime_s from these two quantities:

$$\frac{\text{UnitLength}_{\text{cm}}}{\text{UnitVelocity}_{\text{cm/s}}} = 3.08568 \times 10^{19} \text{Mpc Km}^{-1} \text{s}^{-1},$$

which is close to the number of seconds in $10^{12} \text{Yr} = 3.15533 \times 10^{19} \text{s}$ (the units of time in the code (after being divided by h)). For this reason, the unit time defined as:

$$\text{UnitTime}_s = \frac{\text{UnitLength}_{\text{cm}}}{\text{UnitVelocity}_{\text{cm/s}}},$$

its an approximation of the conversion bewteen seconds and the internal units of the code (10^{12}Yr) and is used to defined most of the other conversion factors in this function.

UnitTime_s , the way it is defined, could actually be used to convert time derived from the dynamical time $\left(\frac{R_{\text{disk}}}{V_{\text{vir}}}\right)$ into seconds. However, for the coincidence mentioned above, the time units of a time derived from t_{dyn} are very close to the code internal units (10^{12}Yr). For this reason through the code it is assumed that t_{dyn} has internal units and its never converted (note that t_{dyn} has an h factor, as the code internal units which despite not being included is UnitTime_s is included in the output of [time_to_present\(\)](#) - so it is consistent).

Assuming that UnitTime_s actually converts seconds to 10^{12}Yr all the following converting factors are correct:

$\text{UnitTime}_{\text{Myr}} = \frac{\text{UnitTime}_s}{\text{SEC_PER_MYR}} = 9.780285 \times 10^{05} \text{Yr}$, where $\text{SEC_PER_MYR} = 3.155 \times 10^7 \text{s}$, converts Yr to Myr.

$G = \frac{\text{GRAVITY} \times \text{UnitMass}_g \times \text{UnitTime}_s^2}{\text{UnitLength}_{\text{cm}}^3} = 43.00708$, where $\text{GRAVITY} = 6.672 \times 10^{-8} \text{cm}^3 \text{g}^{-1} \text{s}^{-2}$, is the gravitational constant in internal units ($\text{Mpc}^3 (10^{10} M_\odot)^{-1} (10^{12} \text{Yr})^{-2}$).

$\text{UnitDensity}_{\text{cgs}} = \frac{\text{UnitMass}_g}{\text{UnitLength}_{\text{cm}}^3} = 6.769898 \times 10^{-31}$, converts density in g cm^{-3} into internal units ($10^{10} M_\odot \text{Mpc}^{-3}$)

$\text{UnitPressure}_{\text{cgs}} = \frac{\text{UnitMass}_{\text{g}}}{\text{UnitLength}_{\text{cm}} \times \text{UnitTime}_{\text{s}}^2} = 6.769898 \times 10^{-21}$, converts pressure in $\text{g cm}^{-1}\text{s}^{-2}$ into internal units ($10^{10} M_{\odot} \text{Mpc}^{-1} (10^{12} \text{Yr})^{-2}$)

$\text{UnitCoolingRate}_{\text{cgs}} = \frac{\text{UnitPressure}_{\text{cgs}}}{\text{UnitTime}_{\text{s}}} = 2.193973 \times 10^{-40}$, converts the cooling rate in $\text{g cm}^{-1}\text{s}^{-3}$ into internal units ($10^{10} M_{\odot} \text{Mpc}^{-1} (10^{12} \text{Yr})^{-3}$)

$\text{UnitEnergy}_{\text{cgs}} = \frac{\text{UnitMass}_{\text{g}} \times \text{UnitLength}_{\text{cm}}^2}{\text{UnitTime}_{\text{s}}^2} = 1.989000 \times 10^{53}$, converts energy in $\text{g cm}^2\text{s}^{-2}$ into internal units ($(10^{10} M_{\odot} \text{Mpc}^2 (10^{12} \text{Yr})^{-2})$)

$\text{Hubble} = \text{HUBBLE} \times \text{UnitTime}_{\text{s}} = 100.0001$, where $\text{HUBBLE} = 3.2407789 \times 10^{-18} \text{h s}^{-1}$, is the hubble constante in ($\text{h Km s}^{-1}\text{Mpc}^{-1}$) or in internal units ($\text{h } 10^{12} \text{Yr}^{-1}$).

Then define a constant: $\text{RhoCrit} = \frac{3 \times \text{Hubble}^2}{8 \times \text{M_PI} \times \text{G}} = 27.75505 \text{h}^2 10^{10} M_{\odot} \text{Mpc}^{-3}$,

Definition at line 829 of file [init.c](#).

References [G](#), [Hubble](#), [RhoCrit](#), [UnitCoolingRate_in_cgs](#), [UnitDensity_in_cgs](#), [UnitEnergy_in_cgs](#), [UnitLength_in_cm](#), [UnitMass_in_g](#), [UnitPressure_in_cgs](#), [UnitTime_in_Megayears](#), [UnitTime_in_s](#), and [UnitVelocity_in_cm_per_s](#).

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

4.9.2.18 void setup_spectrophotometric_model (void)

Reads in the look up tables from a given Stellar Population Synthesis Model. The default model is Bruzual & Charlot 2003. There are different files, each one corresponding to a different metallicity. On each file the tables have the Magnitudes for single bursts of $1 M_{\odot}$ for a range of ages and "inversely" k-corrected in case the code needs to compute observed frame magnitudes. 6 different metallicities are available at `./PhotTables/` with the file names `**_Set_Phot_table_m22.dat/m32/m42/m52/m62/m72` corresponding to $Z=0.0001/z=0.0004/z=0.004/Z=0.008/Z=0.02/Z=0.05/Z=0.1$. Two different prefixes are available DataRelease or SDSS corresponding respectively to BVRIK or ugriz bands (both with all the different metallicities.)

On each file, the three first numbers correspond to: the number of snapshots, the number of bands and the number of age bins. Then the age grid is listed. After that for each snapshot, for each age the value corresponding to a burst inversely k-corrected to that redshift, with that age (and metallicity) is listed.

The basic structure is number of columns corresponding to number of mags, repeated over the number of ages on the initial listed grid, multiplied by the number of snapshots, with the snapshot number listed in between.

`agTableZz[Mag][mettallicity][Snapshot][Age]`.

Definition at line 364 of file [init.c](#).

References [AgeTab](#), [Hubble_h](#), [init_jump_index\(\)](#), [MagTableZz](#), [PhotDir](#), [PhotPrefix](#), [RedshiftTab](#), [This-Task](#), and [UnitTime_in_Megayears](#).

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

4.9.3 Variable Documentation

4.9.3.1 double jumpfac

Definition at line 433 of file [init.c](#).

Referenced by [get_jump_index\(\)](#), and [init_jump_index\(\)](#).

4.9.3.2 int jumptab[NJUMPTAB]

Definition at line 432 of file [init.c](#).

Referenced by [get_jump_index\(\)](#), and [init_jump_index\(\)](#).

4.10 code/init.c

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <string.h>
00004 #include <math.h>
00005 #include <time.h>
00006 #include <sys/time.h>
00007 #include <sys/resource.h>
00008 #include <unistd.h>
00009 #include <gsl/gsl_rng.h>
00010
00011 #include "allvars.h"
00012 #include "proto.h"
00013
00014 /**@file init.c
00015 * @brief Sets up some unit conversion variables; converts SN and AGN feedback
00016 * variables into internal units; and reads in input tables, including
00017 * the desired output snapshots, the photometric and dust tables, the
00018 * cooling functions and reionization tables.
00019 *
00020 * <B>set_units()</B> - THIS IS FUNDAMENTAL TO UNDERSTAND THE UNITS IN
00021 * sets ups some variables used to convert from physical to
00022 * internal units (as UnitDensity_in_cgs). These are obtained from the
00023 * three defined in input.par: UnitLength_in_cm, UnitMass_in_g and
00024 * UnitVelocity_in_cm_per_s.
00025 *
00026 * <B>read_output_snaps()</B> - reads in the list of output snapshots from
00027 * file ./input/desired_output_snaps.txt
00028 *
00029 * <B>read_snap_list()</B> - reads in 1/(z+1) from FileWithSnapList define
00030 * d
00031 * in ./input/input.par for the list of output snapshots. This will then b
00032 * e
00033 * use to create a table with output redshift ZZ[] and ages Age[].
00034 *
00035 *
00036 * <B>read_recgas()</B> - Done if GASRECYCLE OFF - option not supported.
00037 * Reads in Frac[] from gas_m62.dat.
00038 *
00039 * <B>read_file_nrs()</B> - Done if SPECIFYFILENR OFF - the dark matter fi
00040 * les
00041 * to read can be defined in a file, instead of being read sequentially.
00042 * These are defined in FileNrDir, in input.par, and read into
00043 * ListInputFiltrNr[].
00044 *
00045 * <B>read_sfrz()</B> - Done if H2FORMATION OFF - option not supported.
00046 * Reads in Rho[] and H2[][].
00047 *
00048 * <B>read_reionization()</B> - Reads in Reion_z[] and Reion_Mc[] from
00049 * ./input/Mc.txt. These are used if UPDATEREIONIZATION OFF to get Okamoto
00050 * (2008)
00051 * fitting parameters (Mc) for the Hoeft(2006) reionization formalism
00052 * (Guo2010) instead of Gnedin+Kravtsov (Croton2005).
00053 *
00054 * <B>setup_spectrophotometric_model()</B> - Reads in the look up tables
00055 * from a given Stellar Population Synthesis Model (**_Set_Phot_table_m**.
00056 * dat) .

```

```

00051 *      Default - Bruzual & Charlot 2003. Detailed description in
00052 *      add_to_luminosities() in recipe_misc.c. There are different files, each
00053 *      one corresponding to a different metallicity. On each file the tables h
00054 *      ave
00055 *      s
00056 *      and "inversely" k-corrected in case the code needs to compute observed
00057 *      frame magnitudes.
00058 *
00059 *      <B>read_tsud_tables()</B> - Reads in the dust extinction for the same b
00060 *      ands
00061 *      ium
00062 *      (**_Set_Ext_table.dat) and young birth clouds (**_Set_YSExt_table.dat).
00063 *
00064 *      Detailed description at recipe_dust.c
00065 *
00066 *      <B>read_cooling_functions()</B> - calls the functions that read in the
00067 *      cooling functions defined in cool_func.c
00068 *
00069 *      In init.c, but called from other files, are the function to interpolate
00070 *      properties into different tables. They are all the same but interpolate
00071 *      into different properties (have different inputs):
00072 *
00073 *      <B>find_interpolation_point()</B> - Used by update_from_recycle() (not
00074 *      supported) to interpolate into the age table from the SSP look up table
00075 *      s.
00076 *
00077 *      <B>find_interpolated_lum()</B> - Used by add_to_luminosities() to
00078 *      interpolates into the age and metallicity in the SSP tables.
00079 *      <B>find_interpolated_obs_lum()</B> - Same as find_interpolated_lum() but
00080 *      also interpolates in redshift. Not needed if the redshifts for the "inv
00081 *      erted"
00082 *      kcorrections in the SSP tables correspond to the simulations redshifts.
00083 *
00084 *      <B>find_interpolate_reionization()</B> - Called from recipe_infall
00085 *      interpolates into the Reion_z[], Reion_Mc[] table, giving a value of Mc
00086 *      for a given redshift.
00087 *
00088 *      SuperNovae and AGN feedback parameters are converted into internal unit
00089 *      s:
00090 *      \f$ AgnEfficiency = \frac{UnitMass_{\rm g}}{1.58e^{-26}UnitTime_{\rm s}}
00091 *      \f$ EnergySNcode = \frac{EnergySN}{UnitEnergy_{\rm cgs}} h; \f$
00092 *      \f$ EtaSNcode = EtaSN \frac{UnitMass_{\rm g}}{M_{\odot} h}. \f$
00093 *
00094 //Needs to be moved to proto.h
00095 void read_reionization(void);
00096
00097 /**@brief controlling recipe for init.c, calls functions to read in tables and
00098 *      defines some SN and AGN feedback parameters.
00099 *
00100 *      Calls set_units(), read_output_snaps(), read_snap_list(), read_recgas()
00101 *      , read_file_nrs(), read_sfrz(), read_reionization(),

```

```

00102 *      setup_spectrophotometric_model(), read_tsud_tables() and
00103 *      read_cooling_functions().
00104 *
00105 *      Converts EnergySN (->EnergySNcode), EtaSN (->EtaSNcode) and AgnEfficien
00106 *      cy
00107 *      into internal units.
00108 void init(void)
00109 {
00110     int i;
00111     struct rlimit rlim;
00112
00113     getrlimit(RLIMIT_CORE, &rlim);
00114     rlim.rlim_cur = RLIM_INFINITY;
00115     setrlimit(RLIMIT_CORE, &rlim);
00116
00117     random_generator = gsl_rng_alloc(gsl_rng_ranlxd1);
00118
00119     gsl_rng_set(random_generator, 42); /* start-up seed */
00120
00121
00122     set_units();
00123
00124
00125     EnergySNcode = EnergySN / UnitEnergy_in_cgss * Hubble_h;
00126     EtaSNcode = EtaSN * (UnitMass_in_g / SOLAR_MASS) / Hubble_h;
00127 #ifdef PARALLEL
00128     if(ThisTask == 0)
00129         printf("\nEnergySNcode*EtaSNcode= %g\n", EnergySNcode * EtaSNcode);
00130 #else
00131     printf("\nEnergySNcode*EtaSNcode= %g\n", EnergySNcode * EtaSNcode);
00132 #endif
00133
00134     //reads in the desired output snapshots
00135     read_output_snaps();
00136     //reads in the redshifts corresponding to the desired output snaps
00137     read_snap_list();
00138
00139 #ifdef GASRECYCLE
00140     /* read in the table of gas recycling */
00141     read_recgas();
00142 #endif
00143
00144 #ifdef SPECIFYFILENR
00145     /* read in the number of the files to be processed */
00146     read_file_nrs();
00147 #endif
00148
00149 #ifdef H2FORMATION
00150     read_sfrz();
00151 #endif
00152
00153 //ifdef UPDATEREIONIZATION to use Hoeft (2006) reionization
00154     read_reionization();
00155
00156     for(i = 0; i < Snaplistlen; i++)
00157     {
00158         //convert AA[] into redshift - ZZ[]
00159         ZZ[i] = 1 / AA[i] - 1;
00160         //table with time in internal units (1e12Yrs/h)
00161         Age[i] = time_to_present(ZZ[i]);
00162     }
00163
00164     //Values of a for the beginning and end of reionization
00165     a0 = 1.0 / (1.0 + Reionization_z0);
00166     ar = 1.0 / (1.0 + Reionization_zr);
00167

```

```

00168 //read in photometric tables
00169 setup_spectrophotometric_model();
00170
00171 //read in dust tables
00172 read_tsud_tables();
00173
00174 read_cooling_functions();
00175
00176 /* TODO - this conversion is done again in recipe_cooling.c -*/
00177
00178 /* Msun/year into g/s and then into internal units
00179 * of 1e10Msun/h / 1e12Yrs/h */
00180 AgnEfficiency /= (UnitMass_in_g / UnitTime_in_s * 1.58e-26);
00181
00182
00183 }
00184
00185 /**@brief Reads in the list of output snapshots from
00186 * file /input/desired_output_snaps.txt*/
00187 void read_output_snaps(void)
00188 {
00189     int i;
00190
00191 #ifndef GALAXYTREE
00192     char buf[1000];
00193     FILE *fd;
00194
00195     sprintf(buf, "%s", FileWithOutputSamps);
00196
00197     if (!(fd = fopen(buf, "r")))
00198     {
00199         printf("file '%s' not found.\n", buf);
00200         exit(1);
00201     }
00202
00203     for(i = 0; i < NOUT; i++)
00204     {
00205         if(fscanf(fd, " %d ", &ListOutputSamps[i]) != 1)
00206         {
00207             printf("I/O error in file '%s'\n", buf);
00208             exit(1);
00209         }
00210     }
00211     fclose(fd);
00212 #else
00213     for(i = 0; i < NOUT; i++)
00214         ListOutputSamps[i] = i;
00215 #endif
00216 }
00217
00218 /**@brief Reads in 1/(z+1) from FileWithSnapList defined
00219 * in ./input/input.par for the list of output snapshots.
00220 * This will then be used to create a table with output
00221 * redshift ZZ[] and ages Age[].*/
00222 void read_snap_list(void)
00223 {
00224     FILE *fd;
00225     char fname[1000];
00226
00227     sprintf(fname, "%s", FileWithSnapList);
00228
00229     if (!(fd = fopen(fname, "r")))
00230     {
00231         printf("can't read output list in file '%s'\n", fname);
00232         exit(1);
00233     }
00234

```

```

00235     Snaplistlen = 0;
00236     do
00237     {
00238         if(fscanf(fd, " %lg ", &AA[Snaplistlen]) == 1)
00239             Snaplistlen++;
00240         else
00241             break;
00242     }
00243     while(Snaplistlen < MAXSNAPS);
00244
00245     fclose(fd);
00246
00247 #ifdef PARALLEL
00248     if(ThisTask == 0)
00249         printf("found %d defined times in snaplist.\n", Snaplistlen);
00250 #else
00251     printf("found %d defined times in snaplist.\n", Snaplistlen);
00252 #endif
00253 }
00254
00255
00256
00257 /**@brief Reads in the dust extinction tables.
00258 */
00259 * The extinction tables are read for the same bands read for the
00260 * spectrophotometric tables both for the inter-galactic medium
00261 * (**_Set_Ext_table.dat) and young birth clouds (**_Set_YSExt_table.dat).
00262 * Detailed description at recipe_dust.c. The prefix corresponds to different
00263 * magnitude sets: DataRelease or SDSS corresponding respectively to BVRIK or
00264 * ugriz bands.
00265 *
00266 * Extinction curves for the ISM:
00267 *  $\left(\frac{A_{\lambda}}{\lambda}\right)^{-1}$  is read into
00268 *  $\left(\frac{Z_{\rm gas}}{Z_{\odot}}\right)^{-1}$  is read into
00269 * DeltaM[number of magnitudes+1][gas metallicity][redshift] - one extra
00270 * column in the end that corresponds to V_band extinction used for the YS
00271 * component.
00272 *
00273 *
00274 * The optical depth for YS ( $\tau_{\lambda}^{-1}$ ) is calibrated
00275 * from the ISM optical depth in the V-band:
00276 *
00277 *  $\tau_{\lambda}^{-1} = \left(\frac{5500}{\lambda}\right)^{-0.7}$ 
00278 *
00279 *
00280 * where  $\left(\frac{5500}{\lambda}\right)^{-0.7}$  is read
00281 * into CORRECTIONS[number of magnitudes][redshift] from file
00282 * (_YSExt_table.dat)
00283 *
00284 * All the values are inversely k-corrected as the magnitudes in order to
00285 * compute dust corrected observed frame magnitudes.
00286 *
00287 * For each snapshot there are 6 times 100 numbers corresponding to 100 differen
t
00288 * metallicities for each band. The first tree numbers on the file indicate
00289 * the number of grid points (100) and the min and max point of the grid
00290 * (in terms of metallicities).
00291 */
00292 void read_tsud_tables(void)
00293 {
00294     char buf[1000];
00295     FILE *fd;
00296     int i,k,j,l,dummy;
00297     float dumb;
00298
00299     sprintf(buf, "%s/%s_Ext_table.dat", PhotDir, PhotPrefix);
00300     if(fd = fopen(buf, "r"))

```

```

00301      {
00302          printf("file '%s' not found.\n", buf);
00303          exit(0);
00304      }
00305 //Jump the first 3 numbers (# of grid points, min and max z)
00306 fscanf(fd,"%d %e %e",&dummy, &dumb, &dumb);
00307 //loop over the different output redshifts
00308 for(j = 0; j < IZ ; j++)
00309 {
00310     fscanf(fd, " %f ", &dumb);
00311     //loop over the metallicity grid
00312     for(i = 0; i < NZZ_EXT; i++)
00313     {
00314         //read correction for each magnitude
00315         for (l=0; l<NMAG+1; l++)
00316         {
00317             fscanf(fd,"%e",&DeltaM[l][i][j]);
00318         }
00319     }
00320 }
00321 fclose(fd);
00322
00323 sprintf(buf, "%s/%s_YSExt_table.dat",PhotDir,PhotPrefix);
00324 if(!(fd = fopen(buf, "r")))
00325 {
00326     printf("file '%s' not found.\n", buf);
00327     exit(0);
00328 }
00329 //Get a correction for each mag, for each redshift
00330 for(i = 0; i < IZ ; i++)
00331 {
00332     fscanf(fd, " %f ", &dumb);
00333     for (l=0; l<NMAG; l++)
00334     {
00335         fscanf(fd,"%e",&Corrections[l][i]);
00336     }
00337 }
00338 fclose(fd);
00339 }
00340
00341 /**@brief Reads in the look up tables from Stellar Population Synthesis Models.
00342 *
00343 * Reads in the look up tables from a given Stellar Population Synthesis Model.
00344 * The default model is Bruzual & Charlot 2003. There are different files, each
00345 * one corresponding to a different metallicity. On each file the tables have
00346 * the Magnitudes for single bursts of 1\f$M_{\odot}\f$ for a range of ages
00347 * and "inversely" k-corrected in case the code needs to compute observed
00348 * frame magnitudes. 6 different metallicities are available at ./PhotTables/
00349 * with the file names **_Set_Phot_table_m22.dat/m32/m42/m52/m62/m72
00350 * corresponding to z=0.0001/z=0.0004/z=0.004/z=0.008/z=0.02/z=0.05/z=0.1. Two
00351 * different prefixes are available DataRelease or SDSS corresponding respectivel
00352 * to BVRIK or ugriz bands (both with all the different metallicities.)
00353 *
00354 * On each file, the three first numbers correspond to: the number of snapshots,
00355 * the number of bands and the number of age bins. Then the age grid is listed.
00356 * After that for each snapshot, for each age the value corresponding to a burst
00357 * inversely k-corrected to that redshift, with that age (and metallicity) is lis
00358 * ted.
00359 * The basic structure is number of columns corresponding to number of mags,
00360 * repeated over the number of ages on the initial listed grid, multiplied by the
00361 * number of snapshots, with the snapshot number listed in between.
00362 *
00363 * agTableZz[Mag][mettalicity][Snapshot][Age]. */
00364 void setup_spectrophotometric_model(void)

```

```

00365 {
00366     FILE *fd;
00367     int i, j, k, l, p;
00368     char buf[100];
00369     int nfill,n_age_bins,dummy;
00370     char *filenames[] = { "m22", "m32", "m42", "m52", "m62", "m72" };
00371     char dum[1];
00372
00373     /*Loop over the different files corresponding to different metallicities */
00374     for(k = 0; k < IZZ; k++)
00375     {
00376         sprintf(buf, "%s/%s_Phot_table_%s.dat", PhotDir, PhotPrefix, filenames[k]);
00377         if (!(fd = fopen(buf, "r")))
00378         {
00379             printf("file '%s' not found.\n", buf);
00380             exit(0);
00381         }
00382 #ifdef PARALLEL
00383         if (ThisTask == 0)
00384             printf("reading file %s \n",buf);
00385 #else
00386         printf("reading file %s \n",buf);
00387 #endif
00388
00389     // skip header line :
00390     fscanf(fd, "%d %d %d", &dummy, &nfill, &n_age_bins);
00391     /* check that the numbers on top of the file correspond to NMAG and ZL_LEN
00392      * could add a check to the snapshot number (IZ) to be consistent. */
00393     if (nfill != NMAG) {printf("nfill not equal to NMAG !!! ");exit(0);}
00394     if (n_age_bins != ZL_LEN) {printf("n_age_bins not equal to ZL_LEN !!! ");ex
it(0);}
00395
00396     // read ages of SSPs
00397     for(p = 0; p < ZL_LEN; p++)
00398     {
00399         fscanf(fd, " %e ", &AgeTab[p]);
00400         if (AgeTab[p] > 0.0) // avoid taking a log of 0 ...
00401         {
00402             /* converts AgeTab from years to log10(internal time units 1e12 Yrs
/h
00403             *UnitTime_in_Megayears is ~1e6, but not quite.
00404             *look at set_units() at init.c*/
00405             AgeTab[p] = AgeTab[p] / 1.0e6 / UnitTime_in_Megayears * Hubble_h;
00406             AgeTab[p] = log10(AgeTab[p]);
00407         }
00408     }
00409     // read mags at each output redshift
00410     for(p = 0; p < IZ; p++)
00411     {
00412         fscanf(fd, " %f ", &RedshiftTab[p]);
00413         //for each age
00414         for(i = 0; i < ZL_LEN; i++)
00415         {
00416             //read each mag
00417             for(l=0; l<NMAG; l++)
00418             {
00419                 fscanf(fd, "%e", &MagTableZZ[l][k][p][i]);
00420
00421             }
00422         }
00423     }
00424     fclose(fd);
00425
00426 }
00427 init_jump_index();
00428 }
00429 //TODO - This shouldn't be defined here

```

```

00430 #define NJUMPTAB 1000
00431
00432 int jumptab[NJUMPTAB];
00433 double jumpfac;
00434
00435 void init_jump_index(void)
00436 {
00437     double age;
00438     int i, idx;
00439
00440     jumpfac = NJUMPTAB / (AgeTab[ZL_LEN - 1] - AgeTab[1]);
00441
00442     for(i = 0; i < NJUMPTAB; i++)
00443     {
00444         age = AgeTab[1] + i / jumpfac;
00445         idx = 1;
00446         while(AgeTab[idx + 1] < age)
00447             idx++;
00448         jumptab[i] = idx;
00449     }
00450 }
00451
00452
00453 int get_jump_index(double age)
00454 {
00455     return jumptab[(int) ((age - AgeTab[1]) * jumpfac)];
00456 }
00457
00458 /**@brief Used by update_from_recycle() (notsupported) to interpolate
00459 *      into the age table from the SSP look up tables.*/
00460 void find_interpolation_point(double timenow, double timetarget, int *index, doubl
e *f1, double *f2)
00461 {
00462     int idx;
00463     double age, frac;
00464
00465     age = timenow - timetarget;
00466
00467     if(age > 0)
00468     {
00469         age = log10(age);
00470
00471         if(age > AgeTab[ZL_LEN - 1]) /* beyond table, take latest entry */
00472         {
00473             *index = ZL_LEN - 2;
00474             *f1 = 0;
00475             *f2 = 1;
00476         }
00477         else if(age < AgeTab[1]) /* age younger than 1st entry, take 1st entry */
00478         {
00479             *index = 0;
00480             *f1 = 0;
00481             *f2 = 1;
00482         }
00483     }
00484     else
00485     {
00486         /*
00487          idx = 1;
00488          */
00489         idx = get_jump_index(age);
00490         while(AgeTab[idx + 1] < age)
00491             idx++;
00492         *index = idx;
00493         frac = (age - AgeTab[idx]) / (AgeTab[idx + 1] - AgeTab[idx]);
00494         *f1 = 1 - frac;
00495         *f2 = frac;
00496     }
}

```

```

00496      }
00497  else                                /* this lies in the past */
00498  {
00499      *index = 0;
00500      *f1 = 0;
00501      *f2 = 0;
00502  }
00503
00504 }
00505
00506 /**@brief Used by add_to_luminosities() to interpolates into
00507 *       the age and metallicity in the SSP tables.*/
00508 void find_interpolated_lum(double timenow, double timetarget, double metallicity,
00509                               int *metindex,
00510                               int *tabindex, double *f1, double *f2, double *fmet1,
00511                               double *fmet2)
00510 {
00511     int k, i, idx;
00512     double age, frac;
00513     double ft1, ft2, fm1, fm2;
00514     float zz[] = { 0.0001, 0.0004, 0.004, 0.008, 0.02, 0.05 };
00515
00516     age = timenow - timetarget;
00517
00518     if(age > 0)
00519     {
00520         age = log10(age);
00521
00522         if(age > AgeTab[ZL_LEN - 1])          /* beyond table, take latest entry */
00523         {
00524             k = ZL_LEN - 2;
00525             ft1 = 0;
00526             ft2 = 1;
00527         }
00528         else if(age < AgeTab[1]) /* age younger than 1st entry, take 1st entry */
00529         {
00530             k = 0;
00531             ft1 = 0;
00532             ft2 = 1;
00533         }
00534     else
00535     {
00536         /*
00537             idx = 1;
00538         */
00539         idx = get_jump_index(age);
00540         while(AgeTab[idx + 1] < age)
00541             idx++;
00542         k = idx;
00543         frac = (age - AgeTab[idx]) / (AgeTab[idx + 1] - AgeTab[idx]);
00544         ft1 = 1 - frac;
00545         ft2 = frac;
00546     }
00547 }
00548 else                                /* this lies in the past */
00549 {
00550     k = 0;
00551     ft1 = 0;
00552     ft2 = 0;
00553 }
00554
00555 /* Now interpolate also for the metallicity */
00556
00557 for(i = 0; i < IZZ; i++)
00558 {
00559     zz[i] = log10(zz[i]);
00560 }

```

```

00561     metallicity = log10(metallicity);
00562
00563     if(metallicity > zz[IZZ - 1]) /* beyond table, take latest entry */
00564     {
00565         i = IZZ - 2;
00566         fm1 = 0;
00567         fm2 = 1;
00568     }
00569     else if(metallicity < zz[0]) /* age younger than 1st entry, take 1st entry */
00570     {
00571         i = 0;
00572         fm1 = 1;
00573         fm2 = 0;
00574     }
00575     else
00576     {
00577         idx = 0;
00578         while(zz[idx + 1] < metallicity)
00579             idx++;
00580         i = idx;
00581         frac = (metallicity - zz[idx]) / (zz[idx + 1] - zz[idx]);
00582         fm1 = 1 - frac;
00583         fm2 = frac;
00584     }
00585
00586
00587     *metindex = i;
00588     *tabindex = k;
00589
00590     *f1 = ft1;
00591     *f2 = ft2;
00592     *fmet1 = fml;
00593     *fmet2 = fm2;
00594 }
00595
00596
00597 /**@brief Same as find_interpolated_lum() but also interpolates in redshift;
00598 *      Not needed if the redshifts for the "inverted" kcorrections in the
00599 *      SSP tables correspond to the simulation's redshifts.*/
00600 void find_interpolated_obs_lum(double timenow, double timetarget, double metallicity, double redshift,
00601                                     int *metindex, int *tabindex, int *redindex, double
00602                                     *f1, double *f2,
00603                                     double *fmet1, double *fmet2, double *fred1, double
00604                                     *fred2)
00605 {
00606     int k, i, j, idx;
00607     double age, frac;
00608     double ft1, ft2, fm1, fm2, fz1, fz2;
00609     float zz[] = { 0.0001, 0.0004, 0.004, 0.008, 0.02, 0.05 };
00610
00611     age = timenow - timetarget;
00612
00613     if(age > 0)
00614     {
00615         age = log10(age);
00616
00617         if(age > AgeTab[ZL_LEN - 1]) /* beyond table, take latest entry */
00618         {
00619             k = ZL_LEN - 2;
00620             ft1 = 0;
00621             ft2 = 1;
00622         }
00623         else if(age < AgeTab[1]) /* age younger than 1st entry, take 1st entry */
00624         {
00625             k = 0;
00626             ft1 = 0;

```

```

00625         ft2 = 1;
00626     }
00627     else
00628     {
00629         /*
00630             idx = 1;
00631         */
00632         idx = get_jump_index(age);
00633         while(AgeTab[idx + 1] < age)
00634             idx++;
00635         k = idx;
00636         frac = (age - AgeTab[idx]) / (AgeTab[idx + 1] - AgeTab[idx]);
00637         ft1 = 1 - frac;
00638         ft2 = frac;
00639     }
00640 }
00641 else                                /* this lies in the past */
00642 {
00643     k = 0;
00644     ft1 = 0;
00645     ft2 = 0;
00646 }
00647
00648 /* Now interpolate also for the metallicity */
00649
00650 for(i = 0; i < IZZ; i++)
00651 {
00652     zz[i] = log10(zz[i]);
00653 }
00654 metallicity = log10(metallicity);
00655
00656 if(metallicity > zz[IZZ - 1]) /* beyond table, take latest entry */
00657 {
00658     i = IZZ - 2;
00659     fm1 = 0;
00660     fm2 = 1;
00661 }
00662 else if(metallicity < zz[0]) /* age younger than 1st entry, take 1st entry */
00663 {
00664     i = 0;
00665     fm1 = 1;
00666     fm2 = 0;
00667 }
00668 else
00669 {
00670     idx = 0;
00671     while(zz[idx + 1] < metallicity)
00672         idx++;
00673     i = idx;
00674     frac = (metallicity - zz[idx]) / (zz[idx + 1] - zz[idx]);
00675     fm1 = 1 - frac;
00676     fm2 = frac;
00677 }
00678
00679 /* Now interpolate also for the redshift of the observations */
00680
00681 if(redshift > RedshiftTab[IZ - 1]) /* beyond table, take latest entry */
00682 {
00683     j = IZ - 2;
00684     fz1 = 0;
00685     fz2 = 1;
00686 }
00687 else if(redshift < RedshiftTab[0]) /* age younger than 1st entry, take 1st en
try */
00688 {
00689     j = 0;
00690     fz1 = 1;

```

```

00691     fz2 = 0;
00692 }
00693 else
00694 {
00695     idx = 0;
00696     while(RedshiftTab[idx + 1] < redshift)
00697         idx++;
00698     j = idx;
00699     frac = (redshift - RedshiftTab[idx]) / (RedshiftTab[idx + 1] - RedshiftTab[
00700         idx]);
00701     fz1 = 1 - frac;
00702     fz2 = frac;
00703 }
00704
00705 *metindex = i;
00706 *tabindex = k;
00707 *redindex = j;
00708
00709 *f1 = ft1;
00710 *f2 = ft2;
00711 *fmet1 = fml;
00712 *fmet2 = fm2;
00713 *fred1 = fz1;
00714 *fred2 = fz2;
00715 }
00716
00717
00718 /**@brief NEVER USED*/
00719 int find_index(float *xx, int n, float x)
00720 {
00721 //TODO - never used
00722     int j, jl, ju, jm;
00723
00724     jl = 0;
00725     ju = n + 1;
00726
00727     while((ju - jl) > 1)
00728     {
00729         jm = (ju + jl) / 2.;
00730
00731         if(xx[n - 1] > xx[0]) && (x > xx[jm - 1]))
00732             jl = jm;
00733
00734         else
00735             ju = jm;
00736
00737     }
00738
00739     j = jl;
00740     if(j == 0)
00741         j = j + 1;
00742
00743     return j;
00744 }
00745
00746 /**@brief Sets up some variables used to convert from physical to internal
00747 * units (as UnitDensity_in_cgs); These are obtained from the three
00748 * defined in input.par: UnitLength_in_cm (cm to Mpc), UnitMass_in_g
00749 * (g to 1e10Msun) and UnitVelocity_in_cm_per_s (cm/s to km/s).
00750 *
00751 * As defined in input.par, \f$ \rm{UnitLength}_{\rm{cm}} = 3.08568 \times 10^{24} \rm{cm} \f$, converts from cm into Mpc and
00752 * \f$ \rm{UnitVelocity}_{\rm{cm/s}} = 10000 \rm{cm/s} \f$, converts from
00753 * cm/s to Km/s (cm to km). In set_units() an interesting coincidence
00754 * is used to derived \f$ \rm{UnitTime}_{\rm{s}} \f$ from these two quantitie
00755 *
s:

```

```

00756 *
00757 *      \f$ \frac{\rm{UnitLength}_{\rm{cm}}}{\rm{UnitVelocity}_{\rm{cm/s}}} = 3.08568 \times 10^{19} \rm{Mpc} \sim \rm{Km}^{-1} \rm{s}^{-1} \f$,
00758 *
00759 *
00760 *      which is close to the number of seconds in
00761 *      \f$ 10^{12} \rm{Yr} = 3.15533 \times 10^{19} \rm{s} \f$ (the units of time
00762 *      in
00763 *      the code (after being divided by h)). For this reason, the unit time
00764 *      defined as:
00765 *
00766 *      \f$ \rm{UnitTime}_{\rm{s}} = \frac{\rm{UnitLength}_{\rm{cm}}}{\rm{UnitVelocity}_{\rm{cm/s}}} \f$ is
00767 *
00768 *      its an approximation of the conversion bewteen seconds and the
00769 *      internal units of the code \f$(10^{12}) \rm{Yr}\f$ and is used to
00770 *      defined most of the other conversion factors in this function.
00771 *
00772 *      \f$ \rm{UnitTime}_{\rm{s}} \f$, the way it is defined, could actually be used
00773 *      to convert time derived from the dynamical time
00774 *      \f$ \left( \frac{R_{\rm{disk}}}{V_{\rm{vir}}} \right) \f$ into seconds.
00775 *      However, for the coincidence mentioned above, the time units of a time
00776 *      derived from \f$ t_{\rm{dyn}} \f$ are very close to the code internal
00777 *      units \f$(10^{12}) \rm{Yr}\f$. For this reason through the code it is assumed
00778 *      that \f$ t_{\rm{dyn}} \f$ has internal units and its never converted (note
00779 *
00780 *      that \f$ t_{\rm{dyn}} \f$ has an h factor, as the code internal units
00781 *      which despite not being included is \f$ \rm{UnitTime}_{\rm{s}} \f$ is incl
00782 *      uded
00783 *      in the output of time_to_present() - so it is consistent).
00784 *
00785 *      Assuming that \f$ \rm{UnitTime}_{\rm{s}} \f$ actually converts seconds to
00786 *      \f$ 10^{12} \rm{Yr} \f$ all the following converting factors are correct:
00787 *
00788 *      \f$ \rm{UnitTime}_{\rm{Myr}} = \frac{\rm{UnitTime}_{\rm{s}}}{\rm{SEC}} \cdot \frac{\rm{PER}}{\rm{MYR}} =
00789 *          9.780285 \times 10^{05} \rm{Yr} \f$, where \f$ \rm{SEC} \cdot \rm{PER} \cdot \rm{MYR} \f$
00790 *      \f$ = 3.155 \times 10^7 \rm{s} \f$, converts Yr to Myr.
00791 *
00792 *      \f$ G = \frac{\rm{GRAVITY}}{\rm{UnitMass}_{\rm{g}} \cdot \rm{UnitLength}_{\rm{cm}}^3} = 43.00708 \f$, where \f$ \rm{GRAVITY} \f$
00793 *      \f$ = 6.672 \times 10^{-8} \rm{cm}^3 \rm{g}^{-1} \rm{s}^{-2} \f$, is the
00794 *      gravitational constant in internal units
00795 *      \f$ (\rm{Mpc})^3 (10^{10} \dot{M})^{-1} (10^{12} \rm{Yr})^{-2} \f$.
00796 *
00797 *      \f$ \rm{UnitDensity}_{\rm{cgs}} = \frac{\rm{UnitMass}_{\rm{g}}}{\rm{UnitLength}_{\rm{cm}}^3} = 6.769898 \rm{time}^{-3} \rm{s}^{-31} \f$,
00798 *      converts density in \f$ \rm{g/cm}^3 \f$ into internal units
00799 *      \f$ (10^{10} \dot{M})^{-1} \rm{Mpc}^{-3} \f$.
00800 *
00801 *      \f$ \rm{UnitPressure}_{\rm{cgs}} = \frac{\rm{UnitMass}_{\rm{g}} \cdot \rm{UnitLength}_{\rm{cm}}}{\rm{UnitTime}_{\rm{s}}^2} \f$,
00802 *      \f$ = 6.769898 \times 10^{-21} \f$, converts pressure in
00803 *      \f$ \rm{g/cm}^{-1} \rm{s}^{-2} \f$ into internal units
00804 *      \f$ (10^{10} \dot{M})^{-1} \rm{Mpc}^{-1} (10^{12} \rm{Yr})^{-2} \f$.
00805 *
00806 *      \f$ \rm{UnitCoolingRate}_{\rm{cgs}} = \frac{\rm{UnitPressure}_{\rm{cgs}}}{\rm{UnitTime}_{\rm{s}}} = 2.193973 \rm{time}^{-40} \f$,
00807 *      converts the cooling rate in \f$ \rm{g/cm}^{-1} \rm{s}^{-3} \f$ into
00808 *      internal units \f$ (10^{10} \dot{M})^{-1} \rm{Mpc}^{-1} (10^{12} \rm{Yr})^{-3} \f$.
00809 *
00810 *      \f$ \rm{UnitEnergy}_{\rm{cgs}} = \f$
```

```

00813 *      \frac{\rm{UnitMass}}{\rm{tTime}} \times \frac{\rm{UnitLength}}{\rm{cm}}^2 \times \frac{\rm{UnitTime}}{\rm{s}}^2
00814 *      = 1.989000 \times 10^{53} \text{, converts energy in}
00815 *      \text{f$ \rm{g} \sim \rm{cm}^2 \rm{s}^{-2} \rm{f$} into internal units}
00816 *      \text{f$ ((10^{10} \rm{M_{\odot}}) \sim \rm{Mpc}^2) (10^{12} \rm{Yr})^{-2} \rm{f$}}
00817 *
00818 *      \text{f$ \rm{Hubble} = \rm{HUBBLE} \times \rm{UnitTime} / \rm{s} = 100.0001 \rm{f$},}
00819 *      where
00820 *      \text{f$ \rm{Hubble} = 3.2407789 \times 10^{-18} \text{ h} \sim \rm{s}^{-1} \rm{f$}, is the hubble}
00821 *      constante in \text{f$ (\text{h} \sim \rm{Km} \sim \rm{s}^{-1}) \rm{Mpc}^{-1} \rm{f$} or in internal u}
00822 *      nits
00823 *      \text{f$ (\text{h} \sim 10^{12}) \rm{Yr}^{-1} \rm{f$}.}
00824 *      Then define a constant:
00825 *      \text{f$ \rm{RhoCrit} = \frac{3}{8} \times \rm{Hubble}^2 \times \rm{M_{\odot}} \times \rm{PI} \times G = 27.75505 \text{ h}^{2 \sim 10^{10} \rm{M_{\odot}}} \rm{Mpc}^{-3} \rm{f$},}
00826 *
00827 */
00828
00829 void set_units(void)
00830 {
00831
00832
00833     // SEC_PER_MEGAYEAR    3.155e13
00834     // SEC_PER_YEAR        3.155e7
00835
00836     //converts from seconds to 1e12Yrs (actually from km to Mpc)
00837     UnitTime_in_s = UnitLength_in_cm / UnitVelocity_in_cm_per_s;
00838
00839
00840     /*assuming that UnitTime_in_s is number of seconds in 1e12 years it gives
00841     * approximately 1e6. This should never be used!! units in the code are not in
00842     * Myrs*/
00843     UnitTime_in_Megayears = UnitTime_in_s / SEC_PER_MEGAYEAR; //from years to milli
00844     on years=9.780285e+05
00845
00846     G = GRAVITY / pow(UnitLength_in_cm, 3) * UnitMass_in_g * pow(UnitTime_in_s, 2);
00847     //4.300708e+01
00848
00849     UnitDensity_in_cgs = UnitMass_in_g / pow(UnitLength_in_cm, 3); //6.769898e-31
00850
00851     UnitPressure_in_cgs = UnitMass_in_g / UnitLength_in_cm / pow(UnitTime_in_s, 2);
00852     //6.769898e-21
00853
00854     UnitCoolingRate_in_cgs = UnitPressure_in_cgs / UnitTime_in_s; //2.193973e-40
00855
00856     UnitEnergy_in_cgs = UnitMass_in_g * pow(UnitLength_in_cm, 2) / pow(
00857     UnitTime_in_s, 2); //1.989000e+53
00858
00859     /* convert some physical input parameters to internal units */
00860
00861     Hubble = HUBBLE * UnitTime_in_s; //1.000001e+02
00862
00863
00864 #ifdef GASRECYCLE
00865 void read_recgas(void)
00866 {
00867     char buf[1000];
00868     FILE *fd;
00869     int p,i,k,j,l,dummy;
00870     float dumb;

```

```

00871     if (! (fd = fopen ("./gas_m62.dat", "r")))
00872 {
00873     printf("file '%s' not found.\n", "/afs/mpa/data/guoqi/SAM/check/L-Galax
ies/gas_m62.dat");
00874     exit(0);
00875 }
00877
00878 for (p=0; p < ZL_LEN; p++)
00879 {
00880     fscanf(fd, "%f %f", &dumb, &Frac[p]);
00881     /*      printf("Frac reading: dumb %f , frac%f\n",dumb,Frac[p]);*/
00882 }
00883 fclose(fd);
00884
00885 }
00886 #endif
00887
00888
00889 #ifdef SPECIFYFILENR
00890
00891 void read_file_nrs(void)
00892 {
00893     int i;
00894     char buf[1000];
00895     FILE *fd;
00896     sprintf(buf, "%s", FileNrDir);
00897     if(! (fd = fopen(buf, "r")))
00898     {
00899         printf("file '%s' not found.\n", buf);
00900         exit(1);
00901     }
00902
00903     for(i = 0; i < 111; i++) //only 111files in ../input/filenrdir.txt are read in
00904     {
00905         if(fscanf(fd, " %d ", &ListInputFilrNr[i]) != 1)
00906         {
00907             printf("I/O error in file '%s'\n", buf);
00908             exit(1);
00909         }
00910     }
00911     fclose(fd);
00912
00913 }
00914
00915
00916 #endif
00917
00918 #ifdef H2FORMATION
00919
00920 void read_sfrz(void)
00921 {
00922     char buf[1000];
00923     FILE *fd;
00924     int p,i,k,j,l,dummy;
00925     float dumb;
00926     if (! (fd = fopen ("./SFR_Z.dat", "r")))
00927
00928 {
00929     printf("file '%s' not found.\n", "./SFR_Z.dat");
00930     exit(0);
00931 }
00932     printf("rho_len %d, zlen %d \n",RHO_LEN,Z_LEN);
00933     for (p=0; p < RHO_LEN; p++)
00934     {
00935         fscanf(fd, "%f", &Rho[p]);
00936         for (i=0; i<Z_LEN; i++)

```

```

00937         fscanf(fd,"%f",&H2[p][i]);
00938     }
00940
00941     fclose(fd);
00942
00943 }
00944
00945 void find_interpolate_h2(double metalicity, double rho, int *tabindex, int *metindex, double *f1, double *f2, double *fmet1, double *fmet2)
00946 {
00947     double frac;
00948     int idx;
00949     float zz[] = {-2., -1.75, -1.5, -1.25, -1., -0.75, -0.5, -0.25, 0., 0.25, 0.5, 0.75, 1.};
00950
00951     if (rho < Rho[0]) /* rho less than the smalles rho in the table, take the 1st entry*/
00952     {
00953         *tabindex = 0;
00954         *f1=1.;
00955         *f2=0.;
00956     }
00957     else if (rho > Rho[RHO_LEN-1]) /* rho great than the largest rho in the table, take the last entry*/
00958     {
00959         *tabindex = RHO_LEN-2;
00960         *f1=0.;
00961         *f2=1.;
00962     }
00963     else
00964     {
00965         idx=0;
00966         while(Rho[idx+1] < rho)
00967             idx++;
00968
00969         frac=(rho - Rho[idx])/ (Rho[idx+1]-Rho[idx]);
00970         *f1=1-frac;
00971         *f2=frac;
00972         *tabindex=idx;
00973     }
00974
00975     if (metalicity < zz[0]) /* rho less than the smalles rho in the table, take the 1st entry*/
00976     {
00977         *metindex = 0;
00978         *fmet1=1.;
00979         *fmet2=0.;
00980     }
00981     else if (metalicity > zz[Z_LEN-1]) /* rho great than the largest rho in the table, take the last entry*/
00982     {
00983         *metindex = Z_LEN-2;
00984         *fmet1=0.;
00985         *fmet2=1.;
00986     }
00987     else
00988     {
00989         idx=0;
00990         while(zz[idx+1]<metalicity)
00991             idx++;
00992
00993         frac=(-metalicity + zz[idx])/(-zz[idx+1]+zz[idx]);
00994         *fmet1=1-frac;
00995         *fmet2=frac;
00996         *metindex=idx;
00997     }
00998 }
```

```

00999 }
01000 #endif
01001
01002
01003 void read_reionization(void)
01004 {
01005     char buf[1000];
01006     FILE *fd;
01007     int p;
01008     float dumb;
01009     if (!(fd = fopen("./input/Mc.txt", "r")))
01010
01011     {
01012         printf("file '%s' not found.\n", "./tmp/Mc.txt");
01013         exit(0);
01014     }
01015
01016     for (p=0; p < 45; p++)
01017     {
01018         fscanf(fd, "%f", &dumb);
01019         fscanf(fd, "%f", &Reion_z[p]);
01020         fscanf(fd, "%f", &Reion_Mc[p]);
01021
01022     }
01023     Reion_z[45] = Reion_z[44];
01024     Reion_Mc[45] = Reion_Mc[44];
01025
01026
01027     fclose(fd);
01028
01029
01030 }
01031
01032 void find_interpolate_reionization(double zcurr, int *tabindex, double *f1, double *f2)
01033 {
01034     double frac;
01035     int idx;
01036
01037     if (zcurr > Reion_z[0]) /* redshift is higher than the largest z in the table,
take the 1st entry*/
01038     {
01039         *tabindex = 0;
01040         *f1=1.;
01041         *f2=0.;
01042     }
01043     else if (zcurr <= Reion_z[44]) /* redshift smaller than the smallest rho in the
table, take the last entry*/
01044     {
01045         *tabindex = 44;
01046         *f1=1.;
01047         *f2=0.;
01048     }
01049     else
01050     {
01051         idx=0;
01052         while(Reion_z[idx+1] > zcurr)
01053             idx++;
01054
01055         frac=(-zcurr + Reion_z[idx])/(Reion_z[idx]-Reion_z[idx+1]);
01056         *f1=1-frc;
01057         *f2=frc;
01058         *tabindex=idx;
01059     }
01060
01061 }
01062

```

4.11 code/io_tree.c File Reference

Reads in the data from the dark matter simulation merger trees, creates output files and after calculations frees the allocated memory.

Functions

- void [load_tree_table](#) (int filenr)

*Reads all the tree files if USE_MEMORY_TO_MINIMIZE_IO ON, otherwise reads in the headers for trees_** and trees_aux; Opens output files.*
- void [free_tree_table](#) (void)

*Deallocates the arrays used to read in the headers of trees_** and tree_aux (the ones with more than one element); if USE_MEMORY_TO_MINIMIZE_IO ON, deallocates the pointers containing all the input and output data.*
- void [load_tree](#) (int filenr, int nr)

Reads the actual trees into structures to be used in the code: Halo and HaloIDs; the galaxy structures (HaloGal and Gal) and the HaloAux are also allocated.
- void [free_galaxies_and_tree](#) (void)

Frees all the Halo and Galaxy structures in the code.
- size_t [myfread](#) (void *ptr, size_t size, size_t nmemb, FILE *stream)

Reading routine, either from a file into a structure or from a pointer to a structure.
- size_t [myfwrite](#) (void *ptr, size_t size, size_t nmemb, FILE *stream)

Writing routine, either from a structure to a file or from a structure to a pointer.
- int [myfseek](#) (FILE *stream, long offset, int whence)

Routine to seek on either a pointer created to store all the data for a file (if USE_MEMORY_TO_MINIMIZE_IO is ON) or to simple seek on a file (if USE_MEMORY_TO_MINIMIZE_IO is OFF).
- void [load_all_auxdata](#) (int filenr)

If USE_MEMORY_TO_MINIMIZE_IO ON - Routine to read in all the tree_aux data for one file into a pointer (ptr_auxdata) - later myfread() will pass the data onto each tree structure.
- void [load_all_dbids](#) (int filenr)

If USE_MEMORY_TO_MINIMIZE_IO ON - Routine to read in all the tree_dbids data for one file into a pointer (ptr_dbids) - later myfread() will pass the data onto each tree structure.
- void [load_all_treedata](#) (int filenr)

*If USE_MEMORY_TO_MINIMIZE_IO ON - Routine to read in all the trees_** data for one file into a pointer (ptr_treedata) - later myfread() will pass the data onto each tree structure; The galaxy pointers are allocated.*
- void [write_all_galaxy_data](#) (int filenr)

If USE_MEMORY_TO_MINIMIZE_IO ON, after all the trees have been computed and myfwrite as written the data on every tree into a single pointer, this pointer is written out in one go - this function is used for GALAXYTREE ON.

- void [write_galaxy_data_snap](#) (int n, int filenr)

If USE_MEMORY_TO_MINIMIZE_IO ON, after all the trees have been computed and myfwrite as written the data on every tree into a single pointer, this pointer is written out in one go - this function is used for normal SNAP OUTPUT.

- void [write_galaxy_for_momaf](#) (int n, int filenr)

If USE_MEMORY_TO_MINIMIZE_IO ON, after all the trees have been computed and myfwrite as written the data on every tree into a single pointer, this pointer is written out in one go - this function is used for OUTPUT_MOMAF_INPUTS ON.

- void [openallfiles](#) (int filenr)

If NEW_OI ON, this function opens all the tree files and creates file pointers: tree_file, treedbids_file and treeaux_file.

- void [closeallfiles](#) (int filenr)

If NEW_OI ON, this function closes all the files.

- FILE * [open_outputtree_file](#) (int filenr, char mode[])

If NEW_OI ON, this function opens the output file in case GALAXYTREE ON.

- FILE * [open_tree_file](#) (int filenr)

*If NEW_OI ON, this function is called by openallfiles and actually opens trees_**, returning a pointer to the file.*

- FILE * [open_treedbids_file](#) (int filenr)

If NEW_OI ON, this function is called by openallfiles and actually opens tree_dbids, returning a pointer to the file.

- FILE * [open_treeaux_file](#) (int filenr)

If NEW_OI ON, this function is called by openallfiles and actually opens trees_aux, returning a pointer to the file.

4.11.1 Detailed Description

There are three different input files: trees_** - normal tree files; tree_dbids - file containing halo IDs (used if GALAXYTREE ON); tree_aux - file containing the particle data (used if UPDATETYPETWO ON).

There are three different options available for IO in the code:

NEW_IO OFF & USE_MEMORY_TO_MINIMIZE_IO OFF - In this case, the dark matter and output files are opened for each tree and closed after galaxies are written. Files need to be reopened for next tree and the already read and written portions skipped. Uses: [load_tree_table\(\)](#) - reads in headers for trees_** and trees_aux and creates output files, [load_tree\(\)](#) - reads in trees_** and trees_dbids, [free_tree_table\(\)](#) - frees headers, [free_galaxies_and_tree\(\)](#), frees tree files, [my_fread\(\)](#), [my_fwrite\(\)](#), [my_fseek\(\)](#).

NEW_IO ON & USE_MEMORY_TO_MINIMIZE_IO OFF - The reading and writing is done on the same tree based steps, but input and output files are only opened and closed once for each file. This avoids the fseek to search which part of files were already read/ write if we keep opening and closing them. Uses: [load_tree_table\(\)](#), [load_tree\(\)](#), [free_tree_table\(\)](#), [free_galaxies_and_tree\(\)](#), [my_fread\(\)](#), [my_fwrite\(\)](#) (same functionality as before). Uses [open_all_files\(\)](#) and [closeallfiles](#) to open/close all files (done only once for each file).

NEW_IO OFF & USE_MEMORY_TO_MINIMIZE_IO ON - All the trees in a file are read at once and all the galaxy properties stored until all the trees have been processed. Then everything is written in one go. First it uses `load_all_aux_data()`, `load_all_tree_data()` `load_all_dbids_data()` to get all the tree data into arrays. Then uses the normal functions, but with a different `myfread()`, `myfwrite()` and `myfseek()` to search in the arrays instead of in the files. Then it writes everything into the output files in one go. `write_all_galaxy_data()` - everything for the galaxy tree option; `write_galaxy_data_snap()` - separated by desired output snapshots; `write_galaxy_for_momaf()` - momaf output.

Definition in file [io_tree.c](#).

4.11.2 Function Documentation

4.11.2.1 void closeallfiles (int *filenr*)

Definition at line [905](#) of file [io_tree.c](#).

References [output_file](#), [tree_file](#), [treeaux_file](#), and [treedbids_file](#).

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

4.11.2.2 void free_galaxies_and_tree (void)

Definition at line [432](#) of file [io_tree.c](#).

References [Gal](#), [Halo](#), [HaloAux](#), [HaloGal](#), [HaloIDs](#), and [myfree\(\)](#).

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

4.11.2.3 void free_tree_table (void)

Definition at line [248](#) of file [io_tree.c](#).

References [CountIDs_halo](#), [CountIDs_snaptree](#), [myfree\(\)](#), [OffsetIDs_halo](#), [OffsetIDs_snaptree](#), [ptr_auxdata](#), [ptr_dbids](#), [ptr_galaxydata](#), [ptr_galsnapdata](#), [ptr_momafdata](#), [ptr_treedata](#), [TreeFirstHalo](#), [TreeN-gals](#), and [TreeNHalos](#).

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

4.11.2.4 void load_all_auxdata (int *filenr*)

Definition at line [626](#) of file [io_tree.c](#).

References [filled_galaxydata](#), [LastSnapShotNr](#), [mymalloc\(\)](#), [offset_auxdata](#), [ptr_auxdata](#), and [SimulationDir](#).

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

4.11.2.5 void load_all_dbids (int *filenr*)

Definition at line [669](#) of file [io_tree.c](#).

References [LastSnapShotNr](#), [mymalloc\(\)](#), [offset_dbids](#), [ptr_dbids](#), and [SimulationDir](#).

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

4.11.2.6 void load_all_treedata (int *filenr*)

Definition at line 711 of file [io_tree.c](#).

References [filled_galaxydata](#), [filled_galsnapdata](#), [filled_momafdata](#), [LastSnapShotNr](#), [maxstorage_galaxydata](#), [maxstorage_galsnapdata](#), [maxstorage_momafdata](#), [mymalloc\(\)](#), [offset_galaxydata](#), [offset_galsnapdata](#), [offset_momafdata](#), [offset_treedata](#), [ptr_galaxydata](#), [ptr_galsnapdata](#), [ptr_momafdata](#), [ptr_treedata](#), and [SimulationDir](#).

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

4.11.2.7 void load_tree (int *filenr*, int *nr*)

If USE_MEMORY_TO_MINIMIZE_IO & NEW_IO are OFF, the trees_** files are opened every time a tree needs to be read in. Then the code's structure that will have the tree information is read: Halo = (sizeof(struct halo_data) * TreeNHalos[]) are read.

HaloAux structure is allocated = (sizeof(struct halo_aux_data) * TreeNHalos[])

Considering the number of halos allocated for the current tree the size of the structures containing the galaxies with a halo and the galaxies with and without a halo to be allocated is estimated:

For galaxies with a halo - MaxGals = MAXGALFAC * TreeNHalos[] and HaloGal = (sizeof(struct GALAXY) * MaxGals).

For all galaxies - FoF_MaxGals = 10000*15 and Gal = (sizeof(struct GALAXY) * FoF_MaxGals)

If GALAXYTREE ON, HaloIDs structure is read from tree_dbids = sizeof(struct halo_ids_data) * TreeNHalos[]

Definition at line 313 of file [io_tree.c](#).

References [halo_aux_data::DoneFlag](#), [FoF_MaxGals](#), [Gal](#), [Halo](#), [HaloAux](#), [halo_aux_data::HaloFlag](#), [HaloGal](#), [HaloIDs](#), [LastSnapShotNr](#), [MaxGals](#), [myfread\(\)](#), [myfseek\(\)](#), [mymalloc\(\)](#), [halo_aux_data::NGalaxies](#), [Ntrees](#), [offset_dbids](#), [offset_treedata](#), [SimulationDir](#), [tree_file](#), [treedbids_file](#), [TreeFirstHalo](#), and [TreeNHalos](#).

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

4.11.2.8 void load_tree_table (int *filenr*)

If USE_MEMORY_TO_MINIMIZE_IO ON the first step on this file is to call [load_all_dbids\(\)](#), [load_all_auxdata\(\)](#), [load_all_treedata\(\)](#). These will read all the tree data for the current file into pointers instead of doing it once for every tree.

If USE_MEMORY_TO_MINIMIZE_IO OFF the trees are read independently from the files.

Modified versions of myfread/myfwrite/myfseek are called to either read individual trees from the files into structures or from the pointers with all the data for the current file into structures.

For each tree the code reads in the header in trees_**: Ntrees - number of trees in the current file (int); totNHalos - total number of halos summed over all the trees in current file (int); TreeNHalos - number of halos in each tree (Ntrees).

Then output files are opened SA_z**_** - for snapshot output; SA_galtree_** for GALAXYTREE option and SA_**_** for MOMAF.

If UPDATETYPETWO ON the header in tree_aux is also read: NtotHalos - total number of halos in the file (int); TotIds - total number of particle IDs (int); Ntrees - total number of trees (int); TotSnaps - total number of snapshots (int). Define some other quantities: CountIDs_snaptree - number of Ids for each tree

at each snapshot (TotSnaps * Ntrees); OffsetIDs_snaptree (TotSnaps * Ntrees); CountIDs_halo - Number of IDs per halo (NtotHalos); OffsetIDs_halo (int).

Definition at line 90 of file [io_tree.c](#).

References [CountIDs_halo](#), [CountIDs_snaptree](#), [FileNameGalaxies](#), [LastSnapShotNr](#), [ListOutputSnaps](#), [load_all_auxdata\(\)](#), [load_all_dbids\(\)](#), [load_all_treedata\(\)](#), [myfread\(\)](#), [myfseek\(\)](#), [mymalloc\(\)](#), [NtotHalos](#), [Ntrees](#), [offset_auxdata](#), [offset_treedata](#), [OffsetIDs_halo](#), [OffsetIDs_snaptree](#), [OutputDir](#), [SimulationDir](#), [TotGalaxies](#), [TotGalCount](#), [TotIds](#), [TotSnaps](#), [tree_file](#), [treeaux_file](#), [TreeFirstHalo](#), [TreeNgals](#), [TreeNHalos](#), and [ZZ](#).

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

4.11.2.9 size_t myfread (void * *ptr*, size_t *size*, size_t *nmemb*, FILE * *stream*)

If USE_MEMORY_TO_MINIMIZE_IO OFF - this routine simply reads the data from a file into a structure.

If USE_MEMORY_TO_MINIMIZE_IO ON - the reading routines [load_all_dbids\(\)](#), [load_all_auxdata\(\)](#) and [load_all_treedata\(\)](#) will read all the tree data into pointers in one go for each file. Then, whenever each tree is being constructed myfread copies the properties from the pointers containing all the trees in the current file into a structure containing a single tree used by the code - from *ptr_treedata*, *ptr_dbids* and *ptr_auxdata* to *Halo*, *Haloid*s and some arrays (e.g. *PosList*) respectively.

Definition at line 561 of file [io_tree.c](#).

References [offset_auxdata](#), [offset_dbids](#), [offset_treedata](#), [ptr_auxdata](#), [ptr_dbids](#), and [ptr_treedata](#).

Referenced by [load_tree\(\)](#), [load_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.11.2.10 int myfseek (FILE * *stream*, long *offset*, int *whence*)

Definition at line 590 of file [io_tree.c](#).

References [offset_auxdata](#), [offset_dbids](#), [offset_galaxydata](#), [offset_galsnapdata](#), [offset_momafdata](#), and [offset_treedata](#).

Referenced by [load_tree\(\)](#), [load_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.11.2.11 size_t myfwrite (void * *ptr*, size_t *size*, size_t *nmemb*, FILE * *stream*)

If USE_MEMORY_TO_MINIMIZE_IO OFF - this routine simply writes the data in a structure into a file.

If USE_MEMORY_TO_MINIMIZE_IO ON - the myfwrite routine copies the galaxy properties from the current tree being treated into a pointer. This is done for all the trees in each file and after that, the pointer with properties for all the galaxies in each file written. This will be done by [write_all_galaxy_data\(\)](#), [write_all_snap_data\(\)](#) or [write_galaxy_for_momaf\(\)](#) to write either a full tree, snaps or for the MOMAF. The different outputs will be stored in a different pointer until the file is done. Inside this myfwrite the properties are copied into the corresponding pointer depending on the last input parameter: stream = 4 - GALAXYTREE option - pointer *ptr_galaxydata*; stream = >10 && <10000 - SNAP option - pointer *ptr_galsnapdata*[]; stream = >10000 - MOMAF option - pointer *ptr_momafdata*[];

Definition at line 504 of file [io_tree.c](#).

References [filled_galaxydata](#), [filled_galsnapdata](#), [filled_momafdata](#), [maxstorage_galaxydata](#), [maxstorage_galsnapdata](#), [maxstorage_momafdata](#), [offset_galaxydata](#), [offset_galsnapdata](#), [offset_momafdata](#), [ptr_galaxydata](#), [ptr_galsnapdata](#), and [ptr_momafdata](#).

Referenced by [finalize_galaxy_file\(\)](#), [finalize_momaf_file\(\)](#), [openallfiles\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.11.2.12 FILE* open_outputtree_file (int *filenr*, char *mode[]*)

Definition at line 918 of file [io_tree.c](#).

References [FileNameGalaxies](#), and [OutputDir](#).

Referenced by [finalize_galaxy_file\(\)](#), [openallfiles\(\)](#), and [save_galaxy_tree\(\)](#).

4.11.2.13 FILE* open_tree_file (int *filenr*)

Definition at line 936 of file [io_tree.c](#).

References [LastSnapShotNr](#), and [SimulationDir](#).

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

4.11.2.14 FILE* open_treeaux_file (int *filenr*)

Definition at line 972 of file [io_tree.c](#).

References [LastSnapShotNr](#), and [SimulationDir](#).

Referenced by [openallfiles\(\)](#), and [save_galaxy_tree\(\)](#).

4.11.2.15 FILE* open_treedbids_file (int *filenr*)

Definition at line 954 of file [io_tree.c](#).

References [LastSnapShotNr](#), and [SimulationDir](#).

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

4.11.2.16 void openallfiles (int *filenr*)

Definition at line 889 of file [io_tree.c](#).

References [myfwrite\(\)](#), [open_outputtree_file\(\)](#), [open_tree_file\(\)](#), [open_treeaux_file\(\)](#), [open_treedbids_file\(\)](#), [output_file](#), [tree_file](#), [treeaux_file](#), and [treedbids_file](#).

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

4.11.2.17 void write_all_galaxy_data (int *filenr*)

Definition at line 805 of file [io_tree.c](#).

References [FileNameGalaxies](#), [filled_galaxydata](#), [OutputDir](#), and [ptr_galaxydata](#).

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

4.11.2.18 void write_galaxy_data_snap (int *n*, int *filenr*)

Definition at line 832 of file [io_tree.c](#).

References [FileNameGalaxies](#), [filled_galsnapdata](#), [ListOutputSnaps](#), [OutputDir](#), [ptr_galsnapdata](#), and [ZZ](#). Referenced by [finalize_galaxy_file\(\)](#).

4.11.2.19 void write_galaxy_for_momaf(int n, int filenr)

Definition at line 861 of file [io_tree.c](#).

References [FileNameGalaxies](#), [filled_momafdata](#), [ListOutputSnaps](#), [OutputDir](#), and [ptr_momafdata](#). Referenced by [finalize_momaf_file\(\)](#).

4.12 code/io_tree.c

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <string.h>
00004 #include <math.h>
00005 #include <time.h>
00006 #include <sys/types.h>
00007 #include <sys/stat.h>
00008 #include <unistd.h>
00009
00010 #include "allvars.h"
00011 #include "proto.h"
00012
00013 /**@file io_tree.c
00014 * @brief Reads in the data from the dark matter simulation merger
00015 * trees, creates output files and after calculations frees
00016 * the allocated memory.
00017 *
00018 * There are three different input files: trees_** - normal tree
00019 * files; tree_dbids - file containing halo IDs (used if GALAXYTREE
00020 * ON); tree_aux - file containing the particle data (used if
00021 * UPDATETYPETWO ON).
00022 *
00023 * There are three different options available for IO in the code:
00024 *
00025 * <B>NEW_IO OFF & USE_MEMORY_TO_MINIMIZE_IO OFF</B> - In this case,
00026 * the dark matter and output files are opened for each tree and
00027 * closed after galaxies are written. Files need to be reopened for
00028 * next tree and the already read and written portions skipped.
00029 * Uses: load_tree_table() - reads in headers for trees_** and
00030 * trees_aux and creates output files, load_tree() - reads in trees_**
00031 * and trees_dbids, free_tree_table() - frees headers,
00032 * free_galaxies_and_tree(), frees tree files, my_fread(),
00033 * my_fwrite(), my_fseek().
00034 *
00035 * <B>NEW_IO ON & USE_MEMORY_TO_MINIMIZE_IO OFF</B> - The reading
00036 * and writing is done on the same tree based steps, but input and
00037 * output files are only opened and closed once for each file. This
00038 * avoids the fseek to search which part of files were already read/
00039 * write if we keep opening and closing them. Uses: load_tree_table(),
00040 * load_tree(), free_tree_table(), free_galaxies_and_tree(),
00041 * my_fread(), my_fwrite() (same functionality as before). Uses
00042 * open_all_files() and closeallfiles to open/close all files (done
00043 * only once for each file).
00044 *
00045 * <B>NEW_IO OFF & USE_MEMORY_TO_MINIMIZE_IO ON</B> - All the trees
00046 * in a file are read at once and all the galaxy properties stored
00047 * until all the trees have been processed. Then everything is written
00048 * in one go. First it uses load_all_aux_data (), load_all_tree_data()
00049 * load_all_dbids_data() to get all the tree data into arrays. Then

```

```

00050 * uses the normal functions, but with a different myread(),
00051 * myfwrite() and myfseek() to search in the arrays instead of in the
00052 * files. Then it writes everything into the output files in one go.
00053 * write_all_galaxy_data() - everything for the galaxy tree option;
00054 * write_galaxy_data_snap() - separated by desired output snapshots;
00055 * write_galaxy_for_momaf() - momaf output.
00056 */
00057
00058 /**@brief Reads all the tree files if USE_MEMORY_TO_MINIMIZE_IO ON,
00059 * otherwise reads in the headers for trees_** and trees_aux;
00060 * Opens output files.
00061 *
00062 * If USE_MEMORY_TO_MINIMIZE_IO ON the first step on this file is to
00063 * call load_all_dbids(), load_all_auxdata(), load_all_treedata().
00064 * These will read all the tree data for the current file into pointers
00065 * instead of doing it once for every tree.
00066 *
00067 * If USE_MEMORY_TO_MINIMIZE_IO OFF the trees are read independently
00068 * from the files.
00069 *
00070 * Modified versions of myread/myfwrite/myfseek are called to either
00071 * read individual trees from the files into structures or from the
00072 * pointers with all the data for the current file into structures.
00073 *
00074 * For each tree the code reads in the header in trees_**: Ntrees -
00075 * number of trees in the current file (int); totNHalos - total number
00076 * of halos summed over all the trees in current file (int);
00077 * TreeNHalos - number of halos in each tree (Ntrees).
00078 *
00079 * Then output files are opened SA_z**_** - for snapshot output;
00080 * SA_galtree_** for GALAXYTREE option and SA_***.** for MOMAF.
00081 *
00082 * If UPDATETYPE TWO ON the header in tree_aux is also read: NtotHalos -
00083 * total number of halos in the file (int); TotIds - total number of
00084 * particle IDs (int); Ntrees - total number of trees (int); TotSnaps -
00085 * total number of snapshots (int). Define some other quantities:
00086 * CountIDs_snaptree - number of IDs for each tree at each snapshot
00087 * (TotSnaps * Ntrees); OffsetIDs_snaptree (TotSnaps * Ntrees);
00088 * CountIDs_halo - Number of IDs per halo (NtotHalos); OffsetIDs_halo
00089 * (int). */
00090 void load_tree_table(int filenr)
00091 {
00092     int i, n, totNHalos;
00093     char buf[1000];
00094     FILE *fd;
00095
00096 #ifdef USE_MEMORY_TO_MINIMIZE_IO
00097     // read all bytes from the files into in-memory byte arrays
00098     // also define the myread method to use these arrays iso a
00099     // standard file pointer
00100     load_all_dbids(filenr);
00101     load_all_auxdata(filenr);
00102     load_all_treedata(filenr);
00103
00104     fd = (FILE *) 1; // TODO avoid use of magic numbers
00105     offset_treedata = 0;
00106 #else
00107     //open trees_** file
00108 #ifdef NEW_IO
00109     fd = tree_file;
00110 #else
00111 #ifndef MRII
00112     sprintf(buf, "%s/treedata/trees_%03d.%d", SimulationDir, LastSnapShotNr, filenr
00113 );
00113 #else
00114     sprintf(buf, "%s/treedata/trees_sf1_%03d.%d", SimulationDir, LastSnapShotNr, fi
00115 lenr);

```

```

00115 #endif
00116     if(!(fd = fopen(buf, "r")))
00117     {
00118         printf("can't open file place 1 '%s'\n", buf);
00119         exit(1);
00120     }
00121 #endif
00122 #endif
00123
00124 //read header on trees_** file
00125 myfread(&Ntrees, 1, sizeof(int), fd);
00126 myfread(&totNHalos, 1, sizeof(int), fd);
00127
00128 TreeNHalos = mymalloc(sizeof(int) * Ntrees, "treenhalos");
00129 TreeFirstHalo = mymalloc(sizeof(int) * Ntrees, "treefirsthalo");
00130
00131 for(n = 0; n < NOUT; n++) // TODO is this useful when GALAXYTREE
00132     TreeNgals[n] = mymalloc(sizeof(int) * Ntrees, "treengals");
00133 myfread(TreeNHalos, Ntrees, sizeof(int), fd);
00134
00135 #ifndef USE_MEMORY_TO_MINIMIZE_IO
00136 #ifndef NEW_IO
00137     fclose(fd);
00138 #endif
00139 #endif
00140
00141 if(Ntrees)
00142     TreeFirstHalo[0] = 0;
00143 /*Define a variable containing the number you have to jump to
00144 * get from one firshalo to the next. */
00145 for(i = 1; i < Ntrees; i++)
00146     TreeFirstHalo[i] = TreeFirstHalo[i - 1] + TreeNHalos[i - 1];
00147
00148 // create output files - snapshot option
00149 #ifndef GALAXYTREE
00150     for(n = 0; n < NOUT; n++)
00151     {
00152         for(i = 0; i < Ntrees; i++)
00153             TreeNgals[n][i] = 0;
00154
00155         sprintf(buf, "%s/%s_z%1.2f_%d", OutputDir, FileNameGalaxies, ZZ[
00156             ListOutputSnaps[n]], filenr);
00157
00158         if(!(fd = fopen(buf, "w")))
00159         {
00160             printf("can't open file place 2 '%s'\n", buf);
00161             exit(1);
00162         }
00163         fclose(fd);
00164         TotGalaxies[n] = 0;
00165     }
00166 // create output files - GALAXYTREE option
00167 #ifndef NEW_IO
00168     sprintf(buf, "%s/%s_galtree_%d", OutputDir, FileNameGalaxies, filenr);
00169     if(!(fd = fopen(buf, "w")))
00170     {
00171         printf("can't open file place 3 '%s'\n", buf);
00172         exit(1);
00173     }
00174     fclose(fd);
00175 #endif
00176     TotGalCount = 0;
00177 #endif
00178 // create output files - MOMAF option
00179 #ifdef OUTPUT_MOMAF_INPUTS
00180     for(n = 0; n < NOUT; n++)

```

```

00181      {
00182          sprintf(buf, "%s/%s_%d.%d", OutputDir, FileNameGalaxies, filenr,
00183          ListOutputSnaps[n]);
00183          if (!(fd = fopen(buf, "w")))
00184          {
00185              printf("can't open file place4 '%s'\n", buf);
00186              exit(1);
00187          }
00188          fclose(fd);
00189          TotGalaxies[n] = 0;
00190      }
00191 #endif
00192
00193
00194 #ifdef UPDATETYPETWO //open tree_aux file
00195
00196 #ifdef USE_MEMORY_TO_MINIMIZE_IO
00197     fd = (FILE *) 2;
00198     offset_auxdata = 0;
00199 #else
00200 #ifdef NEW_IO
00201     fd = treeaux_file;
00202 #else
00203 #ifndef MRII
00204     sprintf(buf, "%s/treedata/treeaux_%03d.%d", SimulationDir, LastSnapShotNr, filenr);
00205 #else
00206     sprintf(buf, "%s/treedata/treeaux_sf1_%03d.%d", SimulationDir, LastSnapShotNr, filenr);
00207 #endif
00208     if (!(fd = fopen(buf, "r")))
00209     {
00210         printf("Can't open file place 5 '%s'\n", buf);
00211         fflush(stdout);
00212         exit(1);
00213     }
00214 #endif
00215 #endif
00216
00217 //read header from tree_aux file
00218 myfread(&NtotHalos, 1, sizeof(int), fd);
00219 myfread(&TotIds, 1, sizeof(int), fd);
00220 myfread(&Ntrees, 1, sizeof(int), fd);
00221 myfread(&TotSnaps, 1, sizeof(int), fd);
00222
00223 CountIDs_snaptree = mymalloc(sizeof(int) * TotSnaps * Ntrees, "countsIDs_snaptree");
00224 OffsetIDs_snaptree = mymalloc(sizeof(int) * TotSnaps * Ntrees, "offsetIDs_snaptree");
00225 CountIDs_halo = mymalloc(sizeof(int) * NtotHalos, "countsIDs_halo");
00226 OffsetIDs_halo = mymalloc(sizeof(int) * NtotHalos, "offsetIDs_halo");
00227
00228 myfseek(fd, 2 * TotSnaps * sizeof(int), SEEK_CUR);
00229
00230 myfread(CountIDs_snaptree, sizeof(int), TotSnaps * Ntrees, fd);
00231 myfread(OffsetIDs_snaptree, sizeof(int), TotSnaps * Ntrees, fd);
00232 myfread(CountIDs_halo, sizeof(int), NtotHalos, fd);
00233 myfread(OffsetIDs_halo, sizeof(int), NtotHalos, fd);
00234
00235 #ifndef USE_MEMORY_TO_MINIMIZE_IO
00236 #ifndef NEW_IO
00237     fclose(fd);
00238 #endif
00239 #endif
00240 #endif
00241 }
00242

```

```

00243 /**
00244  * @brief Deallocation of arrays used to read in the headers of
00245  * trees_** and tree_aux (the ones with more than one
00246  * element); if USE_MEMORY_TO_MINIMIZE_IO ON, deallocate
00247  * the pointers containing all the input and output data.*/
00248 void free_tree_table(void)
00249 {
00250     int n;
00251
00252 //deallocates header from tree_aux
00253 #ifdef UPDATETYPE TWO
00254     myfree(OffsetIDs_halo);
00255     myfree(CountIDs_halo);
00256     myfree(OffsetIDs_snaptree);
00257     myfree(CountIDs_snaptree);
00258 #endif
00259
00260     for(n = NOUT - 1; n >= 0; n--)
00261         myfree(TreeNgals[n]);
00262
00263 //deallocates header from trees_**
00264 myfree(TreeFirstHalo); //derived from the header of trees_**
00265 myfree(TreeNHalos);
00266
00267 /*deallocates all the arrays ptr containing all the input and output
00268  when minimize IO is ON. */
00269 #ifdef USE_MEMORY_TO_MINIMIZE_IO
00270 #ifdef OUTPUT_MOMAF_INPUTS
00271     for(n = NOUT - 1; n >= 0; n--)
00272         myfree(ptr_momafdata[n]);
00273 #endif
00274 #ifdef GALAXYTREE
00275     myfree(ptr_galaxydata);
00276 #else
00277     for(n = NOUT - 1; n >= 0; n--)
00278         myfree(ptr_galsnapdata[n]);
00279 #endif
00280     myfree(ptr_treedata);
00281     myfree(ptr_auxdata);
00282     myfree(ptr_dbids);
00283 #endif
00284 }
00285
00286
00287 /**
00288  * @brief Reads the actual trees into structures to be used in the
00289  * code: Halo and HaloIDs; the galaxy structures (HaloGal
00290  * and Gal) and the HaloAux are also allocated
00291  *
00292  * If USE_MEMORY_TO_MINIMIZE_IO & NEW_IO are OFF, the trees_** files
00293  * are opened every time a tree needs to be read in. Then the code's
00294  * structure that will have the tree information is read: Halo =
00295  * (sizeof(struct halo_data) * TreeNHalos[])
00296  *
00297  * HaloAux structure is allocated =
00298  * (sizeof(struct halo_aux_data) * TreeNHalos[])
00299  *
00300  * Considering the number of halos allocated for the current tree
00301  * the size of the structures containing the galaxies
00302  * with a halo and the galaxies with and without a halo to be
00303  * allocated is estimated:
00304  *
00305  * For galaxies with a halo - MaxGals = MAXGALFAC * TreeNHalos[] and
00306  * HaloGal = (sizeof(struct GALAXY) * MaxGals).
00307  *
00308  * For all galaxies - FoF_MaxGals = 10000*15 and
00309  * Gal = (sizeof(struct GALAXY) * FoF_MaxGals)

```

```

00310 * If GALAXYTREE ON, HaloIDs structure is read from tree_dbids =
00311 * sizeof(struct halo_ids_data) * TreeNHalos[] */
00312
00313 void load_tree(int filenr, int nr)
00314 {
00315     int i;
00316     FILE *fd;
00317
00318 #ifndef USE_MEMORY_TO_MINIMIZE_IO
00319 #ifndef NEW_IO
00320     char buf[1000];
00321 #endif
00322 #endif
00323
00324 /* If USE_MEMORY_TO_MINIMIZE_IO & NEW_IO are OFF, the trees files are
00325 * opened every time a trees needs to be read in. */
00326 #ifdef USE_MEMORY_TO_MINIMIZE_IO
00327     fd = (FILE *) 1;
00328     offset_treedata = 0;
00329 #else
00330 #ifdef NEW_IO
00331     fd = tree_file; // file is already opened
00332 #else
00333 #ifndef MRII
00334     sprintf(buf, "%s/treedata/trees_%03d.%d", SimulationDir, LastSnapShotNr, filenr);
00335 #else
00336     sprintf(buf, "%s/treedata/trees_sf1_%03d.%d", SimulationDir, LastSnapShotNr, filenr);
00337 #endif //MRII
00338     if (!(fd = fopen(buf, "r")))
00339     {
00340         printf("can't open file '%s'\n", buf);
00341         exit(1);
00342     }
00343 #endif //NEW_IO
00344 #endif //USE_MEMORY_TO_MINIMIZE_IO
00345
00346 #ifndef NEW_IO //file is kept open, no need so seek
00347     myfseek(fd, sizeof(int) * (2 + Ntrees), SEEK_CUR);
00348     myfseek(fd, sizeof(struct halo_data) * TreeFirstHalo[nr], SEEK_CUR);
00349 #endif
00350     Halo = mymalloc(sizeof(struct halo_data) * TreeNHalos[nr], "halo");
00351
00352     myfread(Halo, TreeNHalos[nr], sizeof(struct halo_data), fd);
00353
00354 #ifndef USE_MEMORY_TO_MINIMIZE_IO
00355 #ifndef NEW_IO
00356     fclose(fd);
00357 #endif
00358 #endif
00359
00360 /* Considering the number of halos allocated for the current tree
00361 * estimate the size of the structures containing the galaxies
00362 * with a halo and the galaxies with and without a halo to be
00363 * allocated*/
00364 /*TODO - we could probably use a decent definition of these*/
00365
00366 /*max number of galaxies with a halo in the current tree*/
00367 MaxGals = (int)(MAXGALFAC * TreeNHalos[nr]);
00368 if (MaxGals < 10000)
00369     MaxGals = 10000;
00370 /*maximum number of galaxies in a tree (with and without a halo)*/
00371 FoF_MaxGals = 10000*15;
00372
00373 #ifdef UseFullSfr
00374     printf("usefullsfr\n");

```

```

00375 #endif
00376 //ifdef SAVE_MEMORY
00377 // printf("size of tree %d",sizeof(struct GALAXY));
00378 //exit(0);
00379 //endif
00380
00381 //Allocate HaloAux and Galaxy structures.
00382 HaloAux = mymalloc(sizeof(struct halo_aux_data) * TreeNHalos[nr], "haloaux");
00383 HaloGal = mymalloc(sizeof(struct GALAXY) * MaxGals, "halogal");
00384 Gal = mymalloc(sizeof(struct GALAXY) * FoF_MaxGals, "gal");
00385
00386 for(i = 0; i < TreeNHalos[nr]; i++)
00387 {
00388     HaloAux[i].DoneFlag = 0;
00389     HaloAux[i].HaloFlag = 0;
00390     HaloAux[i].NGalaxies = 0;
00391 }
00392
00393 //If GALAXYTREE ON read in HaloIDs structure from tree_dbids file
00394 #ifdef GALAXYTREE
00395 #ifdef USE_MEMORY_TO_MINIMIZE_IO
00396     fd = (FILE *) 3;
00397     offset_dbids = 0;
00398 #else
00399 #ifdef NEW_IO
00400     fd = treedbids_file;
00401 #else
00402 #ifndef MRII
00403     sprintf(buf, "%s/treedata/tree_dbids_%03d.%d", SimulationDir, LastSnapShotNr,
00404         filenr);
00405 #else
00406     sprintf(buf, "%s/treedata/tree_sfl_dbids_%03d.%d", SimulationDir,
00407         LastSnapShotNr, filenr);
00408 #endif
00409     if(!(fd = fopen(buf, "r")))
00410     {
00411         printf("can't open file '%s'\n", buf);
00412         exit(1);
00413     }
00414 #endif
00415     HaloIDs = mymalloc(sizeof(struct halo_ids_data) * TreeNHalos[nr], "haloIDs");
00416 #ifndef NEW_IO
00417     myfseek(fd, sizeof(struct halo_ids_data) * TreeFirstHalo[nr], SEEK_CUR);
00418 #endif
00419     myfread(HaloIDs, sizeof(struct halo_ids_data), TreeNHalos[nr], fd);
00420
00421 #ifndef USE_MEMORY_TO_MINIMIZE_IO
00422 #ifndef NEW_IO
00423     fclose(fd);
00424 #endif
00425 #endif
00426 #endif
00427 }
00428
00429
00430
00431 /**@brief Frees all the Halo and Galaxy structures in the code. */
00432 void free_galaxies_and_tree(void)
00433 {
00434 #ifdef GALAXYTREE
00435     myfree(HaloIDs);
00436 #endif
00437     myfree(Gal);
00438     myfree(HaloGal);
00439     myfree(HaloAux);

```

```

00440     myfree(Halo);
00441 }
00442
00443
00444
00445
00446
00447
00448 #ifndef USE_MEMORY_TO_MINIMIZE_IO
00449
00450 /**@brief Reading routine, either from a file into a structure or
00451 *      from a pointer to a structure.
00452 *
00453 *      If USE_MEMORY_TO_MINIMIZE_IO OFF - this routine simply
00454 *      reads the data from a file into a structure.
00455 *
00456 *      If USE_MEMORY_TO_MINIMIZE_IO ON - the reading routines
00457 *      load_all_dbids(), load_all_auxdata() and load_all_treedata()
00458 *      will read all the tree data into pointers in one go for each
00459 *      file. Then, whenever each tree is being constructed myfread
00460 *      copies the properties from the pointers containing all the
00461 *      trees in the current file into a structure containing a single
00462 *      tree used by the code - from ptr_treedata, ptr_dbids and ptr_auxdata
00463 *      to Halo, HaloIDs and some arrays (e.x PosList) respectively.
00464 *
00465 */
00466 size_t myfread(void *ptr, size_t size, size_t nmemb, FILE * stream)
00467 {
00468     return fread(ptr, size, nmemb, stream);
00469 }
00470 /**@brief Writing routine, either from a structure to a file or
00471 *      from a structure to a pointer.
00472 *
00473 *      If USE_MEMORY_TO_MINIMIZE_IO OFF - this routine simply
00474 *      writes the data in a structure into a file.
00475 *
00476 *      If USE_MEMORY_TO_MINIMIZE_IO ON - the myfwrite routine copies the
00477 *      galaxy properties from the current tree being treated into a
00478 *      pointer. This is done for all the trees in each file and after
00479 *      that, the pointer with properties for all the galaxies in each
00480 *      file written. This will be done by write_all_galaxy_data(),
00481 *      write_all_snap_data() or write_galaxy_for_momaf() to write either
00482 *      a full tree, snaps or for the MOMAF. The different outputs will be
00483 *      stored in a different pointer until the file is done. Inside this
00484 *      myfwrite the properties are copied into the corresponding pointer
00485 *      depending on the last input parameter:
00486 *      stream = 4 - GALAXYTREE option - pointer ptr_galaxydata;
00487 *      stream = >10 && <10000 - SNAP option - pointer ptr_galsnapdata[];
00488 *      stream = >10000 - MOMAF option - ptr_momafdata[];
00489 */
00490 size_t myfwrite(void *ptr, size_t size, size_t nmemb, FILE * stream)
00491 {
00492     return fwrite(ptr, size, nmemb, stream);
00493 }
00494
00495 /**@brief Routine to seek on either a pointer created to store
00496 *      all the data for a file (if USE_MEMORY_TO_MINIMIZE_IO is ON)
00497 *      or to simple seek on a file (if USE_MEMORY_TO_MINIMIZE_IO is OFF) */
00498 int myfseek(FILE * stream, long offset, int whence)
00499 {
00500     return fseek(stream, offset, whence);
00501 }
00502
00503 #else
00504 size_t myfwrite(void *ptr, size_t size, size_t nmemb, FILE * fd)
00505 {
00506     int n;

```

```

00507
00508     n = (int) fd;
00509
00510     //If GALAXYTREE ON
00511     if(n == 4)
00512     {
00513         if(offset_galaxydata + size * nmemb > maxstorage_galaxydata)
00514         {
00515             printf("out of space\n");
00516             exit(1212);
00517         }
00518         //copy data from structure (ptr) to pointer (ptr_galaxydata)
00519         memcpy(ptr_galaxydata + offset_galaxydata, ptr, size * nmemb);
00520         offset_galaxydata += size * nmemb;
00521         if(offset_galaxydata > filled_galaxydata)
00522             filled_galaxydata = offset_galaxydata;
00523     }
00524     //NORMAL SNAP OUTPUT
00525     if(n >= 10 && n < 10000)
00526     {
00527         n -= 10;
00528
00529         if(offset_galsnapdata[n] + size * nmemb > maxstorage_galsnapdata[n])
00530         {
00531             printf("out of space in galaxy storage for snapfile=%d\n", n);
00532             exit(1212);
00533         }
00534         //copy data from structure (ptr) to pointer (ptr_galsnapdata)
00535         memcpy(ptr_galsnapdata[n] + offset_galsnapdata[n], ptr, size * nmemb);
00536         offset_galsnapdata[n] += size * nmemb;
00537         if(offset_galsnapdata[n] > filled_galsnapdata[n])
00538             filled_galsnapdata[n] = offset_galsnapdata[n];
00539     }
00540 //MOMAF
00541 #ifdef OUTPUT_MOMAF_INPUTS
00542     if (n >= 10000)
00543     {
00544         n -= 10000;
00545         if(offset_momafdata[n] + size * nmemb > maxstorage_momafdata[n])
00546         {
00547             printf("out of space in galaxy storage for momaf file = %d\n", n);
00548             exit(1212);
00549         }
00550         //copy data from structure (ptr) to pointer (ptr_momafdata)
00551         memcpy(ptr_momafdata[n] + offset_momafdata[n], ptr, size * nmemb);
00552         offset_momafdata[n] += size * nmemb;
00553         if(offset_momafdata[n] > filled_momafdata[n])
00554             filled_momafdata[n] = offset_momafdata[n];
00555     }
00556 #endif
00557
00558     return size * nmemb;
00559 }
00560
00561 size_t myfread(void *ptr, size_t size, size_t nmemb, FILE * fd)
00562 {
00563     int n;
00564
00565     n = (int) fd;
00566
00567     if(n == 1)
00568     {
00569         //copy data from pointer (ptr_data) into structure ptr
00570         memcpy(ptr, ptr_treedata + offset_treedata, size * nmemb);
00571         offset_treedata += size * nmemb;
00572     }
00573     if(n == 2)

```

```

00574      {
00575          //copy data from pointer (ptr_auxdata) into structure ptr
00576          memcpy(ptr, ptr_auxdata + offset_auxdata, size * nmemb);
00577          offset_auxdata += size * nmemb;
00578      }
00579
00580      if(n == 3)
00581      {
00582          //copy data from pointer (ptr_dbids) into structure ptr
00583          memcpy(ptr, ptr_dbids + offset_dbids, size * nmemb);
00584          offset_dbids += size * nmemb;
00585      }
00586
00587      return size * nmemb;
00588  }
00589
00590 int myfseek(FILE * fd, long offset, int whence)
00591 {
00592     int n;
00593
00594     n = (int) fd;
00595
00596     if(n == 1)
00597         offset_treedata += offset;
00598
00599     if(n == 2)
00600         offset_auxdata += offset;
00601
00602     if(n == 3)
00603         offset_dbids += offset;
00604
00605     if(n == 4)
00606         offset_galaxydata += offset;
00607
00608     if(n >= 10 && n < 10000)
00609     {
00610         n -= 10;
00611         offset_galsnapdata[n] += offset;
00612     }
00613 #ifdef OUTPUT_MOMAF_INPUTS
00614     if (n >= 10000)
00615     {
00616         n -= 10000;
00617         offset_momafdata[n] += offset;
00618     }
00619 #endif
00620     return 0;
00621 }
00622
00623 /**@brief If USE_MEMORY_TO_MINIMIZE_IO ON - Routine to read in all
00624 *       the tree_aux data for one file into a pointer (ptr_auxdata) -
00625 *       later myfread() will pass the data onto each tree structure*/
00626 void load_all_auxdata(int filenr)
00627 {
00628     FILE *fd;
00629     char buf[1000];
00630     int ret;
00631     struct stat filestatus;
00632     size_t bytes;
00633 #ifndef MRII
00634     sprintf(buf, "%s/treedata/treeaux_%03d.%d", SimulationDir, LastSnapShotNr, file
00635             nr);
00635 #else
00636     sprintf(buf, "%s/treedata/treeaux_sf1_%03d.%d", SimulationDir, LastSnapShotNr,
00637             filenr);
00637 #endif
00638     ret = stat(buf, &filestatus);

```

```

00639
00640     if(ret != 0)                      /* seems not to exist */
00641     {
00642         printf("can't open file '%s'\n", buf);
00643         exit(1);
00644     }
00645
00646     fd = fopen(buf, "r");
00647
00648     bytes = filestatus.st_size;
00649
00650     ptr_auxdata = mymalloc(bytes, "ptr_auxdata");
00651     offset_auxdata = 0;
00652     filled_galaxydata = 0;
00653
00654     printf("reading %s... (%d bytes)\n", buf, (int) bytes);
00655     fflush(stdout);
00656 //read in tree_aux for all the trees in the file
00657     fread(ptr_auxdata, 1, bytes, fd);
00658
00659     printf("done\n");
00660     fflush(stdout);
00661
00662     fclose(fd);
00663 }
00664
00665
00666 /**@brief If USE_MEMORY_TO_MINIMIZE_IO ON - Routine to read in all
00667 *          the tree_dbids data for one file into a pointer (ptr_dbids) -
00668 *          later myfread() will pass the data onto each tree structure*/
00669 void load_all_dbids(int filenr)
0070 {
0071     FILE *fd;
0072     char buf[1000];
0073     int ret;
0074     struct stat filestatus;
0075     size_t bytes;
0076 #ifndef MRII
0077     sprintf(buf, "%s/treedata/tree_dbids_%03d.%d", SimulationDir, LastSnapShotNr, f
0078         ilenr);
0079 #else
0080     sprintf(buf, "%s/treedata/tree_sf1_dbids_%03d.%d", SimulationDir,
0081         LastSnapShotNr, filenr);
0082 #endif
0083     ret = stat(buf, &filestatus);
0084
0085     if(ret != 0)                      /* seems not to exist */
0086     {
0087         printf("can't open file '%s'\n", buf);
0088         exit(1);
0089     }
0090
0091     fd = fopen(buf, "r");
0092
0093     bytes = filestatus.st_size;
0094
0095     ptr_dbids = mymalloc(bytes, "ptr_dbids");
0096     offset_dbids = 0;
0097
0098     printf("reading %s... (%d bytes)\n", buf, (int) bytes);
0099     fflush(stdout);
0100 //read in tree_dbids for all the trees in the file
0101     fread(ptr_dbids, 1, bytes, fd);
0102
0103     printf("done\n");
0104     fflush(stdout);
0105 }
```

```

00704     fclose(fd);
00705 }
00706
00707 /*@brief If USE_MEMORY_TO_MINIMIZE_IO ON - Routine to read in all the
00708 *      trees_** data for one file into a pointer (ptr_treedata) -
00709 *      later myread() will pass the data onto each tree structure;
00710 *      The galaxy pointers are allocated. */
00711 void load_all_treedata(int filenr)
00712 {
00713     FILE *fd;
00714     char buf[1000];
00715     int ret;
00716     struct stat filestatus;
00717     size_t bytes;
00718     float smart_factor, smart_factor_momaf;
00719 #ifndef MRII
00720     sprintf(buf, "%s/treedata/trees_%03d.%d", SimulationDir, LastSnapShotNr, filenr
00721 );
00721 #else
00722     sprintf(buf, "%s/treedata/trees_sf1_%03d.%d", SimulationDir, LastSnapShotNr, fi
00723 lenr);
00723 #endif
00724     ret = stat(buf, &filestatus);
00725
00726     if(ret != 0)           /* seems not to exist */
00727     {
00728         printf("can't open file '%s'\n", buf);
00729         exit(1);
00730     }
00731
00732     fd = fopen(buf, "r");
00733
00734     bytes = filestatus.st_size;
00735
00736     ptr_treedata = mymalloc(bytes, "ptr_treedata");
00737     offset_treedata = 0;
00738
00739     printf("reading %s... (%d bytes)\n", buf, (int) bytes);
00740     fflush(stdout);
00741     //read in tree_** for all the trees in the file
00742     fread(ptr_treedata, 1, bytes, fd);
00743
00744     printf("done\n");
00745     fflush(stdout);
00746
00747     fclose(fd);
00748
00749     /*This factor will be used to get the size to be allocated for the
00750      * pointer containing all the galaxy data. */
00751     smart_factor = ALLOCPARAMETER;
00752 #ifdef OUTPUT_REST_MAGS
00753     smart_factor += 3. * NMAG;
00754 #endif
00755 #ifdef OUTPUT_OBS_MAGS
00756     smart_factor += 6. * NMAG;
00757 #endif
00758 #ifdef NO_PROPS_OUTPUTS
00759     smart_factor -= 49.;
00760 #endif
00761     smart_factor = smart_factor *3./50.;
00762
00763     /* TODO Next definition for smart_factor seems more consistent with other
00764      *      estimates on expected # of galaxies and more flexible to changes
00765      *      in GALAXY_OUTPUT. Introduced after for millimil file 7,
00766      *      maxstorage_galaxydata was exceeded after the*/
00767     smart_factor = MAXGALFAC*((float)sizeof(struct GALAXY_OUTPUT))/sizeof(struct
00768     halo_data);

```

```

00768
00769
00770 #ifdef GALAXYTREE
00771     bytes = filestatus.st_size * smart_factor; // size of galaxy pointer for GALAXY
00772     TREE
00773     ptr_galaxydata = mymalloc(bytes, "ptr_galaxydata");
00774     offset_galaxydata = 0;
00775     filled_galaxydata = 0;
00776     maxstorage_galaxydata = bytes;
00777 #else
00778     bytes = filestatus.st_size * smart_factor / 20.0; // size of galaxy pointer for
00779     SNAP OUTPUT
00780     for(ret = 0; ret < NOUT; ret++)
00781     {
00782         ptr_galsnapdata[ret] = mymalloc(bytes, "ptr_galsnapdata");
00783         offset_galsnapdata[ret] = 0;
00784         filled_galsnapdata[ret] = 0;
00785         maxstorage_galsnapdata[ret] = bytes;
00786     }
00787 #endif
00788
00789 #ifdef OUTPUT_MOMAF_INPUTS
00790     smart_factor_momaf = 3./5. + 6./25. * NMAG;
00791     bytes = filestatus.st_size * smart_factor_momaf / 20.0; // size of galaxy poin
00792     ter for MOMAF
00793     for(ret = 0; ret < NOUT; ret++)
00794     {
00795         ptr_momafdata[ret] = mymalloc(bytes, "ptr_momafdata");
00796         offset_momafdata[ret] = 0;
00797         filled_momafdata[ret] = 0;
00798         maxstorage_momafdata[ret] = bytes;
00799     }
00800
00801 /**@brief If USE_MEMORY_TO_MINIMIZE_IO ON, after all the trees have
00802 *      been computed and myfwrite as written the data on every tree
00803 *      into a single pointer, this pointer is written out in one
00804 *      go - this function is used for GALAXYTREE ON.*/
00805 void write_all_galaxy_data(int filenr)
00806 {
00807     FILE *fd;
00808     char buf[2000];
00809
00810     sprintf(buf, "%s/%s_galtree_%d", OutputDir, FileNameGalaxies, filenr);
00811     if(fd = fopen(buf, "w"))
00812     {
00813         printf("can't open file '%s'\n", buf);
00814         exit(1);
00815     }
00816
00817     printf("writing %s... (%d bytes)\n", buf, (int) filled_galaxydata);
00818     fflush(stdout);
00819
00820     fwrite(ptr_galaxydata, 1, filled_galaxydata, fd);
00821
00822     printf("done\n");
00823     fflush(stdout);
00824
00825     fclose(fd);
00826 }
00827
00828 /**@brief If USE_MEMORY_TO_MINIMIZE_IO ON, after all the trees have
00829 *      been computed and myfwrite as written the data on every tree
00830 *      into a single pointer, this pointer is written out in one
00831 *      go - this function is used for normal SNAP OUTPUT.*/

```

```

00832 void write_galaxy_data_snap(int n, int filenr)
00833 {
00834     char buf[2000];
00835     FILE *fd;
00836
00837     sprintf(buf, "%s/%s_z%1.2f_%d", OutputDir, FileNameGalaxies, ZZ[
00838         ListOutputSaps[n]], filenr);
00839     if (!(fd = fopen(buf, "w")))
00840     {
00841         printf("can't open file '%s'\n", buf);
00842         exit(1);
00843     }
00844
00845     printf("writing %s... (%d bytes)\n", buf, (int) filled_galsnapdata[n]);
00846     fflush(stdout);
00847
00848     fwrite(ptr_galsnapdata[n], 1, filled_galsnapdata[n], fd);
00849
00850     printf("done\n");
00851     fflush(stdout);
00852
00853     fclose(fd);
00854 }
00855
00856 /**@brief If USE_MEMORY_TO_MINIMIZE_IO ON, after all the trees have
00857 *      been computed and myfwrite as written the data on every tree
00858 *      into a single pointer, this pointer is written out in one
00859 *      go - this function is used for OUTPUT_MOMAF_INPUTS ON. */
00860 #ifdef OUTPUT_MOMAF_INPUTS
00861 void write_galaxy_for_momaf(int n, int filenr)
00862 {
00863     char buf[2000];
00864     FILE *fd;
00865     sprintf(buf, "%s/%s_%d.%d", OutputDir, FileNameGalaxies, filenr,
00866             ListOutputSaps[n]);
00867     if (!(fd = fopen(buf, "w")))
00868     {
00869         printf("can't open file '%s'\n", buf);
00870         exit(1);
00871     }
00872     printf("writing %s... (%d bytes)\n", buf, (int) filled_momafdata[n]);
00873     fflush(stdout);
00874     fwrite(ptr_momafdata[n], 1, filled_momafdata[n], fd);
00875     printf("done\n");
00876     fflush(stdout);
00877     fclose(fd);
00878 }
00879#endif
00880#endif //USE_MEMORY_TO_MINIMIZE_IO
00881
00882
00883
00884
00885 /**@brief If NEW_OI ON, this function opens all the tree files and
00886 *      creates file pointers: tree_file, treedbids_file and
00887 *      treeaux_file.*/
00888 #ifdef NEW_IO
00889 void openallfiles(int filenr)
00890 {
00891     char *buf;
00892 #ifdef GALAXYTREE // this should always be the case for NEW_IO, see main.c::check
00893     _options()!!
00894     output_file = open_outputtree_file(filenr, "wb");
00895     // fill header row with 0s

```

```

00896     buf = malloc(sizeof(struct GALAXY_OUTPUT));
00897     memset(buf,0,sizeof(struct GALAXY_OUTPUT));
00898     myfwrite(buf, sizeof(struct GALAXY_OUTPUT), SEEK_CUR, output_file);
00899
00900     tree_file = open_tree_file(filenr);
00901     treedbids_file = open_treedbids_file(filenr);
00902     treeaux_file = open_treeaux_file(filenr);
00903 }
00904 /**@brief If NEW_OI ON, this function closes all the files.*/
00905 void closeallfiles(int filenr)
00906 {
00907     fclose(output_file);
00908     fclose(tree_file);
00909     fclose(treedbids_file);
00910     fclose(treeaux_file);
00911 }
00912 #endif
00913
00914 #ifdef GALAXYTREE
00915
00916 /**@brief If NEW_OI ON, this function opens the output file
00917 * in case GALAXYTREE ON. */
00918 FILE* open_outputtree_file(int filenr, char mode[] )
00919 {
00920     char buf[2000];
00921     FILE *fd;
00922
00923     sprintf(buf, "%s/%s_galtree_%d", OutputDir, FileNameGalaxies, filenr);
00924     if(!(fd = fopen(buf, mode)))
00925     {
00926         printf("io_tree::open_outputtree_file : can't open output file '%s'\n", buf
00927     );
00928         exit(1);
00929     }
00930     return fd;
00931 }
00932 #endif
00933 /**@brief If NEW_OI ON, this function is called by openallfiles and actually
00934 *      opens trees_**, returning a pointer to the file. */
00935
00936 FILE* open_tree_file(int filenr)
00937 {
00938     char buf[2000];
00939     FILE *fd;
00940     #ifndef MRII
00941     sprintf(buf, "%s/treedata/trees_%03d.%d", SimulationDir, LastSnapShotNr, filenr
00942 );
00943     #else
00944     sprintf(buf, "%s/treedata/trees_sf1_%03d.%d", SimulationDir, LastSnapShotNr, fi
00945     lenr);
00946     #endif
00947     if(!(fd = fopen(buf, "rb")))
00948     {
00949         printf("io_tree::open_tree_file : can't open tree file '%s'\n", buf);
00950         exit(1);
00951     }
00952     return fd;
00953 }
00954 /**@brief If NEW_OI ON, this function is called by openallfiles and actually
00955 *      opens tree_dbids, returning a pointer to the file. */
00956 FILE* open_treedbids_file(int filenr)
00957 {
00958     char buf[2000];
00959     FILE *fd;
00960     #ifndef MRII
00961     sprintf(buf, "%s/treedata/tree_dbids_%03d.%d", SimulationDir, LastSnapShotNr,

```

```

        filenr);
00960 #else
00961     sprintf(buf, "%s/treedata/tree_sf1_dbids_%03d.%d", SimulationDir,
00962             LastSnapShotNr, filenr);
00962 #endif
00963     if (!(fd = fopen(buf, "rb")))
00964     {
00965         printf("io_tree::open_treedbids_file : can't open treedbids file '%s'\n", b
00966             uf);
00967         exit(1);
00968     }
00969     return fd;
00969 }
00970 /**@brief If NEW_OI ON, this function is called by openallfiles and actually
00971 *      opens trees_aux, returning a pointer to the file. */
00972 FILE* open_treeaux_file(int filenr)
00973 {
00974     char buf[2000];
00975     FILE *fd;
00976 #ifndef MRII
00977     sprintf(buf, "%s/treedata/treeaux_%03d.%d", SimulationDir, LastSnapShotNr, filen
00978         r);
00978 #else
00979     sprintf(buf, "%s/treedata/treeaux_sf1_%03d.%d", SimulationDir, LastSnapShotNr, f
00980         ilenr);
00980 #endif
00981     if (!(fd = fopen(buf, "rb")))
00982     {
00983         printf("io_tree::open_treeaux_file : Can't open treeaux file '%s'\n", buf);
00984         fflush(stdout);
00985         exit(1);
00986     }
00987     return fd;
00988 }
00989

```

4.13 code/main.c File Reference

The file containing the [main\(\)](#) function for L-Galaxies.

Functions

- int [main](#) (int argc, char **argv)

4.13.1 Detailed Description

Controlling function of L-Galaxies plus Construct Galaxies, Join Galaxies of progenitors, Evolve Galaxies and check Makefile options.

Reads the parameter file given as an argument at run time: [read_parameter_file\(argv\[1\]\)](#).

Checks for consistency between some Makefile options: [check_options\(\)](#).

Runs [init\(\)](#) to initialize some stuff.

For each tree, on each [file](#): reads tree, constructs the galaxies, saves the galaxies, frees memory for galaxies and tree.

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

4.13.2 Function Documentation

4.13.2.1 int main (int argc, char ** argv)

Recursively constructs all the galaxies.

Starting from z=0, the code asks if the main progenitor of that halo has been constructed. Builds the main progenitor if it hasn't been built yet. If the main progenitor has been built then the halos in the FOF of the main progenitor are also built. After all the progenitors (main halos and subhalos have been built), or if there are no progenitors, the same procedure is done for all the halos in the current FOF (build their progenitors). Then the galaxies on the current FOF group can be constructed.

`join_galaxies_of_progenitors` updates the properties of the galaxy from the dark matter halo properties and deals with merging clocks.

This routine is called by `construct_galaxies` for every halo in the FOF being constructed.

When there is no galaxy in the Halo of FirstProgenitor, the `first_occupied` pointer is changed to a subhalo which have the maximum mass.

For a central galaxy `galaxy` it just updates its properties. For satellites it needs to know its most massive (or only progenitor) to keep track of the merging clock. It also finds the central galaxies into which galaxies should merge. If `MERGE01=1`, type 1's can merge if their baryonic mass is bigger than the dark matter mass and type 2's can merge into them. Once the type 1's merge into a type 0 all its satellites will have the merging clock reset into the type 0.

`evolve_galaxies` deals with most of the SA recipes

This is where most of the physical recipes are called, including: `infall_recipe` (gets the fraction of primordial gas that infalled), `add_infall_to_hot` (adds the infalled gas to the hot phase), `reincorporate_gas` (reincorporates gas ejected by SN), `cooling_recipe` (gets the amount of gas that cooled - takes into account AGN feedback), `cool_gas_onto_galaxy` (adds the gas that cooled into the cold phase), `starformation_and_feedback` (normal SF and SN feedback), `deal_with_galaxy_merger` (adds components, grows black hole and deals with SF burst), `disruption` (total and instantaneous disruption of type 2 satellites), `dust` (if galaxy is in an output time, dust extinction is computed).

All these calculations are done in time steps of 1/STEPS the time between each snapshot (STEPS=20).

Check whether makefile options are compatible.

Definition at line 33 of file [main.c](#).

References `add_infall_to_hot()`, `cal_gas_recycle()`, `GALAXY::CentralGal`, `check_options()`, `closeallfiles()`, `construct_galaxies()`, `cool_gas_onto_galaxy()`, `cooling_recipe()`, `GALAXY::CoolingGas`, `GALAXY::CoolingRadius`, `deal_with_galaxy_merger()`, `DiskRadiusMethod`, `disrupt()`, `halo_aux_data::DoneFlag`, `evolve_galaxies()`, `FileNameGalaxies`, `finalize_galaxy_file()`, `finalize_momaf_file()`, `FirstFile`, `halo_aux_data::FirstGalaxy`, `halo_data::FirstHaloInFOFgroup`, `halo_data::FirstProgenitor`, `GALAXY::FirstProgGal`, `FoF_MaxGals`, `free_galaxies_and_tree()`, `free_tree_table()`, `Gal`, `GalCount`, `GALAXY::GasDiskRadius`, `get_disk_radius()`, `Halo`, `HaloAux`, `halo_aux_data::HaloFlag`, `HaloGal`, `GALAXY::HaloNr`, `infall_recipe()`, `init()`, `init_galaxy()`, `join_galaxies_of_progenitors()`, `LastFile`, `GALAXY::Len`, `halo_data::Len`, `ListInputFilrNr`, `ListOutputSnaps`, `load_tree()`, `load_tree_table()`, `MaxGals`, `GALAXY::MergTime`, `halo_data::MostBoundID`, `GALAXY::MostBoundID`, `mu_seed`, `halo_data::NextHaloInFOFgroup`, `halo_data::NextProgenitor`, `GALAXY::NextProgGal`, `halo_aux_data::NGalaxies`, `NTask`, `Ntrees`, `NumGals`, `NumMergers`, `NumToTime()`, `openallfiles()`, `OutputDir`, `halo_data::Pos`, `GALAXY::Pos`, `random_generator`, `read_parameter_file()`, `reincorporate_gas()`, `ReIncorporationFactor`, `save_galaxies()`, `save_galaxy_tree()`, `set_merger_center()`, `GALAXY::Sfr`, `GALAXY::SfrBulge`, `GALAXY::SnapNum`, `halo_data::SnapNum`, `starformation_and_feedback()`, `GALAXY::StellarDiskRadius`, `ThisTask`, `TreeNHalos`, `tsudy()`, `GALAXY::Type`, `update_centralgal()`, `update_type_1()`, `update_type_2()`, `halo_data::Vel`, `GALAXY::Vel`, `halo_data::Vmax`, `GALAXY::Vmax`,

and ZZ.

4.14 code/main.c

```

00001  /** @file main.c
00002  * @brief The file containing the main() function for L-Galaxies.
00003  */
00004 #include <stdio.h>
00005 #include <stdlib.h>
00006 #include <string.h>
00007 #include <math.h>
00008 #include <time.h>
00009 #include <sys/stat.h>
00010
00011 #ifdef PARALLEL
00012 #include <mpi.h>
00013 #endif
00014
00015 #include "allvars.h"
00016 #include "proto.h"
00017
00018
00019 /**@file main.c
00020 * @brief Controlling function of L-Galaxies plus Construct Galaxies,
00021 * Join Galaxies of progenitors, Evolve Galaxies and check
00022 * Makefile options.
00023 *
00024 * Reads the parameter file given as an argument at run time:
00025 * read_parameter_file(argv[1]).
```

~~00026 *~~

```

00027 * Checks for consistency between some Makefile options: check_options().
00028 *
00029 * Runs init() to initialize some stuff.
00030 *
00031 * For each tree, on each file: reads tree, constructs the galaxies, saves
00032 * the galaxies, frees memory for galaxies and tree. */
00033 int main(int argc, char **argv) {
00034     int filenr, tree, halonr, filenrid;
00035     struct stat filestatus;
00036     FILE *fd;
00037     char buf[1000];
00038 #ifdef PARALLEL
00039     time_t start, current;
00040
00041     MPI_Init(&argc, &argv);
00042     MPI_Comm_rank(MPI_COMM_WORLD, &ThisTask);
00043     MPI_Comm_size(MPI_COMM_WORLD, &NTask);
00044 #endif
00045
00046     if (argc != 2) {
00047         printf("\n usage: L-Galaxies <parameterfile>\n\n");
00048         exit(1);
00049     }
00050
00051 /*Reads the parameter file, given as an argument at run time.*/
00052 read_parameter_file(argv[1]);
00053
00054 /* check compatibility of some Makefile Options*/
00055 check_options();
00056
00057 // tsud
00058 mu_seed = -150;
00059
00060 init();
00061

```

```

00062 #ifdef PARALLEL
00063     /* a small delay so that processors dont use the same file */
00064     time(&start);
00065     do
00066         time(&current);
00067     while(difftime(current, start) < 2.0 * ThisTask);
00068 #endiff
00069
00070 /*Loop on dark matter tree files*/
00071 /*Either from a list of files*/
00072 #ifdef SPECIFYFILENR
00073     for(filenrid = FirstFile; filenrid <= LastFile; filenrid++)
00074     {
00075         filenr=ListInputFilrNr[filenrid];
00076     #else
00077         /*Or for all the files from FirstFile to LastFile*/
00078         for (filenr = FirstFile; filenr <= LastFile; filenr++) {
00079     #endiff
00080 #ifdef GALAXYTREE
00081     sprintf(buf, "%s/%s_galtree_%d", OutputDir, FileNameGalaxies, filenr);
00082 #else
00083     sprintf(buf, "%s/%s_z%1.2f_%d", OutputDir, FileNameGalaxies,
00084             ZZ[ListOutputSnaps[0]], filenr);
00085 #endiff
00086     /* if(stat(buf, &filestatus) == 0) // seems to exist
00087     {
00088 #ifndef PARALLEL
00089     printf("\nfile %d exists in %s, skipping\n\n", filenr,OutputDir);
00090 #endiff
00091     continue;
00092 } */
00093
00094 #ifdef PARALLEL
00095     printf("\nTask %d reading file nr %d\n\n", ThisTask, filenr);
00096 #endiff
00097
00098 #ifdef NEW_IO
00099     openallfiles(filenr);
00100 #else
00101     if ((fd = fopen(buf, "w")))
00102         fclose(fd);
00103 #endiff
00104     //read in the headers of hte tree files
00105     load_tree_table(filenr);
00106
00107     //Loop through all the trees in this file
00108     for (tree = 0; tree < Ntrees; tree++)
00109     {
00110         // if ( filenr == 0 && tree == 0) continue;
00111         //reads in the actual trees
00112         load_tree(filenr, tree);
00113         gsl_rng_set(random_generator, filenr * 100000 + tree);
00114         NumMergers = 0;
00115         NumGals = 0;
00116 #ifdef GALAXYTREE
00117         GalCount = 0;
00118 #endiff
00119         //Loop through all the halos in the current tree
00120         for(halonr = 0; halonr < TreeNHalos[tree]; halonr++)
00121             if(HaloAux[halonr].DoneFlag == 0)
00122                 /*we include the parameter tree in function construct_galaxies to l
oop back galaxy for GasRecycle */
00123                 //Do SAM!
00124                 construct_galaxies(halonr,tree);
00125
00126
00127     //Save properties into output file

```

```

00128 #ifdef GALAXYTREE
00129         save_galaxy_tree(filenr, tree);
00130 #else
00131         save_galaxies(filenr, tree);
00132 #endif
00133         //free memory
00134         free_galaxies_and_tree();
00135     }
00136
00137     finalize_galaxy_file(filenr);
00138 #ifdef OUTPUT_MOMAF_INPUTS
00139     finalize_momaf_file(filenr);
00140 #endif
00141 #ifdef NEW_IO
00142     closeallfiles(filenr);
00143 #endif
00144     free_tree_table();
00145
00146     printf("\ndone file %d\n\n", filenr);
00147 }
00148
00149 #ifdef PARALLEL
00150     MPI_Finalize();
00151 #endif
00152     return 0;
00153
00154 }
00155 /**@brief Recursively constructs all the galaxies.
00156 *
00157 * Starting from z=0, the code asks if the main progenitor of that
00158 * halo has been constructed. Builds the main progenitor if it hasn't
00159 * been built yet. If the main progenitor has been built then the
00160 * halos in the FOF of the main progenitor are also built. After all
00161 * the progenitors (main halos and subhalos have been built), or if
00162 * there are no progenitors, the same procedure is done for all the
00163 * halos in the current FOF (build their progenitors). Then the
00164 * galaxies on the current FOF group can be constructed. */
00165 void construct_galaxies(int halonr, int tree) {
00166     static int halosdone = 0;
00167     int prog, fofhalo, ngal,cenngal;
00168     int i;
00169     int p;
00170     HaloAux[halonr].DoneFlag = 1;
00171     halosdone++;
00172
00173     prog = Halo[halonr].FirstProgenitor;
00174
00175     while (prog >= 0) //If halo has a progenitor
00176     {
00177         if(HaloAux[prog].DoneFlag == 0) //If progenitor hasn't been done
00178             yet
00179             construct_galaxies(prog,tree);
00180         prog = Halo[prog].NextProgenitor; //Jump to next halo in progenitors FOF
00181     }
00182
00183     //Now check for the progenitors of all the halos in the current FOF group
00184
00185     fofhalo = Halo[halonr].FirstHaloInFOFgroup; //Starting at the first halo in c
urrent FOF
00186     if (HaloAux[fofhalo].HaloFlag == 0) //If it hasn't been done
00187     {
00188         HaloAux[fofhalo].HaloFlag = 1; //mark has to do
00189         while (fofhalo >= 0) //go through all the halos in current FOF
00190         {
00191             prog = Halo[fofhalo].FirstProgenitor;
00192                 while (prog >= 0) //build its progenitors
00193                 {

```

```

00192             if(HaloAux[prog].DoneFlag == 0)
00193                 construct_galaxies(prog,tree);
00194             prog = Halo[prog].NextProgenitor;
00195         }
00196
00197         fofhalo = Halo[fofhalo].NextHaloInFOFgroup; //Jump to next halo
00198     in FOF
00199 }
00200
00201 /* At this point, the galaxies for all progenitors of this halo have been
00202 * properly constructed. Also, the galaxies of the progenitors of all other
00203 * halos in the same FOF-group have been constructed as well. We can hence go
00204 * ahead and construct all galaxies for the subhalos in this FOF halo, and
00205 * evolve them in time. */
00206
00207
00208     fofhalo = Halo[halonr].FirstHaloInFOFgroup;
00209     if (HaloAux[fofhalo].HaloFlag == 1) //If it is marked as an halo to do
00210     {
00211         ngal = 0;
00212         HaloAux[fofhalo].HaloFlag = 2;
00213
00214 #ifdef MERGE01
00215     cenngal=set_merger_center(fofhalo); //Find type 0 for type 1 to merge int
00216 #endif
00217     /*For all the halos in the current FOF join all the progenitor galaxies tog
00218 ether */
00219     while (fofhalo >= 0)
00220     {
00221 #ifdef MERGE01
00222         ngal = join_galaxies_of_progenitors(fofhalo, ngal,cenngal);
00223 #else
00224         ngal = join_galaxies_of_progenitors(fofhalo, ngal);
00225 #endif
00226         fofhalo = Halo[fofhalo].NextHaloInFOFgroup;
00227     }
00228 //to test, find out the position of the type 0 galaxies.:
00229 #ifdef MERGE01
00230     for (i=0;i<ngal;i++)
00231     {
00232         if (Gal[i].Type == 0)
00233         {
00234             if (i > 0)
00235                 printf("to test,right cenngalid %d, firstinfof id %d inprog
am \n",cenngal,i);
00236             if (i != cenngal)
00237             {
00238                 printf("must be wrong, 1 place i %d cenngal %d total
%d type %d, %d, %d\n",i,cenngal,ngal, Gal[cenngal].Type,Halo[halonr].FirstHaloInF
00239                 OFgroup,tree);
00240                 exit(0);
00241             }
00242             if (i != cenngal)
00243             {
00244                 printf("must be wrong ,snap %d\n",Gal[i].Type);
00245                 exit(0);
00246             }
00247 #endif
00248 //end of test
00249 /*Evolve the Galaxies -> SAM!*/
00250     evolve_galaxies(Halo[halonr].FirstHaloInFOFgroup, ngal,tree,cenngal);
00251 }

```

```

00253
00254 }
00255
00256 /**@brief join_galaxies_of_progenitors updates the properties of the galaxy from
00257 * the dark matter halo properties and deals with merging clocks.
00258 *
00259 * This routine is called by construct_galaxies for every halo in the FOF being
00260 * constructed.
00261 *
00262 * When there is no galaxy in the Halo of FirstProgenitor, the first_occupied
00263 * pointer is changed to a subhalo which have the maximum mass.
00264 *
00265 * For a central galaxy galaxy it just updates its properties. For satellites
00266 * it needs to know its most massive (or only progenitor) to keep track of the
00267 * merging clock. It also finds the central galaxies into which galaxies should
00268 * merge. If MERGE01=1, type 1's can merge if their baryonic mass is bigger than
00269 * the dark matter mass and type 2's can merge into them. Once the type 1's merge

00270 * into a type 0 all its satellites will have the merging clock reset into the
00271 * type 0.
00272 *
00273 */
00274
00275 #ifdef MERGE01
00276 int join_galaxies_of_progenitors(int halonr, int ngalstart, int cenngal)
00277 #else
00278 int join_galaxies_of_progenitors(int halonr, int ngalstart)
00279 #endif
00280 {
00281     int ngal, prog, i, j, first_occupied, lenmax, centralgal, mostmassive;
00282
00283
00284     lenmax = 0;
00285     first_occupied = Halo[halonr].FirstProgenitor;
00286     prog = Halo[halonr].FirstProgenitor;
00287
00288
00289     /* When there is no galaxy in the Halo of FirstProgenitor, the first_occu
00290     pied
00291     * pointer is changed to a subhalo which have the maximum mass (This shou
00292     ld
00293     * only happen in the case that the leaf on the firstprogenitor branch oc
00294     curs
00295     * as a subhalo, in that case no galaxy would be assigned to it). */
00296     if (prog >= 0) //If halo has progenitors
00297     {
00298         if (HaloAux[prog].NGalaxies == 0) //if progenitor has no galaxies
00299
00300             /*Loop through all the progenitors of current halo*/
00301             while (prog >= 0)
00302             {
00303                 for (i = 0; i < HaloAux[prog].NGalaxies; i++)//When nex
00304                 t progenitors have galaxies
00305
00306                 if(HaloGal[HaloAux[prog].FirstGalaxy + i].Type == 0
00307                     || HaloGal[HaloAux[prog].
00308                     FirstGalaxy + i].Type == 1)
00309
00310                     if (Halo[prog].Len > lenmax)
00311                     {
00312                         lenmax = Halo[prog].Len;
00313                         first_occupied = prog;//define the new f
00314                         irst_occupied
00315
00316                     }
00317
00318                 }
00319             }
00320             prog = Halo[prog].NextProgenitor;

```

```

00312             }
00313         }
00314     /* TODO is it true that IF the progenitor halos are sorted by Len
00315      * (apart maybe from firstprogenitor), then the first_occupied's FirstGalaxy
00316      will also be
00317      * the central galaxy of the new halo?*/
00318
00319
00320
00321     lenmax = 0;
00322     prog = Halo[halonr].FirstProgenitor;
00323     mostmassive = Halo[halonr].FirstProgenitor;
00324
00325     /* loop through all the progenitors and get the halo mass and ID
00326      * of the most massive*/
00327     while (prog >= 0)
00328     {
00329         if (Halo[prog].Len > lenmax)
00330         {
00331             lenmax = Halo[prog].Len;
00332             mostmassive = prog;
00333         }
00334         prog = Halo[prog].NextProgenitor;
00335     }
00336
00337     ngal = ngalstart;
00338     prog = Halo[halonr].FirstProgenitor;
00339
00340     while (prog >= 0)
00341     {
00342         for (i = 0; i < HaloAux[prog].NGalaxies; i++)
00343         {
00344             if (ngal >= FoF_MaxGals)
00345             {
00346                 printf(
00347                     "Oops. We reached the maximum number FoF_MaxGals=%d of galaxies in FoF... exiting.\n",
00348                                         FoF_MaxGals);
00349                 exit(1);
00350             }
00351
00352             Gal[ngal] = HaloGal[HaloAux[prog].FirstGalaxy + i];
00353             Gal[ngal].HaloNr = halonr;
00354             Gal[ngal].CoolingRadius = 0.0;
00355             Gal[ngal].CoolingGas = 0.0;
00356
00357             /* for the largest tree in MRII, we cannot record 68 snapshots for galaxy
00358             tree -- memory issue */
00359 #ifdef SAVE_MEMORY
00360             Gal[ngal].Sfr = 0.0;
00361             Gal[ngal].SfrBulge = 0.0;
00362 #endif
00363 #ifdef GALAXYTREE
00364             Gal[ngal].NextProgGal = -1;
00365             Gal[ngal].FirstProgGal = HaloAux[prog].FirstGalaxy + i;
00366 #endif
00367
00368             /* Update Properties of this galaxy with physical properties of halo */
00369             /* this deals with the central galaxies of subhalos */
00370             if(Gal[ngal].Type == 0 || Gal[ngal].Type == 1)
00371             {
00372                 if (prog == first_occupied)
00373                 {
00374                     Gal[ngal].MostBoundID = Halo[halonr].MostBoundID;
00375                     for (j = 0; j < 3; j++) {

```

```

00375                               Gal[ngal].Pos[j] = Halo[halonr].Pos[j];
00376                               Gal[ngal].Vel[j] = Halo[halonr].Vel[j];
00377                           }
00378
00379                               Gal[ngal].Len = Halo[halonr].Len;
00380
00381                               if(halonr == Halo[halonr].FirstHaloInFOFgroup)
00382                                     // TODO this could be place where
00383                                     to set this (FOF-central-)subhalo's
00384                                     // FOFCentralGal property in case
00385                                     that is different from FirstGalaxy
00386                                     update_centralgal(ngal,halonr);
00387                                     else
00388 #ifdef MERGE01
00389                                     update_type_1(ngal,halonr,cenngal,prog);
00390 #else
00391                                     update_type_1(ngal,halonr,0,prog);
00392                                     if (DiskRadiusMethod == 0 ||
00393                                         DiskRadiusMethod == 1) {
00394                                         Gal[ngal].GasDiskRadius = get_disk_radius(halonr, n
00395                                         gal);
00396                                         Gal[ngal].StellarDiskRadius = Gal[ngal].
00397                                         GasDiskRadius;
00398                                     }
00399                                     else
00400                                     {
00401                                         update_type_2(ngal,halonr,prog,mostmassive);
00402                                     }
00403
00404 /* Note: Galaxies that are already type=2 do not need a special t
00405 reatment at this point */
00406
00407                               if (Gal[ngal].Type < 0 || Gal[ngal].Type > 2) {
00408                                 printf("what's that????\n");
00409                                 exit(88);
00410                               }
00411                               ngal++;
00412
00413 } // end for
00414
00415     prog = Halo[prog].NextProgenitor;
00416 } //end while
00417
00418 /* If there are no progenitors with galaxies, a new galaxy is created.
00419 * However, if it's a subhalo, no galaxy is placed, since it would stay
00420 * at zero luminosity. */
00421
00422     if (ngal == 0) {
00423       if (Halo[halonr].FirstHaloInFOFgroup == halonr) {
00424         init_galaxy(ngal, halonr);
00425         ngal++;
00426       }
00427
00428 /* If the baryonic mass of a type 1 is comparable to its dark matter mass
00429 * and its merging clock for it is switched on, its satellites (type 2's)
00430 * will preferably merge onto this type 1 rather than the type 0 */
00431
00432     for(i = ngalstart, centralgal = -1; i < ngal; i++)
00433       if (Gal[i].Type == 0 || Gal[i].Type == 1)
00434       {
00435         if (centralgal != -1) {
00436           printf("can't be!\n");
00437           exit(8);
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01940
01941
01942
01943
01944
01945
01946
01947
01948
01949
01950
01951
01952
01953
01954
01955
01956
01957
01958
01959
01960
01961
01962
01963
01964
01965
01966
01967
01968
01969
01970
01971
01972
01973
01974
01975
01976
01977
01978
01979
01980
01981
01982
01983
01984
01985
01986
01987
01988
01989
01990
01991
01992
01993
01994
01995
01996
01997
01998
01999
02000
02001
02002
02003
02004
02005
02006
02007
02008
02009
02010
02011
02012
02013
02014
02015
02016
02017
02018
02019
02020
02021
02022
02023
02024
02025
02026
02027
02028
02029
02030
02031
02032
02033
02034
02035
02036
02037
02038
02039
02040
02041
02042
02043
02044
02045
02046
02047
02048
02049
02050
02051
02052
02053
02054
02055
02056
02057
02058
02059
02060
02061
02062
02063
02064
02065
02066
02067
02068
02069
02070
02071
02072
02073
02074
02075
02076
02077
02078
02079
02080
02081
02082
02083
02084
02085
02086
02087
02088
02089
02090
02091
02092
02093
02094
02095
02096
02097
02098
02099
02100
02101
02102
02103
02104
02105
02106
02107
02108
02109
02110
02111
02112
02113
02114
02115
02116
02117
02118
02119
02120
02121
02122
02123
02124
02125
02126
02127
02128
02129
02130
02131
02132
02133
```

```

00436             }
00437         centralgal = i;
00438     }
00439
00440     for (i = ngalstart; i < ngal; i++) {
00441         Gal[i].CentralGal = centralgal;
00442         if (centralgal != -1)
00443             for(j = 0; j<3; j++)
00444                 Gal[i].MergCentralPos[j] = Gal[centralgal].Pos[j];
00445
00446     }
00447
00448 /* Satellites whose type 1 has merged into type 0, will be reset to merge
00449 * into the type 0 later. */
00450 #ifdef MERGE01
00451     if (centralgal == -1 && ngal != ngalstart)
00452     {
00453         for (i = ngalstart; i < ngal; i++)
00454         {
00455             Gal[i].CentralGal = cenngal;
00456             for(j = 0; j<3; j++)
00457                 Gal[i].MergCentralPos[j] = Gal[cenngal].Pos[j];
00458
00459         }
00460     }
00461 #endif
00462     return ngal;
00463 }
00464 }
00465
00466 /**@brief evolve_galaxies deals with most of the SA recipes
00467 *
00468 * This is where most of the physical recipes are called, including:
00469 * infall_recipe (gets the fraction of primordial gas that infalled),
00470 * add_infall_to_hot (adds the infalled gas to the hot phase),
00471 * reincorporate_gas (reincorporates gas ejected by SN), cooling_recipe
00472 * (gets the amount of gas that cooled - takes into account AGN feedback),
00473 * cool_gas_onto_galaxy (adds the gas that cooled into the cold phase),
00474 * starformation_and_feedback (normal SF and SN feedback),
00475 * deal_with_galaxy_merger (adds components, grows black hole and deals
00476 * with SF burst), disruption (total and instantaneous disruption of
00477 * type 2 satellites), dust (if galaxy is in an output time, dust
00478 * extinction is computed).
00479 *
00480 * All these calculations are done in time steps of 1/STEPS the time
00481 * between each snapshot (STEPS=20). */
00482
00483 /* note: halonr is here the FOF-background subhalo (i.e. main halo) */
00484 void evolve_galaxies(int halonr, int ngal,int tree,int cenngal)
00485 {
00486     int p,q,ind, nstep, centralgal, merger_centralgal, n, currenthalo;
00487     double infallingGas, coolingGas, deltaT, Zcurr, time,ptime,previoustime;
00488     double starmass;
00489     int ip;
00490
00491     previoustime = NumToTime(Gal[0].SnapNum);
00492     ptime = NumToTime(Halo[halonr].SnapNum);
00493
00494     deltaT = NumToTime(Gal[0].SnapNum) - NumToTime(Halo[halonr].SnapNum);
00495     Zcurr = ZZ[Halo[halonr].SnapNum];
00496
00497     centralgal = Gal[0].CentralGal;
00498
00499
00500     if (Gal[centralgal].Type != 0 || Gal[centralgal].HaloNr != halonr) {
00501         printf("Something wrong here ..... \n");
00502         exit(54);

```

```

00503     }
00504
00505     /* Compute the diff between the hot gas obtained at the end of the previous
00506     * snapshot and the one obtained at the beginning of the new snapshot using
00507     * conservation of the baryons.
00508     * NOTE that it is assumed that the hot gas can be diluted but it is also
00509     * possible that the metallicity increases because the hot gas decreases.
00510     * TODO If we take away gas, shouldn't we take away metals? */
00511     infallingGas = infall_recipe(centralgal, ngal, Zcurr);
00512
00513     for (p = 0; p < ngal; p++) {
00514         if (Gal[p].Type == 2 && (Gal[p].EjectedMass > 1.e-8
00515                                     || Gal[p].MetalsEjectedMass > 1.e-8)) {
00516             printf("wrong in infall main.c \n");
00517             exit(0);
00518         }
00519     }
00520     for (p = 0; p < ngal; p++)
00521         /* for mass=0 type 2 galaxies, we drops it to save the calculation time and m
        emory*/
00522
00523     {
00524         if (Gal[p].Type == 2 && Gal[p].ColdGas + Gal[p].StellarMass < 1.e-8)
00525             Gal[p].Type = 3;
00526     }
00527
00528
00529     /* All the physics are computed in a number of intervals between snapshots
00530     * equal to STEPS */
00531     for (nstep = 0; nstep < STEPS; nstep++) {
00532
00533
00534
00535     /* infall for central galaxy only */
00536     add_infall_to_hot(centralgal, infallingGas / STEPS);
00537
00538         for (p = 0; p < ngal; p++) {
00539             /* don't treat galaxies that have already merged */
00540             if (Gal[p].Type == 3)
00541                 continue;
00542             /* time between snapshots in 1e12 Yrs*/
00543             deltaT = NumToTime(Gal[p].SnapNum) - NumToTime(Halo[halonr].SnapNum);
00544             /* time of the current step*/
00545             time = NumToTime(Gal[p].SnapNum) - (nstep + 0.5) * (deltaT / STEPS);
00546
00547                 if (Gal[p].Type == 0 || Gal[p].Type == 1) {
00548
00549                 /* reincorporation for central galaxy and satellites beyond r200 of
        the main halo*/
00550                 if (ReIncorporationFactor > 0)
00551                     reincorporate_gas(centralgal, p, deltaT / STEPS);
00552
00553
00554
00555                 /* determine cooling gas given halo properties and add it to the co
        ld phase*/
00556                 coolingGas = cooling_recipe(p, deltaT / STEPS);
00557                 cool_gas_onto_galaxy(p, coolingGas);
00558
00559             }
00560
00561
00562
00563             /* stars form and then explode! */
00564             starformation_and_feedback(p, centralgal, time, deltaT / STEPS, halonr)
00565 ;

```

```

00566
00567         }
00568
00569
00570     /* Check for merger events */
00571
00572     for (p = 0; p < ngal; p++) {
00573
00574         if(Gal[p].Type == 2 || (Gal[p].Type == 1 && Gal[p].MergeOn == 1))      /
00575         * satellite galaxy */
00576         {
00577             deltaT = NumToTime(Gal[p].SnapNum) - NumToTime(Halo[halonr].SnapNum
00578         );
00579
00580             Gal[p].MergTime -= deltaT / STEPS;
00581
00582             /* a merger has occurred! */
00583             if (Gal[p].MergTime < 0.0) {
00584                 NumMergers++;
00585
00586                 if(Gal[p].Type == 1)
00587                     for (q=0 ;q < ngal; q++)
00588                     if (Gal[q].Type == 2 && Gal[p].CentralGal == p)
00589                         Gal[q].CentralGal = cengal;
00590
00591                 if (Gal[p].Type ==2)
00592                     merger_centralgal = Gal[p].CentralGal;
00593                 else
00594                     merger_centralgal = cengal;
00595                 deltaT = NumToTime(Gal[p].SnapNum) - NumToTime(Halo[halonr].Sna
00596                 pNum);
00597
00598                 time = NumToTime(Gal[p].SnapNum) - (nstep + 0.5) * (deltaT / ST
00599                 EPS);
00600
00601                 /*TODO should deltaT/STEPS be passed here?*/
00602                 /*TODO - the star formation during mergers will be divided by d
00603                 eltaT, then at save
00604                 * is converted using Unit_time_in_s !!!!!!!!!!!!!*/
00605
00606                 deal_with_galaxy_merger(p, merger_central
00607                 gal, centralgal, time,
00608
00609             }
00610
00611             }/* end move forward in interval STEPS */
00612
00613             /* check the bulge size*/
00614             //checkbulgesize_main(ngal);
00615
00616             /* Disruption of type 2 galaxies. Type 1 galaxies are not disrupted since usual
00617             ly
00618             * bayonic component is more compact than dark matter.*/
00619             #ifdef DISRUPTION
00620             for (p= 0; p < ngal; p++)
00621             if(Gal[p].Type == 2)
00622                 disrupt(p, Gal[p].CentralGal,tree);
00623
00624             /* If this is an output snapshot apply the dust model to each galaxy */
00625             for (n = 0; n < NOUT; n++) {

```

```

00626             if (Halo[halonr].SnapNum == ListOutputSamps[n]) {
00627                 for(p = 0; p < ngal; p++) {
00628                     //dust_model(p, n);
00629                     tsudy(p, n, halonr);
00630                     break;
00631                 }
00632             }
00633
00634             for (p = 0, currenthalo = -1; p < ngal; p++) {
00635                 if (Gal[p].HaloNr != currenthalo) {
00636                     currenthalo = Gal[p].HaloNr;
00637                     HaloAux[currenthalo].FirstGalaxy = NumGals;
00638                     HaloAux[currenthalo].NGalaxies = 0;
00639                 }
00640             /* may be wrong */
00641
00642             if (Gal[p].Type != 3) {
00643                 if (NumGals >= MaxGals) {
00644                     printf("maximum number of galaxies reached... exiting.\n"
00645                         );
00646                     exit(1);
00647                 }
00648             Gal[p].SnapNum = Halo[currenthalo].SnapNum;
00649
00650             HaloGal[NumGals++] = Gal[p];
00651             HaloAux[currenthalo].NGalaxies++;
00652
00653         }
00654     }
00655 }
00656
00657
00658 /*calculate the gas recycle using BC03 table */
00659 #ifdef GASRECYCLE
00660     cal_gas_recycle(ngal);
00661 #endif
00662
00663 }
00664 /**
00665 * @brief Check whether makefile options are compatible.
00666 */
00667 void check_options() {
00668 #ifdef OUTPUT_OBS_MAGS
00669 #ifndef COMPUTE_OBS_MAGS
00670     printf("> Error : option OUTPUT_OBS_MAGS requires option COMPUTE_OBS_MAGS \n");
00671     exit(32);
00672 #endif
00673 #endif
00674
00675 #ifdef OUTPUT_MOMAF_INPUTS
00676 #ifndef COMPUTE_OBS_MAGS
00677     printf("> Error : option OUTPUT_MOMAF_INPUTS requires option COMPUTE_OBS_MAGS \
00678 n");
00679     exit(32);
00680 #endif
00681
00682 #ifdef OUTPUT_L_CONE_INPUTS
00683 #ifndef OUTPUT_OBS_MAGS
00684     printf("> Error : option OUTPUT_L_CONE_INPUTS requires option OUTPUT_OBS_MAGS \
00685 n");
00686     exit(32);
00687 #endif
00688

```

```

00689 #ifdef OUTPUT_L_CONE_INPUTS
00690 #ifndef GALAXYTREE
00691     printf("> Error : option OUTPUT_L_CONE_INPUTS requires option GALAXYTREE \n");
00692     exit(32);
00693 #endif
00694 #endif
00695
00696 #ifdef NEW_IO
00697 #ifndef GALAXYTREE
00698     printf("> Error : option NEW_IO requires option GALAXYTREE \n");
00699     exit(32);
00700 #endif
00701 #ifdef USE_MEMORY_TO_MINIMIZE_IO
00702     printf("> Error : option NEW_IO requires option USE_MEMORY_TO_MINIMIZE_IO to be
00703         false \n");
00704     exit(32);
00705 #endif
00706
00707 #ifdef SAVE_MEMORY
00708 #ifdef UseFullSfr
00709     printf("> Error : SAVE_MEMORY and UseFullSfr could not be switched on together!
00710         \n");
00711     exit(32);
00712 #endif
00713
00714 #ifndef DISRUPTION
00715 #ifdef ICL
00716     printf("> Warning : DISRUPTION off then ICL makes no sense \n");
00717 #endif
00718 #endif
00719 }
00720
00721
00722

```

4.15 code/mymalloc.c File Reference

Functions

- void [print_allocated](#) (void)
- void * [mymalloc](#) (size_t n, char *myflag)
- void [myfree](#) (void *p)

Variables

- static unsigned long [Nblocks](#) = 0
- static void * [Table](#) [MAXBLOCKS]
- static size_t [SizeTable](#) [MAXBLOCKS]
- static size_t [TotMem](#) = 0
- static size_t [HighMarkMem](#) = 0
- static size_t [OldPrintedHighMark](#) = 0

4.15.1 Function Documentation

4.15.1.1 void myfree (void * *p*)

Definition at line 66 of file [mymalloc.c](#).

References [Nblocks](#), [SizeTable](#), [Table](#), and [TotMem](#).

Referenced by [free_galaxies_and_tree\(\)](#), [free_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.15.1.2 void* mymalloc (size_t *n*, char * *myflag*)

Definition at line 26 of file [mymalloc.c](#).

References [HighMarkMem](#), [Nblocks](#), [OldPrintedHighMark](#), [SizeTable](#), [Table](#), and [TotMem](#).

Referenced by [load_all_auxdata\(\)](#), [load_all_dbids\(\)](#), [load_all_treedata\(\)](#), [load_tree\(\)](#), [load_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.15.1.3 void print_allocated (void)

Definition at line 20 of file [mymalloc.c](#).

References [TotMem](#).

4.15.2 Variable Documentation

4.15.2.1 size_t HighMarkMem = 0 [static]

Definition at line 18 of file [mymalloc.c](#).

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

4.15.2.2 unsigned long Nblocks = 0 [static]

Definition at line 12 of file [mymalloc.c](#).

Referenced by [myfree\(\)](#), and [mymalloc\(\)](#).

4.15.2.3 size_t OldPrintedHighMark = 0 [static]

Definition at line 18 of file [mymalloc.c](#).

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

4.15.2.4 size_t SizeTable[MAXBLOCKS] [static]

Definition at line 16 of file [mymalloc.c](#).

Referenced by [myfree\(\)](#), and [mymalloc\(\)](#).

4.15.2.5 void* Table[MAXBLOCKS] [static]

Definition at line 14 of file [mymalloc.c](#).

Referenced by [myfree\(\)](#), and [mymalloc\(\)](#).

4.15.2.6 size_t TotMem = 0 [static]

Definition at line 18 of file [mymalloc.c](#).

Referenced by [myfree\(\)](#), [mymalloc\(\)](#), and [print_allocated\(\)](#).

4.16 code/mymalloc.c

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <string.h>
00004 #include <math.h>
00005
00006 #include "allvars.h"
00007 #include "proto.h"
00008
00009
00010 #define MAXBLOCKS 256
00011
00012 static unsigned long Nblocks = 0;
00013
00014 static void *Table[MAXBLOCKS];
00015
00016 static size_t SizeTable[MAXBLOCKS];
00017
00018 static size_t TotMem = 0, HighMarkMem = 0, OldPrintedHighMark = 0;
00019
00020 void print_allocated(void)
00021 {
00022     printf("allocated = %g MB\n", TotMem / (1024.0 * 1024.0));
00023 }
00024
00025
00026 void *mymalloc(size_t n, char *myflag)
00027 {
00028     if((n % 8) > 0)
00029         n = (n / 8 + 1) * 8;
00030
00031     if(n == 0)
00032         n = 8;
00033
00034     if(Nblocks >= MAXBLOCKS)
00035     {
00036         printf("No blocks left in mymalloc().\n");
00037         printf("Called from %s\n", myflag);
00038         exit(1);
00039     }
00040
00041     SizeTable[Nblocks] = n;
00042     TotMem += n;
00043     if(TotMem > HighMarkMem)
00044     {
00045         HighMarkMem = TotMem;
00046         if(HighMarkMem > OldPrintedHighMark + 10 * 1024.0 * 1024.0)
00047             {
00048                 printf("new high mark = %g MB\n", HighMarkMem / (1024.0 * 1024.0));

```

```

00049         OldPrintedHighMark = HighMarkMem;
00050     }
00051 }
00052
00053 if(! (Table[Nblocks] = malloc(n)))
00054 {
00055     printf("Failed to allocate memory for %g MB.\n", n / (1024.0 * 1024.0) );
00056     printf("Called from %s\n",myflag);
00057     exit(2);
00058 }
00059
00060 Nblocks += 1;
00061
00062 return Table[Nblocks - 1];
00063 }
00064
00065
00066 void myfree(void *p)
00067 {
00068     if(Nblocks == 0)
00069         exit(1);
00070
00071     if(p != Table[Nblocks - 1])
00072     {
00073         printf("Wrong call of myfree() - not the last allocated block!\n");
00074         exit(1);
00075     }
00076
00077     free(p);
00078
00079     Nblocks -= 1;
00080
00081     TotMem -= SizeTable[Nblocks];
00082 }
00083
00084

```

4.17 code/peano.c File Reference

Functions

- int `peano_hilbert_key` (int x, int y, int z, int bits)

Variables

- static char `quadrants` [24][2][2][2]
- static char `rotxmap_table` [24]
- static char `rotymap_table` [24]
- static char `rotx_table` [8] = { 3, 0, 0, 2, 2, 0, 0, 1 }
- static char `roty_table` [8] = { 0, 1, 1, 2, 2, 3, 3, 0 }
- static char `sense_table` [8] = { -1, -1, -1, +1, +1, -1, -1, -1 }

4.17.1 Function Documentation

4.17.1.1 int `peano_hilbert_key` (int x, int y, int z, int bits)

Definition at line 61 of file `peano.c`.

References [quadrants](#), [rotx_table](#), [rotxmap_table](#), [roty_table](#), [rotymap_table](#), and [sense_table](#).

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

4.17.2 Variable Documentation

4.17.2.1 char quadrants[24][2][2][2] [static]

Definition at line 13 of file [peano.c](#).

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

4.17.2.2 char rotx_table[8] = { 3, 0, 0, 2, 2, 0, 0, 1 } [static]

Definition at line 55 of file [peano.c](#).

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

4.17.2.3 char rotxmap_table[24] [static]

Initial value:

```
{ 4, 5, 6, 7, 8, 9, 10, 11,
  12, 13, 14, 15, 0, 1, 2, 3, 17, 18, 19, 16, 23, 20, 21, 22
}
```

Definition at line 47 of file [peano.c](#).

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

4.17.2.4 char roty_table[8] = { 0, 1, 1, 2, 2, 3, 3, 0 } [static]

Definition at line 56 of file [peano.c](#).

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

4.17.2.5 char rotymap_table[24] [static]

Initial value:

```
{ 1, 2, 3, 0, 16, 17, 18, 19,
  11, 8, 9, 10, 22, 23, 20, 21, 14, 15, 12, 13, 4, 5, 6, 7
}
```

Definition at line 51 of file [peano.c](#).

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

4.17.2.6 char sense_table[8] = { -1, -1, -1, +1, +1, -1, -1, -1 } [static]

Definition at line 58 of file [peano.c](#).

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

4.18 code/peano.c

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <string.h>
00004 #include <math.h>
00005 #include <time.h>
00006
00007 #include "allvars.h"
00008 #include "proto.h"
00009
00010
00011
00012
00013 static char quadrants[24][2][2][2] = {
00014     /* rotx=0, roty=0-3 */
00015     {{0, 7}, {1, 6}}, {{3, 4}, {2, 5}},
00016     {{7, 4}, {6, 5}}, {{0, 3}, {1, 2}},
00017     {{4, 3}, {5, 2}}, {{7, 0}, {6, 1}},
00018     {{3, 0}, {2, 1}}, {{4, 7}, {5, 6}},
00019     /* rotx=1, roty=0-3 */
00020     {{1, 0}, {6, 7}}, {{2, 3}, {5, 4}},
00021     {{0, 3}, {7, 4}}, {{1, 2}, {6, 5}},
00022     {{3, 2}, {4, 5}}, {{0, 1}, {7, 6}},
00023     {{2, 1}, {5, 6}}, {{3, 0}, {4, 7}},
00024     /* rotx=2, roty=0-3 */
00025     {{6, 1}, {7, 0}}, {{5, 2}, {4, 3}},
00026     {{1, 2}, {0, 3}}, {{6, 5}, {7, 4}},
00027     {{2, 5}, {3, 4}}, {{1, 6}, {0, 7}},
00028     {{5, 6}, {4, 7}}, {{2, 1}, {3, 0}},
00029     /* rotx=3, roty=0-3 */
00030     {{7, 6}, {0, 1}}, {{4, 5}, {3, 2}},
00031     {{6, 5}, {1, 2}}, {{7, 4}, {0, 3}},
00032     {{5, 4}, {2, 3}}, {{6, 7}, {1, 0}},
00033     {{4, 7}, {3, 0}}, {{5, 6}, {2, 1}},
00034     /* rotx=4, roty=0-3 */
00035     {{6, 7}, {5, 4}}, {{1, 0}, {2, 3}},
00036     {{7, 0}, {4, 3}}, {{6, 1}, {5, 2}},
00037     {{0, 1}, {3, 2}}, {{7, 6}, {4, 5}},
00038     {{1, 6}, {2, 5}}, {{0, 7}, {3, 4}},
00039     /* rotx=5, roty=0-3 */
00040     {{2, 3}, {1, 0}}, {{5, 4}, {6, 7}},
00041     {{3, 4}, {0, 7}}, {{2, 5}, {1, 6}},
00042     {{4, 5}, {7, 6}}, {{3, 2}, {0, 1}},
00043     {{5, 2}, {6, 1}}, {{4, 3}, {7, 0}}
00044 };
00045
00046
00047 static char rotxmap_table[24] = { 4, 5, 6, 7, 8, 9, 10, 11,
00048     12, 13, 14, 15, 0, 1, 2, 3, 17, 18, 19, 16, 23, 20, 21, 22
00049 };
00050
00051 static char rotymap_table[24] = { 1, 2, 3, 0, 16, 17, 18, 19,
00052     11, 8, 9, 10, 22, 23, 20, 21, 14, 15, 12, 13, 4, 5, 6, 7
00053 };
00054
00055 static char rotx_table[8] = { 3, 0, 0, 2, 2, 0, 0, 1 };
00056 static char roty_table[8] = { 0, 1, 1, 2, 2, 3, 3, 0 };
00057
00058 static char sense_table[8] = { -1, -1, -1, +1, +1, -1, -1, -1 };
00059
00060
00061 int peano_hilbert_key(int x, int y, int z, int bits)
00062 {
00063     int i, bitx, bity, bitz, mask, quad, rotation;
00064     char sense, rotx, roty;
00065     int key;

```

```

00066     mask = 1 << (bits - 1);
00067     key = 0;
00068     rotation = 0;
00069     sense = 1;
00070
00071
00072     for(i = 0; i < bits; i++, mask >>= 1)
00073     {
00074         bitx = (x & mask) ? 1 : 0;
00075         bity = (y & mask) ? 1 : 0;
00076         bitz = (z & mask) ? 1 : 0;
00077
00078         quad = quadrants[rotation][bitx][bity][bitz];
00079
00080         key <= 3;
00081         key += (sense == 1) ? (quad) : (7 - quad);
00082
00083         rotx = rotx_table[quad];
00084         roty = roty_table[quad];
00085         sense *= sense_table[quad];
00086
00087         while(rotx > 0)
00088         {
00089             rotation = rotxmap_table[rotation];
00090             rotx--;
00091         }
00092
00093         while(roty > 0)
00094         {
00095             rotation = rotymap_table[rotation];
00096             roty--;
00097         }
00098     }
00099
00100     return key;
00101 }
```

4.19 code/proto.h File Reference

Functions

- size_t **myfread** (void *ptr, size_t size, size_t nmemb, FILE *stream)

Reading routine, either from a file into a structure or from a pointer to a structure.
- size_t **myfwrite** (void *ptr, size_t size, size_t nmemb, FILE *stream)

Writing routine, either from a structure to a file or from a structure to a pointer.
- int **myfseek** (FILE *stream, long offset, int whence)

Routine to seek on either a pointer created to store all the data for a file (if USE_MEMORY_TO_MINIMIZE_IO is ON) or to simple seek on a file (if USE_MEMORY_TO_MINIMIZE_IO is OFF).
- void **load_all_auxdata** (int filenr)

If USE_MEMORY_TO_MINIMIZE_IO ON - Routine to read in all the tree_aux data for one file into a pointer (ptr_auxdata) - later myfread() will pass the data onto each tree structure.
- void **load_all_dbids** (int filenr)

If USE_MEMORY_TO_MINIMIZE_IO ON - Routine to read in all the tree_dbids data for one file into a pointer (ptr_dbids) - later myfread() will pass the data onto each tree structure.

- void `load_all_treedata` (int filenr)

*If USE_MEMORY_TO_MINIMIZE_IO ON - Routine to read in all the trees_** data for one file into a pointer (ptr_treedata) - later `myfread()` will pass the data onto each tree structure; The galaxy pointers are allocated.*

- void `write_all_galaxy_data` (int filenr)

If USE_MEMORY_TO_MINIMIZE_IO ON, after all the trees have been computed and myfwrite as written the data on every tree into a single pointer, this pointer is written out in one go - this function is used for GALAXYTREE ON.

- void `write_galaxy_data_snap` (int n, int filenr)

If USE_MEMORY_TO_MINIMIZE_IO ON, after all the trees have been computed and myfwrite as written the data on every tree into a single pointer, this pointer is written out in one go - this function is used for normal SNAP OUTPUT.

- void `save_galaxy_tree` (int filenr, int tree)

Saves galaxies if GALAXY_TREE=1.

- int `walk` (int nr)

Walks up the main leaf of a tree TODO - is that what it actually does?

- int `peano_hilbert_key` (int x, int y, int z, int bits)

- void `get_coordinates` (float *pos, float *vel, long long ID, int tree, int halonr, int snapnum)

get_coordinates receives the positions and velocities of a type 2 galaxy and updates them with the values from the most bound particle identified at disruption time.

- int `join_galaxies_of_progenitors` (int halonr, int nstart)

- int `join_galaxies_of_progenitors` (int halonr, int nstart, int cenngal)

- void `init` (void)

controlling recipe for `init.c`, calls functions to read in tables and defines some SN and AGN feedback parameters.

- void `set_units` (void)

Sets up some variables used to convert from physical to internal units (as UnitDensity_in_cgs); These are obtained from the three defined in input.par: UnitLength_in_cm (cm to Mpc), UnitMass_in_g (g to 1e10Msun) and UnitVelocity_in_cm_per_s (cm/s to km/s).

- void `load_tree_table` (int filenr)

*Reads all the tree files if USE_MEMORY_TO_MINIMIZE_IO ON, otherwise reads in the headers for trees_- ** and trees_aux; Opens output files.*

- void `load_tree` (int filenr, int nr)

Reads the actual trees into structures to be used in the code: Halo and HaloIDs; the galaxy structures (HaloGal and Gal) and the HaloAux are also allocated.

- void `save_galaxies` (int filenr, int tree)

Saves the Galaxy_Output structure for all the galaxies in the current tree into the current output file (one for each input dark matter file) for the chosen snapshots.

- void `prepare_galaxy_for_output` (int n, int filenr, int tree, struct **GALAXY** *g, struct **GALAXY_OUTPUT** *o)

Copies all the relevant properties from the Galaxy structure into the Galaxy output structure, some units are corrected.

- void **fix_units_for_ouput** (struct **GALAXY_OUTPUT** *o)
Removes h from units of galaxy properties.
- void **free_galaxies_and_tree** (void)
Frees all the Halo and Galaxy structures in the code.
- void **free_tree_table** (void)
*Deallocates the arrays used to read in the headers of trees_** and tree_aux (the ones with more than one element); if USE_MEMORY_TO_MINIMIZE_IO ON, deallocates the pointers containing all the input and output data.*
- void **print_allocated** (void)
- void **read_parameter_file** (char *fname)
- void * **mymalloc** (size_t n, char *myflag)
- void **myfree** (void *p)
- void **finalize_galaxy_file** (int filenr)
Writes an header in the output files.
- void **starformation_and_feedback** (int p, int centralgal, double time, double dt, int halonr)
Main recipe, calculates the fraction of cold gas turned into stars due to star formation; the fraction of mass instantaneously recycled and returned to the cold gas; the fraction of gas reheated from cold to hot, ejected from hot to external and returned from ejected to hot due to SN feedback.
- void **do_major_merger_starburst** (int t, int centralgal, double time, double deltaT, int halonr)
Merger burst recipe used before Croton2006 - in a major merger all the cold gas from the central and satellite galaxies is consumed to form a bulge.
- void **add_galaxies_together** (int t, int p)
Adds all the components of the satellite galaxy into its central companion.
- void **add_to_luminosities** (int p, double mstars, double time, double metallicity)
Whenever star formation occurs, calculates the luminosity corresponding to the mass of stars formed, considering the metallicity and age of the material.
- void **init_galaxy** (int p, int halonr)
Initializes the Galaxy Structure by setting all its elements to zero.
- double **infall_recipe** (int centralgal, int ngal, double Zcurr)
- void **add_infall_to_hot** (int centralgal, double infallingGas)
The gas that infalled is added to the hot gas of the central galaxy.
- double **cooling_recipe** (int centralgal, double dt)
main cooling recipe, where the cooling rates are calculated
- void **cool_gas_onto_galaxy** (int centralgal, double mcool)
updates the fractions of hot and cold gas due to cooling.
- void **reincorporate_gas** (int centralgal, int p, double dt)

reincorporates ejected gas back into the central galaxy hot halo

- void `deal_with_galaxy_merger` (int p, int merger_centralgal, int centralgal, double time, double deltaT, int halonr)

Deals with all the physics triggered by mergers.

- double `dmax` (double x, double y)

gets the maximum of two numbers.

- double `do_reionization` (int centralgal, double Zcurr)

- double `do_AGN_heating` (double coolingGas, int centralgal, double dt, double x)

calculates the energy released by black holes due to passive accretion, that will be used to reduced the cooling.

- double `NumToTime` (int snapnum)

Converts snapnum into age - in Myr.

- double `RedshiftObs` (int snapnum)

Converts snapnum into redshift.

- void `update_from_star_formation` (int p, double stars, double metallicity)

Updates the different components due to star formation: mass and metals in stars and cold gas and stellar spin.

- void `construct_galaxies` (int halonr, int tree)

- void `evolve_galaxies` (int halonr, int ngal, int tree, int cenngal)

- void `update_from_feedback` (int p, int centralgal, double reheated_mass, double ejected_mass, double ejected_sat, double metallicity)

Updates cold, hot and external gas components due to SN reheating and ejection.

- double `estimate_merging_time` (int halonr, int mostmassive, int p)

Calculates the merging time whenever a galaxy becomes a satellite.

- double `get_virial_velocity` (int halonr, int gal)

Calculates the virial velocity from the virial mass and radius.

- double `get_virial_radius` (int halonr, int gal)

Calculates virial radius from a critical overdensity.

- double `get_virial_mass` (int halonr, int gal)

*Calculates the virial mass: $M_{\text{crit}200}$ for central halos with $M_{\text{crit}200}$ or $\text{Len} * \text{PartMass}$ for central halos without.*

- double `collisional_starburst_recipe` (double mass_ratio, int merger_centralgal, int centralgal, double time, double deltaT, int halonr)

Merger burst recipe from Somerville 2001 (used after Croton2006).

- void `make_bulge_from_burst` (int p)

In a major merger, both disks are destroyed and all the mass transferred to the bulge.

- void `grow_black_hole` (int halonr, int merger_centralgal, double mass_ratio, double dt)

Grows black holes, through accretion from cold gas during mergers, as in Kauffmann & Haehnelt (2000) - Quasar Mode.

- void [check_disk_instability](#) (int p)
Checks for disk stability using the Mo, Mao & White (1998) criteria.
- double [lum_to_mag](#) (double lum)
Converts luminosities into magnitudes.
- double [slab_model](#) (float tau, float teta)
- double [get_metallicity](#) (double gas, double metals)
Gets metallicity for a given mass of material and metals.
- double [get_disk_radius](#) (int halonr, int p)
This routine is no longer called (after Guo2010).
- void [get_gas_disk_radius](#) (int p)
Updates the gas disk radius.
- void [get_stellar_disk_radius](#) (int p)
Updates the stellar disk radius.
- void [read_output_snaps](#) (void)
Reads in the list of output snapshots from file /input/desired_output_snaps.txt.
- void [read_snap_list](#) (void)
- void [setup_spectrophotometric_model](#) (void)
Reads in the look up tables from Stellar Population Synthesis Models.
- void [read_tsud_tables](#) (void)
Reads in the dust extinction tables.
- void [tsudy](#) (int p, int magbin, int halonr)
main routine where the extinction is calculated
- void [read_cooling_functions](#) (void)
- double [get_metaldependent_cooling_rate](#) (double logTemp, double logZ)
- double [get_rate](#) (int tab, double logTemp)
- double [time_to_present](#) (double z)
- double [integrand_time_to_present](#) (double a, void *param)
- void [find_interpolation_point](#) (double timenow, double timetarget, int *index, double *f1, double *f2)
Used by update_from_recycle() (not supported) to interpolate into the age table from the SSP look up tables.
- void [find_interpolated_lum](#) (double timenow, double timetarget, double metallicity, int *metindex, int *tabindex, double *f1, double *f2, double *fmet1, double *fmet2)
Used by add_to_luminosities() to interpolates into the age and metallicity in the SSP tables.
- void [find_interpolated_obs_lum](#) (double timenow, double timetarget, double metallicity, double redshift, int *metindex, int *tabindex, int *redindex, double *f1, double *f2, double *fmet1, double *fmet2, double *fred1, double *fred2)

Same as [find_interpolated_lum\(\)](#) but also interpolates in redshift; Not needed if the redshifts for the "inverted" kcorrections in the SSP tables correspond to the simulation's redshifts.

- void [init_jump_index](#) (void)
- int [get_jump_index](#) (double age)
- void [disrupt](#) (int p, int centralgal, int tree)
- double [peri_radius](#) (int p, int centralgal)

Calculates the distance of the satellite to the pericentre of the main dark matter halo.

- double [dmin](#) (double x, double y)

Returns the minimum of two numbers.

- double [hot_retain_sat](#) (int i, int centralgal, double infallingMass)

Gradual stripping of hot gas (ejected sgas stripping is proportional) from type I satellites.

- void [check_options](#) ()
- size_t [momaf_fwrite](#) (void *ptr, size_t size, size_t nmemb, FILE *stream)
- void [write_galaxy_for_momaf](#) (int n, int filenr)

If USE_MEMORY_TO_MINIMIZE_IO ON, after all the trees have been computed and myfwrite as written the data on every tree into a single pointer, this pointer is written out in one go - this function is used for OUTPUT_MOMAF_INPUTS ON.

- void [finalize_momaf_file](#) (int filenr)

- void [prepare_galaxy_for_momaf](#) (int n, int filenr, int tree, struct [GALAXY](#) *g, struct [MOMAF_INPUTS](#) *o)

- void [read_recgas](#) (void)

- void [cal_gas_recycle](#) (int ngal)

- double [update_from_recycle](#) (int p, double time, double previoustime)

- double [get_initial_disk_radius](#) (int halonr, int p)

Initiates the value of the disk radius.

- void [update_bulge_from_disk](#) (int p, double stars)

- double [bulge_from_disk](#) (double frac)

- double [func_size](#) (double x, double a)

- void [bulgesize_from_merger](#) (double mass_ratio, int merger_centralgal, int p, double Mcstar, double Mcbulge, double Mcgas, double frac)

Calculates the bulge size after a merger.

- double [diskmass_r](#) (double rmin, double rd, double Sigma0, double dr)

Returns the mass of a disk at a certain radius.

- double [bulgemass_r](#) (double rmin, double re, double Ie, double dr)

Returns the mass of a bulge at a certain radius.

- double [sat_radius](#) (int p)

Calculates the half mass radius of satellite galaxies.

- void [read_file_nrs](#) (void)

- void [read_sfrz](#) (void)

- double [cal_H2](#) (int p)

Model the formation of molecular gas.

- void [find_interpolate_h2](#) (double metalicity, double rho, int *tabindex, int *metindex, double *f1, double *f2, double *fmet1, double *fmet2)
- double [Energy_in_Reheat](#) (int p)
- void [update_type_2](#) (int ngal, int halonr, int prog, int mostmassive)

Updates properties of type 2 galaxies.
- void [update_centralgal](#) (int ngal, int halonr)

Updates properties of central galaxies.
- void [update_hotgas](#) (int ngal, int centralgal)

Not Used - update_hot_frac instead.
- void [update_type_1](#) (int ngal, int halonr, int cenngal, int prog)

Updates properties of type 1 galaxies.
- void [update_ICL](#) (int p, int q)
- void [checkbulgesize_main](#) (int ngal)
- void [update_hot_frac](#) (int p, double reincorporated, float HotGas)
- int [set_merger_center](#) (int fofhalo)

Calculates the central galaxies for type 1's (needed if MERGE01=1).
- void [openallfiles](#) (int filenr)

If NEW_OI ON, this function opens all the tree files and creates file pointers: tree_file, treedbids_file and treeaux_file.
- void [closeallfiles](#) (int filenr)

If NEW_OI ON, this function closes all the files.
- FILE * [open_outputtree_file](#) (int filenr, char mode[])

If NEW_OI ON, this function opens the output file in case GALAXYTREE ON.
- FILE * [open_tree_file](#) (int filenr)

*If NEW_OI ON, this function is called by openallfiles and actually opens trees_**, returning a pointer to the file.*
- FILE * [open_treedbids_file](#) (int filenr)

If NEW_OI ON, this function is called by openallfiles and actually opens tree_dbids, returning a pointer to the file.
- FILE * [open_treeaux_file](#) (int filenr)

If NEW_OI ON, this function is called by openallfiles and actually opens trees_aux, returning a pointer to the file.

4.19.1 Function Documentation

4.19.1.1 void add_galaxies_together (int t, int p)

All the components of the satellite galaxy are added to the correspondent component of the central galaxy. Cold gas spin is updated and a bulge is formed at the central galaxy, with the stars of the satellite if BulgeFormationInMinorMergersOn=1.

Definition at line 434 of file [recipe_mergers.c](#).

References [GALAXY::BlackHoleMass](#), [BulgeFormationInMinorMergersOn](#), [GALAXY::BulgeMass](#), [GALAXY::ColdGas](#), [GALAXY::dObsLum](#), [GALAXY::dObsLumBulge](#), [GALAXY::dObsYLum](#), [GALAXY::dObsYLumBulge](#), [GALAXY::EjectedMass](#), [Gal](#), [GALAXY::GasSpin](#), [GALAXY::HaloSpin](#), [GALAXY::HotGas](#), [GALAXY::ICLLum](#), [GALAXY::ICM](#), [GALAXY::Lum](#), [GALAXY::LumBulge](#), [GALAXY::MassWeightAge](#), [GALAXY::MergeSat](#), [GALAXY::MetalsBulgeMass](#), [GALAXY::MetalsColdGas](#), [GALAXY::MetalsEjectedMass](#), [GALAXY::MetalsHotGas](#), [GALAXY::MetalsICM](#), [GALAXY::MetalsStellarMass](#), [GALAXY::ObsICL](#), [GALAXY::ObsLum](#), [GALAXY::ObsLumBulge](#), [GALAXY::ObsYLum](#), [GALAXY::ObsYLumBulge](#), [GALAXY::Sfr](#), [GALAXY::SfrBulge](#), [GALAXY::StarMerge](#), [GALAXY::StellarMass](#), [GALAXY::YLum](#), and [GALAXY::YLumBulge](#).

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

4.19.1.2 void add_infall_to_hot (int centralgal, double infallingGas)

Definition at line 639 of file [recipe_infall.c](#).

References [Gal](#), [GALAXY::HotGas](#), and [GALAXY::MetalsHotGas](#).

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

4.19.1.3 void add_to_luminosities (int p, double mstars, double time, double metallicity)

The semi-analytic code uses look up tables produced by Evolutionary Population Synthesis Models to convert the mass formed on every star formation episode into a luminosity. Each of These tables corresponds to a simple stellar population i.e, a population with a single metallicity. For a given IMF, metatilicity and age, the tables give the luminosity for a $10^{11} M_{\odot}$ burst. The default model uses a Chabrier IMF and stellar populations from Bruzual & Charlot 2003 with 6 different metallicities.

The magnitudes are immediately calculated for each output bin, so that we know the age of each population that contributed to a galaxy total population: the age between creation and output. Apart from the different ages of the populations at a given output bin, if the option COMPUTE_OBS_MAGS is turned on, then we also need to know the K-corrections (going the opposite directions as in observations) that will affect each population.

For each metallicity there is a look up table which has the different magnitudes for each age and then this is k-corrected to all the snapshots.

If MetallicityOption = 0 -> only solar metallicity. If MetallicityOption = 1 -> 6 metallicities.

Definition at line 341 of file [recipe_misc.c](#).

References [GALAXY::dObsLum](#), [GALAXY::dObsYLum](#), [find_interpolated_lum\(\)](#), [Gal](#), [Hubble_h](#), [ListOutputSnaps](#), [GALAXY::Lum](#), [MagTableZz](#), [GALAXY::MassWeightAge](#), [MetallicityOption](#), [NumToTime\(\)](#), [GALAXY::ObsLum](#), [GALAXY::ObsYLum](#), [RecycleFraction](#), [UnitTime_in_Megayears](#), and [GALAXY::YLum](#).

Referenced by [collisional_starburst_recipe\(\)](#), [do_major_merger_starburst\(\)](#), and [starformation_and_-feedback\(\)](#).

4.19.1.4 double bulge_from_disk (double frac)

4.19.1.5 double bulgemass_r (double r, double rb, double Mbudge, double dr)

Bulge profile -> de Vaucouleurs type $r^{\wedge}\{1/4\}$ law

Definition at line 270 of file [recipe_disrupt.c](#).

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

4.19.1.6 void bulgesize_from_merger (double *mass_ratio*, int *merger_centralgal*, int *p*, double *Mcstar*, double *Mcbulge*, double *Mcgas*, double *frac*)

For any type of merger calculates the new bulge size using Eq. 33 in Guo2010:

$$C \frac{GM_{\text{new,bulge}}^2}{R_{\text{new,bulge}}} = C \frac{GM_1^2}{R_1} + C \frac{GM_2^2}{R_2} + \alpha_{\text{inter}} \frac{GM_1 M_2}{R_1 + R_2}.$$

This implementation assumed that the new bulge occupies the same space as the components that formed it.

Definition at line 1013 of file [recipe_mergers.c](#).

References [GALAXY::BulgeMass](#), [GALAXY::BulgeSize](#), [GALAXY::ColdGas](#), [Gal](#), [GALAXY::GasDiskRadius](#), [GALAXY::HaloNr](#), [GALAXY::StellarDiskRadius](#), [GALAXY::StellarMass](#), and [ThreshMajorMerger](#).

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

4.19.1.7 void cal_gas_recycle (int *ngal*)

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

4.19.1.8 double cal_H2 (int *p*)

Created by Qi, but never used. The table is given by Krumholz.

Definition at line 627 of file [recipe_misc.c](#).

References [GALAXY::ColdGas](#), [find_interpolate_h2\(\)](#), [Gal](#), [GALAXY::GasDiskRadius](#), [get_metallicity\(\)](#), [H2](#), and [GALAXY::MetalsColdGas](#).

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

4.19.1.9 void check_disk_instability (int *p*)

Calculates the stability of the stellar disk as discussed in Mo, Mao & White (1998). For unstable stars, the required amount is transferred to the bulge to make the disk stable again. Mass, metals and luminosities updated. After Guo2010 the bulge size is followed and needs to be updated. Eq 34 & 35 in Guo2010 are used.

Definition at line 654 of file [recipe_starformation_and_feedback.c](#).

References [GALAXY::BulgeMass](#), [GALAXY::BulgeSize](#), [GALAXY::dObsLum](#), [GALAXY::dObsLumBulge](#), [GALAXY::dObsYLum](#), [GALAXY::dObsYLumBulge](#), [G](#), [Gal](#), [get_metallicity\(\)](#), [GALAXY::InfallVmax](#), [GALAXY::Lum](#), [GALAXY::LumBulge](#), [GALAXY::MetalsBulgeMass](#), [GALAXY::MetalsStellarMass](#), [GALAXY::ObsLum](#), [GALAXY::ObsLumBulge](#), [GALAXY::ObsYLum](#), [GALAXY::ObsYLumBulge](#), [GALAXY::StellarDiskRadius](#), [GALAXY::StellarMass](#), [GALAXY::Type](#), [update_bulge_from_disk\(\)](#), [GALAXY::Vmax](#), [GALAXY::YLum](#), and [GALAXY::YLumBulge](#).

Referenced by [deal_with_galaxy_merger\(\)](#), and [starformation_and_feedback\(\)](#).

4.19.1.10 void check_options()

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

4.19.1.11 void checkbulgesize_main(int *ngal*)**4.19.1.12 void closeallfiles(int *filenr*)**

Definition at line [905](#) of file [io_tree.c](#).

References [output_file](#), [tree_file](#), [treeaux_file](#), and [treedbids_file](#).

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

4.19.1.13 double collisional_starburst_recipe(double *mass_ratio*, int *merger_centralgal*, int *centralgal*, double *time*, double *deltaT*, int *halonr*)

If StarBurstRecipe = 1 (since Croton2006), the Somerville 2001 model of bursts is used. The burst can happen for both major and minor mergers, with a fraction of the added cold gas from the satellite and central being consumed. SN Feedback from starformation is computed and the sizes of bulge and disk followed (not done for the other burst mode).

Definition at line [765](#) of file [recipe_mergers.c](#).

References [add_to_luminosities\(\)](#), [GALAXY::ColdGas](#), [EjectionOn](#), [EjectPreVelocity](#), [EjectSlope](#), [Energy_in_Reheat\(\)](#), [EnergySNcode](#), [EtaSNcode](#), [FeedbackEjectionEfficiency](#), [FeedbackEpsilon](#), [FeedbackRecipe](#), [FeedbackReheatingEpsilon](#), [FracZtoHot](#), [Gal](#), [get_metallicity\(\)](#), [Halo](#), [GALAXY::HaloNr](#), [GALAXY::InfallVmax](#), [ListOutputSamps](#), [GALAXY::MetalsColdGas](#), [GALAXY::MetalsHotGas](#), [GALAXY::Pos](#), [ReheatPreVelocity](#), [ReheatSlope](#), [GALAXY::Rvir](#), [GALAXY::Sfr](#), [GALAXY::SnapNum](#), [halo_data::SnapNum](#), [GALAXY::StarMerge](#), [ThreshMajorMerger](#), [GALAXY::Type](#), [update_from_feedback\(\)](#), [update_from_star_formation\(\)](#), [GALAXY::Vmax](#), [GALAXY::Vvir](#), [Yield](#), and [ZZ](#).

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

4.19.1.14 void construct_galaxies(int *halonr*, int *tree*)

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

4.19.1.15 void cool_gas_onto_galaxy(int *p*, double *mcool*)

`cool_gas_onto_galaxy` updates the fractions of hot and cold gas due to cooling. This is done for the mass, metals and, after Guo2010, spin components

Definition at line [276](#) of file [recipe_cooling.c](#).

References [GALAXY::ColdGas](#), [DiskRadiusMethod](#), [Gal](#), [get_gas_disk_radius\(\)](#), [get_metallicity\(\)](#), [GALAXY::HotGas](#), [GALAXY::MetalsColdGas](#), [GALAXY::MetalsHotGas](#), and [update_hot_frac\(\)](#).

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

4.19.1.16 double cooling_recipe(int *centralgal*, double *dt*)

Definition at line [71](#) of file [recipe_cooling.c](#).

References [AGNrecipeOn](#), [CoolingCutoff](#), [GALAXY::CoolingGas](#), [GALAXY::CoolingRadius](#), [CoolingVelocityCutOff](#), [do_AGN_heating\(\)](#), [Gal](#), [get_metaldependent_cooling_rate\(\)](#), [GALAXY::HotGas](#), [GALAXY::HotRadius](#), [GALAXY::MetalsHotGas](#), [ReionizationOn](#), [GALAXY::Rvir](#), [UnitDensity_in_cgss](#), [UnitTime_in_s](#), [GALAXY::Vvir](#), and [GALAXY::XrayLum](#).

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

4.19.1.17 void deal_with_galaxy_merger (int *p*, int *merger_centralgal*, int *centralgal*, double *time*, double *deltaT*, int *halonr*)

Deals with the physics triggered by mergers, according to the mass fraction of the merger ($M_{\text{sat}}/M_{\text{central}} >< 0.3$). Add the cold and stellar phase of the satellite galaxy to the central one, form a bulge at the central galaxy with the stars from the satellite in a minor merger if [BulgeFormationInMinorMergersOn=1](#). Grows black hole through accretion from cold gas "quasar mode". If [StarBurstRecipe = 0](#), all the cold gas is consumed to form stars and these are transferred to the bulge of the new galaxy formed (only for major mergers). If [StarBurstRecipe = 1](#), the Somerville 2001 model of bursts is used, SN Feedback from starformation is computed and the sizes of bulge and disk followed. When a major merger occurs, the disk of both merging galaxies is completely destroyed to form a bulge.

Definition at line 260 of file [recipe_mergers.c](#).

References [add_galaxies_together\(\)](#), [AGNrecipeOn](#), [GALAXY::BulgeMass](#), [GALAXY::BulgeSize](#), [bulgesize_from_merger\(\)](#), [check_disk_instability\(\)](#), [GALAXY::ColdGas](#), [collisional_starburst_recipe\(\)](#), [DiskRadiusMethod](#), [do_major_merger_starburst\(\)](#), [GALAXY::FirstProgGal](#), [Gal](#), [GALAXY::GasDiskRadius](#), [get_gas_disk_radius\(\)](#), [get_stellar_disk_radius\(\)](#), [grow_black_hole\(\)](#), [HaloGal](#), [make_bulge_from_burst\(\)](#), [GALAXY::MergeOn](#), [GALAXY::NextProgGal](#), [StarBurstRecipe](#), [GALAXY::StellarDiskRadius](#), [GALAXY::StellarMass](#), [ThreshMajorMerger](#), [TrackDiskInstability](#), and [GALAXY::Type](#).

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

4.19.1.18 double diskmass_r (double *r*, double *rd*, double *Sigma0*, double *dr*)

Disk profile -> exponential

Definition at line 263 of file [recipe_disrupt.c](#).

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

4.19.1.19 void disrupt (int *p*, int *centralgal*, int *tree*)

Definition at line 50 of file [recipe_disrupt.c](#).

References [GALAXY::CentralGal](#), [GALAXY::ColdGas](#), [GALAXY::DisruptOn](#), [GALAXY::FirstProgGal](#), [Gal](#), [HaloGal](#), [GALAXY::HotGas](#), [GALAXY::ICLLum](#), [GALAXY::ICM](#), [GALAXY::Lum](#), [GALAXY::MetalsColdGas](#), [GALAXY::MetalsHotGas](#), [GALAXY::MetalsICM](#), [GALAXY::MetalsStellarMass](#), [GALAXY::Mvir](#), [GALAXY::NextProgGal](#), [GALAXY::ObsICL](#), [GALAXY::ObsLum](#), [peri_radius\(\)](#), [GALAXY::Rvir](#), [sat_radius\(\)](#), [GALAXY::StellarMass](#), and [GALAXY::Type](#).

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

4.19.1.20 double dmax (double *x*, double *y*)

Definition at line 549 of file [recipe_misc.c](#).

4.19.1.21 double dmin (double *x*, double *y*)

Definition at line 143 of file [cool_func.c](#).

4.19.1.22 double do_AGN_heating (double *coolingGas*, int *centralgal*, double *dt*, double *x*)

`do_AGN_heating` calculates the amount of energy released by black holes due to passive accretion, Which is then used to reduced the cooling.

There is one parameter, `AgnEfficiency`, which is the efficiency of AGN passive accretion and consequently of cooling flow reheating, Eqs. 10, 11 & 12 in Croton 2006. The standard recipe is `AGNRecipeOn` =1 (empirical) with options 2 and 3 representing Bondi-Hoyle and cold cloud accretion. The three should be identical and the use of empirical avoids people shouting about dutty-cycles being incoistent with Bondi-Holy & etc.

Definition at line 178 of file [recipe_cooling.c](#).

References `AgnEfficiency`, `AGNRecipeOn`, `GALAXY::BlackHoleMass`, `G`, `Gal`, `get_metallicity()`, `GALAXY::HotGas`, `GALAXY::HotRadius`, `GALAXY::MetalsHotGas`, `GALAXY::Mvir`, `GALAXY::Rvir`, `UnitEnergy_in_cgs`, `UnitMass_in_g`, `UnitTime_in_s`, and `GALAXY::Vvir`.

Referenced by `cooling_recipe()`.

4.19.1.23 void do_major_merger_starburst (int *p*, int *centralgal*, double *time*, double *deltaT*, int *halonr*)

If `StarBurstRecipe` = 0 (introduced by Kauffmann 1999), the burst only occurs for major mergers. In this case all the cold gas in the satellite and central galaxies is consumed to form stars and these are transferred to the bulge of the new galaxy formed. There is no proper feedback from star formation, just the gas properties updated.

Definition at line 626 of file [recipe_mergers.c](#).

References `add_to_luminosities()`, `GALAXY::CentralGal`, `GALAXY::ColdGas`, `EjectionOn`, `Feedback-Recipe`, `Gal`, `get_metallicity()`, `Halo`, `GALAXY::HaloNr`, `ListOutputSnaps`, `GALAXY::MetalsColdGas`, `GALAXY::MetalsEjectedMass`, `GALAXY::MetalsHotGas`, `GALAXY::MetalsStellarMass`, `GALAXY::Pos`, `RecycleFraction`, `GALAXY::Rvir`, `GALAXY::Sfr`, `GALAXY::SnapNum`, `halo_data::SnapNum`, `StarBurstsInMajorMergersOn`, `GALAXY::StellarMass`, `Yield`, and `ZZ`.

Referenced by `deal_with_galaxy_merger()`.

4.19.1.24 double do_reionization (int *centralgal*, double *Zcurr*)

reionization recipie described in Gnedin (2000), using the fitting

Definition at line 599 of file [recipe_infall.c](#).

References `a0`, `find_interpolate_reionization()`, `Gal`, `GALAXY::Mvir`, `Omega`, and `Reion_Mc`.

Referenced by `infall_recipe()`.

4.19.1.25 double Energy_in_Reheat (int *p*)

Referenced by `collisional_starburst_recipe()`, and `starformation_and_feedback()`.

4.19.1.26 double estimate_merging_time (int *halonr*, int *mostmassive*, int *p*)

Binney & Tremaine 1987 - 7.26 merging time for satellites due to dynamical friction. After Delucia2007 *2, shown to agree with Kolchin2008 simulations in Delucia2010. This is set when a galaxy becomes a type 2 or being a type 1 $M_{\text{star}} > M_{\text{vir}}$. In DeLucia2007 they could only merge into a type 0, now (after guo2010) they can merge into a type 1.

Definition at line 197 of file [recipe_mergers.c](#).

References [halo_data::Descendant](#), [halo_data::FirstProgenitor](#), [G](#), [Gal](#), [get_virial_mass\(\)](#), [get_virial_radius\(\)](#), [get_virial_velocity\(\)](#), [Halo](#), [GALAXY::Len](#), [GALAXY::Pos](#), [halo_data::SnapNum](#), [GALAXY::StellarMass](#), [GALAXY::Type](#), and [ZZ](#).

Referenced by [update_type_1\(\)](#), and [update_type_2\(\)](#).

4.19.1.27 void evolve_galaxies (int *halonr*, int *ngal*, int *tree*, int *cenngal*)

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

4.19.1.28 void finalize_galaxy_file (int *filenr*)

If GALAXYTREE=1 three numbers are written: 1 (int); size_of_struct(Galaxy_Output) (int); and TotGalCount(int). If GALAXYTREE=0 then the header is: Ntrees (int); total number of galaxies for the snapshot corresponding to the current file -> TotGalaxies[n] (int); and the number of galaxies on each tree on the current snapshot -> TreeNgals[n] (int*Ntrees).

If USE_MEMORY_TO_MINIMIZE_IO ON, finalize_galaxy_file also copies the galaxy data stored in pointers into the output files, so that it is all done in one go for a given file. This is done using either write_galaxy_data_snap (for SNAP output), write_all_galaxy_data (for GALAXYTREE option) or write_galaxy_for_momaf (for MOMAF option).

Definition at line 211 of file [save.c](#).

References [FileNameGalaxies](#), [ListOutputSamps](#), [myfwrite\(\)](#), [Ntrees](#), [offset_galaxydata](#), [offset_galsnapdata](#), [open_outputtree_file\(\)](#), [output_file](#), [OutputDir](#), [TotGalaxies](#), [TotGalCount](#), [TreeNgals](#), [write_all_galaxy_data\(\)](#), [write_galaxy_data_snap\(\)](#), and [ZZ](#).

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

4.19.1.29 void finalize_momaf_file (int *filenr*)

Definition at line 287 of file [save.c](#).

References [FileNameGalaxies](#), [ListOutputSamps](#), [myfwrite\(\)](#), [offset_momafdata](#), [OutputDir](#), [TotGalaxies](#), and [write_galaxy_for_momaf\(\)](#).

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

4.19.1.30 void find_interpolate_h2 (double *metality*, double *rho*, int * *tabindex*, int * *metindex*, double * *f1*, double * *f2*, double * *fmet1*, double * *fmet2*)

Definition at line 945 of file [init.c](#).

References [Rho](#).

Referenced by [cal_H2\(\)](#).

4.19.1.31 void find_interpolated_lum (double *timenow*, double *timetarget*, double *metallicity*, int * *metindex*, int * *tabindex*, double * *f1*, double * *f2*, double * *fmet1*, double * *fmet2*)

Definition at line 508 of file [init.c](#).

References [AgeTab](#), and [get_jump_index\(\)](#).

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

4.19.1.32 void find_interpolated_obsLum (double *timenow*, double *timetarget*, double *metallicity*, double *redshift*, int * *metindex*, int * *tabindex*, int * *redindex*, double * *f1*, double * *f2*, double * *fmet1*, double * *fmet2*, double * *fred1*, double * *fred2*)

Definition at line 600 of file [init.c](#).

References [AgeTab](#), [get_jump_index\(\)](#), and [RedshiftTab](#).

4.19.1.33 void find_interpolation_point (double *timenow*, double *timetarget*, int * *index*, double * *f1*, double * *f2*)

Definition at line 460 of file [init.c](#).

References [AgeTab](#), and [get_jump_index\(\)](#).

4.19.1.34 void fix_units_for_output (struct GALAXY_OUTPUT * *o*)

If desired (makefile option FIX_OUTPUT_UNITS is set), the output properties of the galaxies can be scaled to physical units excluding any factors of h == Hubble/100 km/s/Mpc.

Definition at line 703 of file [save.c](#).

References [GALAXY_OUTPUT::BlackHoleMass](#), [GALAXY_OUTPUT::BulgeMass](#), [GALAXY_OUTPUT::BulgeSize](#), [GALAXY_OUTPUT::CentralMvir](#), [GALAXY_OUTPUT::ColdGas](#), [GALAXY_OUTPUT::CoolingRadius](#), [GALAXY_OUTPUT::EjectedMass](#), [GALAXY_OUTPUT::GasDiskRadius](#), [GALAXY_OUTPUT::GasSpin](#), [GALAXY_OUTPUT::HotGas](#), [GALAXY_OUTPUT::HotRadius](#), [Hubble_h](#), [GALAXY_OUTPUT::ICM](#), [GALAXY_OUTPUT::MetalsBulgeMass](#), [GALAXY_OUTPUT::MetalsColdGas](#), [GALAXY_OUTPUT::MetalsEjectedMass](#), [GALAXY_OUTPUT::MetalsHotGas](#), [GALAXY_OUTPUT::MetalsICM](#), [GALAXY_OUTPUT::MetalsStellarMass](#), [GALAXY_OUTPUT::Mvir](#), [GALAXY_OUTPUT::Pos](#), [GALAXY_OUTPUT::Rvir](#), [GALAXY_OUTPUT::StellarDiskRadius](#), [GALAXY_OUTPUT::StellarMass](#), and [GALAXY_OUTPUT::StellarSpin](#).

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

4.19.1.35 void free_galaxies_and_tree (void)

Definition at line 432 of file [io_tree.c](#).

References [Gal](#), [Halo](#), [HaloAux](#), [HaloGal](#), [HaloIDs](#), and [myfree\(\)](#).

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

4.19.1.36 void free_tree_table (void)

Definition at line 248 of file [io_tree.c](#).

References [CountIDs_halo](#), [CountIDs_snaptree](#), [myfree\(\)](#), [OffsetIDs_halo](#), [OffsetIDs_snaptree](#), [ptr_auxdata](#), [ptr_dbids](#), [ptr_galaxydata](#), [ptr_galsnapdata](#), [ptr_momafdata](#), [ptr_treedata](#), [TreeFirstHalo](#), [TreeN-gals](#), and [TreeNHalos](#).

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

4.19.1.37 double func_size (double x, double a)

4.19.1.38 void get_coordinates (float * pos, float * vel, long long ID, int tree, int halonr, int snapnum)

Pos and Vel for the type 2 galaxy are updated with PosList and VelList, containing the corresponding properties for the most bound dark matter particle identified at disruption time. The velocity is converted into a peculiar velocity: $v_p = \sqrt{AA}v$.

Definition at line 331 of file [save.c](#).

References [AA](#), [CountIDs_halo](#), [IdList](#), [OffsetIDs_halo](#), [PosList](#), [TreeFirstHalo](#), and [VelList](#).

Referenced by [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.19.1.39 double get_disk_radius (int halonr, int p)

Definition at line 20 of file [recipe_misc.c](#).

References [DiskRadiusMethod](#), [Gal](#), [Halo](#), [GALAXY::Rvir](#), and [GALAXY::Vvir](#).

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

4.19.1.40 void get_gas_disk_radius (int p)

The gas disk is assumed to be thin, in centrifugal equilibrium and to have an exponential density profile:

$$\Sigma(R_{\text{gas}}) = \Sigma_{\text{gas}0} e^{-\frac{R_{\text{gas}}}{R_{\text{gas},d}}},$$

where $R_{\text{gas},d}$ is the scale length of the gas disk and $\Sigma_{\text{gas}0}$ is the corresponding central surface density.

Assuming a flat circular velocity curve (galaxy with a negligible self-gravity) in an isothermal dark matter halo, the gas disk scale length is given by:

$$R_{\text{gas},d} = \frac{J_{\text{gas}}/M_{\text{gas}}}{2V_{\text{max}}},$$

assuming conservation of the angular momentum of the cooling gas and that the maximum circular velocity of satellite galaxies does not change after infall (inner dark matter regions are compact and don't change).

Definition at line 61 of file [recipe_misc.c](#).

References [Gal](#), [GALAXY::GasDiskRadius](#), [GALAXY::GasSpin](#), [GALAXY::InfallVmax](#), [GALAXY::Type](#), and [GALAXY::Vmax](#).

Referenced by [cool_gas_onto_galaxy\(\)](#), and [deal_with_galaxy_merger\(\)](#).

4.19.1.41 double get_initial_disk_radius (int halonr, int p)

First determination of radius in Guo2010 (same as in Delucia2007), after this, the disks are updated using [get_gas_disk_radius](#) and [get_stellar_disk_radius](#). Two options are available:

If DiskRadiusMethod = 0 then $R_{\text{disk}} = \frac{R_{\text{vir}}}{10}$

If DiskRadiusMethod = 1 or 2 then the Mo, Mao & White (1998) formalism is used with a Bullock style λ :

$$R_d = \frac{1}{\sqrt{2}} \frac{j_d}{m_d} \lambda r_{200}$$

and using the Milky Way as an approximate guide $R_{\text{disk}} = 3R_d$.

Definition at line 121 of file [recipe_misc.c](#).

References [DiskRadiusMethod](#), [Gal](#), [Halo](#), [GALAXY::Rvir](#), and [GALAXY::Vvir](#).

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

4.19.1.42 int get_jump_index (double age)

Definition at line 453 of file [init.c](#).

References [AgeTab](#), [jumpfac](#), and [jumptab](#).

Referenced by [find_interpolated_lum\(\)](#), [find_interpolated_obs_lum\(\)](#), and [find_interpolation_point\(\)](#).

4.19.1.43 double get_metaldependent_cooling_rate (double logTemp, double logZ)

Definition at line 89 of file [cool_func.c](#).

References [get_rate\(\)](#), and [metallicities](#).

Referenced by [cooling_recipe\(\)](#), and [test\(\)](#).

4.19.1.44 double get_metallicity (double gas, double metals)

Definition at line 297 of file [recipe_misc.c](#).

Referenced by [cal_H2\(\)](#), [check_disk_instability\(\)](#), [collisional_starburst_recipe\(\)](#), [cool_gas_onto_galaxy\(\)](#), [do_AGN_heating\(\)](#), [do_major_merger_starburst\(\)](#), [grow_black_hole\(\)](#), [reincorporate_gas\(\)](#), [starformation_and_feedback\(\)](#), and [update_from_feedback\(\)](#).

4.19.1.45 double get_rate (int tab, double logTemp)

Definition at line 119 of file [cool_func.c](#).

References [CoolRate](#).

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

4.19.1.46 void get_stellar_disk_radius (int p)

The stellar disk is assumed to be thin, in centrifugal equilibrium and to have an exponential density profile:

$$\Sigma(R_\star) = \Sigma_{\star 0} e^{-\frac{R_\star}{R_{\star,d}}},$$

where $R_{\star,d}$ is the scale length of the gas disk and $\Sigma_{\star 0}$ is the corresponding central surface density.

Assuming a flat circular velocity curve (galaxy with a negligible self-gravity) in an isothermal dark matter halo, the gas disk scale length is given by:

$$R_{\star,d} = \frac{J_\star/M_\star}{2V_{\max}},$$

assuming that the maximum circular velocity of satellite galaxies does not change after infall (inner dark matter regions are compact and don't change).

Definition at line 93 of file [recipe_misc.c](#).

References [Gal](#), [GALAXY::InfallVmax](#), [GALAXY::StellarDiskRadius](#), [GALAXY::StellarSpin](#), [GALAXY::Type](#), and [GALAXY::Vmax](#).

Referenced by [deal_with_galaxy_merger\(\)](#), [starformation_and_feedback\(\)](#), and [update_from_star_formation\(\)](#).

4.19.1.47 double get_virial_mass (int halonr, int p)

Definition at line 561 of file [recipe_misc.c](#).

References [Halo](#), [halo_data::Len](#), [halo_data::M_Crit200](#), and [PartMass](#).

Referenced by [estimate_merging_time\(\)](#), [get_virial_radius\(\)](#), [get_virial_velocity\(\)](#), [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), and [update_centralgal\(\)](#).

4.19.1.48 double get_virial_radius (int halonr, int p)

Calculates virial radius using: $M_{\text{vir}} = \frac{4}{3}\pi R_{\text{vir}}^3 \rho_c \Delta_c$.

From which, assuming $\Delta_c = 200$, $R_{\text{vir}} = \left(\frac{3M_{\text{vir}}}{4\pi 200\rho_c} \right)^{1/3}$

Definition at line 592 of file [recipe_misc.c](#).

References [G](#), [get_virial_mass\(\)](#), [Halo](#), [Hubble](#), [Omega](#), [OmegaLambda](#), [halo_data::SnapNum](#), and [ZZ](#).

Referenced by [estimate_merging_time\(\)](#), [get_virial_velocity\(\)](#), [init_galaxy\(\)](#), and [update_centralgal\(\)](#).

4.19.1.49 double get_virial_velocity (int halonr, int p)

Calculates virial velocity: $V_{\text{vir}} = \frac{GM_{\text{vir}}}{R_{\text{vir}}}$

Definition at line 577 of file [recipe_misc.c](#).

References [G](#), [get_virial_mass\(\)](#), and [get_virial_radius\(\)](#).

Referenced by [estimate_merging_time\(\)](#), [init_galaxy\(\)](#), and [update_centralgal\(\)](#).

4.19.1.50 void grow_black_hole (int halonr, int merger_centralgal, double mass_ratio, double dt)

Grows black hole through accretion from cold gas during mergers, as in Kauffmann & Haehnelt (2000). I have made an addition here - the black hole can grow during minor mergers but at a reduced rate and i have included evolution with redshift

Definition at line 401 of file [recipe_mergers.c](#).

References [BlackHoleGrowthRate](#), [GALAXY::BlackHoleMass](#), [GALAXY::ColdGas](#), [Gal](#), [get_metallicity\(\)](#), [GALAXY::MetalsColdGas](#), and [GALAXY::Vvir](#).

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

4.19.1.51 double hot_retain_sat (int *i*, int *centralgal*, double *infallingMass*)

This is caused both by tidal and ram-pressure stripping. This function return the actual mass of hot gas that the type 1 retains.

TIDAL STRIPPING Hot gas is tidally stripped at the same rate at which dark matter is striped:

$$\frac{M_{\text{hot}}(R_{\text{tidal}})}{M_{\text{hot,infall}}} = \frac{M_{\text{DM}}}{M_{\text{DM,infall}}}$$

Since the hot gas distribution is assumed to be $\rho \propto r^{-2}$ this means $M_{\text{hot}}(r) \propto r$. Therefore, the tidal radius beyond gas is stripped is given by:

$$R_{\text{tidal}} = \left(\frac{M_{\text{DM}}}{M_{\text{DM,infall}}} \right) R_{\text{DM,infall}}$$

RAM PRESSURE STRIPPING Let $R_{r.p.}$ represent the distance from the centre of the satellite at which ram pressure striping equals its self-gravity. Then:

$\rho_{\text{sat}}(R_{r.p.})V_{\text{sat}}^2 = \rho_{\text{par}}(R_{\text{orbit}})V_{\text{orbit}}^2$ Where the four terms represent respectively the density of the satellite at $R_{r.p.}$, the virial velocity of the satellite at infall, the density of the parent halo at the radius of the satellite and the orbit velocity of the satellite (given by V_c of the parent halo)

The striping radius is given by

$$R_{\text{strip}} = \min(R_{\text{tidal}}, R_{r.p.})$$

Definition at line 696 of file `recipe_infall.c`.

References `Gal`, `Halo`, `GALAXY::HaloNr`, `GALAXY::HotFrac`, `GALAXY::HotGas`, `GALAXY::HotRadius`, `GALAXY::Len`, `GALAXY::Mvir`, `PartMass`, `GALAXY::Pos`, `GALAXY::Rvir`, `halo_data::SnapNum`, `GALAXY::Type`, and `ZZ`.

Referenced by `infall_recipe()`.

4.19.1.52 double infall_recipe (int *centralgal*, int *ngal*, double *Zcurr*)

Definition at line 60 of file `recipe_infall.c`.

References `BaryonFrac`, `GALAXY::BlackHoleMass`, `GALAXY::CentralGal`, `GALAXY::ColdGas`, `do_reionization()`, `GALAXY::EjectedMass`, `EjectionRecipe`, `Gal`, `Halo`, `GALAXY::HaloNr`, `hot_retain_sat()`, `GALAXY::HotFrac`, `GALAXY::HotGas`, `GALAXY::HotRadius`, `GALAXY::ICM`, `GALAXY::MetalsEjectedMass`, `GALAXY::MetalsHotGas`, `GALAXY::MetalsICM`, `GALAXY::Mvir`, `GALAXY::Pos`, `ReionizationOn`, `GALAXY::Rvir`, `SatelliteRecipe`, `halo_data::SnapNum`, `GALAXY::StellarMass`, `GALAXY::Type`, `update_hot_frac()`, `update_ICL()`, and `ZZ`.

Referenced by `main()`.

4.19.1.53 void init (void)

Calls `set_units()`, `read_output_snaps()`, `read_snap_list()`, `read_recgas()`, `read_file_nrs()`, `read_sfrz()`, `read_reionization()`, `setup_spectrophotometric_model()`, `read_tsud_tables()` and `read_cooling_functions()`.

Converts EnergySN (->EnergySNcode), EtaSN (->EtaSNcode) and AgnEfficiency into internal units.

Definition at line 108 of file `init.c`.

References `a0`, `AA`, `Age`, `AgnEfficiency`, `ar`, `EnergySN`, `EnergySNcode`, `EtaSN`, `EtaSNcode`, `Hubble_h`, `random_generator`, `read_cooling_functions()`, `read_file_nrs()`, `read_output_snaps()`, `read_recgas()`, `read_reionization()`, `read_sfrz()`, `read_snap_list()`, `read_tsud_tables()`, `Reionization_z0`, `Reionization_zr`, `set_units()`, `setup_spectrophotometric_model()`, `Snaplistlen`, `ThisTask`, `time_to_present()`, `UnitEnergy_in_cgs`, `UnitMass_in_g`, `UnitTime_in_s`, and `ZZ`.

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

4.19.1.54 void init_galaxy (int *p*, int *halonr*)

Definition at line 146 of file [recipe_misc.c](#).

References [GALAXY::BlackHoleMass](#), [GALAXY::BulgeMass](#), [GALAXY::BulgeSize](#), [GALAXY::ColdGas](#), [GALAXY::CoolingRadius](#), [GALAXY::DescendantGal](#), [GALAXY::DisruptOn](#), [GALAXY::dObsLum](#), [GALAXY::dObsLumBulge](#), [GALAXY::dObsLumDust](#), [GALAXY::dObsYLum](#), [GALAXY::dObsYLumBulge](#), [GALAXY::EjectedMass](#), [GALAXY::FirstProgGal](#), [Gal](#), [GALAXY::GasDiskRadius](#), [GALAXY::GasSpin](#), [get_initial_disk_radius\(\)](#), [get_virial_mass\(\)](#), [get_virial_radius\(\)](#), [get_virial_velocity\(\)](#), [Halo](#), [GALAXY::HaloNr](#), [GALAXY::HaloSpin](#), [GALAXY::HotFrac](#), [GALAXY::HotGas](#), [GALAXY::HotRadius](#), [GALAXY::ICLLum](#), [GALAXY::ICM](#), [GALAXY::Inclination](#), [GALAXY::InfallSnap](#), [GALAXY::InfallVmax](#), [halo_data::Len](#), [GALAXY::Len](#), [GALAXY::Lum](#), [GALAXY::LumBulge](#), [GALAXY::LumDust](#), [GALAXY::MassWeightAge](#), [GALAXY::MergCentralPos](#), [GALAXY::MergeSat](#), [GALAXY::MergTime](#), [GALAXY::MetalsBulgeMass](#), [GALAXY::MetalsColdGas](#), [GALAXY::MetalsEjectedMass](#), [GALAXY::MetalsHotGas](#), [GALAXY::MetalsICM](#), [GALAXY::MetalsStellarMass](#), [halo_data::MostBoundID](#), [GALAXY::MostBoundID](#), [GALAXY::Mvir](#), [GALAXY::NextProgGal](#), [GALAXY::ObsICL](#), [GALAXY::ObsLum](#), [GALAXY::ObsLumBulge](#), [GALAXY::ObsLumDust](#), [GALAXY::ObsYLum](#), [GALAXY::ObsYLumBulge](#), [GALAXY::OriMergTime](#), [halo_data::Pos](#), [GALAXY::Pos](#), [random_generator](#), [GALAXY::Rvir](#), [GALAXY::Sfr](#), [GALAXY::SfrBulge](#), [halo_data::SnapNum](#), [GALAXY::SnapNum](#), [halo_data::Spin](#), [GALAXY::StarMerge](#), [GALAXY::StellarDiskRadius](#), [GALAXY::StellarMass](#), [GALAXY::StellarSpin](#), [GALAXY::Type](#), [halo_data::Vel](#), [GALAXY::Vel](#), [halo_data::Vmax](#), [GALAXY::Vmax](#), [GALAXY::Vvir](#), [GALAXY::XrayLum](#), [GALAXY::YLum](#), and [GALAXY::YLumBulge](#).

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

4.19.1.55 void init_jump_index (void)

Definition at line 435 of file [init.c](#).

References [AgeTab](#), [jumpfac](#), and [jumptab](#).

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

4.19.1.56 double integrand_time_to_present (double *a*, void * *param*)

Definition at line 42 of file [age.c](#).

References [Omega](#), and [OmegaLambda](#).

4.19.1.57 int join_galaxies_of_progenitors (int *halonr*, int *nstart*)

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

4.19.1.58 int join_galaxies_of_progenitors (int *halonr*, int *nstart*, int *cenngal*)

4.19.1.59 void load_all_auxdata (int *filenr*)

Definition at line 626 of file [io_tree.c](#).

References [filled_galaxydata](#), [LastSnapShotNr](#), [mymalloc\(\)](#), [offset_auxdata](#), [ptr_auxdata](#), and [SimulationDir](#).

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

4.19.1.60 void load_all_dbids (int *filenr*)

Definition at line [669](#) of file [io_tree.c](#).

References [LastSnapShotNr](#), [mymalloc\(\)](#), [offset_dbids](#), [ptr_dbids](#), and [SimulationDir](#).

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

4.19.1.61 void load_all_treedata (int *filenr*)

Definition at line [711](#) of file [io_tree.c](#).

References [filled_galaxydata](#), [filled_galsnapdata](#), [filled_momafdata](#), [LastSnapShotNr](#), [maxstorage_galaxydata](#), [maxstorage_galsnapdata](#), [maxstorage_momafdata](#), [mymalloc\(\)](#), [offset_galaxydata](#), [offset_galsnapdata](#), [offset_momafdata](#), [offset_treedata](#), [ptr_galaxydata](#), [ptr_galsnapdata](#), [ptr_momafdata](#), [ptr_treedata](#), and [SimulationDir](#).

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

4.19.1.62 void load_tree (int *filenr*, int *nr*)

If USE_MEMORY_TO_MINIMIZE_IO & NEW_IO are OFF, the trees_** files are opened every time a tree needs to be read in. Then the code's structure that will have the tree information is read: Halo = (sizeof(struct halo_data) * TreeNHalos[]) are read.

HaloAux structure is allocated = (sizeof(struct halo_aux_data) * TreeNHalos[])

Considering the number of halos allocated for the current tree the size of the structures containing the galaxies with a halo and the galaxies with and without a halo to be allocated is estimated:

For galaxies with a halo - MaxGals = MAXGALFAC * TreeNHalos[] and HaloGal = (sizeof(struct GALAXY) * MaxGals).

For all galaxies - FoF_MaxGals = 10000*15 and Gal = (sizeof(struct GALAXY) * FoF_MaxGals)

If GALAXYTREE ON, HaloIDs structure is read from tree_dbids = sizeof(struct halo_ids_data) * TreeNHalos[]

Definition at line [313](#) of file [io_tree.c](#).

References [halo_aux_data::DoneFlag](#), [FoF_MaxGals](#), [Gal](#), [Halo](#), [HaloAux](#), [halo_aux_data::HaloFlag](#), [HaloGal](#), [HaloIDs](#), [LastSnapShotNr](#), [MaxGals](#), [myfread\(\)](#), [myfseek\(\)](#), [mymalloc\(\)](#), [halo_aux_data::NGalaxies](#), [Ntrees](#), [offset_dbids](#), [offset_treedata](#), [SimulationDir](#), [tree_file](#), [treedbids_file](#), [TreeFirstHalo](#), and [TreeNHalos](#).

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

4.19.1.63 void load_tree_table (int *filenr*)

If USE_MEMORY_TO_MINIMIZE_IO ON the first step on this file is to call [load_all_dbids\(\)](#), [load_all_auxdata\(\)](#), [load_all_treedata\(\)](#). These will read all the tree data for the current file into pointers instead of doing it once for every tree.

If USE_MEMORY_TO_MINIMIZE_IO OFF the trees are read independently from the files.

Modified versions of myread/myfwrite/myfseek are called to either read individual trees from the files into structures or from the pointers with all the data for the current file into structures.

For each tree the code reads in the header in trees_**: Ntrees - number of trees in the current file (int); totNHalos - total number of halos summed over all the trees in current file (int); TreeNHalos - number of halos in each tree (Ntrees).

Then output files are opened SA_z**_** - for snapshot output; SA_galtree_** for GALAXYTREE option and SA_**_** for MOMAF.

If UPDATETYPETWO ON the header in tree_aux is also read: NtotHalos - total number of halos in the file (int); TotIds - total number of particle IDs (int); Ntrees - total number of trees (int); TotSnaps - total number of snapshots (int). Define some other quantities: CountIDs_snaptree - number of IDs for each tree at each snapshot (TotSnaps * Ntrees); OffsetIDs_snaptree (TotSnaps * Ntrees); CountIDs_halo - Number of IDs per halo (NtotHalos); OffsetIDs_halo (int).

Definition at line 90 of file [io_tree.c](#).

References [CountIDs_halo](#), [CountIDs_snaptree](#), [FileNameGalaxies](#), [LastSnapShotNr](#), [ListOutputSnaps](#), [load_all_auxdata\(\)](#), [load_all_dbids\(\)](#), [load_all_treedata\(\)](#), [myread\(\)](#), [myfseek\(\)](#), [mymalloc\(\)](#), [NtotHalos](#), [Ntrees](#), [offset_auxdata](#), [offset_treedata](#), [OffsetIDs_halo](#), [OffsetIDs_snaptree](#), [OutputDir](#), [SimulationDir](#), [TotGalaxies](#), [TotGalCount](#), [TotIds](#), [TotSnaps](#), [tree_file](#), [treeaux_file](#), [TreeFirstHalo](#), [TreeNgals](#), [TreeNHalos](#), and [ZZ](#).

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

4.19.1.64 double lum_to_mag (double *lum*)

Converts luminosities into magnitudes: $M = -2.5\log_{10}(L)$

Definition at line 614 of file [recipe_misc.c](#).

Referenced by [prepare_galaxy_for_momaf\(\)](#), and [prepare_galaxy_for_output\(\)](#).

4.19.1.65 void make_bulge_from_burst (int *p*)

Definition at line 711 of file [recipe_mergers.c](#).

References [GALAXY::BulgeMass](#), [GALAXY::dObsLum](#), [GALAXY::dObsLumBulge](#), [GALAXY::dObsYLum](#), [GALAXY::dObsYLumBulge](#), [Gal](#), [GALAXY::Lum](#), [GALAXY::LumBulge](#), [GALAXY::MetalsBulgeMass](#), [GALAXY::MetalsStellarMass](#), [GALAXY::ObsLum](#), [GALAXY::ObsLumBulge](#), [GALAXY::ObsYLum](#), [GALAXY::ObsYLumBulge](#), [GALAXY::Sfr](#), [GALAXY::SfrBulge](#), [GALAXY::StellarMass](#), [GALAXY::YLum](#), and [GALAXY::YLumBulge](#).

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

4.19.1.66 size_t momaf_fwrite (void * *ptr*, size_t *size*, size_t *nmemb*, FILE * *stream*)

4.19.1.67 size_t myread (void * *ptr*, size_t *size*, size_t *nmemb*, FILE * *stream*)

If USE_MEMORY_TO_MINIMIZE_IO OFF - this routine simply reads the data from a file into a structure.

If USE_MEMORY_TO_MINIMIZE_IO ON - the reading routines [load_all_dbids\(\)](#), [load_all_auxdata\(\)](#) and [load_all_treedata\(\)](#) will read all the tree data into pointers in one go for each file. Then, whenever each tree is being constructed myread copies the properties from the pointers containing all the trees in

the current file into a structure containing a single tree used by the code - from ptr_treedata, ptr_dbids and ptr_auxdata to Halo, HaloIDs and some arrays (e.x PosList) respectively.

Definition at line 561 of file [io_tree.c](#).

References [offset_auxdata](#), [offset_dbids](#), [offset_treedata](#), [ptr_auxdata](#), [ptr_dbids](#), and [ptr_treedata](#).

Referenced by [load_tree\(\)](#), [load_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.19.1.68 void myfree (void * *p*)

Definition at line 66 of file [mymalloc.c](#).

References [Nblocks](#), [SizeTable](#), [Table](#), and [TotMem](#).

Referenced by [free_galaxies_and_tree\(\)](#), [free_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.19.1.69 int myfseek (FILE * *stream*, long *offset*, int *whence*)

Definition at line 590 of file [io_tree.c](#).

References [offset_auxdata](#), [offset_dbids](#), [offset_galaxydata](#), [offset_galsnapdata](#), [offset_momafdata](#), and [offset_treedata](#).

Referenced by [load_tree\(\)](#), [load_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.19.1.70 size_t myfwrite (void * *ptr*, size_t *size*, size_t *nmem*, FILE * *stream*)

If USE_MEMORY_TO_MINIMIZE_IO OFF - this routine simply writes the data in a structure into a file.

If USE_MEMORY_TO_MINIMIZE_IO ON - the myfwrite routine copies the galaxy properties from the current tree being treated into a pointer. This is done for all the trees in each file and after that, the pointer with properties for all the galaxies in each file written. This will be done by [write_all_galaxy_data\(\)](#), [write_all_snap_data\(\)](#) or [write_galaxy_for_momaf\(\)](#) to write either a full tree, snaps or for the MOMAF. The different outputs will be stored in a different pointer until the file is done. Inside this myfwrite the properties are copied into the corresponding pointer depending on the last input parameter: stream = 4 - GALAXYTREE option - pointer *ptr_galaxydata*; stream = >10 && <10000 - SNAP option - pointer *ptr_galsnapdata*[]; stream = >10000 - MOMAF option - *ptr_momafdata*[];

Definition at line 504 of file [io_tree.c](#).

References [filled_galaxydata](#), [filled_galsnapdata](#), [filled_momafdata](#), [maxstorage_galaxydata](#), [maxstorage_galsnapdata](#), [maxstorage_momafdata](#), [offset_galaxydata](#), [offset_galsnapdata](#), [offset_momafdata](#), [ptr_galaxydata](#), [ptr_galsnapdata](#), and [ptr_momafdata](#).

Referenced by [finalize_galaxy_file\(\)](#), [finalize_momaf_file\(\)](#), [openallfiles\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.19.1.71 void* mymalloc (size_t *n*, char * *myflag*)

Definition at line 26 of file [mymalloc.c](#).

References [HighMarkMem](#), [Nblocks](#), [OldPrintedHighMark](#), [SizeTable](#), [Table](#), and [TotMem](#).

Referenced by [load_all_auxdata\(\)](#), [load_all_dbids\(\)](#), [load_all_treedata\(\)](#), [load_tree\(\)](#), [load_tree_table\(\)](#), [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.19.1.72 double NumToTime (int snapnum)

Definition at line 535 of file [recipe_misc.c](#).

References [Age](#).

Referenced by [add_to_luminosities\(\)](#), [main\(\)](#), [update_type_1\(\)](#), and [update_type_2\(\)](#).

4.19.1.73 FILE* open_outputtree_file (int filenr, char mode[])

Definition at line 918 of file [io_tree.c](#).

References [FileNameGalaxies](#), and [OutputDir](#).

Referenced by [finalize_galaxy_file\(\)](#), [openallfiles\(\)](#), and [save_galaxy_tree\(\)](#).

4.19.1.74 FILE* open_tree_file (int filenr)

Definition at line 936 of file [io_tree.c](#).

References [LastSnapShotNr](#), and [SimulationDir](#).

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

4.19.1.75 FILE* open_treeaux_file (int filenr)

Definition at line 972 of file [io_tree.c](#).

References [LastSnapShotNr](#), and [SimulationDir](#).

Referenced by [openallfiles\(\)](#), and [save_galaxy_tree\(\)](#).

4.19.1.76 FILE* open_treedbids_file (int filenr)

Definition at line 954 of file [io_tree.c](#).

References [LastSnapShotNr](#), and [SimulationDir](#).

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

4.19.1.77 void openallfiles (int filenr)

Definition at line 889 of file [io_tree.c](#).

References [myfwrite\(\)](#), [open_outputtree_file\(\)](#), [open_tree_file\(\)](#), [open_treeaux_file\(\)](#), [open_treedbids_file\(\)](#), [output_file](#), [tree_file](#), [treeaux_file](#), and [treedbids_file](#).

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

4.19.1.78 int peano_hilbert_key (int x, int y, int z, int bits)

Definition at line 61 of file [peano.c](#).

References [quadrants](#), [rotx_table](#), [rotxmap_table](#), [roty_table](#), [rotymap_table](#), and [sense_table](#).

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

4.19.1.79 double peri_radius (int *p*, int *centralgal*)

Definition at line 169 of file [recipe_disrupt.c](#).

References [Gal](#), [Halo](#), [GALAXY::HaloNr](#), [GALAXY::Pos](#), [GALAXY::SnapNum](#), [GALAXY::Vel](#), [GALAXY::Vvir](#), and [ZZ](#).

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

4.19.1.80 void prepare_galaxy_for_momaf (int *n*, int *filenr*, int *tree*, struct GALAXY * *g*, struct MOMAF_INPUTS * *o*)

Definition at line 741 of file [save.c](#).

References [GALAXY::dObsLumBulge](#), [GALAXY::dObsLumDust](#), [MOMAF_INPUTS::dObsMagBulge](#), [MOMAF_INPUTS::dObsMagDust](#), [GALAXY::GalID](#), [MOMAF_INPUTS::GalID](#), [halo_ids_data::HaloID](#), [MOMAF_INPUTS::HaloID](#), [HaloIDs](#), [GALAXY::HaloNr](#), [lum_to_mag\(\)](#), [GALAXY::MergCentralPos](#), [GALAXY::MergTime](#), [GALAXY::ObsLumBulge](#), [GALAXY::ObsLumDust](#), [MOMAF_INPUTS::ObsMagBulge](#), [MOMAF_INPUTS::ObsMagDust](#), [GALAXY::OriMergTime](#), [GALAXY::Pos](#), [MOMAF_INPUTS::Pos](#), [GALAXY::SnapNum](#), [MOMAF_INPUTS::SnapNum](#), [GALAXY::Type](#), [GALAXY::Vel](#), [MOMAF_INPUTS::Vel](#), and [ZZ](#).

Referenced by [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.19.1.81 void prepare_galaxy_for_output (int *n*, int *filenr*, int *tree*, struct GALAXY * *g*, struct GALAXY_OUTPUT * *o*)

Definition at line 382 of file [save.c](#).

References [GALAXY::BlackHoleMass](#), [GALAXY_OUTPUT::BlackHoleMass](#), [BoxSize](#), [GALAXY::BulgeMass](#), [GALAXY_OUTPUT::BulgeMass](#), [GALAXY::BulgeSize](#), [GALAXY_OUTPUT::BulgeSize](#), [GALAXY_OUTPUT::CentralMvir](#), [GALAXY::ColdGas](#), [GALAXY_OUTPUT::ColdGas](#), [GALAXY::CoolingRadius](#), [GALAXY_OUTPUT::CoolingRadius](#), [GALAXY::DescendantGal](#), [GALAXY_OUTPUT::DescendantGal](#), [GALAXY::DisruptOn](#), [GALAXY_OUTPUT::DisruptOn](#), [GALAXY::dObsLum](#), [GALAXY::dObsLumBulge](#), [GALAXY::dObsLumDust](#), [GALAXY_OUTPUT::dObsMag](#), [GALAXY_OUTPUT::dObsMagBulge](#), [GALAXY_OUTPUT::dObsMagDust](#), [GALAXY::EjectedMass](#), [GALAXY_OUTPUT::EjectedMass](#), [halo_data::FileNr](#), [GALAXY_OUTPUT::FileTreeNr](#), [halo_aux_data::FirstGalaxy](#), [halo_data::FirstHaloInFOFgroup](#), [GALAXY::FirstProgGal](#), [GALAXY_OUTPUT::FirstProgGal](#), [fix_units_for_ouput\(\)](#), [GALAXY_OUTPUT::FOFCentralGal](#), [GALAXY::GalID](#), [GALAXY_OUTPUT::GalID](#), [GALAXY::GasDiskRadius](#), [GALAXY_OUTPUT::GasDiskRadius](#), [GALAXY::GasSpin](#), [GALAXY_OUTPUT::GasSpin](#), [get_virial_mass\(\)](#), [Halo](#), [HaloAux](#), [HaloGal](#), [halo_ids_data::Haloid](#), [GALAXY_OUTPUT::HaloID](#), [HaloIDs](#), [GALAXY_OUTPUT::HaloIndex](#), [GALAXY::HaloNr](#), [Hashbits](#), [GALAXY::HotGas](#), [GALAXY_OUTPUT::HotGas](#), [GALAXY::HotRadius](#), [GALAXY_OUTPUT::HotRadius](#), [Hubble_h](#), [GALAXY::ICLLum](#), [GALAXY::ICM](#), [GALAXY_OUTPUT::ICM](#), [GALAXY::InfallSnap](#), [GALAXY_OUTPUT::InfallSnap](#), [GALAXY::InfallVmax](#), [GALAXY_OUTPUT::InfallVmax](#), [GALAXY::LastProgGal](#), [GALAXY_OUTPUT::LastProgGal](#), [halo_data::Len](#), [GALAXY::Len](#), [GALAXY_OUTPUT::Len](#), [GALAXY::Lum](#), [lum_to_mag\(\)](#), [GALAXY::LumBulge](#), [GALAXY::LumDust](#), [GALAXY_OUTPUT::Mag](#), [GALAXY_OUTPUT::MagBulge](#), [GALAXY_OUTPUT::MagDust](#), [GALAXY_OUTPUT::MagICL](#), [GALAXY::MainLeaf](#), [GALAXY_OUTPUT::MainLeafId](#), [GALAXY::MassWeightAge](#), [GALAXY_OUTPUT::MassWeightAge](#), [GALAXY::MergCentralPos](#), [GALAXY_OUTPUT::MergeOn](#), [GALAXY::MergeOn](#), [GALAXY_OUTPUT::MergTime](#), [GALAXY::MergTime](#), [GALAXY::MetalsBulgeMass](#), [GALAXY_OUTPUT::MetalsBulgeMass](#), [GALAXY::MetalsColdGas](#), [GALAXY_OUTPUT::MetalsColdGas](#), [GALAXY::MetalsEjectedMass](#), [GALAXY_OUTPUT::MetalsEjectedMass](#), [GALAXY::MetalsHotGas](#),

GALAXY_OUTPUT::MetalsHotGas, GALAXY::MetalsICM, GALAXY_OUTPUT::MetalsICM,
 GALAXY::MetalsStellarMass, GALAXY_OUTPUT::MetalsStellarMass, GALAXY_-
 OUTPUT::MMSubID, GALAXY::Mvir, GALAXY_OUTPUT::Mvir, halo_data::NextHaloInFOFgroup,
 GALAXY::NextProgGal, GALAXY_OUTPUT::NextProgGal, halo_aux_data::NGalaxies,
 GALAXY::ObsICL, GALAXY::ObsLum, GALAXY::ObsLumBulge, GALAXY::ObsLumDust,
 GALAXY_OUTPUT::ObsMag, GALAXY_OUTPUT::ObsMagBulge, GALAXY_-
 OUTPUT::ObsMagDust, GALAXY_OUTPUT::ObsMagICL, GALAXY_OUTPUT::OriMergTime,
 GALAXY::OriMergTime, peano_hilbert_key(), GALAXY_OUTPUT::PeanoKey, GALAXY::Pos,
 GALAXY_OUTPUT::Pos, GALAXY_OUTPUT::Redshift, GALAXY::Rvir, GALAXY_OUTPUT::Rvir,
 GALAXY::Sfr, GALAXY_OUTPUT::Sfr, GALAXY::SfrBulge, GALAXY_OUTPUT::SfrBulge,
 GALAXY::SnapNum, GALAXY_OUTPUT::SnapNum, GALAXY::StellarDiskRadius, GALAXY_-
 OUTPUT::StellarDiskRadius, GALAXY::StellarMass, GALAXY_OUTPUT::StellarMass,
 GALAXY::StellarSpin, GALAXY_OUTPUT::StellarSpin, halo_data::SubhaloIndex, GALAXY_-
 OUTPUT::SubID, GALAXY::TreeRoot, GALAXY_OUTPUT::TreeRootId, GALAXY_OUTPUT::Type,
 GALAXY::Type, UnitMass_in_g, UnitTime_in_Megayears, UnitTime_in_s, GALAXY::Vel, GALAXY_-
 OUTPUT::Vel, GALAXY::Vmax, GALAXY_OUTPUT::Vmax, GALAXY::Vvir, GALAXY_-
 OUTPUT::Vvir, GALAXY::XrayLum, GALAXY_OUTPUT::XrayLum, and ZZ.

Referenced by [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.19.1.82 void print_allocated (void)

Definition at line 20 of file [mymalloc.c](#).

References [TotMem](#).

4.19.1.83 void read_cooling_functions (void)

Definition at line 41 of file [cool_func.c](#).

References [CoolFunctionsDir](#), [CoolRate](#), [metallicities](#), [name](#), and [ThisTask](#).

Referenced by [init\(\)](#), and [test\(\)](#).

4.19.1.84 void read_file_nrs (void)

Definition at line 891 of file [init.c](#).

References [FileNrDir](#), and [ListInputFilrNr](#).

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

4.19.1.85 void read_output_snaps (void)

Definition at line 187 of file [init.c](#).

References [FileWithOutputSnaps](#), and [ListOutputSnaps](#).

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

4.19.1.86 void read_parameter_file (char * *fname*)

Definition at line 18 of file [read_parameters.c](#).

References [AgnEfficiency](#), [AGNRecipeOn](#), [BaryonFrac](#), [BlackHoleGrowthRate](#), [BoxSize](#), [BulgeFormationInMinorMergersOn](#), [CoolFunctionsDir](#), [CoolingCutoff](#), [CoolingVelocityCutOff](#), [DiskRadiusMethod](#), [EjectionOn](#), [EjectionRecipe](#), [EjectPreVelocity](#), [EjectSlope](#), [EnergySN](#), [EtaSN](#), [FeedbackEjectionEfficiency](#), [FeedbackEpsilon](#), [FeedbackRecipe](#), [FeedbackReheatingEpsilon](#), [FileNameGalaxies](#), [FileNrDir](#), [FilesPerSnapshot](#), [FileWithOutputSamps](#), [FileWithSnapList](#), [FirstFile](#), [FracZtoHot](#), [Hashbits](#), [Hubble_h](#), [LastFile](#), [LastSnapShotNr](#), [MetallicityOption](#), [Omega](#), [OmegaLambda](#), [OutputDir](#), [PartMass](#), [PhotDir](#), [PhotPrefix](#), [RecycleFraction](#), [ReheatPreVelocity](#), [ReheatSlope](#), [ReIncorporationFactor](#), [ReIncorporationRecipe](#), [Reionization_z0](#), [Reionization_zr](#), [ReionizationOn](#), [SatelliteRecipe](#), [SfrAlpha](#), [SfrEfficiency](#), [SfrLawPivotVelocity](#), [SfrLawSlope](#), [SimulationDir](#), [SNinReheat](#), [StarBurstRecipe](#), [StarBurstsInMajorMergersOn](#), [StarFormationRecipe](#), [ThisTask](#), [ThreshMajorMerger](#), [TrackDiskInstability](#), [UnitLength_in_cm](#), [UnitMass_in_g](#), [UnitVelocity_in_cm_per_s](#), and [Yield](#).

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

4.19.1.87 void read_recgas (void)

Definition at line [865](#) of file [init.c](#).

References [Frac](#).

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

4.19.1.88 void read_sfrz (void)

Definition at line [920](#) of file [init.c](#).

References [H2](#), and [Rho](#).

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

4.19.1.89 void read_snap_list (void)

Definition at line [222](#) of file [init.c](#).

References [AA](#), [FileWithSnapList](#), [Snaplistlen](#), and [ThisTask](#).

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

4.19.1.90 void read_tsud_tables (void)

The extinction tables are read for the same bands read for the spectrophotometric tables both for the intergalactic medium (**_Set_Ext_table.dat) and young birth clouds (**_Set_YSExt_table.dat). Detailed description at [recipe_dust.c](#). The prefix corresponds to different magnitude sets: DataRelease or SDSS corresponding respectively to BVRIK or ugriz bands.

Extinction curves for the ISM: $\left(\frac{A_\lambda}{A_v}\right)_{Z_\odot} \left(\frac{Z_{\text{gas}}}{Z_\odot}\right)^s$ are read into DeltaM[number of magnitudes+1][gas metallicity][redshift] - one extra column in the end that corresponds to V_band extinction used for the YS component.

The optical depth for YS (τ_λ^{BC}) is calibrated from the ISM optical depth in the V-band:

$$\tau_\lambda^{BC} = \tau_v^{\text{ISM}} \left(\frac{1}{\mu} - 1\right) \left(\frac{\lambda}{5500}\right)^{-0.7},$$

where $\left(\frac{\lambda}{5500}\right)^{-0.7}$ is read into CORRECTIONS[number of magnitudes][redshift] from file (_YSExt_table.dat)

All the values are inversely k-corrected as the magnitudes in order to compute dust corrected observed frame magnitudes.

For each snapshot there are 6 times 100 numbers corresponding to 100 different metallicities for each band. The first tree numbers on the file indicate the number of grid points (100) and the min and max point of the grid (in terms of metallicitites).

Definition at line 292 of file [init.c](#).

References [Corrections](#), [DeltaM](#), [PhotDir](#), and [PhotPrefix](#).

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

4.19.1.91 double RedshiftObs (int snapnum)

Definition at line 542 of file [recipe_misc.c](#).

References [ZZ](#).

4.19.1.92 void reincorporate_gas (int centralgal, int p, double dt)

Definition at line 34 of file [recipe_reincorporation.c](#).

References [GALAXY::EjectedMass](#), [Gal](#), [get_metallicity\(\)](#), [Halo](#), [GALAXY::HaloNr](#), [GALAXY::HotGas](#), [GALAXY::MetalsEjectedMass](#), [GALAXY::MetalsHotGas](#), [GALAXY::Pos](#), [ReIncorporationFactor](#), [ReIncorporationRecipe](#), [GALAXY::Rvir](#), [halo_data::SnapNum](#), [GALAXY::Type](#), [update_hot_frac\(\)](#), [GALAXY::Vvir](#), and [ZZ](#).

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

4.19.1.93 double sat_radius (int p)

Definition at line 199 of file [recipe_disrupt.c](#).

References [GALAXY::BulgeMass](#), [bulgemass_r\(\)](#), [GALAXY::BulgeSize](#), [GALAXY::ColdGas](#), [diskmass_r\(\)](#), [Gal](#), [GALAXY::GasDiskRadius](#), [GALAXY::StellarDiskRadius](#), and [GALAXY::StellarMass](#).

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

4.19.1.94 void save_galaxies (int filenr, int tree)

If UPDATETYPETWO=1 then the positions and velocities of type 2 galaxies are updated from the most bound dark matter particle. After that the [GALAXY_OUTPUT](#) structure is created and written. input: int file number (current file where the output is being written), int tree number (tree being currently treated).

If USE_MEMORY_TO_MINIMIZE_IO ON, this write statements in this routine copy the galaxy data from the working structures into pointers until that has been done for all the tree in a given file.

Definition at line 58 of file [save.c](#).

References [CountIDs_snaptree](#), [FileNameGalaxies](#), [get_coordinates\(\)](#), [HaloGal](#), [IdList](#), [LastSnapShotNr](#), [ListOutputSamps](#), [myfread\(\)](#), [myfree\(\)](#), [myfseek\(\)](#), [myfwrite\(\)](#), [mymalloc\(\)](#), [Nids](#), [NtotHalos](#), [Ntrees](#), [NumGals](#), [offset_auxdata](#), [offset_galsnapdata](#), [offset_momafdata](#), [OffsetIDs](#), [OffsetIDs_snaptree](#), [OutputDir](#), [MOMAF_INPUTS::Pos](#), [PosList](#), [prepare_galaxy_for_momaf\(\)](#), [prepare_galaxy_for_output\(\)](#), [SimulationDir](#), [MOMAF_INPUTS::SnapNum](#), [TotGalaxies](#), [TotIds](#), [TotSnaps](#), [TreeNgals](#), [MOMAF_INPUTS::Vel](#), [VelList](#), and [ZZ](#).

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

4.19.1.95 void save_galaxy_tree (int *filenr*, int *tree*)

This routine is similar to save_galaxies except for the initial part where galaxies are ordered.

If USE_MEMORY_TO_MINIMIZE_IO ON, this write statements in this routine copy the galaxy data from the working structures into pointers until that has been done for all the tree in a given file.

Definition at line [812](#) of file [save.c](#).

References [CountIDs_snaptree](#), [GALAXY::DescendantGal](#), [GALAXY::Done](#), [FileNameGalaxies](#), [GALAXY::FirstProgGal](#), [GalaxiesInOrder](#), [GALAXY::GalID](#), [get_coordinates\(\)](#), [HaloGal](#), [IdList](#), [GALAXY::LastProgGal](#), [LastSnapShotNr](#), [ListOutputSnaps](#), [GALAXY::MainLeaf](#), [myfread\(\)](#), [myfree\(\)](#), [myfseek\(\)](#), [myfwrite\(\)](#), [mymalloc\(\)](#), [GALAXY::NextProgGal](#), [Nids](#), [NtotHalos](#), [Ntrees](#), [NumGals](#), [offset_auxdata](#), [offset_galaxydata](#), [offset_momafdata](#), [OffsetIDs](#), [OffsetIDs_snaptree](#), [open_outputtree_file\(\)](#), [open_treaux_file\(\)](#), [output_file](#), [OutputDir](#), [MOMAF_INPUTS::Pos](#), [PosList](#), [prepare_galaxy_for_momaf\(\)](#), [prepare_galaxy_for_output\(\)](#), [MOMAF_INPUTS::SnapNum](#), [TotGalaxies](#), [TotGalCount](#), [TotIds](#), [TotSnaps](#), [treaux_file](#), [GALAXY::TreeRoot](#), [MOMAF_INPUTS::Vel](#), [VelList](#), and [walk\(\)](#).

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

4.19.1.96 int set_merger_center (int *fofhalo*)

If MERGE01=1 (Guo2010), get id of central galaxy, since type 1's can have their merger clock started before they become type 2's if M_star>M_vir. Introduced for millennium 2, where they can have very small masses, still be followed and never merge. At this moment the centre is still not known, reason why this function is needed. Also, if the type 1 merges, all the type 2's associated with it will need to know the "new" central galaxy they are merging into.

Definition at line [101](#) of file [recipe_mergers.c](#).

References [halo_aux_data::FirstGalaxy](#), [halo_data::FirstProgenitor](#), [Halo](#), [HaloAux](#), [HaloGal](#), [halo_data::Len](#), [GALAXY::Len](#), [halo_data::NextHaloInFOFgroup](#), [halo_data::NextProgenitor](#), [halo_aux_data::NGalaxies](#), and [GALAXY::Type](#).

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

4.19.1.97 void set_units (void)

As defined in input.par, $\text{UnitLength}_{\text{cm}} = 3.08568 \times 10^{24} \text{ cm}$, converts from cm into Mpc and $\text{UnitVelocity}_{\text{cm/s}} = 10000 \text{ cm/s}$, converts from cm/s to Km/s (cm to km). In [set_units\(\)](#) an interesting coincidence is used to derived UnitTime_s from these two quantities:

$$\frac{\text{UnitLength}_{\text{cm}}}{\text{UnitVelocity}_{\text{cm/s}}} = 3.08568 \times 10^{19} \text{ Mpc Km}^{-1} \text{ s}^{-1},$$

which is close to the number of seconds in $10^{12} \text{ Yr} = 3.15533 \times 10^{19} \text{ s}$ (the units of time in the code (after being divided by h)). For this reason, the unit time defined as:

$$\text{UnitTime}_s = \frac{\text{UnitLength}_{\text{cm}}}{\text{UnitVelocity}_{\text{cm/s}}},$$

its an approximation of the conversion bewteen seconds and the internal units of the code (10^{12} Yr) and is used to defined most of the other conversion factors in this function.

UnitTime_s , the way it is defined, could actually be used to convert time derived from the dynamical time $\left(\frac{R_{\text{disk}}}{V_{\text{vir}}}\right)$ into seconds. However, for the coincidence mentioned above, the time units of a time derived from t_{dyn} are very close to the code internal units (10^{12} Yr). For this reason through the code it is assumed

that t_{dyn} has internal units and its never converted (note that t_{dyn} has an h factor, as the code internal units which despite not being included is `UnitTime_s` is included in the output of `time_to_present()` - so it is consistent).

Assuming that `UnitTime_s` actually converts seconds to 10^{12}Yr all the following converting factors are correct:

$\text{UnitTime}_{\text{Myr}} = \frac{\text{UnitTime}_s}{\text{SEC_PER_MYR}} = 9.780285 \times 10^{05} \text{Yr}$, where $\text{SEC_PER_MYR} = 3.155 \times 10^7 \text{s}$, converts Yr to Myr.

$G = \frac{\text{GRAVITY} \times \text{UnitMass}_g \times \text{UnitTime}_s^2}{\text{UnitLength}_{\text{cm}}^3} = 43.00708$, where $\text{GRAVITY} = 6.672 \times 10^{-8} \text{cm}^3 \text{g}^{-1} \text{s}^{-2}$, is the gravitational constant in internal units $(\text{Mpc}^3 (10^{10} M_\odot)^{-1} (10^{12} \text{Yr})^{-2})$.

$\text{UnitDensity}_{\text{cgs}} = \frac{\text{UnitMass}_g}{\text{UnitLength}_{\text{cm}}^3} = 6.769898 \times 10^{-31}$, converts density in g cm^{-3} into internal units $(10^{10} M_\odot \text{Mpc}^{-3})$

$\text{UnitPressure}_{\text{cgs}} = \frac{\text{UnitMass}_g}{\text{UnitLength}_{\text{cm}} \times \text{UnitTime}_s^2} = 6.769898 \times 10^{-21}$, converts pressure in $\text{g cm}^{-1} \text{s}^{-2}$ into internal units $(10^{10} M_\odot \text{Mpc}^{-1} (10^{12} \text{Yr})^{-2})$

$\text{UnitCoolingRate}_{\text{cgs}} = \frac{\text{UnitPressure}_{\text{cgs}}}{\text{UnitTime}_s} = 2.193973 \times 10^{-40}$, converts the cooling rate in $\text{g cm}^{-1} \text{s}^{-3}$ into internal units $(10^{10} M_\odot \text{Mpc}^{-1} (10^{12} \text{Yr})^{-3})$

$\text{UnitEnergy}_{\text{cgs}} = \frac{\text{UnitMass}_g \times \text{UnitLength}_{\text{cm}}^2}{\text{UnitTime}_s^2} = 1.989000 \times 10^{53}$, converts energy in $\text{g cm}^2 \text{s}^{-2}$ into internal units $((10^{10} M_\odot \text{Mpc}^2 (10^{12} \text{Yr})^{-2})$

$\text{Hubble} = \text{HUBBLE} \times \text{UnitTime}_s = 100.0001$, where $\text{HUBBLE} = 3.2407789 \times 10^{-18} \text{h s}^{-1}$, is the hubble constante in $(h \text{ Km s}^{-1} \text{Mpc}^{-1})$ or in internal units $(h 10^{12} \text{Yr}^{-1})$.

Then define a constant: $\text{RhoCrit} = \frac{3 \times \text{Hubble}^2}{8 \times \text{PI} \times G} = 27.75505 h^2 10^{10} M_\odot \text{Mpc}^{-3}$,

Definition at line 829 of file `init.c`.

References [G](#), [Hubble](#), [RhoCrit](#), [UnitCoolingRate_in_cgs](#), [UnitDensity_in_cgs](#), [UnitEnergy_in_cgs](#), [UnitLength_in_cm](#), [UnitMass_in_g](#), [UnitPressure_in_cgs](#), [UnitTime_in_Megayears](#), [UnitTime_in_s](#), and [UnitVelocity_in_cm_per_s](#).

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

4.19.1.98 void setup_spectrophotometric_model (void)

Reads in the look up tables from a given Stellar Population Synthesis Model. The default model is Bruzual & Charlot 2003. There are different files, each one corresponding to a different metallicity. On each file the tables have the Magnitudes for single bursts of $1 M_\odot$ for a range of ages and "inversely" k-corrected in case the code needs to compute observed frame magnitudes. 6 different metallicities are available at `./PhotTables/` with the file names `**_Set_Phot_table_m22.dat/m32/m42/m52/m62/m72` corresponding to $Z=0.0001/z=0.0004/z=0.004/Z=0.008/Z=0.02/Z=0.05/Z=0.1$. Two different prefixes are available DataRelease or SDSS corresponding respectively to BVRIK or ugriz bands (both with all the different metallicities.)

On each file, the three first numbers correspond to: the number of snapshots, the number of bands and the number of age bins. Then the age grid is listed. After that for each snapshot, for each age the value corresponding to a burst inversely k-corrected to that redshift, with that age (and metallicity) is listed.

The basic structure is number of columns corresponding to number of mags, repeated over the number of ages on the initial listed grid, multiplied by the number of snapshots, with the snapshot number listed in between.

`agTableZz[Mag][mettallicity][Snapshot][Age]`.

Definition at line 364 of file [init.c](#).

References [AgeTab](#), [Hubble_h](#), [init_jump_index\(\)](#), [MagTableZz](#), [PhotDir](#), [PhotPrefix](#), [RedshiftTab](#), [This-Task](#), and [UnitTime_in_Megayears](#).

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

4.19.1.99 double slab_model (float tau, float theta)

4.19.1.100 void starformation_and_feedback (int p, int centralgal, double time, double dt, int halonr)

Variables: reff=Rdisk, tdyn=Rdisk/Vmax, strdot=Mstar_dot, stars=strdot*dt

Definition at line 83 of file [recipe_starformation_and_feedback.c](#).

References [add_to_luminosities\(\)](#), [cal_H2O](#), [GALAXY::CentralGal](#), [check_disk_instability\(\)](#), [GALAXY::ColdGas](#), [DiskRadiusMethod](#), [EjectionOn](#), [EjectPreVelocity](#), [EjectSlope](#), [Energy_in_Reheat\(\)](#), [EnergySNcode](#), [EtaSNcode](#), [FeedbackEjectionEfficiency](#), [FeedbackEpsilon](#), [Feedback-Recipe](#), [FeedbackReheatingEpsilon](#), [FracZtoHot](#), [Gal](#), [GALAXY::GasDiskRadius](#), [get_metallicity\(\)](#), [get_stellar_disk_radius\(\)](#), [Halo](#), [GALAXY::HaloNr](#), [GALAXY::InfallVmax](#), [ListOutputSnaps](#), [GALAXY::MetalsColdGas](#), [GALAXY::MetalsHotGas](#), [GALAXY::Pos](#), [ReheatPreVelocity](#), [Reheat-Slope](#), [GALAXY::Rvir](#), [GALAXY::Sfr](#), [SfrAlpha](#), [SfrEfficiency](#), [SfrLawPivotVelocity](#), [SfrLawSlope](#), [GALAXY::SnapNum](#), [halo_data::SnapNum](#), [StarFormationRecipe](#), [GALAXY::StellarMass](#), [TrackDiskInstability](#), [GALAXY::Type](#), [update_from_feedback\(\)](#), [update_from_star_formation\(\)](#), [GALAXY::Vmax](#), [GALAXY::Vvir](#), [Yield](#), and [ZZ](#).

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

4.19.1.101 double time_to_present (double z)

Definition at line 19 of file [age.c](#).

References [Hubble](#).

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

4.19.1.102 void tsudy (int p, int magbin, int halonr)

Definition at line 57 of file [recipe_dust.c](#).

References [GALAXY::ColdGas](#), [Corrections](#), [DeltaM](#), [GALAXY::dObsLum](#), [GALAXY::dObsLumBulge](#), [GALAXY::dObsLumDust](#), [GALAXY::dObsYLum](#), [GALAXY::dObsYLumBulge](#), [Gal](#), [gasdev\(\)](#), [GALAXY::GasDiskRadius](#), [Halo](#), [GALAXY::Inclination](#), [ListOutputSnaps](#), [GALAXY::Lum](#), [GALAXY::LumBulge](#), [GALAXY::LumDust](#), [GALAXY::MetalsColdGas](#), [mu_seed](#), [GALAXY::ObsLum](#), [GALAXY::ObsLumBulge](#), [GALAXY::ObsLumDust](#), [GALAXY::ObsYLum](#), [GALAXY::ObsYLumBulge](#), [GALAXY::SnapNum](#), [GALAXY::YLum](#), [GALAXY::YLumBulge](#), and [ZZ](#).

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

4.19.1.103 void update_bulge_from_disk (int p, double stars)

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

4.19.1.104 void update_centralgal (int *ngal*, int *halonr*)

M_{vir} , R_{vir} and V_{vir} are only updated for type 0's. Once galaxies become satellites these quantities stay unchanged, so will be the values at infall.

If type = 0 then the HotRadius is the Viral Radius, which will be used in the cooling recipe.

Other infall information will not be used for central galaxies so we do not care whether they carry the correct values.

Definition at line 668 of file `recipe_misc.c`.

References [Gal](#), [get_virial_mass\(\)](#), [get_virial_radius\(\)](#), [get_virial_velocity\(\)](#), [Halo](#), [GALAXY::HaloSpin](#), [GALAXY::HotRadius](#), [GALAXY::InfallSnap](#), [GALAXY::InfallVmax](#), [GALAXY::MergeOn](#), [GALAXY::Mvir](#), [GALAXY::Rvir](#), [halo_data::SnapNum](#), [GALAXY::Type](#), [halo_data::Vmax](#), and [GALAXY::Vvir](#).

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

4.19.1.105 void update_from_feedback (int *p*, int *centralgal*, double *reheated_mass*, double *ejected_mass*, double *ejected_sat*, double *metallicity*)

Definition at line 445 of file `recipe_starformation_and_feedback.c`.

References [GALAXY::CentralGal](#), [GALAXY::ColdGas](#), [GALAXY::EjectedMass](#), [EjectionOn](#), [EjectionRecipe](#), [FeedbackRecipe](#), [Gal](#), [get_metallicity\(\)](#), [Halo](#), [GALAXY::HaloNr](#), [GALAXY::HotGas](#), [GALAXY::HotRadius](#), [GALAXY::MetalsColdGas](#), [GALAXY::MetalsEjectedMass](#), [GALAXY::MetalsHotGas](#), [GALAXY::Pos](#), [GALAXY::Rvir](#), [halo_data::SnapNum](#), [GALAXY::Type](#), [update_hot_frac\(\)](#), and [ZZ](#).

Referenced by [collisional_starburst_recipe\(\)](#), and [starformation_and_feedback\(\)](#).

4.19.1.106 double update_from_recycle (int *p*, double *time*, double *previoustime*)

4.19.1.107 void update_from_star_formation (int *p*, double *stars*, double *metallicity*)

Definition at line 422 of file `recipe_starformation_and_feedback.c`.

References [GALAXY::BulgeMass](#), [GALAXY::ColdGas](#), [DiskRadiusMethod](#), [Gal](#), [GALAXY::GasSpin](#), [get_stellar_disk_radius\(\)](#), [GALAXY::MetalsColdGas](#), [GALAXY::MetalsStellarMass](#), [RecycleFraction](#), [GALAXY::StellarMass](#), and [GALAXY::StellarSpin](#).

Referenced by [collisional_starburst_recipe\(\)](#), and [starformation_and_feedback\(\)](#).

4.19.1.108 void update_hot_frac (int *p*, double *reincorporated*, float *HotGas*)

Definition at line 754 of file `recipe_infall.c`.

References [Gal](#), [GALAXY::HotFrac](#), [GALAXY::HotRadius](#), [GALAXY::Len](#), [GALAXY::Mvir](#), [PartMass](#), and [GALAXY::Rvir](#).

Referenced by [cool_gas_onto_galaxy\(\)](#), [infall_recipe\(\)](#), [reincorporate_gas\(\)](#), and [update_from_feedback\(\)](#).

4.19.1.109 void update_hotgas (int *ngal*, int *centralgal*)

Updates the fraction of hot gas attached on each dark matter particles of the subhalo.

Definition at line 840 of file [recipe_misc.c](#).

References [GALAXY::CentralGal](#), [Gal](#), [Halo](#), [GALAXY::HaloNr](#), [GALAXY::HotFrac](#), [GALAXY::HotGas](#), [GALAXY::Len](#), [PartMass](#), [GALAXY::Pos](#), [GALAXY::Rvir](#), [halo_data::SnapNum](#), [GALAXY::Type](#), and [ZZ](#).

4.19.1.110 void update_ICL(int *p*, int *q*)

Definition at line 779 of file [recipe_infall.c](#).

References [Gal](#), [GALAXY::ICLLum](#), and [GALAXY::ObsICL](#).

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

4.19.1.111 void update_type_1(int *ngal*, int *halonr*, int *cenngal*, int *prog*)

If the galaxy has just become a satellite (its type hasn't yet been reset from 0 to 1), the fraction of hot gas to dark matter halo mass is recorded $HotFrac = \frac{M_{\text{hot}}}{M_{\text{vir}}}$.

If MERGE01 = 1, then a dynamical friction decay time scale is calculated for type 1's (as is done for type 2 - introduced for millennium II where the increased resolution means type 1 always retain some dark matter and orbit around for a long time). This is only calculated when the baryonic mass of the type 1 becomes larger than its dark matter mass. The code finds the type 0 to which this galaxy should merge and then sets up the merging clock.

Definition at line 699 of file [recipe_misc.c](#).

References [GALAXY::ColdGas](#), [halo_data::Descendant](#), [estimate_merging_time\(\)](#), [halo_data::FirstHaloInFOFgroup](#), [Gal](#), [Halo](#), [GALAXY::HotFrac](#), [GALAXY::HotGas](#), [GALAXY::Len](#), [GALAXY::MergeOn](#), [GALAXY::MergTime](#), [GALAXY::Mvir](#), [NumToTime\(\)](#), [GALAXY::OriMergTime](#), [PartMass](#), [GALAXY::SnapNum](#), [GALAXY::StellarMass](#), and [GALAXY::Type](#).

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

4.19.1.112 void update_type_2(int *ngal*, int *halonr*, int *prog*, int *mostmassive*)

Sets HotFrac and Hot Radius to 0, since all the hot gas has been stripped. Calls estimate_merging_time to get the merging time scale, calculated for the orbital decay due to dynamical friction, since this galaxy has lost its dark matter halo and its position cannot be tracked.

Definition at line 814 of file [recipe_misc.c](#).

References [estimate_merging_time\(\)](#), [Gal](#), [Halo](#), [GALAXY::HotFrac](#), [GALAXY::HotRadius](#), [GALAXY::MergeOn](#), [GALAXY::MergTime](#), [NumToTime\(\)](#), [GALAXY::OriMergTime](#), [GALAXY::SnapNum](#), and [GALAXY::Type](#).

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

4.19.1.113 int walk(int *nr*)

Definition at line 993 of file [save.c](#).

References [GALAXY::Done](#), [GALAXY::FirstProgGal](#), [GalaxiesInOrder](#), [GalCount](#), [GALAXY::GalID](#), [HaloGal](#), [GALAXY::LastProgGal](#), [GALAXY::MainLeaf](#), [GALAXY::NextProgGal](#), [GALAXY::TreeRoot](#), and [walk\(\)](#).

Referenced by [save_galaxy_tree\(\)](#), and [walk\(\)](#).

4.19.1.114 void write_all_galaxy_data (int filenr)

Definition at line 805 of file [io_tree.c](#).

References [FileNameGalaxies](#), [filled_galaxydata](#), [OutputDir](#), and [ptr_galaxydata](#).

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

4.19.1.115 void write_galaxy_data_snap (int n, int filenr)

Definition at line 832 of file [io_tree.c](#).

References [FileNameGalaxies](#), [filled_galsnapdata](#), [ListOutputSamps](#), [OutputDir](#), [ptr_galsnapdata](#), and [ZZ](#).

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

4.19.1.116 void write_galaxy_for_momaf (int n, int filenr)

Definition at line 861 of file [io_tree.c](#).

References [FileNameGalaxies](#), [filled_momafdata](#), [ListOutputSamps](#), [OutputDir](#), and [ptr_momafdata](#).

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

4.20 code/proto.h

```

00001
00002 #include "allvars.h"
00003
00004 size_t myfread(void *ptr, size_t size, size_t nmemb, FILE *stream);
00005 size_t myfwrite(void *ptr, size_t size, size_t nmemb, FILE *stream);
00006 int myfseek(FILE *stream, long offset, int whence);
00007 void load_all_auxdata(int filenr);
00008 void load_all_dbids(int filenr);
00009 void load_all_treedata(int filenr);
00010 void write_all_galaxy_data(int filenr);
00011 void write_galaxy_data_snap(int n, int filenr);
00012
00013 void save_galaxy_tree(int filenr, int tree);
00014 int walk(int nr);
00015
00016 int peano_hilbert_key(int x, int y, int z, int bits);
00017
00018 void get_coordinates(float *pos, float *vel, long long ID, int tree, int halonr,
    int snapnum);
00019 #ifndef MERGE01
00020 int join_galaxies_of_progenitors(int halonr, int nstart);
00021 #else
00022 int join_galaxies_of_progenitors(int halonr, int nstart, int cenngal);
00023 #endif
00024 void init(void);
00025 void set_units(void);
00026
00027 void load_tree_table(int filenr);
00028 void load_tree(int filenr, int nr);
00029 void save_galaxies(int filenr, int tree);
00030
00031 void prepare_galaxy_for_output(int n, int filenr, int tree, struct GALAXY *g, st
    ruct GALAXY_OUTPUT *o);
00032 void fix_units_for_output(struct GALAXY_OUTPUT *o);
00033

```

```

00034 void free_galaxies_and_tree(void);
00035 void free_tree_table(void);
00036 void print_allocated(void);
00037
00038 void read_parameter_file(char *fname);
00039 void *mymalloc(size_t n, char *myflag);
00040 void myfree(void *p);
00041
00042 void finalize_galaxy_file(int filenr);
00043
00044 void starformation_and_feedback(int p, int centralgal, double time, double dt, int halonr);
00045 void do_major_merger_starburst(int t, int centralgal, double time, double deltaT, int halonr);
00046 void add_galaxies_together(int t, int p);
00047 void add_to_luminosities(int p, double mstars, double time, double metallicity);
00048 void init_galaxy(int p, int halonr);
00049 double infall_recipe(int centralgal, int ngal, double Zcurr);
00050 void add_infall_to_hot(int centralgal, double infallingGas);
00051 double cooling_recipe(int centralgal, double dt);
00052 void cool_gas_onto_galaxy(int centralgal, double mcool);
00053 void reincorporate_gas(int centralgal, int p, double dt);
00054 void deal_with_galaxy_merger(int p, int merger_centralgal, int centralgal, double time, double deltaT, int halonr);
00055 double dmax(double x, double y);
00056 double do_reionization(int centralgal, double Zcurr);
00057 double do_AGN_heating(double coolingGas, int centralgal, double dt, double x);
00058 double NumToTime(int snapnum);
00059 double RedshiftObs(int snapnum);
00060 void update_from_star_formation(int p, double stars, double metallicity);
00061
00062 //*****
00063 //*****
00064 /* new model */
00065 void construct_galaxies(int halonr, int tree);
00066 void evolve_galaxies(int halonr, int ngal, int tree, int cenngal);
00067 void update_from_feedback(int p, int centralgal, double reheated_mass, double ejected_mass, double ejected_sat, double metallicity);
00068 double estimate_merging_time(int halonr, int mostmassive, int p);
00069
00070 double get_virial_velocity(int halonr, int gal);
00071 double get_virial_radius(int halonr, int gal);
00072 double get_virial_mass(int halonr, int gal);
00073 double collisional_starburst_recipe(double mass_ratio, int merger_centralgal, int centralgal, double time, double deltaT, int halonr);
00074
00075 /*de lucia's */
00076 //void construct_galaxies(int halonr);
00077 //void evolve_galaxies(int halonr, int ngal);
00078 //void update_from_feedback(int p, int centralgal, double reheated_mass, double ejected_mass, double metallicity);
00079 //double estimate_merging_time(int halonr, int mostmassive);
00080
00081 //double get_virial_velocity(int halonr);
00082 //double get_virial_radius(int halonr);
00083 //double get_virial_mass(int halonr);
00084 //void collisional_starburst_recipe(double mass_ratio, int merger_centralgal, int centralgal, double time, double deltaT, int halonr);
00085
00086 //*****
00087 //*****
00088
00089 void make_bulge_from_burst(int p);
00090 void grow_black_hole(int halonr, int merger_centralgal, double mass_ratio, double dt);
00091 void check_disk_instability(int p);
00092

```

```

00093 double lum_to_mag(double lum);
00094
00095
00096 double slab_model(float tau, float teta);
00097 double get_metallicity(double gas, double metals);
00098
00099 double get_disk_radius(int halonr, int p);
00100 void get_gas_disk_radius(int p);
00101 void get_stellar_disk_radius(int p);
00102
00103 void read_output_snaps(void);
00104 void read_snap_list(void);
00105 void setup_spectrophotometric_model(void);
00106
00107 // begin tsud
00108 void read_tsud_tables(void);
00109 void tsudy(int p, int magbin, int halonr);
00110 // end tsud
00111
00112 void read_cooling_functions(void);
00113 double get_metaldependent_cooling_rate(double logTemp, double logZ);
00114 double get_rate(int tab, double logTemp);
00115
00116 double time_to_present(double z);
00117 double integrand_time_to_present(double a, void *param);
00118 void find_interpolation_point(double timenow, double timetarget, int *index, double *f1, double *f2);
00119 void find_interpolated_lum(double timenow, double timetarget, double metallicity, int *metindex, int *tabindex, double *f1, double *f2, double *fmet1, double *fmet2);
00120 void find_interpolated_obs_lum(double timenow, double timetarget, double metallicity, double redshift, int *metindex, int *tabindex, int *redindex, double *f1, double *f2, double *fmet1, double *fmet2, double *fred1, double *fred2);
00121
00122 void init_jump_index(void);
00123 int get_jump_index(double age);
00124
00125 void disrupt(int p, int centralgal, int tree);
00126 double peri_radius(int p, int centralgal);
00127
00128 double dmin(double x, double y);
00129 double hot_retain_sat(int i, int centralgal, double infallingMass);
00130 void check_options();
00131
00132 #ifdef OUTPUT_MOMAF_INPUTS
00133 size_t momaf_fwrite(void *ptr, size_t size, size_t nmemb, FILE * stream);
00134 void write_galaxy_for_momaf(int n, int filenr);
00135 void finalize_momaf_file(int filenr);
00136 void prepare_galaxy_for_momaf(int n, int filenr, int tree, struct GALAXY *g, struct MOMAF_INPUTS *o);
00137 #endif
00138
00139 void read_recgas(void);
00140 void cal_gas_recycle(int ngal);
00141 double update_from_recycle(int p, double time, double previoustime);
00142 double get_initial_disk_radius(int halonr, int p);
00143 void update_bulge_from_disk(int p, double stars);
00144 double bulge_from_disk(double frac);
00145 double func_size(double x, double a);
00146 void bulgesize_from_merger(double mass_ratio, int merger_centralgal, int p, double Mcstar, double Mcbulge, double Mcgas, double frac);
00147 double diskmass_r(double rmin, double rd, double Sigma0, double dr);
00148 double bulgemass_r(double rmin, double re, double Ie, double dr);
00149 double sat_radius(int p);
00150
00151 #ifdef SPECIFYFILENR
00152 void read_file_nrs(void);

```

```

00153 #endif
00154
00155 #ifdef H2FORMATION
00156
00157 /*for H2 Formation*/
00158 void read_sfrz(void);
00159 double cal_H2(int p);
00160 void find_interpolate_h2(double metalicity, double rho, int *tabindex, int *metin
    dex, double *f1, double *f2, double *fmet1, double *fmet2);
00161 #endif
00162
00163 double Energy_in_Reheat(int p);
00164 void update_type_2(int ngal, int halonr, int prog, int mostmassive);
00165 void update_centralgal(int ngal, int halonr);
00166 void update_hotgas(int ngal, int centralgal);
00167 void update_type_1(int ngal, int halonr, int cenngal, int prog);
00168 void update_ICL(int p, int q);
00169
00170
00171 /* check funs */
00172 void checkbulgesize_main(int ngal);
00173 void update_hot_frac(int p, double reincorporated, float HotGas);
00174 #ifdef MERGE01
00175 int set_merger_center(int fofhalo);
00176 #endif
00177
00178
00179 // ~ NEW_IO io_tree functions ~
00180 #ifdef NEW_IO
00181 void openallfiles(int filenr);
00182 void closeallfiles(int filenr);
00183 #endif
00184 #ifdef GALAXYTREE
00185 // Open the output tree file
00186 FILE* open_outputtree_file(int filenr, char mode[]);
00187 #endif
00188 FILE* open_tree_file(int filenr);
00189 FILE* open_treedbids_file(int filenr);
00190 FILE* open_treeaux_file(int filenr);
00191
00192

```

4.21 code/read_parameters.c File Reference

reads all the parameters in input.par into global variables that can be used by the code.

Functions

- void [read_parameter_file](#) (char *fname)

4.21.1 Detailed Description

Definition in file [read_parameters.c](#).

4.21.2 Function Documentation

4.21.2.1 void read_parameter_file (char * *fname*)

Definition at line 18 of file [read_parameters.c](#).

References [AgnEfficiency](#), [AGNRecipeOn](#), [BaryonFrac](#), [BlackHoleGrowthRate](#), [BoxSize](#), [BulgeFormationInMinorMergersOn](#), [CoolFunctionsDir](#), [CoolingCutoff](#), [CoolingVelocityCutOff](#), [DiskRadiusMethod](#), [EjectionOn](#), [EjectionRecipe](#), [EjectPreVelocity](#), [EjectSlope](#), [EnergySN](#), [EtaSN](#), [FeedbackEjectionEfficiency](#), [FeedbackEpsilon](#), [FeedbackRecipe](#), [FeedbackReheatingEpsilon](#), [FileNameGalaxies](#), [FileNrDir](#), [FilesPerSnapshot](#), [FileWithOutputSamps](#), [FileWithSnapList](#), [FirstFile](#), [FracZtoHot](#), [Hashbits](#), [Hubble_h](#), [LastFile](#), [LastSnapShotNr](#), [MetallicityOption](#), [Omega](#), [OmegaLambda](#), [OutputDir](#), [PartMass](#), [PhotDir](#), [PhotPrefix](#), [RecycleFraction](#), [ReheatPreVelocity](#), [ReheatSlope](#), [ReIncorporationFactor](#), [ReIncorporationRecipe](#), [Reionization_z0](#), [Reionization_zr](#), [ReionizationOn](#), [SatelliteRecipe](#), [SfrAlpha](#), [SfrEfficiency](#), [SfrLawPivotVelocity](#), [SfrLawSlope](#), [SimulationDir](#), [SNinReheat](#), [StarBurstRecipe](#), [StarBurstsInMajorMergersOn](#), [StarFormationRecipe](#), [ThisTask](#), [ThreshMajorMerger](#), [TrackDiskInstability](#), [UnitLength_in_cm](#), [UnitMass_in_g](#), [UnitVelocity_in_cm_per_s](#), and [Yield](#).

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

4.22 code/read_parameters.c

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <string.h>
00004 #include <math.h>
00005
00006 #include "allvars.h"
00007 #include "proto.h"
00008
00009
00010 #define DOUBLE 1
00011 #define STRING 2
00012 #define INT 3
00013 #define MAXTAGS 300
00014
00015 /*@@file read_parameters.c reads all the parameters in input.par into global variables
00016 *      that can be used by the code. */
00017
00018 void read_parameter_file(char *fname)
00019 {
00020     FILE *fd;
00021     char buf[400], buf1[400], buf2[400], buf3[400];
00022     int i, j, nt = 0;
00023     int id[MAXTAGS];
00024     void *addr[MAXTAGS];
00025     char tag[MAXTAGS][50];
00026     int errorFlag = 0;
00027
00028
00029 #ifdef PARALLEL
00030     if(ThisTask == 0)
00031         {
00032 #endif
00033 #ifdef UseFullSfr
00034     printf("\nUseFullSfr set in Makefile\n");
00035 #endif
00036 #ifdef PARALLEL
00037     }
00038 #endif

```

```
00039
00040 #ifdef PARALLEL
00041     if(ThisTask == 0)
00042         printf("\nreading parameter file:\n\n");
00043 #else
00044     printf("\nreading parameter file:\n\n");
00045 #endif
00046
00047     strcpy(tag[nt], "OutputDir");
00048     addr[nt] = OutputDir;
00049     id[nt++] = STRING;
00050
00051     strcpy(tag[nt], "FileNameGalaxies");
00052     addr[nt] = FileNameGalaxies;
00053     id[nt++] = STRING;
00054
00055     strcpy(tag[nt], "SimulationDir");
00056     addr[nt] = SimulationDir;
00057     id[nt++] = STRING;
00058
00059     strcpy(tag[nt], "FileWithOutputSamps");
00060     addr[nt] = FileWithOutputSamps;
00061     id[nt++] = STRING;
00062
00063 #ifdef SPECIFYFILENR
00064     strcpy(tag[nt], "FileNrDir");
00065     addr[nt] = FileNrDir;
00066     id[nt++] = STRING;
00067 #endif
00068
00069     strcpy(tag[nt], "PhotDir");
00070     addr[nt] = PhotDir;
00071     id[nt++] = STRING;
00072
00073     strcpy(tag[nt], "PhotPrefix");
00074     addr[nt] = PhotPrefix;
00075     id[nt++] = STRING;
00076
00077     strcpy(tag[nt], "CoolFunctionsDir");
00078     addr[nt] = CoolFunctionsDir;
00079     id[nt++] = STRING;
00080
00081     strcpy(tag[nt], "FileWithSnapList");
00082     addr[nt] = FileWithSnapList;
00083     id[nt++] = STRING;
00084
00085     strcpy(tag[nt], "FilesPerSnapshot");
00086     addr[nt] = &FilesPerSnapshot;
00087     id[nt++] = INT;
00088
00089     strcpy(tag[nt], "LastSnapShotNr");
00090     addr[nt] = &LastSnapShotNr;
00091     id[nt++] = INT;
00092
00093     strcpy(tag[nt], "FirstFile");
00094     addr[nt] = &FirstFile;
00095     id[nt++] = INT;
00096
00097     strcpy(tag[nt], "LastFile");
00098     addr[nt] = &LastFile;
00099     id[nt++] = INT;
00100
00101     strcpy(tag[nt], "CoolingVelocityCutOff");
00102     addr[nt] = &CoolingVelocityCutOff;
00103     id[nt++] = DOUBLE;
00104
00105     strcpy(tag[nt], "ThreshMajorMerger");
```

```
00106     addr[nt] = &ThreshMajorMerger;
00107     id[nt++] = DOUBLE;
00108
00109     strcpy(tag[nt], "SfrAlpha");
00110     addr[nt] = &SfrAlpha;
00111     id[nt++] = DOUBLE;
00112
00113     strcpy(tag[nt], "SfrLawPivotVelocity");
00114     addr[nt] = &SfrLawPivotVelocity;
00115     id[nt++] = DOUBLE;
00116
00117     strcpy(tag[nt], "SfrLawSlope");
00118     addr[nt] = &SfrLawSlope;
00119     id[nt++] = DOUBLE;
00120
00121
00122     strcpy(tag[nt], "ReheatPreVelocity");
00123     addr[nt] = &ReheatPreVelocity;
00124     id[nt++] = DOUBLE;
00125
00126     strcpy(tag[nt], "ReheatSlope");
00127     addr[nt] = &ReheatSlope;
00128     id[nt++] = DOUBLE;
00129
00130     strcpy(tag[nt], "EjectPreVelocity");
00131     addr[nt] = &EjectPreVelocity;
00132     id[nt++] = DOUBLE;
00133
00134     strcpy(tag[nt], "EjectSlope");
00135     addr[nt] = &EjectSlope;
00136     id[nt++] = DOUBLE;
00137
00138     strcpy(tag[nt], "SNinReheat");
00139     addr[nt] = &SNinReheat;
00140     id[nt++] = INT;
00141
00142     strcpy(tag[nt], "SatelliteRecipe");
00143     addr[nt] = &SatelliteRecipe;
00144     id[nt++] = INT;
00145
00146     strcpy(tag[nt], "ReIncorporationRecipe");
00147     addr[nt] = &ReIncorporationRecipe;
00148     id[nt++] = INT;
00149
00150     strcpy(tag[nt], "FeedbackEpsilon");
00151     addr[nt] = &FeedbackEpsilon;
00152     id[nt++] = DOUBLE;
00153
00154
00155     strcpy(tag[nt], "RecycleFraction");
00156     addr[nt] = &RecycleFraction;
00157     id[nt++] = DOUBLE;
00158
00159     strcpy(tag[nt], "ReIncorporationFactor");
00160     addr[nt] = &ReIncorporationFactor;
00161     id[nt++] = DOUBLE;
00162
00163     strcpy(tag[nt], "UnitVelocity_in_cm_per_s");
00164     addr[nt] = &UnitVelocity_in_cm_per_s;
00165     id[nt++] = DOUBLE;
00166
00167     strcpy(tag[nt], "UnitLength_in_cm");
00168     addr[nt] = &UnitLength_in_cm;
00169     id[nt++] = DOUBLE;
00170
00171     strcpy(tag[nt], "UnitMass_in_g");
00172     addr[nt] = &UnitMass_in_g;
```

```
00173     id[nt++] = DOUBLE;
00174
00175     strcpy(tag[nt], "Hubble_h");
00176     addr[nt] = &Hubble_h;
00177     id[nt++] = DOUBLE;
00178
00179     strcpy(tag[nt], "StarBurstsInMajorMergersOn");
00180     addr[nt] = &StarBurstsInMajorMergersOn;
00181     id[nt++] = INT;
00182
00183     strcpy(tag[nt], "BulgeFormationInMinorMergersOn");
00184     addr[nt] = &BulgeFormationInMinorMergersOn;
00185     id[nt++] = INT;
00186
00187     strcpy(tag[nt], "MetallicityOption");
00188     addr[nt] = &MetallicityOption;
00189     id[nt++] = INT;
00190
00191
00192     strcpy(tag[nt], "EjectionOn");
00193     addr[nt] = &EjectionOn;
00194     id[nt++] = INT;
00195
00196     strcpy(tag[nt], "CoolingCutoff");
00197     addr[nt] = &CoolingCutoff;
00198     id[nt++] = INT;
00199
00200     strcpy(tag[nt], "ReionizationOn");
00201     addr[nt] = &ReionizationOn;
00202     id[nt++] = INT;
00203
00204     strcpy(tag[nt], "StarFormationRecipe");
00205     addr[nt] = &StarFormationRecipe;
00206     id[nt++] = INT;
00207
00208
00209     strcpy(tag[nt], "StarBurstRecipe");
00210     addr[nt] = &StarBurstRecipe;
00211     id[nt++] = INT;
00212
00213     strcpy(tag[nt], "FeedbackRecipe");
00214     addr[nt] = &FeedbackRecipe;
00215     id[nt++] = INT;
00216
00217     strcpy(tag[nt], "EjectionRecipe");
00218     addr[nt] = &EjectionRecipe;
00219     id[nt++] = INT;
00220
00221     strcpy(tag[nt], "DiskRadiusMethod");
00222     addr[nt] = &DiskRadiusMethod;
00223     id[nt++] = INT;
00224
00225     strcpy(tag[nt], "TrackDiskInstability");
00226     addr[nt] = &TrackDiskInstability;
00227     id[nt++] = INT;
00228
00229     strcpy(tag[nt], "AGNrecipeOn");
00230     addr[nt] = &AGNrecipeOn;
00231     id[nt++] = INT;
00232
00233     strcpy(tag[nt], "BaryonFrac");
00234     addr[nt] = &BaryonFrac;
00235     id[nt++] = DOUBLE;
00236
00237     strcpy(tag[nt], "Omega");
00238     addr[nt] = &Omega;
00239     id[nt++] = DOUBLE;
```

```
00240     strcpy(tag[nt], "OmegaLambda");
00241     addr[nt] = &OmegaLambda;
00242     id[nt++] = DOUBLE;
00243
00244     strcpy(tag[nt], "PartMass");
00245     addr[nt] = &PartMass;
00246     id[nt++] = DOUBLE;
00247
00248     strcpy(tag[nt], "EnergySN");
00249     addr[nt] = &EnergySN;
00250     id[nt++] = DOUBLE;
00251
00252     strcpy(tag[nt], "EtaSN");
00253     addr[nt] = &EtaSN;
00254     id[nt++] = DOUBLE;
00255
00256     strcpy(tag[nt], "Yield");
00257     addr[nt] = &Yield;
00258     id[nt++] = DOUBLE;
00259
00260     strcpy(tag[nt], "FracZtoHot");
00261     addr[nt] = &FracZtoHot;
00262     id[nt++] = DOUBLE;
00263
00264     strcpy(tag[nt], "SfrEfficiency");
00265     addr[nt] = &SfrEfficiency;
00266     id[nt++] = DOUBLE;
00267
00268     strcpy(tag[nt], "FeedbackReheatingEpsilon");
00269     addr[nt] = &FeedbackReheatingEpsilon;
00270     id[nt++] = DOUBLE;
00271
00272     strcpy(tag[nt], "FeedbackEjectionEfficiency");
00273     addr[nt] = &FeedbackEjectionEfficiency;
00274     id[nt++] = DOUBLE;
00275
00276     strcpy(tag[nt], "BlackHoleGrowthRate");
00277     addr[nt] = &BlackHoleGrowthRate;
00278     id[nt++] = DOUBLE;
00279
00280     strcpy(tag[nt], "AgnEfficiency");
00281     addr[nt] = &AgnEfficiency;
00282     id[nt++] = DOUBLE;
00283
00284     strcpy(tag[nt], "Reionization_z0");
00285     addr[nt] = &Reionization_z0;
00286     id[nt++] = DOUBLE;
00287
00288     strcpy(tag[nt], "Reionization_zr");
00289     addr[nt] = &Reionization_zr;
00290     id[nt++] = DOUBLE;
00291
00292 #ifdef GALAXYTREE
00293     strcpy(tag[nt], "Hashbits");
00294     addr[nt] = &Hashbits;
00295     id[nt++] = INT;
00296
00297     strcpy(tag[nt], "BoxSize");
00298     addr[nt] = &BoxSize;
00299     id[nt++] = DOUBLE;
00300
00301 #endif
00302
00303 if((fd = fopen(fname, "r")))
00304 {
00305     while(!feof(fd))
00306     {
```

```

00307     *buf = 0;
00308     fgets(buf, 200, fd);
00309     if(sscanf(buf, "%s%s%s", buf1, buf2, buf3) < 2)
00310         continue;
00311     if(buf1[0] == '%')
00312         continue;
00313
00314     for(i = 0, j = -1; i < nt; i++)
00315         if(strcmp(buf1, tag[i]) == 0)
00316             {
00317                 j = i;
00318                 tag[i][0] = 0;
00319                 break;
00320             }
00321
00322     if(j >= 0)
00323     {
00325 #ifdef PARALLEL
00326         if(ThisTask == 0)
00327             printf("%35s\t%10s\n", buf1, buf2);
00328 #else
00329         printf("%35s\t%10s\n", buf1, buf2);
00330 #endif
00331         switch (id[j])
00332         {
00333             case DOUBLE:
00334                 *((double *) addr[j]) = atof(buf2);
00335                 break;
00336             case STRING:
00337                 strcpy(addr[j], buf2);
00338                 break;
00339             case INT:
00340                 *((int *) addr[j]) = atoi(buf2);
00341                 break;
00342         }
00343     }
00344     else
00345     {
00346         printf("Error in file %s: Tag '%s' not allowed or multiple define
d.\n", fname, buf1);
00347         errorFlag = 1;
00348     }
00349 }
00350 fclose(fd);
00351
00352 i = strlen(OutputDir);
00353 if(i > 0)
00354     if(OutputDir[i - 1] != '/')
00355         strcat(OutputDir, "/");
00356 }
00357 else
00358 {
00359     printf("Parameter file %s not found.\n", fname);
00360     errorFlag = 1;
00361 }
00362
00363
00364 for(i = 0; i < nt; i++)
00365 {
00366     if(*tag[i])
00367     {
00368         printf("Error. I miss a value for tag '%s' in parameter file '%s'.\n",
tag[i], fname);
00369         errorFlag = 1;
00370     }
00371 }

```

```

00372
00373     if(errorFlag)
00374         exit(1);
00375
00376 }
```

4.23 code/recipe_cooling.c File Reference

[recipe_cooling.c](#) calculates the amount of mass that cools from the hot to the cold phase at each timestep (this fraction is then reduced by AGN heating) and updates the hot and cold gas fractions accordingly.

Functions

- double [cooling_recipe](#) (int p, double dt)
main cooling recipe, where the cooling rates are calculated
- double [do_AGN_heating](#) (double coolingGas, int centralgal, double dt, double x)
calculates the energy released by black holes due to passive accretion, that will be used to reduced the cooling.
- void [cool_gas_onto_galaxy](#) (int p, double mcool)
updates the fractions of hot and cold gas due to cooling.

4.23.1 Detailed Description

This recipe calculates the amount of mass that cools from the hot to the cold phase at each timestep. Two different infalling regimes are assumed depending on the redshift and mass of the halos. This is the recipe proposed by (White & Rees, 1978) and that has since been used in most SA.

Before only central galaxies could cool gas, so R_hot was just set equal to Rvir. After Guo2010, satellites can have cooling, but since they are loosing dark matter, Rvir is not a good approximation of R_Hot so Gal.HotRadius was introduced.

-> At early times and for low mass halos, the cooling radius can be larger then the virial radius. In this case, despite the infalling gas being shock heated to the virial temperature, it condenses within a halo dynamical time and a quasi-static atmosphere cannot form. Single line on the code, with no calculations needed, just that $t_{\text{dyn},h} = R_{\text{vir}}/V_{\text{vir}}$ and therefore $\dot{M}_{\text{cool}} = 0.5 \frac{m_{\text{hot}}}{t_{\text{dyn}}} = 0.5 m_{\text{hot}} \frac{V_{\text{vir}}}{R_{\text{vir}}}$

-> For massive halos and at late times, the cooling radius lies within the virial radius and the infalling gas does form a quasi-static hot atmosphere that extends to the virial radius. This gas can cool at later times and its accretion into central regions is modeled through a cooling flow.

$$\dot{M}_{\text{cool}} = m_{\text{hot}} \frac{r_{\text{cool}}}{R_{\text{vir}}} \frac{1}{t_{\text{cool}}} \text{ eq5 (no 0.5 factor...)}$$

In both cases the condensed gas settles into a central disk where star formation occurs. The cooling rate is given by the cooling-AGN heating.

$$\text{BH quiet accretion rate: } \dot{m}_{\text{BH,R}} = k_{\text{AGN}} \left(\frac{m_{\text{BH}}}{10^8 M_{\odot}} \right) \left(\frac{f_{\text{hot}}}{0.1} \right) \left(\frac{V_{\text{vir}}}{200 \text{ km s}^{-1}} \right)^3$$

$$\text{Luminosity from quiet accretion: } L_{\text{BH}} = \eta \dot{m}_{\text{BH,R}} c^2,$$

$$\text{Corresponding reduction in cooling: } \dot{m}'_{\text{cool}} = \dot{m}_{\text{cool}} - \frac{L_{\text{BH}}}{\frac{1}{2} V_{\text{vir}}^2}$$

Definition in file [recipe_cooling.c](#).

4.23.2 Function Documentation

4.23.2.1 void cool_gas_onto_galaxy (int *p*, double *mcool*)

`cool_gas_onto_galaxy` updates the fractions of hot and cold gas due to cooling. This is done for the mass, metals and, after Guo2010, spin components

Definition at line 276 of file `recipe_cooling.c`.

References `GALAXY::ColdGas`, `DiskRadiusMethod`, `Gal`, `get_gas_disk_radius()`, `get_metallicity()`, `GALAXY::HotGas`, `GALAXY::MetalsColdGas`, `GALAXY::MetalsHotGas`, and `update_hot_frac()`.

Referenced by `main()`.

4.23.2.2 double cooling_recipe (int *p*, double *dt*)

Definition at line 71 of file `recipe_cooling.c`.

References `AGNRecipeOn`, `CoolingCutoff`, `GALAXY::CoolingGas`, `GALAXY::CoolingRadius`, `CoolingVelocityCutOff`, `do_AGN_heating()`, `Gal`, `get_metaldependent_cooling_rate()`, `GALAXY::HotGas`, `GALAXY::HotRadius`, `GALAXY::MetalsHotGas`, `ReionizationOn`, `GALAXY::Rvir`, `UnitDensity_in_cgs`, `UnitTime_in_s`, `GALAXY::Vvir`, and `GALAXY::XrayLum`.

Referenced by `main()`.

4.23.2.3 double do_AGN_heating (double *coolingGas*, int *centralgal*, double *dt*, double *x*)

`do_AGN_heating` calculates the amount of energy released by black holes due to passive accretion, Which is then used to reduced the cooling.

There is one parameter, `AgnEfficiency`, which is the efficiency of AGN passive accretion and consequently of cooling flow reheating, Eqs. 10, 11 & 12 in Croton 2006. The standard recipe is `AGNRecipeOn =1` (empirical) with options 2 and 3 representing Bondi-Hoyle and cold cloud accretion. The three should be identical and the use of empirical avoids people shouting about dutty-cycles being incoistent with Bondi-Holy & etc.

Definition at line 178 of file `recipe_cooling.c`.

References `AgnEfficiency`, `AGNRecipeOn`, `GALAXY::BlackHoleMass`, `G`, `Gal`, `get_metallicity()`, `GALAXY::HotGas`, `GALAXY::HotRadius`, `GALAXY::MetalsHotGas`, `GALAXY::Mvir`, `GALAXY::Rvir`, `UnitEnergy_in_cgs`, `UnitMass_in_g`, `UnitTime_in_s`, and `GALAXY::Vvir`.

Referenced by `cooling_recipe()`.

4.24 code/recipe_cooling.c

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <string.h>
00004 #include <math.h>
00005 #include <time.h>
00006
00007 #include "allvars.h"
00008 #include "proto.h"
00009
00010 /** @file recipe_cooling.c
00011 *  @brief recipe_cooling.c calculates the amount of mass that cools
00012 *  from the hot to the cold phase at each timestep (this fraction

```

```

00013 * is then reduced by AGN heating) and updates the hot and cold gas
00014 * fractions accordingly.
00015 *
00016 * This recipe calculates the amount of mass that cools from the hot
00017 * to the cold phase at each timestep. Two different infalling regimes
00018 * are assumed depending on the redshift and mass of the halos. This is
00019 * the recipe proposed by (White & Rees, 1978) and that has since been
00020 * used in most SA.
00021 *
00022 * Before only central galaxies could cool gas, so R_hot was just set
00023 * equal to Rvir. After Guo2010, satellites can have cooling, but since
00024 * they are loosing dark matter, Rvir is not a good approximation of R_Hot
00025 * so Gal.HotRadius was introduced.
00026 *
00027 * -> At early times and for low mass halos, the cooling radius can be
00028 * larger then the virial radius. In this case, despite the infalling
00029 * gas being shock heated to the virial temperature, it condenses within
00030 * a halo dynamical time and a quasi-static atmosphere cannot form.
00031 * Single line on the code, with no calculations needed, just that
00032 *  $\dot{t}_{\text{dyn}} = R_{\text{vir}} / V_{\text{vir}}$ 
00033 * and therefore
00034 *  $\dot{M}_{\text{cool}} = 0.5 \frac{m_{\text{hot}}}{t_{\text{dyn}}} = 0.5 m_{\text{hot}} \frac{V_{\text{vir}}}{R_{\text{vir}}}$ 
00035 *
00036 *
00037 *
00038 * -> For massive halos and at late times, the cooling radius lies within
00039 * the virial radius and the infalling gas does form a quasi-static hot
00040 * atmosphere that extends to the virial radius. This gas can cool at
00041 * later times and its accretion into central regions is modeled through
00042 * a cooling flow.
00043 *
00044 *  $\dot{M}_{\text{cool}} = m_{\text{hot}} \frac{r_{\text{cool}}}{R_{\text{vir}}} \frac{1}{t_{\text{cool}}}$ 
00045 * eq5 (no 0.5 factor...)
00046 *
00047 *
00048 *
00049 * In both cases the condensed gas settles into a central disk where
00050 * star formation occurs. The cooling rate is given by the cooling-AGN
00051 * heating.
00052 *
00053 * BH quiet accretion rate:
00054 *  $\dot{m}_{\text{BH}} = k_{\text{AGN}} \left( \frac{m_{\text{BH}}}{10^8} M_{\odot} \right)^{0.1} \left( \frac{f_{\text{cool}}}{100} \right)^3$ 
00055 *
00056 *
00057 *
00058 *
00059 Luminosity from quiet accretion:
00060  $L_{\text{BH}} = \eta \dot{m}_{\text{BH}} c^2$ 
00061 *
00062 Corresponding reduction in cooling:
00063 *  $\dot{m}'_{\text{cool}} = \dot{m}_{\text{cool}} - \frac{L_{\text{BH}}}{V_{\text{vir}}^2}$ 
00064 *
00065 *
00066 *
00067 */
00068
00069 /** @brief main cooling recipe, where the cooling rates are calculated */
00070
00071 double cooling_recipe(int p, double dt)
00072 {
00073     double Vvir, Rvir, x, lambda, tcool, rcool, temp, tot_hotMass, tot_metals, HotRa
00074     dius;
00075     double coolingGas, logZ, rho_rcool, rho0, cool_rate1, cool_rate2, age;
00076
00077     tot_hotMass = Gal[p].HotGas;
00078     tot_metals = Gal[p].MetalsHotGas;

```

```

00079
00080     /* cooling cut off is an old option to artificially shutdown
00081     * star formation in massive halos. Useful before Croton2006,
00082     * now=0.*/
00083     if((tot_hotMass > 1.0e-6) && ((CoolingCutoff == 0) || (Gal[p].Vvir <
00084         CoolingVelocityCutOff)))
00085     {
00086         /* TODO - Should Rvir be used at all in this recipe after Guo2010?
00087         * probably always HotRadius*/
00088         Vvir = Gal[p].Vvir;
00089         Rvir = Gal[p].Rvir;
00090         //put h
00091         tcool = Rvir / Vvir; // tcool = t_dynamical = Rvir/Vvir
00092
00093         /* temp -> Temperature of the Gas in Kelvin, obtained from
00094         * hidrostatic equilibrium KT=0.5*mu_p*(Vc)^2 assuming Vvir~Vc */
00095         temp = 35.9 * Vvir * Vvir;
00096
00097         if (Gal[p].Type == 0)
00098             HotRadius = Gal[p].Rvir;
00099         else
00100             HotRadius = Gal[p].HotRadius;
00101
00102         if(tot_metals > 0)
00103             logZ = log10(tot_metals / tot_hotMass);
00104         else
00105             logZ = -10.0;
00106
00107         //eq. 3 and 4 Guo2010
00108         lambda = get_metaldependent_cooling_rate(log10(temp), logZ);
00109         x = PROTONMASS * BOLTZMANN * temp / lambda; // now this has units sec g/cm^
00110         3
00111         x /= (UnitDensity_in_cg * UnitTime_in_s); // now in internal units
00112         rho_rcool = x / (0.28086 * tcool);
00113         /* an isothermal density profile for the hot gas is assumed here */
00114         rho0 = tot_hotMass / (4 * M_PI * HotRadius);
00115         rcool = sqrt(rho0 / rho_rcool);
00116
00117         if (Gal[p].CoolingRadius < rcool)
00118             Gal[p].CoolingRadius = rcool;
00119
00120         //if Hotradius is used, when galaxies become type 1's there will be a discont
00121         inuity in the cooling
00122         if(rcool > Rvir) // INFALL DOMINATED REGIME
00123             //coolingGas = tot_hotMass; - Delucia 2007
00124             /*comes in to keep the continuity (Delucia2004) */
00125             //put h
00126             coolingGas = tot_hotMass / (HotRadius / Vvir) * dt;
00127         else // HOT PHASE REGIME -> TODO - WHERE IS THE 0.5 factor of eq 5
00128             /*coolingGas = (tot_hotMass / Rvir) * (rcool / tcool) * dt * 0.5; */
00129             coolingGas = (tot_hotMass / HotRadius) * (rcool / tcool) * dt ;
00130
00131         //Photoionizing background
00132         if (log10(temp) < 4.0)
00133             coolingGas = 0.;
00134
00135         if(coolingGas > tot_hotMass)
00136             coolingGas = tot_hotMass;
00137         else if(coolingGas < 0.0)
00138             coolingGas = 0.0;
00139
00140         /* Suppress cooling due to AGN feedback, the gas is not actually heated,
00141         * just the amount of cooling reduced. */
00142         if(AGNRecipeOn > 0)
00143             coolingGas -= do_AGN_heating(coolingGas, p, dt, x);
00144
00145

```

```

00143     if(coolingGas < 0.0)
00144         coolingGas = 0.0;
00145
00146     }
00147
00148     else
00149     {
00150         coolingGas = 0.0;
00151     }
00152
00153 /* a direct way of implementing reionization, automatic shutdown cooling
00154 * in small halos -> not used */
00155 if(ReionizationOn == 2 && Gal[p].Vvir < 50.)
00156 {
00157     coolingGas = 0.0;
00158 }
00159
00160 /* determine the xray luminosity of any cooling gas in this snapshot
00161 * (White & Frenk 1991 eq21) */
00162 if(coolingGas > 0.0)
00163     Gal[p].XrayLum =
00164     log10(2.5 * (coolingGas / dt) * 6.31 * Gal[p].Vvir * Gal[p].Vvir) + 35.0;
00165 else
00166     Gal[p].XrayLum = 0.0;
00167 Gal[p].CoolingGas += coolingGas;
00168
00169 return coolingGas;
00170
00171 }
00172
00173
00174
00175 /** @brief calculates the energy released by black holes due to passive accretion
00176     * that will be used to reduced the cooling.*/
00177
00178 double do_AGN_heating(double coolingGas, int centralgal, double dt, double x)
00179 {
00180     double AGNrate, AGNheating, AGNaccreted, AGNcoeff, metallicity, EDDrate, FreeFa
11Radius;
00181
00182     /** @brief do_AGN_heating calculates the amount of energy released by
00183     * black holes due to passive accretion, Which is then used to reduced
00184     * the cooling.
00185     *
00186     * There is one parameter, AgnEfficiency, which is the efficiency of AGN
00187     * passive accretion and consequently of cooling flow reheating, Eqs. 10,
00188     * 11 & 12 in Croton 2006. The standard recipe is AGNRecipeOn =1 (empirical)
00189     * with options 2 and 3 representing Bondi-Hoyle and cold cloud accretion.
00190     * The three should be identical and the use of empirical avoids people
00191     * shouting about dutty-cycles beying incoisistent with Bondi-Holy & etc.
00192     */
00193
00194     if(Gal[centralgal].HotGas > 0.0)
00195     {
00196
00197         if(AGNRecipeOn == 2)
00198         {
00199             /* Bondi-Hoyle accretion recipe -- efficiency = 0.15
00200             * Eq. 29 in Croton 2006 */
00201             AGNrate = (2.5 * M_PI * G) * (0.75 * 0.6 * x) * Gal[centralgal].
BlackHoleMass * 0.15;
00202         }
00203         else if(AGNRecipeOn == 3)
00204         {
00205             /* Cold cloud accretion recipe -- trigger: Rff = 50 Rdisk,
* and accretion rate = 0.01% cooling rate

```

```

00207             * Eq. 25 in Croton 2006 */
00208             FreeFallRadius =
00209                 Gal[centralgal].HotGas / (6.0 * 0.6 * x * Gal[centralgal].Rvir * Gal[centralgal].Vvir) / Gal[centralgal].HotRadius * Gal[centralgal].Rvir;
00210             if(Gal[centralgal].BlackHoleMass > 0.0 && FreeFallRadius < Gal[centralgal].GasDiskRadius * 50.0)
00211                 AGNrate = 0.0001 * coolingGas / dt;
00212             else
00213                 AGNrate = 0.0;
00214         }
00215     else
00216     {
00217         /* empirical (standard) accretion recipe - Eq. 10 in Croton 2006 */
00218         AGNrate =
00219             AgnEfficiency / (UnitMass_in_g / UnitTime_in_s * SEC_PER_YEAR / SOLAR_MASS)
00220             * (Gal[centralgal].BlackHoleMass / 0.01) * pow(Gal[centralgal].Vvir / 200.0, 3.0)
00221             * ((Gal[centralgal].HotGas / Gal[centralgal].HotRadius * Gal[centralgal].Rvir / Gal[centralgal].Mvir) / 0.1);
00222     }
00223
00224
00225     /* Eddington rate */
00226     EDDrate = 1.3e48 * Gal[centralgal].BlackHoleMass / (UnitEnergy_in_cgs / UnitTime_in_s) / 9e10;
00227
00228     /* accretion onto BH is always limited by the Eddington rate */
00229     if(AGNrate > EDDrate)
00230         AGNrate = EDDrate;
00231
00232     /* accreted mass onto black hole */
00233     AGNaccreted = AGNrate * dt;
00234
00235     /* cannot accrete more mass than is available! */
00236     if(AGNaccreted > Gal[centralgal].HotGas)
00237         AGNaccreted = Gal[centralgal].HotGas;
00238
00239
00240     /* coefficient to heat the cooling gas back to the virial temperature of the halo */
00241     /* 1.34e5 = sqrt(2*eta*c^2), eta=0.1 (standard efficiency) and c in km/s */
00242     /* Eqs. 11 & 12 in Croton 2006 */
00243     AGNcoeff = (1.34e5 / Gal[centralgal].Vvir) * (1.34e5 / Gal[centralgal].Vvir);
00244
00245     /* cooling mass that can be suppressed from AGN heating */
00246     AGNheating = AGNcoeff * AGNaccreted;
00247
00248
00249     /* limit heating to cooling rate */
00250     if(AGNheating > coolingGas)
00251     {
00252         AGNaccreted = coolingGas / AGNcoeff;
00253         AGNheating = coolingGas;
00254     }
00255
00256
00257     /* accreted mass onto black hole */
00258     metallicity = get_metallicity(Gal[centralgal].HotGas, Gal[centralgal].MetalsHotGas);
00259     Gal[centralgal].BlackHoleMass += AGNaccreted;
00260     Gal[centralgal].HotGas -= AGNaccreted;
00261     Gal[centralgal].MetalsHotGas -= metallicity * AGNaccreted;
00262
00263 }
00264

```

```

00265     else
00266         AGNheating = 0.0;
00267
00268
00269     return AGNheating;
00270
00271 }
00272
00273
00274 /** @brief updates the fractions of hot and cold gas due to cooling. */
00275
00276 void cool_gas_onto_galaxy(int p, double mcool)
00277 {
00278     double metallicity,Mdisk,tmp;
00279     int i;
00280
00281     /** @brief cool_gas_onto_galaxy updates the fractions of hot and cold gas
00282      * due to cooling. This is done for the mass, metals and, after Guo2010,
00283      * spin components */
00284
00285     Mdisk=Gal[p].ColdGas;
00286     if (mcool>Gal[p].HotGas)
00287         mcool = Gal[p].HotGas;
00288
00289     /* add the fraction 1/STEPS of the total cooling gas to the cold disk */
00290     if(mcool > 0.0)
00291     {
00292
00293         tmp=-mcool;
00294
00295         if (Gal[p].Type == 1 && Gal[p].HotGas > 0.0)
00296             update_hot_frac(p,tmp, Gal[p].HotGas);
00297
00298         if(mcool < Gal[p].HotGas)
00299         {
00300             metallicity = get_metallicity(Gal[p].HotGas, Gal[p].MetalsHotGas);
00301             Gal[p].ColdGas += mcool;
00302             Gal[p].MetalsColdGas += metallicity * mcool;
00303             Gal[p].HotGas -= mcool;
00304             Gal[p].MetalsHotGas -= metallicity * mcool;
00305         }
00306     else
00307     {
00308         Gal[p].ColdGas += Gal[p].HotGas;
00309         Gal[p].MetalsColdGas += Gal[p].MetalsHotGas;
00310         Gal[p].HotGas = 0.0;
00311         Gal[p].MetalsHotGas = 0.0;
00312     }
00313
00314     if (DiskRadiusMethod == 2)
00315     {
00316         if (Gal[p].ColdGas != 0.0)
00317
00318             for (i=0;i<3;i++)
00319                 Gal[p].GasSpin[i]=(Gal[p].GasSpin[i]*Mdisk+Gal[p].HaloSpin[i]*mcool
00320                 )/(Gal[p].ColdGas);
00321
00322         get_gas_disk_radius(p);
00323     }
00324 }
00325
00326 }
00327

```

4.25 code/recipe_disrupt.c File Reference

[recipe_disrupt.c](#) checks if a type 2 satellite galaxy should or not be disrupted due to tidal forces

Functions

- void [disrupt](#) (int p, int centralgal, int tree)
- double [peri_radius](#) (int p, int centralgal)

Calculates the distance of the satellite to the pericentre of the main dark matter halo.

- double [sat_radius](#) (int p)

Calculates the half mass radius of satellite galaxies.

- double [diskmass_r](#) (double r, double rd, double Sigma0, double dr)

Returns the mass of a disk at a certain radius.

- double [bulgemass_r](#) (double r, double rb, double Mbulge, double dr)

Returns the mass of a bulge at a certain radius.

4.25.1 Detailed Description

This routine takes into account the tidal effects that satellite galaxies experience while orbiting a central companion. Since the baryonic component is more compact and denser than the dark matter it assumes that only type 2 satellites are affected (those that have already lost their dark matter halo). It assumes that the disruption is complete and instantaneous with all the satellite material being transferred into the central galaxy.

The satellite is assumed to orbit a singular isothermal potential:

$$\phi(R) = V_{\text{vir}}^2 \ln R \quad (\text{Eq. 28 Guo2010})$$

Assuming conservation of energy and angular momentum along the orbit, its pericentric distance (closest point to the centre along the orbit) can be estimated from:

$$\left(\frac{R}{R_{\text{peri}}}\right)^2 = \frac{\ln R / R_{\text{peri}} + \frac{1}{2}(V/V_{\text{vir}})^2}{\frac{1}{2}(V_t/V_{\text{vir}})^2} \quad (\text{Eq. 29 Guo2010}).$$

The main halo density at this point is compared with the baryonic mass (cold gas + stellar) density of the satellite within its half mass radius. If

$$\frac{M_{\text{DM,halo}}(R_{\text{peri}})}{R_{\text{peri}}^3} \equiv \rho_{\text{DM,halo}} > \rho_{\text{sat}} \equiv \frac{M_{\text{sat}}}{R_{\text{sat,half}}^3} \quad (\text{Eq. 30 Guo2010})$$

the galaxy is disrupted.

Definition in file [recipe_disrupt.c](#).

4.25.2 Function Documentation

4.25.2.1 double bulgemass_r (double r, double rb, double Mbulge, double dr)

Bulge profile -> de Vaucouleurs type $r^{1/4}$ law

Definition at line 270 of file [recipe_disrupt.c](#).

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

4.25.2.2 double diskmass_r (double *r*, double *rd*, double *Sigma0*, double *dr*)

Disk profile -> exponential

Definition at line 263 of file [recipe_disrupt.c](#).

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

4.25.2.3 void disrupt (int *p*, int *centralgal*, int *tree*)

Definition at line 50 of file [recipe_disrupt.c](#).

References [GALAXY::CentralGal](#), [GALAXY::ColdGas](#), [GALAXY::DisruptOn](#), [GALAXY::FirstProgGal](#), [Gal](#), [HaloGal](#), [GALAXY::HotGas](#), [GALAXY::ICLlum](#), [GALAXY::ICM](#), [GALAXY::Lum](#), [GALAXY::MetalsColdGas](#), [GALAXY::MetalsHotGas](#), [GALAXY::MetalsICM](#), [GALAXY::MetalsStellarMass](#), [GALAXY::Mvir](#), [GALAXY::NextProgGal](#), [GALAXY::ObsICL](#), [GALAXY::ObsLum](#), [peri_radius\(\)](#), [GALAXY::Rvir](#), [sat_radius\(\)](#), [GALAXY::StellarMass](#), and [GALAXY::Type](#).

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

4.25.2.4 double peri_radius (int *p*, int *centralgal*)

Definition at line 169 of file [recipe_disrupt.c](#).

References [Gal](#), [Halo](#), [GALAXY::HaloNr](#), [GALAXY::Pos](#), [GALAXY::SnapNum](#), [GALAXY::Vel](#), [GALAXY::Vvir](#), and [ZZ](#).

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

4.25.2.5 double sat_radius (int *p*)

Definition at line 199 of file [recipe_disrupt.c](#).

References [GALAXY::BulgeMass](#), [bulgemass_r\(\)](#), [GALAXY::BulgeSize](#), [GALAXY::ColdGas](#), [diskmass_r\(\)](#), [Gal](#), [GALAXY::GasDiskRadius](#), [GALAXY::StellarDiskRadius](#), and [GALAXY::StellarMass](#).

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

4.26 code/recipe_disrupt.c

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <string.h>
00004 #include <math.h>
00005 #include <time.h>
00006
00007 #include "allvars.h"
00008 #include "proto.h"
00009
00010 /** @file recipe_disrupt.c
00011 * @brief recipe_disrupt.c checks if a type 2 satellite galaxy should
00012 * or not be disrupted due to tidal forces
00013 *
00014 * This routine takes into account the tidal effects that satellite
00015 * galaxies experience while orbiting a central companion. Since the

```

```

00016 * baryonic component is more compact and denser than the dark matter
00017 * it assumes that only type 2 satellites are affected (those that have
00018 * already lost their dark matter halo). It assumes that the disruption
00019 * is complete and instantaneous with all the satellite material being
00020 * transferred into the central galaxy.
00021 *
00022 * The satellite is assumed to orbit a singular isothermal potential:
00023 *
00024 *  $\phi(R) = V^2 \ln(R)$  (Eq. 28 Guo2010)
00025 *
00026 * Assuming conservation of energy and angular momentum along the
00027 * orbit, its pericentric distance (closest point to the centre along
00028 * the orbit) can be estimated from:
00029 *
00030 * 
$$\left( \frac{R_{\text{peri}}}{R} \right)^2 = \frac{\ln R / R_{\text{peri}} + \frac{1}{2} (V / V_{\text{vir}})^2}{\left( \frac{1}{2} (V_{\text{t}} / V_{\text{vir}})^2 \right)}$$

00031 * (Eq. 29 Guo2010).
00032 *
00033 *
00034 *
00035 * The main halo density at this point is compared with the baryonic
00036 * mass (cold gas + stellar) density of the satellite within its half
00037 * mass radius. If
00038 *
00039 * 
$$\frac{M_{\text{DM, halo}} / R_{\text{peri}}}{\rho_{\text{halo}}} \gtrsim \frac{M_{\text{sat}} / R_{\text{sat, half}}}{\rho_{\text{sat}}}$$

00040 * (Eq. 30 Guo2010)
00041 *
00042 *
00043 *
00044 * the galaxy is disrupted.
00045 *
00046 *
00047 * */
00048
00049
00050 void disrupt(int p, int centralgal,int tree)
00051 {
00052     double rho_sat, rho_cen;
00053     double cen_mass,r_sat,radius,alpha;
00054     struct GALAXY *g;
00055     int outputbin,j;
00056     int q;
00057
00058     /* If the main halo density at the pericentre (closest point in the orbit
00059     * to the central galaxy) is larger than the satellite's density at the
00060     * half mass radius, the satellite is completely disrupted. Note that since
00061     * the satellite is a type 2 the only mass components remaining and
00062     * contributing to the density are the cold gas and stellar mass. */
00063
00064     centralgal=Gal[p].CentralGal;
00065     //TO DO - PARAMETER = 1, not used and shouldn't exist
00066     alpha=1.;
00067
00068
00069     /* Radius calculated at the peri-point */
00070     radius=peri_radius(p, centralgal);
00071
00072     if (radius < 0)
00073     {
00074         printf("must be wrong \n");
00075         exit(0);
00076     }
00077
00078     /* Calculate the density of the main central halo at radius (the peri-centre).
00079     * The tidal forces are caused by the dark matter of the main halo, hence Mvir
00080     * is used. */
00081     cen_mass=Gal[centralgal].Mvir*radius/Gal[centralgal].Rvir;
00082     rho_cen=cen_mass/pow(radius,3);

```

```

00083 /* Calculate the density of the satellite's baryonic material */
00084 if (Gal[p].StellarMass>0)
00085 {
00086     /* Calculate the rho according to the real geometry */
00087     r_sat = sat_radius(p);
00088     /* Or use radius at the mean radius of the stellar mass */
00089     /* r_sat=(Gal[p].BulgeMass*Gal[p].BulgeSize+(Gal[p].StellarMass-Gal[p].Bulg
00090 emass)*
00091         *Gal[p].StellarDiskRadius/3*1.68)
00092         /(Gal[p].StellarMass); */
00093
00094     /* to calculate the density */
00095     //rho_sat=Gal[p].StellarMass/pow(r_sat,2);
00096     rho_sat=(Gal[p].StellarMass+Gal[p].ColdGas)/pow(r_sat,3);
00097 }
00098 else
00099     rho_sat=0.0;
00100
00101 /* If density of the main halo is larger than that of the satellite baryonic
00102 * component, complete and instantaneous disruption is assumed. Galaxy becomes
00103 * a type 3 and all its material is transferred to the central galaxy. */
00104 if (rho_cen > alpha * rho_sat)
00105 {
00106     Gal[p].Type = 3;
00107 #ifdef GALAXYTREE
00108     // TODO why only for GALAXYTREE?
00109     // Isn't this physics, independent of what one stores at the end?
00110
00111     q = Gal[Gal[p].CentralGal].FirstProgGal;
00112     if (q >= 0) {
00113         // add progenitors of Gal[p] to the list of progenitors of Gal[p].Centra
00114         lGal
00115             while (HaloGal[q].NextProgGal >= 0)
00116                 q = HaloGal[q].NextProgGal;
00117
00118             HaloGal[q].NextProgGal = Gal[p].FirstProgGal;
00119         } // TODO if !(q>=0) shouldn't we set
00120         // Gal[Gal[p].CentralGal].FirstProgGal to Gal[p].FirstProgGal ???
00121
00122     q = HaloGal[q].NextProgGal;
00123     HaloGal[q].DisruptOn = 1;
00124 #endif
00125     /* Put gas component to the central galaxy and stellar material
00126     * into the ICM */
00127     Gal[centralgal].HotGas += Gal[p].HotGas; // just a check - type 2's shouldn
00128     't have this
00129     Gal[centralgal].MetalsHotGas += Gal[p].MetalsHotGas;
00130     Gal[centralgal].HotGas += Gal[p].ColdGas; //cold gas assumed to be reheated
00131     in the process?
00132     Gal[centralgal].MetalsHotGas += Gal[p].MetalsColdGas;
00133     Gal[centralgal].ICM += Gal[p].StellarMass;
00134     Gal[centralgal].MetalsICM += Gal[p].MetalsStellarMass;
00135     //Gal[centralgal].HotICM +=Gal[p].HotGas;
00136     //Gal[centralgal].MetalsHotICM +=Gal[p].MetalsHotGas;
00137
00138     if (Gal[p].HotGas > 1.e-8)
00139     {
00140         printf("wrongla \n");
00141         exit(0);
00142     }
00143 /* Add satellite's luminosity into the luminosity of the ICL
00144     * component of the central galaxy. */
00145 #ifdef ICL
00146     for(outputbin = 0; outputbin < NOUT; outputbin++)
00147     {

```

```

00146         for(j = 0; j < NMAG; j++)
00147             {
00148
00149 #ifdef OUTPUT_REST_MAGS
00150             Gal[centralgal].ICLLum[j][outputbin] += Gal[p].Lum[j][outputbin];
00151 #endif
00152
00153 #ifdef COMPUTE_OBS_MAGS
00154             Gal[centralgal].ObsICL[j][outputbin] += Gal[p].ObsLum[j][outputbin]
00155 ;
00156 #endif
00157         }
00158     }
00159 #endif
00160
00161     }
00162
00163
00164 }
00165
00166 /** @brief Calculates the distance of the satellite to the pericentre of the
00167 * main dark matter halo. */
00168
00169 double peri_radius(int p, int centralgal)
00170 {
00171     int i;
00172     double a, b,v[3],r[3],x,x0;
00173     for (i=0;i<3;i++)
00174     {
00175         r[i]=(Gal[p].Pos[i]-Gal[centralgal].Pos[i])/(1 + ZZ[Halo[Gal[centralgal].HaloNr].SnapNum]); /*sqrt(Gal[p].MergTime/Gal[p].OriMergTime);
00176         v[i]=Gal[p].Vel[i]-Gal[centralgal].Vel[i];
00177     }
00178     b=1/2.* (v[0]*v[0]+v[1]*v[1]+v[2]*v[2])/pow(Gal[centralgal].Vvir,2);
00179     a=1/2.* (v[0]*v[0]+v[1]*v[1]+v[2]*v[2]-pow((r[0]*v[0]+r[1]*v[1]+r[2]*v[2]),2)/(r
00180 [0]*r[0]+r[1]*r[1]+r[2]*r[2]))/pow(Gal[centralgal].Vvir,2);
00181     x=sqrt(b/a);
00182     x0=1000;
00183     while (abs(x0-x)>=1.e-8)
00184     {
00185         x0=x;
00186         x=sqrt((log(x0)+b)/a);
00187     }
00188     if (x == 0)
00189     {
00190         printf("wrong in peri_radius \n");
00191         exit(0);
00192     }
00193
00194     return sqrt(r[0]*r[0]+r[1]*r[1]+r[2]*r[2])/x;
00195 }
00196
00197
00198 /** @brief Calculates the half mass radius of satellite galaxies */
00199 double sat_radius(int p)
00200 {
00201     double r,rd,rb,Mdisk,Sigma0,Ie,rmin,rmax,rbin,M, tmprmin,tmprmax;
00202     double Mgas, Mbulge,rgd,Sigma0_g,rmi,rma, dr,totmass;
00203     int N=50,i;
00204     //disk profile, rd is known
00205     rd=Gal[p].StellarDiskRadius/3.;
00206     rb=Gal[p].BulgeSize;
00207     Mdisk=Gal[p].StellarMass-Gal[p].BulgeMass;
00208     rgd = Gal[p].GasDiskRadius/3.;
00209     Mgas = Gal[p].ColdGas;

```

```

00210     Mbudge = Gal[p].BulgeMass;
00211     totmass = Gal[p].StellarMass + Gal[p].ColdGas;
00212
00213     //defining a maximum search radius for half mass?
00214     tmpmax=(rd*1.68 > rb)? rd*1.68:rb;
00215     rmax = (tmpmax > rgd * 1.68)? tmpmax : rgd * 1.68;
00216
00217     i = 0;
00218     rmin = 1.e-6;
00219     if (rmax == 0 || rmax < rmin)
00220         return(rmax);
00221     rbin=(rmax-rmin)/N;
00222
00223     if (rd == 0 && rgd == 0)
00224         return(rb);
00225     if (rd ==0 && rb == 0)
00226         return(rgd*1.68);
00227     if (rb==0 && rgd==0)
00228         return(rd*1.68);
00229
00230     /* Disk surface densities, used to calculate the total mass inside
00231      * a certain radius. */
00232     Sigma0 = Mdisk / (2 * M_PI * rd *rd);
00233     Sigma0_g = Mgas / (2 * M_PI *rgd * rgd);
00234     if (rgd == 0)
00235     {
00236         Sigma0_g =0;
00237         rgd = 1.;
00238     }
00239     if (rd == 0)
00240     {
00241         Sigma0 = 0;
00242         rd = 1. ;
00243     }
00244
00245     /* increases the search radius until it encompasses half the total mass taking
00246      * into account the stellar disk, stellar bulge and cold gas disk. */
00247     M = 0.0;
00248     do
00249     {
00250         rmi = (rmin) + i* rbin;
00251         rma = (rmin) + rbin * (i+1);
00252         dr = rma - rmi;
00253         r = (rma + rmi) /2;
00254         M = M + diskmass_r(r,rd,Sigma0,dr)+bulgemass_r(r,rb,Mbudge,dr)+diskmass_r(r
00255             ,rgd,Sigma0_g,dr);
00256         i++;
00257     }
00258     while(M < 0.5*totmass);
00259     return (r);
00260
00261 /** @brief Returns the mass of a disk at a certain radius.
00262      *          Disk profile -> exponential */
00263 double diskmass_r(double r, double rd, double Sigma0,double dr)
00264 {
00265     return 2 * M_PI * Sigma0 * dr * r * exp(-r / rd);
00266 }
00267
00268 /** @brief Returns the mass of a bulge at a certain radius.
00269      *          Bulge profile -> de Vaucouleurs type  $r^{1/4}$  law */
00270 double bulgemass_r(double r, double rb, double Mbudge,double dr)
00271 {
00272     return Mbudge / (4 * M_PI * pow(rb,3)) * 1 / pow(r / rb, 2) * 1 / pow(1 + r/rb
00273         , 2) * 4 * M_PI * r * r *dr,
00274 }
```

4.27 code/recipe_dust.c File Reference

[recipe_dust.c](#) is used to compute dust extinction as described in Delucia2007 + redshift dependence as Kitzbichler & White 2007.

Functions

- void [tsudy](#) (int p, int magbin, int halonr)
main routine where the extinction is calculated
- float [gasdev](#) (long *idum)
computes a gaussian random deviate to calculate a random inclination for extinction.
- float [ran1](#) (long *idum)

4.27.1 Detailed Description

There are 2 extinction sources: Extinction from a diffuse inter-stellar medium (ISM) (Devriendt1999); Extinction from molecular clouds in young stars (YS) (Charlot2000); Both were introduced in Delucia2007.

The optical depth of dust in each component τ_λ^z (ISM) and τ_λ^{BC} (YS) is used to compute extinction assuming a slab geometry for the dust and a random inclination of the disk to the line of sight.

Extinction curves for the ISM: $\left(\frac{A_\lambda}{A_v}\right)_{Z_\odot} \left(\frac{Z_{\text{gas}}}{Z_\odot}\right)^s$ are read in from lookup tables into DeltaM[number of magnitudes+1][gas metallicity][redshift] at [init.c](#) (read_tsud_tables) from file (_Ext_table.dat) - one extra column in the end that corresponds to V-band extinction used for the YS component.

The optical depth for the ISM at a given λ can be written as:

$$\tau_\lambda^{ISM} = \left(\frac{A_\lambda}{A_v}\right)_{Z_\odot} \left(\frac{Z_{\text{gas}}}{Z_\odot}\right)^s \left(\frac{\langle N_H \rangle}{2.1 \times 10^{21} \text{ atoms cm}^{-2}}\right),$$

where the mean column density of Hydrogen is:

$$\langle N_H \rangle = \frac{M_{\text{cold}}}{1.4 m_p \pi (a R_D)^2} \text{ atoms cm}^{-2}.$$

The optical depth for YS (τ_λ^{BC}) is calibrated from the ISM optical depth in the V-band:

$$\tau_\lambda^{BC} = \tau_v^{ISM} \left(\frac{1}{\mu} - 1\right) \left(\frac{\lambda}{5500}\right)^{-0.7},$$

where $\left(\frac{\lambda}{5500}\right)^{-0.7}$ is read into CORRECTIONS[number of magnitudes][redshift] from file (_YSExt_table.dat)

Definition in file [recipe_dust.c](#).

4.27.2 Function Documentation

4.27.2.1 float [gasdev](#) (long * *idum*)

Definition at line 207 of file [recipe_dust.c](#).

References [ran1\(\)](#).

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

4.27.2.2 float ran1(long * idum)

Definition at line 230 of file [recipe_dust.c](#).

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

4.27.2.3 void tsudy(int p, int magbin, int halonr)

Definition at line 57 of file [recipe_dust.c](#).

References [GALAXY::ColdGas](#), [Corrections](#), [DeltaM](#), [GALAXY::dObsLum](#), [GALAXY::dObsLumBulge](#), [GALAXY::dObsLumDust](#), [GALAXY::dObsYLum](#), [GALAXY::dObsYLumBulge](#), [Gal](#), [gasdev\(\)](#), [GALAXY::GasDiskRadius](#), [Halo](#), [GALAXY::Inclination](#), [ListOutputSnaps](#), [GALAXY::Lum](#), [GALAXY::LumBulge](#), [GALAXY::LumDust](#), [GALAXY::MetalsColdGas](#), [mu_seed](#), [GALAXY::ObsLum](#), [GALAXY::ObsLumBulge](#), [GALAXY::ObsLumDust](#), [GALAXY::ObsYLum](#), [GALAXY::ObsYLumBulge](#), [GALAXY::SnapNum](#), [GALAXY::YLum](#), [GALAXY::YLumBulge](#), and [ZZ](#).

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

4.28 code/recipe_dust.c

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <string.h>
00004 #include <math.h>
00005 #include <time.h>
00006
00007 #include "allvars.h"
00008 #include "proto.h"
00009
00010 /** @file recipe_dust.c
00011 * @brief recipe_dust.c is used to compute dust extinction as described
00012 * in Delucia2007 + redshift dependence as Kitzbichler & White 2007.
00013
00014 * There are 2 extinction sources:
00015 * Extinction from a diffuse inter-stellar medium (ISM) (Devriendt1999);
00016 * Extinction from molecular clouds in young stars (YS) (Charlot2000);
00017 * Both were introduced in Delucia2007.
00018
00019 * The optical depth of dust in each component
00020 *  $\tau \propto z^{-\lambda}$  (ISM)
00021 * and  $\tau \propto z^{-\lambda} \times \rm{BC}$  (YS)
00022 * is used to compute extinction assuming a slab geometry for the dust and
00023 * a random inclination of the disk to the line of sight.
00024
00025 * Extinction curves for the ISM:
00026 *  $\tau = \left( \frac{A_{\lambda}}{A_v} \right) \left( \frac{N_H}{Z_{\rm gas}} \right)^{-1}$ 
00027 * are read in from lookup tables into
00028 * DeltaM[number of magnitudes+1][gas metallicity][redshift] at init.c
00029 * (read_tsud_tables) from file (_Ext_table.dat) - one extra column in the
00030 * end that corresponds to V_band extinction used for the YS component.
00031 *
00032 *
00033 * The optical depth for the ISM at a given  $\lambda$  can be written as:
00034
00035 
$$\tau(\lambda) = \left( \frac{A_{\lambda}}{A_v} \right) \left( \frac{N_H}{Z_{\rm gas}} \right)^{-1} \times 2.1 \times 10^{21} \times N_{\rm atoms} \times 10^{-2}$$

00036 where the mean column density of Hydrogen is:
00037
00038
00039
00040

```

```

00041      \f$\langle N_{\mathrm{H}} \rangle = \frac{M_{\mathrm{\{cold\}}}}{1.4} \cdot m_p \cdot \pi
00042      (\mathrm{a} \cdot R_{\mathrm{\{mathrm{D}\}}})^2 \cdot \mathrm{atoms} \cdot \mathrm{cm}^{-2}.\f$
00043
00044
00045 * The optical depth for YS ( $\tau_{\mathrm{BC}}$ ) is calibrated
00046 * from the ISM optical depth in the V-band:
00047 *
00048 *  $\tau_{\mathrm{BC}} = \tau_{\mathrm{ISM}} \left( \frac{1}{\mu - 1} \right) \left( \frac{5500 \AA}{\lambda} \right)^{-0.7}$ 
00049 *
00050 *
00051 * where  $\tau_{\mathrm{ISM}}$  is read
00052 * into CORRECTIONS[number of magnitudes][redshift] from file
00053 * (_YSExt_table.dat)
00054 */
00055
00056 /** @brief main routine where the extinction is calculated */
00057 void tsudy(int p, int magbin,int halonr)
00058 {
00059     double nh, Zg, tau, alam, sec, Lum_disk,cosinc;
00060     double tauv,taubc,tauvbc,mu,dly;
00061     int izz,k,iz;
00062     float gasdev(long *idum);
00063
00064 /* Log (Cold Gas metallicity) in solar units (/0.02) is interpolated into
00065 * the grid of the ISM extinction curves lookup table. lzzmin=-3, lzzmax= 1,
00066 * NZZ_EXT=100 */
00067 if(Gal[p].ColdGas > 0.0)
00068 {
00069     Zg = log10(Gal[p].MetalsColdGas/Gal[p].ColdGas/0.02);
00070     if (Zg>lzzmin)
00071     {
00072         izz = (int)((Zg-lzzmin)/(lzzmax-lzzmin)* NZZ_EXT);
00073     }
00074     else
00075     {
00076         izz = 0;
00077     }
00078     if (izz >= NZZ_EXT)
00079     {
00080         izz = NZZ_EXT-1;
00081     }
00082
00083 /* 0.94 = 2.83/3. - 3 to get scale lenght and 2.83 = 1.68^2 */
00084 nh = Gal[p].ColdGas / (M_PI * pow(Gal[p].GasDiskRadius*0.94,2)*1.4);
00085 /* now convert from  $10^{10} \mathrm{M}_{\odot}/\mathrm{h}$  /  $(\mathrm{Mpc}/\mathrm{h})^2$  to  $(2.1 \cdot 10^{21} \mathrm{atoms/cm}^2)$  */
00086 nh = nh / 3252.37; // 3252.37 =  $10^{(3.5122)}$  ... ha ha !
00087 /*redshift dependence*/
00088 nh = nh*pow(1+ZZ[halonr].SnapNum, -0.4);
00089
00090
00091     cosinc = cos(Gal[p].Inclination);
00092     // TODO in recipe_misc we avoid having cosinc too small, why here again (or
00093     // why there)?
00094     if (cosinc < 0.1) cosinc = 0.1; // minimum inclination
00095     sec = 1.0/cosinc;
00096
00097     /* mu for YS extinction, given by a Gaussian with centre 0.3 (MUCENTER)
00098     * and width 0.2 (MUWIDTH), truncated at 0.1 and 1. */
00099     mu = -1.;
00100     while (mu < 0)
00101     {
00102         mu = gasdev(&mu_seed) * MUWIDTH + MUCENTER;
00103         if (mu <0.1 || mu>1.0) mu = -1.;
00104     }
00105
00106 #ifdef OUTPUT_REST_MAGS

```

```

00107     iz = 0; //index of redshift - 0 for rest frame
00108     tauv = DeltaM[NMAG][izz][iz] * nh; // rest-frame V-band -> used to calibrat
e YS extinction
00109     for (k=0;k<NMAG;k++)
00110     {
00111         tau = DeltaM[k][izz][iz] * nh ; // tau for all bands
00112         if (tau > 0.0)
00113         {
00114             tau = tau * sec;
00115             alam = (1.0 - exp(-tau))/tau;
00116             Lum_disk = Gal[p].Lum[k][magbin] - Gal[p].
LumBulge[k][magbin];
00117             Gal[p].LumDust[k][magbin] = Gal[p].LumBulge[k][magbin] + Lum_disk *
alam;
00118             // now remove light from young stars absorbed by birth clouds
00119             tauvbc = tauv * (1./mu - 1.);
00120             taubc = tauvbc * Corrections[k][0];
00121             dly = (Gal[p].YLum[k][magbin] - Gal[p].YLumBulge[k][magbin]) *
alam * (1. - exp(-taubc)) + Gal[p].YLumBulge[k][magbin] * (1. - ExpTauBCBulge);
00122             Gal[p].LumDust[k][magbin] -= dly;
00123         }
00124     else
00125     {
00126         Gal[p].LumDust[k][magbin] = Gal[p].Lum[k][magbin];
00127         // now remove light from young stars absorbed by birth clouds
00128         tauvbc = tauv * (1./mu - 1.);
00129         taubc = tauvbc * Corrections[k][0];
00130         dly = (Gal[p].YLum[k][magbin] - Gal[p].YLumBulge[k][magbin]) *
(1. - exp(-taubc)) + Gal[p].YLumBulge[k][magbin] * (1. - ExpTauBCBulge);
00131         Gal[p].LumDust[k][magbin] -= dly;
00132     }
00133 }
00134 #endif
00135 #ifdef OUTPUT_OBS_MAGS
00136     iz = 63 - ListOutputSamps[magbin]; // index of redshift
00137     tauv = DeltaM[NMAG][izz][0] * nh; // rest-frame V-band tau -> used to calibrat
e YS extinction
00138     for (k=0;k<NMAG;k++)
00139     {
00140         tau = DeltaM[k][izz][iz] * nh ;
00141         if (tau > 0.0)
00142         {
00143             tau = tau * sec;
00144             alam = (1.0 - exp(-tau))/tau;
00145             Lum_disk = Gal[p].ObsLum[k][magbin] - Gal[p].
ObsLumBulge[k][magbin];
00146             Gal[p].ObsLumDust[k][magbin] = Gal[p].ObsLumBulge[k][magbin] + Lum_
disk * alam;
00147             // now remove light from young stars absorbed by birth clouds
00148             tauvbc = tauv * (1./mu - 1.);
00149             taubc = tauvbc * Corrections[k][iz];
00150             dly = (Gal[p].ObsYLum[k][magbin] - Gal[p].ObsYLumBulge[k][magbi
n]) * alam * (1. - exp(-taubc)) + \
Gal[p].ObsYLumBulge[k][magbin] * (1. - ExpTauBCBulge);
00151             Gal[p].ObsLumDust[k][magbin] -= dly;
00152     }
00153 }
00154     else
00155     {
00156         Gal[p].ObsLumDust[k][magbin] = Gal[p].ObsLum[k][magbin];
00157         // now remove light from young stars absorbed by birth clouds
00158         tauvbc = tauv * (1./mu - 1.);
00159         taubc = tauvbc * Corrections[k][iz];
00160         dly = (Gal[p].ObsYLum[k][magbin] - Gal[p].ObsYLumBulge[k][magbi
n]) * (1. - exp(-taubc)) + \
Gal[p].ObsYLumBulge[k][magbin] * (1. - ExpTauBCBulge);
00161         Gal[p].ObsLumDust[k][magbin] -= dly;
00162     }
00163 }
```

```

00164
00165 #ifdef OUTPUT_MOMAF_INPUTS
00166     // compute same thing at iz + 1
00167     tau = DeltaM[k][izz][iz+1] * nh ;
00168     if (tau > 0.0)
00169     {
00170         tau             = tau * sec;
00171         alam           = (1.0 - exp(-tau))/tau;
00172         Lum_disk        = Gal[p].dObsLum[k][magbin] - Gal[p].
00173             dObsLumBulge[k][magbin];
00174         Gal[p].dObsLumDust[k][magbin] = Gal[p].dObsLumBulge[k][magbin] + Lu
00175             m_disk * alam;
00176         // now remove light from young stars absorbed by birth clouds
00177         taubc   = tauvbc * Corrections[k][iz+1];
00178         dly     = (Gal[p].dObsYLum[k][magbin] - Gal[p].dObsYLumBulge[k][mag
00179             bin]) * alam * (1. - exp(-taubc)) + \
00180             Gal[p].dObsYLumBulge[k][magbin] * (1. - ExpTauBCBulge);
00181         Gal[p].dObsLumDust[k][magbin] -= dly;
00182     }
00183     else
00184     {
00185         Gal[p].dObsLumDust[k][magbin] = Gal[p].dObsLum[k][magbin];
00186         // now remove light from young stars absorbed by birth clouds
00187         taubc   = tauvbc * Corrections[k][iz+1];
00188         dly     = (Gal[p].dObsYLum[k][magbin] - Gal[p].dObsYLumBulge[k][mag
00189             bin]) * (1. - exp(-taubc)) + \
00190             Gal[p].dObsYLumBulge[k][magbin] * (1. - ExpTauBCBulge);
00191     }
00192 }
00193 }
00194
00195 #define IA 16807
00196 #define IM 2147483647
00197 #define AM (1.0/IM)
00198 #define IQ 127773
00199 #define IR 2836
00200 #define NTAB 32
00201 #define NDIV (1+(IM-1)/NTAB)
00202 #define EPS 1.2e-7
00203 #define RNMX (1.0-EPS)
00204
00205 /** @brief computes a gaussian random deviate to calculate a random
00206 *      inclination for extinction. */
00207 float gasdev(long *idum)
00208 {
00209     float ran1(long *idum);
00210     static int iset=0;
00211     static float gset;
00212     float fac,rsq,v1,v2;
00213
00214     if (iset == 0) {
00215         do {
00216             v1=2.0*ran1(idum)-1.0;
00217             v2=2.0*ran1(idum)-1.0;
00218             rsq=v1*v1+v2*v2;
00219         } while (rsq >= 1.0 || rsq == 0.0);
00220         fac=sqrt(-2.0*log(rsq)/rsq);
00221         gset=v1*fac;
00222         iset=1;
00223         return v2*fac;
00224     } else {
00225         iset=0;
00226         return gset;

```

```

00227         }
00228     }
00229
00230 float ran1(long *idum)
00231 {
00232     int j;
00233     long k;
00234     static long iy=0;
00235     static long iv[NTAB];
00236     float temp;
00237
00238     if (*idum <= 0 || !iy) {
00239         if (-(*idum) < 1) *idum=1;
00240         else *idum = -(*idum);
00241         for (j=NTAB+7; j>=0; j--) {
00242             k=(+idum)/IQ;
00243             *idum=IA*(*idum-k*IQ)-IR*k;
00244             if ((*idum < 0) *idum += IM;
00245             if (j < NTAB) iv[j] = *idum;
00246         }
00247         iy=iv[0];
00248     }
00249     k=(+idum)/IQ;
00250     *idum=IA*(*idum-k*IQ)-IR*k;
00251     if ((*idum < 0) *idum += IM;
00252     j=iy/NDIV;
00253     iy=iv[j];
00254     iv[j] = *idum;
00255     if ((temp=AM*iy) > RNMX) return RNMX;
00256     else return temp;
00257 }
00258 #undef IA
00259 #undef IM
00260 #undef AM
00261 #undef IQ
00262 #undef IR
00263 #undef NTAB
00264 #undef NDIV
00265 #undef EPS
00266 #undef RNMX
00267
00268

```

4.29 code/recipe_infall.c File Reference

`recipe_infall.c` calculates the amount of gas that infalls into the galaxy hot gas component at each time step.

Functions

- double `infall_recipe` (int centralgal, int ngal, double Zcurr)
- double `do_reionization` (int centralgal, double Zcurr)
- void `add_infall_to_hot` (int centralgal, double infallingGas)

The gas that infalled is added to the hot gas of the central galaxy.

- double `hot_retain_sat` (int i, int centralgal, double infallingMass)

Gradual stripping of hot gas (ejected sgas stripping is proportional) from type I satellites.

- void `update_hot_frac` (int p, double dmass, float HotGas)
- void `update_ICL` (int p, int q)

4.29.1 Detailed Description

This is derived from the baryonic fraction taking reionization into account.

this is where the gas components of newly accreted satellites are treated.

two things need to be done and are mixed in the code: calculate the total baryonic fraction to get infall mass; the different gas components of central galaxies are transferred into centrals. The baryonic fraction is only calculated taking into account material inside Rvir.

There are basically 3 options for the way satellite components are added into centrals, two for SatelliteRecipe=1 and one for =0. SatelliteRecipe deals with whether type 1 are considered centrals of type 2's. Processes only happen if galaxies are inside Rvir.

SatelliteRecipe ==1 & EjectionRecipe == 2 -> Type 1's keep an ejected component. Type 1's are striped of hot and ejected gas gradually and later in the code. A fraction of the hot and ejected gas in the type 2's is added to the type 1 and the rest to the type 0. If satellites are outside Rvir, type 1 keep all its components and receive everything from type 2's.

SatelliteRecipe ==1 & EjectionRecipe == 1 -> If satellites are inside Rvir of type 0, the ejected component of both type 1's and type 2's is added into the type 0 (Ej=1). Type 1's are striped of hot gas gradually and later in the code. A fraction of the hot gas in the type 2's is added to the type 1 and the rest to the type 0. If satellites are outside Rvir, type 1 keep all its components and receive everything from type 2's.

SatelliteRecipe ==0 ==> If inside Rvir, hot and ejected gas from satellites both 1 and 2 is instantaneously striped and added to type 0.

If SatelliteRecipe ==1, further down the gradual striping is applied to type 1's (both to hot and ejected gas - for the ejected gas it only makes sense if Ej=1).

done for the snapshot not for the timestep

TODO this recipe now has two different physics inside: infall and treatment of mergers, these shoudl be separated.

Definition in file [recipe_infall.c](#).

4.29.2 Function Documentation

4.29.2.1 void add_infall_to_hot (int centralgal, double infallingGas)

Definition at line [639](#) of file [recipe_infall.c](#).

References [Gal](#), [GALAXY::HotGas](#), and [GALAXY::MetalsHotGas](#).

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

4.29.2.2 double do_reionization (int centralgal, double Zcurr)

reionization recipie described in Gnedin (2000), using the fitting

Definition at line [599](#) of file [recipe_infall.c](#).

References [a0](#), [find_interpolate_reionization\(\)](#), [Gal](#), [GALAXY::Mvir](#), [Omega](#), and [Reion_Mc](#).

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

4.29.2.3 double hot_retain_sat (int *i*, int *centralgal*, double *infallingMass*)

This is caused both by tidal and ram-pressure stripping. This function return the actual mass of hot gas that the type 1 retains.

TIDAL STRIPPING Hot gas is tidally stripped at the same rate at which dark matter is striped:

$$\frac{M_{\text{hot}}(R_{\text{tidal}})}{M_{\text{hot,infall}}} = \frac{M_{\text{DM}}}{M_{\text{DM,infall}}}$$

Since the hot gas distribution is assumed to be $\rho \propto r^{-2}$ this means $M_{\text{hot}}(r) \propto r$. Therefore, the tidal radius beyond gas is stripped is given by:

$$R_{\text{tidal}} = \left(\frac{M_{\text{DM}}}{M_{\text{DM,infall}}} \right) R_{\text{DM,infall}}$$

RAM PRESSURE STRIPPING Let $R_{r.p.}$ represent the distance from the centre of the satellite at which ram pressure striping equals its self-gravity. Then:

$\rho_{\text{sat}}(R_{r.p.})V_{\text{sat}}^2 = \rho_{\text{par}}(R_{\text{orbit}})V_{\text{orbit}}^2$ Where the four terms represent respectively the density of the satellite at $R_{r.p.}$, the virial velocity of the satellite at infall, the density of the parent halo at the radius of the satellite and the orbit velocity of the satellite (given by V_c of the parent halo)

The striping radius is given by

$$R_{\text{strip}} = \min(R_{\text{tidal}}, R_{r.p.})$$

Definition at line 696 of file [recipe_infall.c](#).

References [Gal](#), [Halo](#), [GALAXY::HaloNr](#), [GALAXY::HotFrac](#), [GALAXY::HotGas](#), [GALAXY::HotRadius](#), [GALAXY::Len](#), [GALAXY::Mvir](#), [PartMass](#), [GALAXY::Pos](#), [GALAXY::Rvir](#), [halo_data::SnapNum](#), [GALAXY::Type](#), and [ZZ](#).

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

4.29.2.4 double infall_recipe (int *centralgal*, int *ngal*, double *Zcurr*)

Definition at line 60 of file [recipe_infall.c](#).

References [BaryonFrac](#), [GALAXY::BlackHoleMass](#), [GALAXY::CentralGal](#), [GALAXY::ColdGas](#), [do_reionization\(\)](#), [GALAXY::EjectedMass](#), [EjectionRecipe](#), [Gal](#), [Halo](#), [GALAXY::HaloNr](#), [hot_retain_sat\(\)](#), [GALAXY::HotFrac](#), [GALAXY::HotGas](#), [GALAXY::HotRadius](#), [GALAXY::ICM](#), [GALAXY::MetalsEjectedMass](#), [GALAXY::MetalsHotGas](#), [GALAXY::MetalsICM](#), [GALAXY::Mvir](#), [GALAXY::Pos](#), [ReionizationOn](#), [GALAXY::Rvir](#), [SatelliteRecipe](#), [halo_data::SnapNum](#), [GALAXY::StellarMass](#), [GALAXY::Type](#), [update_hot_frac\(\)](#), [update_ICL\(\)](#), and [ZZ](#).

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

4.29.2.5 void update_hot_frac (int *p*, double *dmass*, float *HotGas*)

Definition at line 754 of file [recipe_infall.c](#).

References [Gal](#), [GALAXY::HotFrac](#), [GALAXY::HotRadius](#), [GALAXY::Len](#), [GALAXY::Mvir](#), [PartMass](#), and [GALAXY::Rvir](#).

Referenced by [cool_gas_onto_galaxy\(\)](#), [infall_recipe\(\)](#), [reincorporate_gas\(\)](#), and [update_from_feedback\(\)](#).

4.29.2.6 void update_ICL (int *p*, int *q*)

Definition at line 779 of file [recipe_infall.c](#).

References [Gal](#), [GALAXY::ICLLum](#), and [GALAXY::ObsICL](#).

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

4.30 code/recipe_infall.c

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <string.h>
00004 #include <math.h>
00005 #include <time.h>
00006
00007 #include "allvars.h"
00008 #include "proto.h"
00009
00010
00011 /**@file recipe_infall.c
00012 * @brief recipe_infall.c calculates the amount of gas that infalls
00013 * into the galaxy hot gas component at each time step. This
00014 * is derived from the baryonic fraction taking reionization
00015 * into account.
00016 *
00017 * this is where the gas components of newly accreted satellites
00018 * are treated.
00019 *
00020 * two things need to be done and are mixed in the code: calculate
00021 * the total baryonic fraction to get infall mass; the different
00022 * gas components of central galaxies are transferred into centrals.
00023 * The baryonic fraction is only calculated taking into account
00024 * material inside Rvir.
00025 *
00026 * There are basically 3 options for the way satellite components are
00027 * added into centrals, two for SatelliteRecipe=1 and one for =0. Satellite
00028 * Recipe
00029 * deals with whether type 1 are considered centrals of type 2's.
00030 * Processes only happen if galaxies are inside Rvir.
00031 * SatelliteRecipe ==1 & EjectionRecipe == 2 -> Type 1's keep an ejected
00032 * component.
00033 * Type 1's are striped of hot and ejected gas gradually and later in the
00034 * code.
00035 * A fraction of the hot and ejected gas in the type 2's is
00036 * added to the type 1 and the rest to the type 0. If satellites are outs
00037 * ide
00038 * Rvir, type 1 keep all its components and receive everything from type
00039 * 2's.
00040 * SatelliteRecipe ==1 & EjectionRecipe == 1 -> If satellites are inside
00041 * Rvir of type 0, the ejected component of both type 1's and type 2's is
00042 * added into the type 0 (Ej=1). Type 1's are striped of hot gas graduall
00043 * y
00044 * and later in the code. A fraction of the hot gas in the type 2's is
00045 * added to the type 1 and the rest to the type 0. If satellites are outs
00046 * ide
00047 * Rvir, type 1 keep all its components and receive everything from type
00048 * 2's.
00049 * SatelliteRecipe ==0 ==> If inside Rvir, hot and ejected gas from satel
00050 * lites
00051 * both 1 and 2 is instantaneously striped and added to type 0.
00052 * If SatelliteRecipe ==1, further down the gradual striping is applied t
00053 * o

```

```

00049 *           type 1's (both to hot and ejected gas - for the ejected gas it only ma
00050 *           kes sense
00051 *
00052 *
00053 *
00054 *           done for the snapshot not for the timestep
00055 *
00056 *           TODO this recipe now has two different physics inside: infall and treat
00057 *           ment
00058 *           of mergers, these shoudl be separated.
00059 *           */
00060 double infall_recipe(int centralgal, int ngal, double Zcurr)
00061 {
00062     int i;
00063     double tot_stellarMass, tot_coldMass, tot_ejected, tot_hotMass, tot_ejectedMeta
00064     ls, tot_hotMassMetals,
00065     tot_BHMass, reionization_modifier, hot_onto_central,hotmetals_onto_central, t
00066     ot_ICM, tot_ICMMetals;
00067     double infallingMass,hotremain,hotsat,hotmetals;
00068     double ejectedmetals,totmass;
00069     double dis,hottype1;
00070     double hotupdate,ejectedupdate;
00071
00072 /* need to add up all the baryonic mass associated with the full halo to check
00073 * what baryonic fraction is missing/in excess. That will give the mass of gas
00074 * that need to be added/subtracted to the hot phase, gas that infalled.*/
00075 tot_stellarMass = 0.0;
00076 tot_coldMass = 0.0;
00077 tot_hotMass = 0.0;
00078 tot_ejected = 0.0;
00079 tot_hotMassMetals = 0.0;
00080 tot_ejectedMetals = 0.0;
00081 tot_BHMass = 0.0;
00082 tot_ICM = 0.0;
00083 tot_ICMMetals = 0.0;
00084 /* added to the central galaxies */
00085 hot_onto_central=0.0;
00086 hotmetals_onto_central=0.0;
00087
00088 /* SatelliteRecipe == 1 => Guo2010 non instantaneous treatment of
00089 * gas stripping from satellites (thsi means )+ ejected and hot gas of type 2's
00090 * can
00091 * be split between type 0 and type 1*/
00092
00093 if (SatelliteRecipe == 1 && EjectionRecipe == 2 )
00094 {
00095     for(i = 0; i < ngal; i++) /* Loop over all galaxies in the FoF-halo
00096 */
00097 {
00098     if(Gal[i].Type == 2)
00099         update_ICL(Gal[i].CentralGal, i);
00100
00101     /* only galaxies within rvir contribute to total baryon mass in central
00102     halo */
00103     dis=
00104         sqrt(pow(Gal[centralgal].Pos[0] - Gal[Gal[i].CentralGal].Pos[0], 2.0)
00105         +
00106             pow(Gal[centralgal].Pos[1] - Gal[Gal[i].CentralGal].Pos[1], 2.0)
00107             +
00108                 pow(Gal[centralgal].Pos[2] - Gal[Gal[i].CentralGal].Pos[2], 2.0)

```

```

) / (1 + ZZ[Halo[Gal[centralgal].HaloNr].SnapNum]);
00106
00107
00108     if ( dis < Gal[centralgal].Rvir )
00109     {
00110         tot_stellarMass += Gal[i].StellarMass;
00111         tot_coldMass += Gal[i].ColdGas;
00112
00113         tot_hotMass += Gal[i].HotGas;
00114         tot_hotMassMetals += Gal[i].MetalsHotGas;
00115
00116         tot_BHMass += Gal[i].BlackHoleMass;
00117         tot_ICM += Gal[i].ICM;
00118         tot_ICMMetals += Gal[i].MetalsICM;
00119     }
00120
00121     if(i == centralgal)
00122     {
00123         tot_ejected += Gal[i].EjectedMass;
00124         tot_ejectedMetals += Gal[i].MetalsEjectedMass;
00125
00126     }
00127     /*if galaxy is a satellite*/
00128     else
00129     {
00130         if (dis < Gal[centralgal].Rvir)
00131         {
00132             /*if galaxy is a satellite its gas is collect in the hot gas phase o
f the central
00133             * for the infall calculation.*/
00134             tot_hotMass += Gal[i].EjectedMass;
00135             tot_hotMassMetals += Gal[i].MetalsEjectedMass;
00136
00137             /*check this comment*/
00138
00139             /* satellite ejected gas can either be added to the hot phase o
f the
00140                     central galaxies or ejected reservoir of the cen
tral galaxies.*/
00141             // TBD should this be tot_ejected+= .. etc is it important (only
y used in infallingMass calculation)
00142
00143             /*If galaxy is a type 2 satellite inside Rvir of type 0*/
00144             if (Gal[i].Type == 2)
00145             {
00146                 /* hot gas of type 2 galaxies is added to their merger cent
er - part for the type 1 and
00147                     part for the type 0.*/
00148                 hotupdate = Gal[Gal[i].CentralGal].HotRadius / Gal[Gal[i].
CentralGal].Rvir * Gal[i].HotGas;
00149                 ejectedupdate = Gal[Gal[i].CentralGal].HotRadius / Gal[Gal[
i].CentralGal].Rvir * Gal[i].EjectedMass;
00150
00151                 /*If the central galaxy of this type 2 is a type 1, it will
receive
00152                     * a fraction hotupdate of hot gas fro mthe type 2, so the
hot frac
00153                     * of the type 1 needs to be updated. */
00154                 if (Gal[Gal[i].CentralGal].Type == 1)
00155                     update_hot_frac(Gal[i].CentralGal,hotupdate,Gal[Gal[
i].CentralGal].HotGas);
00156
00157                     /*type 1's still have their own ejected gas before becoming
a type2 */
00158                     /*slip type 2 ejected gas between type 0 and type 1*/
00159                     Gal[Gal[i].CentralGal].EjectedMass += ejectedupdate;
                     Gal[Gal[i].CentralGal].MetalsEjectedMass += Gal[i].
MetalsEjectedMass * Gal[Gal[i].CentralGal].HotRadius / Gal[Gal[i].CentralGal].

```

```

Rvir;
00160           Gal[centralgal].EjectedMass += Gal[i].EjectedMass - ejected
00161           update;
00161           Gal[centralgal].MetalsEjectedMass += Gal[i].
00161           MetalsEjectedMass * (1-Gal[Gal[i].CentralGal].HotRadius / Gal[Gal[i].CentralGal].
00161           Rvir);
00162           /*slip type 2 hot gas between type 0 and type 1*/
00163           Gal[Gal[i].CentralGal].HotGas +=hotupdate;
00164           Gal[Gal[i].CentralGal].MetalsHotGas +=Gal[i].MetalsHotGas *
00165           Gal[Gal[i].CentralGal].HotRadius / Gal[Gal[i].CentralGal].Rvir;
00166           Gal[centralgal].HotGas += Gal[i].HotGas - hotupdate;
00167           Gal[centralgal].MetalsHotGas += Gal[i].MetalsHotGas - Gal[i].
00167           MetalsHotGas * Gal[Gal[i].CentralGal].HotRadius / Gal[Gal[i].CentralGal].Rvir;
00168           /*give type 2 ICM to type 1*/
00169           Gal[Gal[i].CentralGal].ICM +=Gal[i].ICM;
00170           Gal[Gal[i].CentralGal].MetalsICM +=Gal[i].MetalsICM;
00171           Gal[i].HotGas = Gal[i].MetalsHotGas = Gal[i].HotFrac =
00172           Gal[i].EjectedMass = Gal[i].MetalsEjectedMass = Gal[i].HotRadius = Gal[i].ICM=
00172           Gal[i].MetalsICM=0.0;
00173           }
00174           }
00175           }
00176           /*if galaxy outside Rvir*/
00177           else
00178           {
00179           /*if type 2 outside Rvir*/
00180           if (Gal[i].Type == 2)
00181           {
00182               update_hot_frac(Gal[i].CentralGal, Gal[i].HotGas, Gal[Gal[i].
00182               CentralGal].HotGas);
00184               Gal[Gal[i].CentralGal].HotGas +=Gal[i].HotGas;
00186               Gal[Gal[i].CentralGal].MetalsHotGas +=Gal[i].MetalsHotGas;
00187
00188
00189               Gal[Gal[i].CentralGal].EjectedMass +=Gal[i].EjectedMass;
00190               Gal[Gal[i].CentralGal].MetalsEjectedMass +=Gal[i].
00190               MetalsEjectedMass;
00191               Gal[Gal[i].CentralGal].ICM +=Gal[i].ICM;
00192               Gal[Gal[i].CentralGal].MetalsICM +=Gal[i].MetalsICM;
00193               Gal[i].EjectedMass = Gal[i].MetalsEjectedMass = Gal[i].
00194               HotGas =Gal[i].MetalsHotGas= Gal[i].HotFrac = Gal[i].HotRadius =Gal[i].ICM=Gal[i].
00194               MetalsICM= 0.0;
00195               }
00196               }
00197               }
00198               }
00199               }
00200
00201           }
00202
00203       }
00204
00205
00206 /*If gradual striping and type 2's only eject into type 0.*/
00207 /*if galaxy is a type 2 all its ejected gas goes into the type 0
00208 * but the hot gas is split between type 1 and 0*/
00209   if (SatelliteRecipe == 1 && EjectionRecipe == 1 )
00210   {
00211     for(i = 0; i < ngal; i++)    /* Loop over all galaxies in the FoF-halo */
00212     {
00213
00214 #ifdef ICL

```

```

00215     if(Gal[i].Type == 2)
00216         update_ICL(centralgal, i);
00217 #endif
00218
00219
00220
00221     dis=
00222     sqrt (pow(Gal[centralgal].Pos[0] - Gal[Gal[i].CentralGal].Pos[0], 2.0)
00223     +
00224         pow(Gal[centralgal].Pos[1] - Gal[Gal[i].CentralGal].Pos[1], 2.0)
00225         +
00226             pow(Gal[centralgal].Pos[2] - Gal[Gal[i].CentralGal].Pos[2], 2.0)
00227     )/(1 + ZZ[Halos[Gal[centralgal].HaloNr].SnapNum]);
00228
00229     /* only galaxies within rvir contribute to total baryon mass in central
00230     halo */
00231     if ( dis < Gal[centralgal].Rvir )
00232     {
00233         tot_stellarMass += Gal[i].StellarMass;
00234         tot_coldMass += Gal[i].ColdGas;
00235
00236         tot_hotMass += Gal[i].HotGas;
00237         tot_hotMassMetals += Gal[i].MetalsHotGas;
00238
00239         tot_BHMass += Gal[i].BlackHoleMass;
00240         tot_ICM += Gal[i].ICM;
00241         tot_ICMMetals += Gal[i].MetalsICM;
00242     }
00243
00244     /* satellite ejected gas can either be added to the hot phase
00245     * or ejected resevor of the central galaxies.*/
00246
00247     if(i == centralgal)
00248     {
00249         tot_ejected += Gal[i].EjectedMass;
00250         tot_ejectedMetals += Gal[i].MetalsEjectedMass;
00251     }
00252
00253     else
00254     {
00255         /*if it is a satellite inside Rvir*/
00256         if (dis < Gal[centralgal].Rvir)
00257         {
00258             /*should be before*/
00259             tot_ejected += Gal[i].EjectedMass;
00260             tot_ejectedMetals += Gal[i].MetalsEjectedMass;
00261
00262             /* if      if EjectionRecipe == 1, then ejecteedmass=0, also works
00263             here */
00264             /* all the ejected gas of satellite is added to central galaxy no matte
00265             r
00266             * it its type 1 or 2.*/
00267             Gal[centralgal].EjectedMass += Gal[i].EjectedMass;
00268             Gal[centralgal].MetalsEjectedMass += Gal[i].MetalsEjectedMass;
00269
00270             /*only for type 2's as the gradual striping of hot gas in type 1's
00271             will be
00272             * treated later. */
00273             if (Gal[i].Type == 2)
00274                 { /* a fraction of hot gas of type 2 galaxies is added to the
00275                 ir merger center */
00276
00277                 /*TODO this will give the fraction of gas from type 2 t
00278                 hat goes into type 0 and type 1.
00279                         * Is this Rvir of the type 1? if it is should be of ty
00280                         pe 0*/

```

```

00272             hotupdate = Gal[Gal[i].CentralGal].HotRadius / Gal[Gal[i].
00273                 CentralGal].Rvir * Gal[i].HotGas;;
00274             if (Gal[Gal[i].CentralGal].Type == 1)
00275                 update_hot_frac(Gal[i].CentralGal,hotupdate,Gal[Gal[i].Ce
00276                     ntralGal].HotGas);
00277             /*if galaxy is a type 2 orbiting a type 1 a fraction of its
00278                 gas goes to the type 1
00279                 * and a fraction to the type 0*/
00280                 Gal[Gal[i].CentralGal].HotGas +=hotupdate;
00281                 Gal[Gal[i].CentralGal].MetalsHotGas +=Gal[i].MetalsHotGas *
00282                     Gal[Gal[i].CentralGal].HotRadius / Gal[Gal[i].CentralGal].Rvir;
00283                 Gal[centralgal].HotGas += Gal[i].HotGas - hotupdate;
00284                 Gal[centralgal].MetalsHotGas += Gal[i].MetalsHotGas - Gal[i]
00285                     .MetalsHotGas * Gal[Gal[i].CentralGal].HotRadius / Gal[Gal[i].CentralGal].Rvir;
00286             /*TODO if central is type 1 all the ICM of type 2 goes to i
00287                 t?, not consistent with hot gas.*/
00288                 Gal[Gal[i].CentralGal].ICM +=Gal[i].ICM;
00289                 Gal[Gal[i].CentralGal].MetalsICM +=Gal[i].MetalsICM;
00290             Gal[i].HotGas = Gal[i].MetalsHotGas = Gal[i].HotFrac =
00291                 Gal[i].EjectedMass = Gal[i].MetalsEjectedMass = Gal[i].HotRadius = Gal[i].ICM=
00292                 Gal[i].MetalsICM=0.0;
00293         }
00294     /* if      if EjectionRecipe == 1, then ejecteedmass=0, also works
00295         here */
00296     Gal[i].EjectedMass = Gal[i].MetalsEjectedMass = 0.0;
00297 }
00298 else //outside Rvir
00299 {
00300 /* If galaxy is a type 2 outside Rvir of type 0, then all its gas com
00301     ponents
00302     * will be added to the type 1. If its a type 1 outside Rvir it reta
00303     ins all
00304     * its gas components so that case is not here. */
00305     if (Gal[i].Type == 2)
00306     {
00307         update_hot_frac(Gal[i].CentralGal,Gal[i].HotGas,Gal[Gal[i].
00308             CentralGal].HotGas);
00309         Gal[Gal[i].CentralGal].HotGas +=Gal[i].HotGas;
00310         Gal[Gal[i].CentralGal].MetalsHotGas +=Gal[i].MetalsHotGas;
00311         Gal[Gal[i].CentralGal].EjectedMass +=Gal[i].EjectedMass;
00312         Gal[Gal[i].CentralGal].MetalsEjectedMass +=Gal[i].
00313             MetalsEjectedMass;
00314         Gal[Gal[i].CentralGal].ICM +=Gal[i].ICM;
00315         Gal[Gal[i].CentralGal].MetalsICM +=Gal[i].MetalsICM;
00316         Gal[i].EjectedMass = Gal[i].MetalsEjectedMass = Gal[i].
00317             HotGas =Gal[i].MetalsHotGas= Gal[i].HotFrac = Gal[i].HotRadius =Gal[i].ICM=Gal[i].
00318             MetalsICM= 0.0;
00319     }
00320 }
00321 /*instantaneous striping of gas from satellites and no ejection of type 2 into
00322     type 1,

```

```

00323     * still there is the condition on Rvir that determines that if a galaxy is a n
00324     * ewly
00325     * accreted type 2 outside Rvir of type 0, its gas will go into the type 1. If
00326     * its
00327     * a type 1 outside Rvir of type 0, it retains all its gas.*/
00328     if(SatelliteRecipe == 0)
00329     {
00330         for(i = 0; i < ngal; i++)      /* Loop over all galaxies in the FoF-halo */
00331         {
00332 #ifdef ICL
00333             if(Gal[i].Type == 2)
00334                 update_ICL(Gal[i].CentralGal, i);
00335 #endif
00336
00337             dis=
00338                 sqrt(pow(Gal[centralgal].Pos[0] - Gal[Gal[i].CentralGal].Pos[0], 2.0)
00339                 +
00340                 pow(Gal[centralgal].Pos[1] - Gal[Gal[i].CentralGal].Pos[1], 2.0)
00341                 +
00342                 pow(Gal[centralgal].Pos[2] - Gal[Gal[i].CentralGal].Pos[2], 2.0)
00343                 )/(1 + ZZ[Halo[Gal[centralgal].HaloNr].SnapNum]);
00344             /* If galaxy is a satellite inside Rvir of type 0 it will lose all its
00345             hot and ejected gas
00346             * into the type 0. only galaxies within Rvir contribute to the baryoni
00347             c fraction of the central halo.*/
00348             if ( dis < Gal[centralgal].Rvir )
00349             {
00350                 /*sum components to compute infall (the hot gas component will be a
00351                 added to hot_onto_central,
00352                 * should be done here). */
00353                 tot_stellarMass += Gal[i].StellarMass;
00354                 tot_coldMass += Gal[i].ColdGas;
00355
00356                 tot_hotMass += Gal[i].HotGas;
00357                 tot_hotMassMetals += Gal[i].MetalsHotGas;
00358
00359                 tot_BHMass += Gal[i].BlackHoleMass;
00360                 tot_ICM += Gal[i].ICM;
00361                 tot_ICMMetals += Gal[i].MetalsICM;
00362
00363                 if(i == centralgal)
00364                 {
00365                     tot_ejected += Gal[i].EjectedMass;
00366                     tot_ejectedMetals += Gal[i].MetalsEjectedMass;
00367                 }
00368                 else
00369                 {
00370                     /* collect satellite ejected gas into hot halo */
00371                     tot_hotMass += Gal[i].EjectedMass;
00372                     tot_hotMassMetals += Gal[i].MetalsEjectedMass;
00373
00374                     /* reset all hot and ejected satellite (type>0) masses to zero
00375                     */
00376                     Gal[i].HotGas = Gal[i].MetalsHotGas = Gal[i].EjectedMass = Gal[i].
00377                     MetalsEjectedMass = Gal[i].HotFrac = Gal[i].HotRadius = Gal[i].ICM = Gal[i].
00378                     MetalsICM = 0.0;
00379                 }
00380             }
00381             else
00382             {
00383                 /* If galaxy is a type 2 outside Rvir of type 0, then all its g
00384                 as components
00385                 * will be added to the type 1. If its a type 1 outside Rvir it
00386                 retains all

```

```

00377             * its gas components so that case is not here. */
00378     if (Gal[i].Type == 2)
00379     {
00380
00381         Gal[Gal[i].CentralGal].HotGas +=Gal[i].HotGas;
00382         Gal[Gal[i].CentralGal].MetalsHotGas +=Gal[i].MetalsHotGas;
00383
00384         Gal[Gal[i].CentralGal].EjectedMass +=Gal[i].EjectedMass;
00385         Gal[Gal[i].CentralGal].MetalsEjectedMass +=Gal[i].
00386             MetalsEjectedMass;
00387         Gal[Gal[i].CentralGal].ICM +=Gal[i].ICM;
00388         Gal[Gal[i].CentralGal].MetalsICM +=Gal[i].MetalsICM;
00389
00390         Gal[i].EjectedMass = Gal[i].MetalsEjectedMass = Gal[i].
00391         HotGas =Gal[i].MetalsHotGas= Gal[i].HotFrac = Gal[i].HotRadius =Gal[i].ICM=Gal[i].
00392             .MetalsICM= 0.0;
00393
00394     }
00395 }
00396 }
00397
00398
00399
00400
00401
00402 /* The baryonic fraction is conserved by adding/subtracting the infallingMass
00403 * calculated here to/from the hot gas of the central galaxy of the FOF
00404 * - include reionization if necessary. This is done in main.c where the infall
00405 * recipe is called.
00406 * If ReionizationOn=1, the impact of reionization on the fraction of infalling
00407 * gas is computed. There are two options depending on UPDATEREIONIZATION (ifnd
00408 ef
00409 * use gnedin, if defined use Hoeft (2006)). In both cases reionization as the
00410 effect
00411 * of reducing the fraction of baryons that collapse into dark matter halos,
00412 * reducing the amount of infalling gas. */
00413 if(ReionizationOn == 1)
00414     reionization_modifier = do_reionization(centralgal, Zcurr);
00415 else
00416     reionization_modifier = 1.0;
00417
00418 infallingMass =
00419     reionization_modifier * BaryonFrac * Gal[centralgal].Mvir - (tot_stellarMass
00420     + tot_coldMass +
00421                                         tot_hotMass + to
00422     t_ejected + tot_BHMass+tot_ICM);
00423
00424
00425 /* GRADUAL STRIPPING - If SatelliteRecipe == 1, there is a gradual
00426 * striping of the gas from accreted satellites (type 1) so only a
00427 * fraction of their hot gas contributes to hot_onto_central (hot gas
00428 * that will end up in the central galaxy), same for hotmetals_onto_central.
00429 * The hot gas is striped at the same rate that dark matter is striped. */
00430 if(SatelliteRecipe == 1)
00431 {
00432     for(i = 0; i < ngal; i++) /* Loop over all galaxies in the FoF-halo */
00433     {
00434         dis=
00435         sqrt(pow(Gal[centralgal].Pos[0] - Gal[Gal[i].CentralGal].Pos[0], 2.0)

```

```

+
00436      +          pow(Gal[centralgal].Pos[1] - Gal[Gal[i].CentralGal].Pos[1], 2.0)
00437      +
00438      +          pow(Gal[centralgal].Pos[2] - Gal[Gal[i].CentralGal].Pos[2], 2.0)
00439      )/(1 + ZZ[Halo[Gal[centralgal].HaloNr].SnapNum]);
00440
00441      /*the hot gas of the central galaxy goes to hot_onto_central */
00442      if(i == centralgal)
00443      {
00444          hot_onto_central +=Gal[i].HotGas;
00445          hotmetals_onto_central +=Gal[i].MetalsHotGas;
00446      }
00447      else
00448      {
00449          /*If galaxy is a type 1 and its already inside Rvir of type 0,
00450          then the
00451          * gradual striping of the gas is computed, only hotsat-hotrema
00452          * in is added
00453          * to hot_onto_central. If galaxy is a type 2, it already lost
00454          * all the gas.*/
00455          if (Gal[i].Type == 1 && dis < Gal[centralgal].Rvir && Gal[i].
00456          HotGas > 0.0)
00457          {
00458              /*gradually stripping of satellite hot gas*/
00459
00460              hotsat=Gal[i].HotGas;
00461              hotmetals=Gal[i].MetalsHotGas;
00462
00463              /*fraction of hotgas that the type 1 will retain*/
00464              hotremain=hot_retain_sat(i,centralgal,infallingMass);
00465
00466              /*update mass in hot gas and in metals in hots gas in the type
00467              1 after striping*/
00468              Gal[i].MetalsHotGas = Gal[i].MetalsHotGas/Gal[i].HotGas*hotremain
00469              in;
00470              Gal[i].HotGas = hotremain;
00471              if (hotsat < hotremain)
00472              {
00473                  printf("wrong here remain hotgas %f hotremain %f\n",hotsat,
00474                  hotremain);
00475                  printf("Gal.type %d\n",Gal[i].Type);
00476                  exit(0);
00477
00478              /*the fraction of gas that is striped goes into hot_onto_centera
00479              l to be added
00480              * to the type 0*, same for metals*/
00481              hot_onto_central +=hotsat-hotremain;
00482              hotmetals_onto_central +=hotmetals-Gal[i].MetalsHotGas;
00483
00484              /*the ejected gas mass and metals of satellites are striped int
00485              o the central
00486              *
00487              * galaxy in a fraction proportional to the striping of hot gas.*/
00488              Gal[centralgal].EjectedMass += Gal[i].EjectedMass*(hotsat-hotre
00489              main)/hotsat;
00490              Gal[centralgal].MetalsEjectedMass += Gal[i].MetalsEjectedMass*(
00491              hotsat-hotremain)/hotsat;
00492              /*Only makes sense if EjectionRecipe=2, otherwise type 1 have n
00493              o ejected*/
00494              Gal[i].EjectedMass -= Gal[i].EjectedMass*(hotsat-hotremain)/hot
00495              sat;
00496              Gal[i].MetalsEjectedMass -= Gal[i].MetalsEjectedMass*(hotsat-ho
00497              tremain)/hotsat;

```

```

00485             }
00486         }
00487     }
00488 }
00489 }
00490
00491
00492 /* if satelliterecipe == 0 then all hot gas and ejected gas of newly accreted s
00493 * satellites
00494 * sat. contributes to the hot component of the type 0 - hot_onto_central
00495 * this should be together with the 0 option before*/
00496 if (SatelliteRecipe == 0)
00497 {
00498     Gal[centralgal].ICM += tot_ICM;
00499     Gal[centralgal].MetalsICM += tot_ICMMetals;
00500     hot_onto_central = tot_hotMass;
00501     hotmetals_onto_central =tot_hotMassMetals;
00502 }
00503 /* update hot mass on centrals*/
00504 /* No need to update the ejected mas with tot_ejected. If satellite recipe =0,
00505 * the ejected gas of satellites is added to the hot phase if satrecipe=1 alre
00506 * ady done.*/
00507 Gal[centralgal].HotGas = hot_onto_central;
00508 Gal[centralgal].MetalsHotGas = hotmetals_onto_central;
00509 /* some book keeping and checks below */
00510 if(Gal[centralgal].MetalsHotGas > Gal[centralgal].HotGas)
00511 {
00512     Gal[centralgal].MetalsHotGas = Gal[centralgal].HotGas;
00513 }
00514
00515 if(Gal[centralgal].HotGas < 0.0)
00516     Gal[centralgal].HotGas = Gal[centralgal].MetalsHotGas = 0.0;
00517
00518 if(Gal[centralgal].MetalsHotGas < 0.0)
00519     Gal[centralgal].MetalsHotGas = 0.0;
00520
00521 if(Gal[centralgal].MetalsEjectedMass > Gal[centralgal].EjectedMass)
00522 {
00523     Gal[centralgal].MetalsEjectedMass = Gal[centralgal].EjectedMass;
00524 }
00525 if(Gal[centralgal].EjectedMass < 0.0)
00526 {
00527     Gal[centralgal].EjectedMass = Gal[centralgal].MetalsEjectedMass = 0.0;
00528 }
00529 if(Gal[centralgal].MetalsEjectedMass < 0.0)
00530     Gal[centralgal].MetalsEjectedMass = 0.0;
00531
00532 return infallingMass;
00533
00534 }
00535
00536
00537
00538
00539 #ifndef UPDATEREIONIZATION
00540 double do_reionization(int centralgal, double Zcurr)
00541 {
00542     double alpha, a, f_of_a, a_on_a0, a_on_ar, Mfiltering, Mjeans, Mchar, mass_to_u
00543     se, modifier;
00544     double Tvir, Vchar, omegaZ, xZ, deltacritZ, HubbleZ;
00545     /** reionization recipie described in Gnedin (2000), using the fitting */
00546     /* formulas given by Kravtsov et al (2004) Appendix B, used after Delucia 2004
00547 */

```

```

00547
00548     /* here are two parameters that Kravtsov et al keep fixed. */
00549     /* alpha gives the best fit to the Gnedin data */
00550     alpha = 6.0;
00551     Tvir = 1e4;
00552
00553     /* calculate the filtering mass */
00554
00555     a = 1.0 / (1.0 + Zcurr);
00556     a_on_a0 = a / a0;
00557     a_on_ar = a / ar;
00558
00559     if(a <= a0)
00560         f_of_a = 3.0 * a / ((2.0 * alpha) * (5.0 + 2.0 * alpha)) * pow(a_on_a0, alpha
00561     );
00562     else if((a > a0) && (a < ar))
00563         f_of_a =
00564             (3.0 / a) * a0 * a0 * (1.0 / (2.0 + alpha) - 2.0 * pow(a_on_a0, -0.5) / (5.
00565             0 + 2.0 * alpha)) +
00566             a * a / 10.0 - (a0 * a0 / 10.0) * (5.0 - 4.0 * pow(a_on_a0, -0.5));
00567     else
00568         f_of_a =
00569             (3.0 / a) * (a0 * a0 * (1.0 / (2.0 + alpha) - 2.0 * pow(a_on_a0, -0.5) / (5.
00570             .0 + 2.0 * alpha)) +
00571                 (ar * ar / 10.0) * (5.0 - 4.0 * pow(a_on_ar, -0.5)) - (a0 *
00572                 a0 / 10.0) * (5.0 -
00573                     4.0 *
00574                         pow(a_on_a0,
00575                             -0.5)) +
00576                         a * ar / 3.0 - (ar * ar / 3.0) * (3.0 - 2.0 * pow(a_on_ar, -0.
00577                         5)));
00578
00579     /* this is in units of 10^10Msun/h, note mu=0.59 and mu^-1.5 = 2.21 */
00580     Mjeans = 25.0 * pow(Omega, -0.5) * 2.21;
00581     Mfiltering = Mjeans * pow(f_of_a, 1.5);
00582
00583     /* calculate the characteristic mass corresponding to a halo temperature of 10^
00584     4K */
00585     Vchar = sqrt(Tvir / 36.0);
00586     omegaZ = Omega * (pow(1.0 + Zcurr, 3.0) / (Omega * pow(1.0 + Zcurr, 3.0) +
00587         OmegaLambda));
00588     xZ = omegaZ - 1.0;
00589     deltaritz = 18.0 * M_PI * M_PI + 82.0 * xZ - 39.0 * xZ * xZ;
00590     HubbleZ = Hubble * sqrt(Omega * pow(1.0 + Zcurr, 3.0) + OmegaLambda);
00591     Mchar = Vchar * Vchar * Vchar / (G * HubbleZ * sqrt(0.5 * deltaritz));
00592
00593     /* we use the maximum of Mfiltering and Mchar */
00594     mass_to_use = dmax(Mfiltering, Mchar);
00595     modifier = 1.0 / pow(1.0 + 0.26 * (mass_to_use / Gal[centralgal].Mvir), 3.0);
00596
00597     return modifier;
00598
00599 double do_reionization(int centralgal, double Zcurr)
00600 {
00601     double x0,x,delta_c,delta_0,a,a0,alpha,tau,Mc,modifier;
00602     int tabindex;
00603     double f1, f2;

```

```

00604 /* we employ the reionization recipie described in Hoeft (2006), however use t
00605 he fitting
00606 * from Okamoto et al. 2008*/
00607 alpha = 2.0;
00608 a = 1. / (1 + Zcurr);
00609 a0 = 1.;
00610
00611 x = - (1 - Omega) * a * a * a / (Omega + (1-Omega) * a * a * a);
00612 delta_c = (178 + 82 *x - 39 * x * x) / (1. + x);
00613
00614 x0 = - (1 - Omega) * a0 * a0 * a0 / (Omega + (1-Omega) * a0 * a0 * a0);
00615 delta_0 = (178 + 82 *x0 - 39 * x0 * x0) / (1. + x0);
00616
00617 tau = 0.73 * pow(Zcurr + 1,0.18)* exp(-pow(0.25 * Zcurr, 2.1));
00618
00619 Mc = pow(tau / (1 + Zcurr),3./2) * sqrt(delta_0 / delta_c);
00620
00621 /* if use Okamoto et al. 2008*/
00622
00623 find_interpolate_reionization( Zcurr, &tabindex, &f1, &f2);
00624 Mc = f1*log10(Reion_Mc[tabindex])+f2*log10(Reion_Mc[tabindex+1]);
00625 Mc = pow(10, Mc-10);
00626
00627
00628 modifier = pow(1 + (pow(2, alpha/3.) -1) * pow(Mc / Gal[centralgal].Mvir, alpha
00629 ), -3./alpha);
00630
00631 return modifier;
00632
00633 }
00634
00635 #endif
00636
00637 /** The gas that infalled is added to the hot gas of the central galaxy. */
00638
00639 void add_infall_to_hot(int centralgal, double infallingGas)
00640 {
00641
00642 /* add the infalling gas to the central galaxy hot component */
00643 Gal[centralgal].HotGas += infallingGas;
00644
00645 //infalling gas is metals free
00646 if(Gal[centralgal].HotGas < 0.0)
00647 /*TODO infallingGas may be < 0 [GL?]
00648 /*TODO - if the baryonic fraction calculated
00649 requires that we take out more than the hot gas what should we do?
00650 take out the cold gas? If infalling<0 should we take out cold gas or ej
00651 ected?*/
00652 {
00653     Gal[centralgal].HotGas = Gal[centralgal].MetalsHotGas = 0.0;
00654 } else if(Gal[centralgal].MetalsHotGas < 0.0)//TODO is this needed?
00655 Gal[centralgal].MetalsHotGas = 0.0;
00656
00657 }
00658
00659
00660 /** Gradual stripping of hot gas (ejected sgas stripping is proportional)
00661 * from type 1 satellites. This is caused both by tidal and ram-pressure
00662 * stripping. This function return the actual mass of hot gas that the
00663 * type 1 retains.
00664 *
00665 * TIDAL STRIPPING
00666 * Hot gas is tidally stripped at the same rate at which dark matter is
00667 * striped:

```

```

00668 *
00669 * \f$ \frac{M_{\rm{hot}}(R_{\rm{tidal}})}{M_{\rm{DM}}(M_{\rm{DM,infall}})} = 
00670 * \frac{M_{\rm{hot}}}{M_{\rm{DM}}} \frac{M_{\rm{DM,infall}}}{M_{\rm{DM}}} \f$ 
00671 *
00672 * Since the hot gas distribution is assumed to be  $\rho \propto r^{-2}$ 
00673 * this means  $M_{\rm{hot}}(r) \propto r$ . Therefore, the tidal
00674 * radius beyond which gas is stripped is given by:
00675 *
00676 * \f$ R_{\rm{tidal}} = 
00677 * \left( \frac{M_{\rm{DM}}}{M_{\rm{DM,infall}}} \right) R_{\rm{DM,infall}} \f$ 
00678 *
00679 * RAM PRESSURE STRIPPING
00680 * Let  $R_{\rm{r.p.}}$  represent the distance from the centre of the satellite
00681 * at which ram pressure stripping equals its self-gravity. Then:
00682 *
00683 * \f$ \rho_{\rm{sat}}(R_{\rm{r.p.}}) V^2 = 
00684 * \rho_{\rm{par}}(R_{\rm{orbit}}) V^2 \f$ 
00685 * Where the four terms represent respectively the density of the satellite
00686 * at  $R_{\rm{r.p.}}$ , the virial velocity of the satellite at infall,
00687 * the density of the parent halo at the radius of the satellite and the
00688 * orbit velocity of the satellite (given by  $V_{\rm{c}}$  of the parent halo\f$ )
00689 *
00690 * The stripping radius is given by
00691 *
00692 * \f$ R_{\rm{strip}} = \min(R_{\rm{tidal}}, R_{\rm{r.p.}}) \f$ 
00693 *
00694 *
00695 */
00696 double hot_retain_sat(int i, int centralgal,double infallingMass)
00697 {
00698     double hotremain,frac,tmp;
00699     double R_2,R_1,Rt,factor;
00700     double RetainFrac;
00701
00702     /*Calculate tidal stripping radius*/
00703     if (Gal[centralgal].Type != 0)
00704         exit(0);
00705     Rt=Gal[i].Len*PartMass/Gal[i].Mvir*Gal[i].Rvir;
00706
00707     /*Ram pressure stripping radius calculation*/
00708     /*First calculate the orbital radius of the satellite R_orbit*/
00709     R_1=sqrt(pow(Gal[centralgal].Pos[0]-Gal[i].Pos[0],2.)+pow(Gal[centralgal].Pos[1]
00710             -Gal[i].Pos[1],2.))
00711             +pow(Gal[centralgal].Pos[2]-Gal[i].Pos[2],2.))/(1+ZZ[Halo[Gal[c
00712             entralgal].HaloNr].SnapNum]);
00713
00714     /*If the central galaxy has no hot gas, it exerts no ram pressure stripping on
00715     the
00716     * satellite. */
00717     if (Gal[centralgal].HotGas+infallingMass <1.e-6)
00718         R_2=Gal[i].HotRadius;
00719     else
00720     {
00721         /*R_r.p./R_orbit*/
00722         RetainFrac=sqrt(Gal[i].HotFrac/(Gal[centralgal].HotGas+infallingMass)*Gal[c
00723             entralgal].Mvir)
00724                     *(Gal[i].Mvir/Gal[i].Rvir)/(Gal[centralgal].Mvir/
00725             Gal[centralgal].Rvir);
00726         /*R_r.p. */
00727         R_2=RetainFrac*R_1;
00728     }
00729
00730     /*Get the smaller of tidal and ram pressure stripping radii.*/
00731     if (R_2 > Rt)
00732         R_2=Rt;
00733

```

```

00729 /*if the striping radius is larger then hot radius there is
00730   * no stripping*/
00731   if (R_2>Gal[i].HotRadius || Gal[i].HotGas < 1.e-8)
00732   {
00733     hotremain=Gal[i].HotGas;
00734
00735     return hotremain;
00736   }
00737 /* If stripping radius is smaller than the hot radius */
00738 else
00739 {
00740   /* Assuming M_hot(r) proportional to r, the remaining hot gas
00741    * is given by: */
00742   hotremain=Gal[i].HotGas*R_2/Gal[i].HotRadius;
00743   /* hot radius is updated to the stripping radius*/
00744   Gal[i].HotRadius=R_2;
00745
00746   return hotremain;
00747 }
00748 }
00749
00750
00751 /*updates hot radius and hot frac. Hot frac is Mhot/Mvir
00752 * duplicated with update_hot_gas??*/
00753 /*p - galaxy id, dmass - hot gas added, previous amount of hot gas.*/
00754 void update_hot_frac(int p, double dmass, float HotGas)
00755 {
00756   if (HotGas > 1.0e-8)
00757     Gal[p].HotFrac *=(dmass+HotGas)/HotGas;
00758   else
00759   {
00760     Gal[p].HotFrac = dmass/(Gal[p].Len*PartMass);
00761
00762     /*TODO, why the condition on 1e-8, shouldn't this be
00763      * if Hotrad< Gal[p].Len*PartMass/Gal[p].Mvir*Gal[p].Rvir; */
00764     if (Gal[p].HotRadius < 1.e-8)
00765       Gal[p].HotRadius = Gal[p].Len*PartMass/Gal[p].Mvir*Gal[p].Rvir;
00766     if (Gal[p].HotRadius > Gal[p].Rvir)
00767       Gal[p].HotRadius = Gal[p].Rvir;
00768   }
00769   if (Gal[p].HotFrac < 0.0)
00770   {
00771     Gal[p].HotFrac = 0.0;
00772   }
00773
00774 }
00775
00776 #ifdef ICL
00777
00778
00779 void update_ICL(int p, int q)
00780 {
00781   int j, outputbin;
00782   for(outputbin = 0; outputbin < NOUT; outputbin++)
00783   {
00784     for(j = 0; j < NMAG; j++)
00785     {
00786
00787     #ifdef OUTPUT_REST_MAGS
00788       Gal[p].ICLLum[j][outputbin] += Gal[q].ICLLum[j][outputbin];
00789       Gal[q].ICLLum[j][outputbin] = 0.0;
00790     #endif
00791     #ifdef COMPUTE_OBS_MAGS
00792       Gal[p].ObsICL[j][outputbin] += Gal[q].ObsICL[j][outputbin];
00793       Gal[q].ObsICL[j][outputbin] = 0.0;
00794     #endif
00795     ;

```

```

00796      }
00797      }
00798  }
00799 }
00800 #endif

```

4.31 code/recipe_mergers.c File Reference

Calculates the merging time, the central galaxy (for type 1's), adds galaxies together, calculates SF from bursts and grows black holes.

Functions

- int [set_merger_center](#) (int fofhalo)

Calculates the central galaxies for type 1's (needed if MERGE01=1).
- double [estimate_merging_time](#) (int halonr, int mother_halonr, int p)

Calculates the merging time whenever a galaxy becomes a satellite.
- void [deal_with_galaxy_merger](#) (int p, int merger_centralgal, int centralgal, double time, double deltaT, int halonr)

Deals with all the physics triggered by mergers.
- void [grow_black_hole](#) (int halonr, int merger_centralgal, double mass_ratio, double dt)

Grows black holes, through accretion from cold gas during mergers, as in Kauffmann & Haehnelt (2000) - Quasar Mode.
- void [add_galaxies_together](#) (int t, int p)

Adds all the components of the satellite galaxy into its central companion.
- void [do_major_merger_starburst](#) (int p, int centralgal, double time, double deltaT, int halonr)

Merger burst recipe used before Croton2006 - in a major merger all the cold gas from the central and satellite galaxies is consumed to form a bulge.
- void [make_bulge_from_burst](#) (int p)

In a major merger, both disks are destroyed and all the mass transferred to the bulge.
- double [collisional_starburst_recipe](#) (double mass_ratio, int merger_centralgal, int centralgal, double time, double deltaT, int halonr)

Merger burst recipe from Somerville 2001 (used after Croton2006).
- void [bulgesize_from_merger](#) (double mass_ratio, int merger_centralgal, int p, double Mcstar, double Mcbulge, double Mcgas, double frac)

Calculates the bulge size after a merger.

4.31.1 Detailed Description

set_merger_center - calculates the central galaxy for type 1's, since if MERGE01=1 (makefile) type 1's can also merge. Therefore, they need a merger central galaxy and will also have a merger clock (needed for millennium two, since due to the high resolution, haloes are very difficult to disrupt and orbit around forever).

estimate_merging_time sets up a merger clock. Originally this was done only for type 2 galaxies. The positions of galaxies in the model are normally given by the position of their dark matter halo. However, when galaxies become satellites, their dark matter haloes are stripped by the central object to the point where there is none left. At this point, the dark matter of the satellite becomes part of the main halo, but the galaxy's position should continue to evolve due to the dynamical friction force caused by the dark matter around it.

The code does keep track of the position of the most bounded particle when the satellite's halo was disrupted, but this is not used to track the galaxy position. Instead a clock is set, giving the time left until the satellite merges with the central galaxy. Before, this was only done for type 2's (satellites that lost a halo). Guo2010 included the MERGE01 option, that sets the clock also for type 1's (satellites with a halo), since for the highest resolution millennium 2, their haloes can be very small and orbit forever around the central companion.

This time is computed using the Chandrasekhar's formula for dynamical friction, as in Binney & Tremaine 1987:

$$F_{\text{df}} = -\frac{4\pi G^2 m_{\text{sat}}^2 \ln(\Lambda) \rho B(x)}{v_{\text{rel}}^2}.$$

Which gives (B&T Eq. 7.26):

$$t_{\text{df}} \approx 1.17 \frac{V_{\text{vir}} r_{\text{sat}}^2}{G m_{\text{sat}} \ln(\Lambda)},$$

that is afterwards multiplied by 2 (after Delucia2007 to fit the data). When the merging time reaches zero, the satellite is assumed to merge with the central galaxy.

deal_with_galaxy_merger deals with the process, according to the mass fraction of the merger. Major if $M_{\text{sat}}/M_{\text{central}} > 0.3$ and minor otherwise. It calls

- **add_galaxies_together** - Add the cold and stellar phase of the merged galaxy to the central one. Also form a bulge at the central galaxy with the stars from the satellite in a minor merger if BulgeFormationInMinorMergersOn=1 (Major mergers are dealt later).
- Then calls **grow_black_hole** - Grows black hole through accretion from cold gas during mergers (due to the instabilities triggered), as in Kauffmann & Haehnelt (2000). This is commonly referred as the quasar mode, main responsible for the black hole growth. After Croton2006 this mode is active even in minor mergers: $\Delta m_{\text{BH},Q} = M_{\text{BH},\text{min}} \frac{f_{\text{BH}}(m_{\text{sat}}/m_{\text{central}}) m_{\text{cold}}}{1 + (280 \text{ km s}^{-1}/V_{\text{vir}})^2}$.
- Finally the burst of star formation due to the merger is treated. There are two options based on StarBurstRecipe input parameter:
 - If StarBurstRecipe = 0 (introduced by Kauffmann 1999), the burst only occurs for major mergers. In this case all the cold gas in the satellite and central galaxies is consumed to form stars and these are transferred to the bulge of the new galaxy formed.
 - If StarBurstRecipe = 1 (since Croton2006), the Somerville 2001 model of bursts is used. The burst can happen for both major and minor mergers, with a fraction of the added cold gas from the satellite and central being consumed: $\dot{m}_{\star}^{\text{burst}} = 0.56 \left(\frac{m_{\text{sat}}}{m_{\text{central}}} \right)^{0.7} m_{\text{gas}}$. SN Feedback from starformation is computed and the sizes of bulge and disk followed.
- When a major merger occurs, the disk of both merging galaxies is completely destroyed to form a

bulge. In either type of mergers, the bulge size is updated using Eq. 33 in Guo2010: $C \frac{GM_{\text{new,bulge}}^2}{R_{\text{new,bulge}}} = C \frac{GM_1^2}{R_1} + C \frac{GM_2^2}{R_2} + \alpha_{\text{inter}} \frac{GM_1 M_2}{R_1 + R_2}$

Definition in file [recipe_mergers.c](#).

4.31.2 Function Documentation

4.31.2.1 void add_galaxies_together (int *t*, int *p*)

All the components of the satellite galaxy are added to the correspondent component of the central galaxy. Cold gas spin is updated and a bulge is formed at the central galaxy, with the stars of the satellite if `BulgeFormationInMinorMergersOn=1`.

Definition at line 434 of file [recipe_mergers.c](#).

References `GALAXY::BlackHoleMass`, `BulgeFormationInMinorMergersOn`, `GALAXY::BulgeMass`, `GALAXY::ColdGas`, `GALAXY::dObsLum`, `GALAXY::dObsLumBulge`, `GALAXY::dObsYLum`, `GALAXY::dObsYLumBulge`, `GALAXY::EjectedMass`, `Gal`, `GALAXY::GasSpin`, `GALAXY::HaloSpin`, `GALAXY::HotGas`, `GALAXY::ICLLum`, `GALAXY::ICM`, `GALAXY::Lum`, `GALAXY::LumBulge`, `GALAXY::MassWeightAge`, `GALAXY::MergeSat`, `GALAXY::MetalsBulgeMass`, `GALAXY::MetalsColdGas`, `GALAXY::MetalsEjectedMass`, `GALAXY::MetalsHotGas`, `GALAXY::MetalsICM`, `GALAXY::MetalsStellarMass`, `GALAXY::ObsICL`, `GALAXY::ObsLum`, `GALAXY::ObsLumBulge`, `GALAXY::ObsYLum`, `GALAXY::ObsYLumBulge`, `GALAXY::Sfr`, `GALAXY::SfrBulge`, `GALAXY::StarMerge`, `GALAXY::StellarMass`, `GALAXY::YLum`, and `GALAXY::YLumBulge`.

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

4.31.2.2 void bulgesize_from_merger (double *mass_ratio*, int *merger_centralgal*, int *p*, double *Mcstar*, double *Mcbulge*, double *Mcgas*, double *frac*)

For any type of merger calculates the new bulge size using Eq. 33 in Guo2010:

$$C \frac{GM_{\text{new,bulge}}^2}{R_{\text{new,bulge}}} = C \frac{GM_1^2}{R_1} + C \frac{GM_2^2}{R_2} + \alpha_{\text{inter}} \frac{GM_1 M_2}{R_1 + R_2}.$$

This implementation assumed that the new bulge occupies the same space as the components that formed it.

Definition at line 1013 of file [recipe_mergers.c](#).

References `GALAXY::BulgeMass`, `GALAXY::BulgeSize`, `GALAXY::ColdGas`, `Gal`, `GALAXY::GasDiskRadius`, `GALAXY::HaloNr`, `GALAXY::StellarDiskRadius`, `GALAXY::StellarMass`, and `ThreshMajorMerger`.

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

4.31.2.3 double collisional_starburst_recipe (double *mass_ratio*, int *merger_centralgal*, int *centralgal*, double *time*, double *deltaT*, int *halonr*)

If `StarBurstRecipe = 1` (since Croton2006), the Somerville 2001 model of bursts is used. The burst can happen for both major and minor mergers, with a fraction of the added cold gas from the satellite and central being consumed. SN Feedback from starformation is computed and the sizes of bulge and disk followed (not done for the other burst mode).

Definition at line 765 of file [recipe_mergers.c](#).

References [add_to_luminosities\(\)](#), [GALAXY::ColdGas](#), [EjectionOn](#), [EjectPreVelocity](#), [EjectSlope](#), [Energy_in_Reheat\(\)](#), [EnergySNcode](#), [EtaSNcode](#), [FeedbackEjectionEfficiency](#), [FeedbackEpsilon](#), [FeedbackRecipe](#), [FeedbackReheatingEpsilon](#), [FracZtoHot](#), [Gal](#), [get_metallicity\(\)](#), [Halo](#), [GALAXY::HaloNr](#), [GALAXY::InfallVmax](#), [ListOutputSnaps](#), [GALAXY::MetalsColdGas](#), [GALAXY::MetalsHotGas](#), [GALAXY::Pos](#), [ReheatPreVelocity](#), [ReheatSlope](#), [GALAXY::Rvir](#), [GALAXY::Sfr](#), [GALAXY::SnapNum](#), [halo_data::SnapNum](#), [GALAXY::StarMerge](#), [ThreshMajorMerger](#), [GALAXY::Type](#), [update_from_feedback\(\)](#), [update_from_star_formation\(\)](#), [GALAXY::Vmax](#), [GALAXY::Vvir](#), [Yield](#), and [ZZ](#).

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

4.31.2.4 void deal_with_galaxy_merger (int *p*, int *merger_centralgal*, int *centralgal*, double *time*, double *deltaT*, int *halonr*)

Deals with the physics triggered by mergers, according to the mass fraction of the merger ($M_{\text{sat}}/M_{\text{central}} >< 0.3$). Add the cold and stellar phase of the satellite galaxy to the central one, form a bulge at the central galaxy with the stars from the satellite in a minor merger if [BulgeFormationInMinorMergersOn=1](#). Grows black hole through accretion from cold gas "quasar mode". If [StarBurstRecipe = 0](#), all the cold gas is consumed to form stars and these are transferred to the bulge of the new galaxy formed (only for major mergers). If [StarBurstRecipe = 1](#), the Somerville 2001 model of bursts is used, SN Feedback from starformation is computed and the sizes of bulge and disk followed. When a major merger occurs, the disk of both merging galaxies is completely destroyed to form a bulge.

Definition at line 260 of file [recipe_mergers.c](#).

References [add_galaxies_together\(\)](#), [AGNRecipeOn](#), [GALAXY::BulgeMass](#), [GALAXY::BulgeSize](#), [bulgesize_from_merger\(\)](#), [check_disk_instability\(\)](#), [GALAXY::ColdGas](#), [collisional_starburst_recipe\(\)](#), [DiskRadiusMethod](#), [do_major_merger_starburst\(\)](#), [GALAXY::FirstProgGal](#), [Gal](#), [GALAXY::GasDiskRadius](#), [get_gas_disk_radius\(\)](#), [get_stellar_disk_radius\(\)](#), [grow_black_hole\(\)](#), [HaloGal](#), [make_bulge_from_burst\(\)](#), [GALAXY::MergeOn](#), [GALAXY::NextProgGal](#), [StarBurstRecipe](#), [GALAXY::StellarDiskRadius](#), [GALAXY::StellarMass](#), [ThreshMajorMerger](#), [TrackDiskInstability](#), and [GALAXY::Type](#).

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

4.31.2.5 void do_major_merger_starburst (int *p*, int *centralgal*, double *time*, double *deltaT*, int *halonr*)

If [StarBurstRecipe = 0](#) (introduced by Kauffmann 1999), the burst only occurs for major mergers. In this case all the cold gas in the satellite and central galaxies is consumed to form stars and these are transferred to the bulge of the new galaxy formed. There is no proper feedback from star formation, just the gas properties updated.

Definition at line 626 of file [recipe_mergers.c](#).

References [add_to_luminosities\(\)](#), [GALAXY::CentralGal](#), [GALAXY::ColdGas](#), [EjectionOn](#), [FeedbackRecipe](#), [Gal](#), [get_metallicity\(\)](#), [Halo](#), [GALAXY::HaloNr](#), [ListOutputSnaps](#), [GALAXY::MetalsColdGas](#), [GALAXY::MetalsEjectedMass](#), [GALAXY::MetalsHotGas](#), [GALAXY::MetalsStellarMass](#), [GALAXY::Pos](#), [RecycleFraction](#), [GALAXY::Rvir](#), [GALAXY::Sfr](#), [GALAXY::SnapNum](#), [halo_data::SnapNum](#), [StarBurstsInMajorMergersOn](#), [GALAXY::StellarMass](#), [Yield](#), and [ZZ](#).

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

4.31.2.6 double estimate_merging_time (int *halonr*, int *mother_halonr*, int *p*)

Binney & Tremaine 1987 - 7.26 merging time for satellites due to dynamical friction. After Delucia2007 *2, shown to agree with Kolchin2008 simulations in Delucia2010. This is set when a galaxy becomes a type 2 or being a type 1 $M_{\text{star}} > M_{\text{vir}}$. In DeLucia2007 they could only merge into a type 0, now (after guo2010) they can merge into a type 1.

Definition at line 197 of file [recipe_mergers.c](#).

References [halo_data::Descendant](#), [halo_data::FirstProgenitor](#), [G](#), [Gal](#), [get_virial_mass\(\)](#), [get_virial_radius\(\)](#), [get_virial_velocity\(\)](#), [Halo](#), [GALAXY::Len](#), [GALAXY::Pos](#), [halo_data::SnapNum](#), [GALAXY::StellarMass](#), [GALAXY::Type](#), and [ZZ](#).

Referenced by [update_type_1\(\)](#), and [update_type_2\(\)](#).

4.31.2.7 void grow_black_hole (int *halonr*, int *merger_centralgal*, double *mass_ratio*, double *dt*)

Grows black hole through accretion from cold gas during mergers, as in Kauffmann & Haehnelt (2000). I have made an addition here - the black hole can grow during minor mergers but at a reduced rate and i have included evolution with redshift

Definition at line 401 of file [recipe_mergers.c](#).

References [BlackHoleGrowthRate](#), [GALAXY::BlackHoleMass](#), [GALAXY::ColdGas](#), [Gal](#), [get_metallicity\(\)](#), [GALAXY::MetalsColdGas](#), and [GALAXY::Vvir](#).

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

4.31.2.8 void make_bulge_from_burst (int *p*)

Definition at line 711 of file [recipe_mergers.c](#).

References [GALAXY::BulgeMass](#), [GALAXY::dObsLum](#), [GALAXY::dObsLumBulge](#), [GALAXY::dObsYLum](#), [GALAXY::dObsYLumBulge](#), [Gal](#), [GALAXY::Lum](#), [GALAXY::LumBulge](#), [GALAXY::MetalsBulgeMass](#), [GALAXY::MetalsStellarMass](#), [GALAXY::ObsLum](#), [GALAXY::ObsLumBulge](#), [GALAXY::ObsYLum](#), [GALAXY::ObsYLumBulge](#), [GALAXY::Sfr](#), [GALAXY::SfrBulge](#), [GALAXY::StellarMass](#), [GALAXY::YLum](#), and [GALAXY::YLumBulge](#).

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

4.31.2.9 int set_merger_center (int *fofhalo*)

If MERGE01=1 (Guo2010), get id of central galaxy, since type 1's can have their merger clock started before they become type 2's if $M_{\text{star}} > M_{\text{vir}}$. Introduced for millennium 2, where they can have very small masses, still be followed and never merge. At this moment the centre is still not known, reason why this function is needed. Also, if the type 1 merges, all the type 2's associated with it will need to know the "new" central galaxy they are merging into.

Definition at line 101 of file [recipe_mergers.c](#).

References [halo_aux_data::FirstGalaxy](#), [halo_data::FirstProgenitor](#), [Halo](#), [HaloAux](#), [HaloGal](#), [halo_data::Len](#), [GALAXY::Len](#), [halo_data::NextHaloInFOFgroup](#), [halo_data::NextProgenitor](#), [halo_aux_data::NGalaxies](#), and [GALAXY::Type](#).

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

4.32 code/recipe_mergers.c

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <string.h>
00004 #include <math.h>
00005 #include <time.h>
00006
00007 #include "allvars.h"
00008 #include "proto.h"
00009
00010 /** @file recipe_mergers.c
00011 * @brief Calculates the merging time, the central galaxy (for type 1's),
00012 * adds galaxies together, calculates SF from bursts and grows
00013 * black holes.
00014 *
00015 *
00016 * <B>set_merger_center</B> - calculates the central galaxy for type 1's,
00017 * since if MERGE01=1 (makefile) type 1's can also merge. Therefore,
00018 * they need a merger central galaxy and will also have a merger clock
00019 * (needed for millennium two, since due to the high resolution, haloes
00020 * are very difficult to disrupt and orbit around forever).
00021 *
00022 *
00023 *
00024 * <B>estimate_merging_time</B> sets up a merger clock. Originally this
00025 * was done only for type 2 galaxies. The positions of galaxies in the
00026 * model are normally given by the position of their dark matter halo.
00027 * However, when galaxies become satellites, their dark matter haloes
00028 * are stripped by the central object to the point where there is none
00029 * left. At this point, the dark matter of the satellite becomes part of
00030 * the main halo, but the galaxy's position should continue to evolve
00031 * due to the dynamical friction force caused by the dark matter around
00032 * it. \n
00033 * The code does keep track of the position of the most bounded particle
00034 * when the satellite's halo was disrupted, but this is not used to track
00035 * the galaxy position. Instead a clock is set, giving the time left until
00036 * the satellite merges with the central galaxy. Before, this was only
00037 * done for type 2's (satellites that lost a halo). Guo2010 included the
00038 * MERGE01 option, that sets the clock also for type 1's (satellites with
00039 * a halo), since for the highest resolution millennium 2, their haloes
00040 * can be very small and orbit forever around the central companion.\n
00041 * This time is computed using the Chandrasekhar's formula for dynamical
00042 * friction, as in Binney & Tremaine 1987:
00043 *
00044 * \f$F_{\rm df} = -\frac{4\pi G^2 m_{\rm sat} \ln(\Lambda)}{v_{\rm rel}^2} \f$
00045 *
00046 *
00047 *
00048 * Which gives (B&T Eq. 7.26):
00049 *
00050 * \f$t_{\rm df} \approx 1.17 \frac{V_{\rm vir}^2 r_{\rm sat}}{(G M_{\rm sat} \ln(\Lambda))} \f$
00051 *
00052 *
00053 * that is afterwards multiplied by 2 (after Delucia2007 to fit the
00054 * data). When the merging time reaches zero, the satellite is assumed to
00055 * merge with the central galaxy.
00056 *
00057 *
00058 *
00059 * <B>deal_with_galaxy_merger</B> deals with the process, according to
00060 * the mass fraction of the merger. Major if
00061 * \f$M_{\rm sat}/M_{\rm central} > 0.3\f$ and minor otherwise. It calls
00062 * - add_galaxies_together - Add the cold and stellar phase of the merged
00063 * galaxy to the central one. Also form a bulge at the central galaxy
00064 * with the stars from the satellite in a minor merger if
00065 * BulgeFormationInMinorMergersOn=1 (Major mergers are dealt later).

```

```

00066 *      - Then calls grow_black_hole - Grows black hole through accretion from
00067 *      cold gas during mergers (due to the instabilities triggered), as in
00068 *      Kauffmann & Haehnelt (2000). This is commonly referred as the quasar
00069 *      mode, main responsible for the black hole growth. After Croton2006 this
00070 *      mode is active even in minor mergers:
00071 *      \f$\Delta m_{\rm BH,Q}=M_{\rm min} \frac{f_{\rm BH}(m_{\rm sat}/m_{\rm central})}{(1+(280/\mathit{km},s)^{-1}/V_{\rm vir})^2}.\f$ 
00072 *
00073 *
00074
00075 *      - Finally the burst of star formation due to the merger is treated.
00076 *      There are two options based on StarBurstRecipe input parameter:
00077 *          - If StarBurstRecipe = 0 (introduced by Kauffmann 1999), the
00078 *              burst only occurs for major mergers. In this case all the cold
00079 *              gas in the satellite and central galaxies is consumed to form
00080 *              stars and these are transferred to the bulge of the new galaxy
00081 *              formed.
00082 *          - If StarBurstRecipe = 1 (since Croton2006), the Somerville 2001
00083 *              model of bursts is used. The burst can happen for both major and
00084 *              minor mergers, with a fraction of the added cold gas from the
00085 *              satellite and central being consumed:
00086 *              \f$\dot{m}_{\star}^{\rm burst} = 0.56 \left(\frac{m_{\rm sat}}{m_{\rm central}}\right)^{0.7} m_{\rm gas}.\f$.
00087 *
00088 *
00089 *      SN Feedback from starformation is computed and the sizes of bulge
00090 *      and disk followed.
00091 *
00092 *      - When a major merger occurs, the disk of both merging galaxies is
00093 *      completely destroyed to form a bulge. In either type of mergers, the
00094 *      bulge size is updated using Eq. 33 in Guo2010:
00095 *      \f$C \frac{GM^2_{\rm new,bulge}}{R_{\rm new,bulge}} = C \frac{GM^2_{\rm 1}}{R_1} + C \frac{GM^2_{\rm 2}}{R_2} + \alpha_{\rm inter} \frac{GM_{\rm 1}M_{\rm 2}}{R_1 + R_2}\f$*
00096 *
00097 *
00098
00099 /** @brief Calculates the central galaxies for type 1's (needed if MERGE01=1). */

00100
00101 int set_merger_center(int fofhalo)
00102 {
00103     /** @brief If MERGE01=1 (Guo2010), get id of central galaxy, since type 1's
00104     *      can have their merger clock started before they become type 2's
00105     *      if M_star>M_vir. Introduced for millennium 2, where they can have
00106     *      very small masses, still be followed and never merge. At this
00107     *      moment the centre is still not known, reason why this function is
00108     *      needed. Also, if the type 1 merges, all the type 2's associated with
00109     *      it will need to know the "new" central galaxy they are merging into.
00110 */
00111     int ngal, prog, i, j, first_occupied, type, halonr ;
00112
00113     double satmvir, satrvir, lenmax;
00114     ngal = 0;
00115
00116     halonr=fofhalo;
00117
00118     while(halonr >= 0)
00119     {
00120         lenmax = 0;
00121         first_occupied = Halo[halonr].FirstProgenitor;
00122         prog = Halo[halonr].FirstProgenitor;
00123
00124         if(prog >=0)
00125         {
00126             if(HaloAux[prog].NGalaxies == 0)
00127                 while(prog >= 0)
00128                 {
00129                     for(i = 0; i < HaloAux[prog].NGalaxies; i++)

```

```

00130
00131     {
00132         if(HaloGal[HaloAux[prog].FirstGalaxy +
00133             i].Type == 0
00134             || HaloGal[HaloAux[prog]
00135                 ].FirstGalaxy + i].Type == 1)
00136             {
00137                 Len;
00138                 if(Halo[prog].Len > lenmax)
00139                     {
00140                         lenmax = Halo[prog].
00141                         first_occupied = prog;
00142                     }
00143             }
00144             prog = Halo[prog].NextProgenitor;
00145
00146
00147
00148     while(prog >= 0)
00149     {
00150         for(i = 0; i < HaloAux[prog].NGalaxies; i++)
00151         {
00152             type = HaloGal[HaloAux[prog].FirstGalaxy + i].Type;
00153             /* this deals with the central galaxies of subhalos */
00154             if(type == 0 || type == 1)
00155             {
00156                 if(prog == first_occupied)
00157                     {
00158                         if(halonr == Halo[halonr].FirstHaloInFO
00159                             Fgroup)
00160                             {
00161                                 if(halonr == Halo[halonr].FirstHaloInFO
00162                                     Fgroup)
00163                                     {
00164                                         if(halonr == Halo[halonr].FirstHaloInFO
00165                                             Fgroup)
00166                                             {
00167                                                 if(halonr == Halo[halonr].FirstHaloInFO
00168                                                     Fgroup)
00169                                                     {
00170                                                         ngal++;
00171
00172                                         }
00173
00174                                         prog = Halo[prog].NextProgenitor;
00175
00176                                     }
00177
00178             if(ngal == 0)
00179             {
00180                 if(Halo[halonr].FirstHaloInFOFgroup == halonr)
00181                     {
00182                         if(halonr == Halo[halonr].FirstHaloInFO
00183                             Fgroup)
00184                             {
00185                                 if(halonr == Halo[halonr].FirstHaloInFO
00186                                     Fgroup)
00187                                     {
00188                                         halonr = Halo[halonr].NextHaloInFOFgroup;
00189
00190                                         }
00191                                         printf("wrong in finding the central fof %d ngal %d\n",
00192                                             fofhalo,ngal);
00192                                         exit(0);

```

```

00193 }
00194
00195 /** @brief Calculates the merging time whenever a galaxy becomes a satellite*/
00196
00197 double estimate_merging_time(int halonr, int mother_halonr,int p)
00198 {
00199     int central_halonr,central_curr;
00200     double coulomb, mergtime, SatelliteMass, SatelliteRadius, MotherHaloRvir;
00201
00202     /** @brief Binney & Tremaine 1987 - 7.26 merging time for satellites due to
00203     *      dynamical friction. After Delucia2007 *2, shown to agree with
00204     *      Kolchin2008 simulations in Delucia2010. This is set when a galaxy
00205     *      becomes a type 2 or being a type 1 \f$M_{\rm star}\f$>M_{\rm vir}\f$.
00206     *
00207     *      In DeLucia2007 they could only merge into a type 0, now (after
00208     *      guo2010) they can merge into a type 1. */
00209
00210     /* recipe updated for more accurate merging time (see BT eq 7.26),
00211        now satellite radius at previous timestep is included */
00212     central_halonr = Halo[Halo[halonr].Descendant].FirstProgenitor;
00213     if (Gal[p].Type == 1)
00214         central_halonr=mother_halonr;
00215     if(central_halonr == halonr)
00216     {
00217         printf("can't be...!\n");
00218         exit(1);
00219     }
00220
00221
00222     coulomb = log(Halo[mother_halonr].Len / ((double) Halo[halonr].Len) + 1);
00224
00225     /* should include stellar+cold gas in SatelliteMass! */
00226     SatelliteMass = get_virial_mass(halonr,p)+Gal[p].StellarMass;
00227
00228     SatelliteRadius = sqrt(pow(Halo[central_halonr].Pos[0] - Halo[halonr].Pos[0], 2
00229 .0) +
00230                         pow(Halo[central_halonr].Pos[1] - Halo[halonr].Pos[1], 2
00231 .0) +
00232                         pow(Halo[central_halonr].Pos[2] - Halo[halonr].Pos[2], 2
00233 .0));
00234
00235     /* convert to physical length */
00236     SatelliteRadius /= (1 + ZZ[Halo[halonr].SnapNum]);
00237
00238     MotherHaloRvir = get_virial_radius(mother_halonr,p);
00239     if(SatelliteRadius > MotherHaloRvir)
00240         SatelliteRadius = MotherHaloRvir;
00241
00242     if(SatelliteMass > 0.0)
00243     {
00244         mergtime =
00245             1.17 * SatelliteRadius * SatelliteRadius * get_virial_velocity(mother_halonr,p) /
00246             (coulomb * G *
00247             SatelliteMass); // Binney & Tremaine Eq.7.26
00248
00249     /* change introduced by Delucia2007 to fit observations */
00250     mergtime = 2.*mergtime;
00251
00252     else
00253         mergtime = -99999.9;
00254
00255
00256     return mergtime;

```

```

00254
00255 }
00256
00257
00258 /** @brief Deals with all the physics triggered by mergers */
00259
00260 void deal_with_galaxy_merger(int p, int merger_centralgal, int centralgal, double
00261           time, double delta_t,
00262           int halonr)
00263 {
00264 /** @brief Deals with the physics triggered by mergers, according to the mass
00265  * fraction of the merger \f$ M_{\rm sat} / M_{\rm central} > < 0.3 \f$.
00266  * Add the cold and stellar phase of the satellite galaxy to the central
00267  * one, form a bulge at the central galaxy with the stars from the
00268  * satellite in a minor merger if BulgeFormationInMinorMergersOn=1.
00269  * Grows black hole through accretion from cold gas "quasar mode". If
00270  * StarBurstRecipe = 0, all the cold gas is consumed to form stars and
00271  * these are transferred to the bulge of the new galaxy formed (only
00272  * for major mergers). If StarBurstRecipe = 1, the Somerville 2001 model
00273  * of bursts is used, SN Feedback from starformation is computed and
00274  * the sizes of bulge and disk followed. When a major merger occurs,
00275  * the disk of both merging galaxies is completely destroyed to form
00276  * a bulge. */
00277
00278
00279     double mi, ma, mass_ratio, Mcstar, Mcgas, Mcbulge;
00280     double frac;
00281 #ifdef GALAXYTREE
00282     int q;
00283
00284     q = Gal[merger_centralgal].FirstProgGal;
00285     if(q >= 0)
00286     {
00287         while(HaloGal[q].NextProgGal >= 0)
00288             q = HaloGal[q].NextProgGal;
00289
00290         HaloGal[q].NextProgGal = Gal[p].FirstProgGal;
00291     }
00292
00293     q = HaloGal[q].NextProgGal;
00294     HaloGal[q].MergeOn = 2;
00295     if (Gal[p].Type == 1)
00296         HaloGal[q].MergeOn = 3;
00297 #endif
00298
00299
00300 /* flag galaxy as finished */
00301 Gal[p].Type = 3;
00302
00303 /* calculate mass ratio of merging galaxies */
00304 if(Gal[p].StellarMass + Gal[p].ColdGas <
00305     Gal[merger_centralgal].StellarMass + Gal[merger_centralgal].ColdGas)
00306 {
00307     mi = Gal[p].StellarMass + Gal[p].ColdGas;
00308     ma = Gal[merger_centralgal].StellarMass + Gal[merger_centralgal].ColdGas;
00309 }
00310 else
00311 {
00312     mi = Gal[merger_centralgal].StellarMass + Gal[merger_centralgal].ColdGas;
00313     ma = Gal[p].StellarMass + Gal[p].ColdGas;
00314 }
00315
00316 if(ma > 0)
00317     mass_ratio = mi / ma;
00318 else
00319     mass_ratio = 1.0;

```

```

00320
00321
00322 /* record the gas and stellar component mass of merger central galaxy
00323 * before merger */
00324 Mcstar=Gal[merger_centralgal].StellarMass;
00325 Mcbulge=Gal[merger_centralgal].BulgeMass;
00326 Mcgas=Gal[merger_centralgal].ColdGas;
00327
00328 /* Add the cold and stellar phase of the merged galaxy to the central one.
00329 Also form a bulge if BulgeFormationInMinorMergersOn is set on, but only
00330 with the stars from the satellite. In a major merger (dealt at the burst
00331 make_bulge_from_burst) all the stars in the central and satellite end up
00332 in the central bulge.*/
00333 add_galaxies_together(merger_centralgal, p);
00334
00335
00336 /* grow black hole through accretion from cold disk during mergers, as in
00337 * Kauffmann & Haehnelt (2000) + minor mergers - Quasar Mode */
00338 if(AGNrecipeOn > 0)
00339 grow_black_hole(halonr, merger_centralgal, mass_ratio, deltaT);
00340
00341
00342 /* StarBurstRecipe=0 --> in a major merger it is assumed that a burst converts
00343 all the cold gas into stars
00344 StarBurstRecipe=1 --> similar to Somerville et al. 2001 */  

00345 if(StarBurstRecipe == 0 && mass_ratio > ThreshMajorMerger)
00346 {
00347     /* all the cold gas available from the two merging galaxies (now in
00348     * the central one) is used up in a single star formation episode.
00349     * Stellar mass and cold gas are updated as well as metals in different
00350     * components from the yield. Magnitudes updated. No feedback. Only for
00351     * major mergers*/
00352     do_major_merger_starburst(merger_centralgal, centralgal, time, deltaT, halo
00353 nr);
00354     /* The disk is completely destroyed and all the stars moved to the bulge */

00355     make_bulge_from_burst(merger_centralgal);
00356 }
00357 else if(StarBurstRecipe == 1)
00358 {
00359     /* Starburst as in Somerville 2001, with feedback computed inside. */
00360     frac=collisional_starburst_recipe(mass_ratio, merger_centralgal, centralgal
00361 , time, deltaT, halonr);
00362     bulgesize_from_merger(mass_ratio, merger_centralgal, p, Mcstar, Mcbulge, Mcgas, f
00363 rac);
00364     if(mass_ratio > ThreshMajorMerger)
00365         make_bulge_from_burst(merger_centralgal);
00366 }
00367
00368 /* If we are in the presence of a minor merger, check disk stability (the disk
00369 * is completely destroyed in major mergers)*/
00370 if(TrackDiskInstability)
00371     if(mass_ratio < ThreshMajorMerger && Gal[merger_centralgal].StellarMass > 0.0
00372 )
00373     check_disk_instability(merger_centralgal);
00374
00375 /* Not supported option to shrink bulge sizes in gas rich mergers */
00376 #ifdef SHRINKINRICHMERGER
00377 if (Mcgas+Mcstar+Gal[p].ColdGas+Gal[p].StellarMass > 1.e-8 )
00378 {
00379     Gal[merger_centralgal].BulgeSize /= 1+pow((Mcgas+Gal[p].ColdGas)/(Mcgas+Mc
00380 star+Gal[p].ColdGas+Gal[p].StellarMass)/0.15,1.);
00381
00382     if (Gal[merger_centralgal].BulgeSize < 1.e-8)

```

```

00380         Gal[merger_centralgal].BulgeSize = 1.e-8;
00381     }
00382 #endiff
00383
00384
00385     if ((Gal[merger_centralgal].BulgeMass > 0.0 && Gal[merger_centralgal].BulgeSize == 0.0) || (Gal[merger_centralgal].BulgeMass == 0.0 && Gal[merger_centralgal].BulgeSize > 0.0))
00386     {
00387         printf("central: stellarmass %f, bulgemass %f, bulgesize %f, coldgas %f,gas
00388 disk %f,stellardisk %f \n",Gal[merger_centralgal].StellarMass, Gal[merger_centralgal].BulgeMass, Gal[merger_centralgal].BulgeSize, Gal[merger_centralgal].ColdGas,
00389 Gal[merger_centralgal].GasDiskRadius, Gal[merger_centralgal].StellarDiskRadius);
00390         exit(0);
00391     }
00392     if (DiskRadiusMethod == 2)
00393     {
00394         get_gas_disk_radius(merger_centralgal);
00395         get_stellar_disk_radius(merger_centralgal);
00396     }
00397
00398 /** @brief Grows black holes, through accretion from cold gas during mergers,
00399 *           as in Kauffmann & Haehnelt (2000) - Quasar Mode. */
00400
00401 void grow_black_hole(int halonr, int merger_centralgal, double mass_ratio, double
00402 dt)
00403 {
00404     double BHaccrete, metallicity;
00405
00406     /** @brief Grows black hole through accretion from cold gas during mergers,
00407      *           as in Kauffmann & Haehnelt (2000). I have made an addition here -
00408      *           the black hole can grow during minor mergers but at a reduced rate
00409      *           and i have included evolution with redshift */
00410
00411     if(Gal[merger_centralgal].ColdGas > 0.0)
00412     {
00413         BHaccrete = BlackHoleGrowthRate * mass_ratio
00414             / (1.0 + pow(280.0 / Gal[merger_centralgal].Vvir, 2.0)) * Gal[merger_ce
00415 ntralgal].ColdGas;
00416         /* redshift dependent accretion, not published */
00417         /* BHaccrete = BlackHoleGrowthRate * (1.0 + ZZ[Halo[halonr].SnapNum]) * mas
00418 s_ratio */
00419
00420         /* cannot accrete more gas than is available! */
00421         if(BHaccrete > Gal[merger_centralgal].ColdGas)
00422             BHaccrete = Gal[merger_centralgal].ColdGas;
00423
00424         metallicity = get_metallicity(Gal[merger_centralgal].ColdGas, Gal[merger_ce
00425 ntralgal].MetalsColdGas);
00426         Gal[merger_centralgal].BlackHoleMass += BHaccrete;
00427         Gal[merger_centralgal].ColdGas -= BHaccrete;
00428         Gal[merger_centralgal].MetalsColdGas -= metallicity * BHaccrete;
00429
00430
00431 /** @brief Adds all the components of the satellite galaxy into its
00432 *           central companion. */
00433
00434 void add_galaxies_together(int t, int p)
00435 {
00436     /** @brief All the components of the satellite galaxy are added to the
00437      *           correspondent component of the central galaxy. Cold gas spin

```

```

00438      *           is updated and a bulge is formed at the central galaxy, with
00439      *           the stars of the satellite if  BulgeFormationInMinorMergersOn=
00440      1.*/
00440  int outputbin, j;
00441  float tspin[3],tmass,pmass;
00442
00443 #ifdef UseFullSfr
00444  int snap;
00445 #endif
00446
00447 /* t central, p satellite */
00448
00449 /* angular momentum transfer between gas*/
00450 tmass= Gal[t].ColdGas;
00451 pmass= Gal[p].ColdGas;
00452
00453 Gal[t].ColdGas += Gal[p].ColdGas;
00454 Gal[t].MetalsColdGas += Gal[p].MetalsColdGas;
00455 Gal[t].StellarMass += Gal[p].StellarMass;
00456 Gal[t].MetalsStellarMass += Gal[p].MetalsStellarMass;
00457 Gal[t].StarMerge +=Gal[p].StarMerge;
00458
00459 Gal[t].MergeSat +=Gal[p].StellarMass;
00460
00461 /* update the ICL */
00462 Gal[t].ICM += Gal[p].ICM;
00463 Gal[t].MetalsICM +=Gal[p].MetalsICM;
00464
00465
00466
00467 /*update the gas spin*/
00468 for(j=0;j<3;j++)
00469   tspin[j]=Gal[t].GasSpin[j]*tmass+Gal[t].HaloSpin[j]*pmass;
00470   if (Gal[t].ColdGas != 0)
00471     for (j=0;j<3;j++)
00472       Gal[t].GasSpin[j]=tspin[j]/(Gal[t].ColdGas);
00473
00474 /* Type 1's can now merge, so they can still have hot and ejected gas */
00475 Gal[t].HotGas += Gal[p].HotGas;
00476 Gal[t].MetalsHotGas += Gal[p].MetalsHotGas;
00477 Gal[t].EjectedMass += Gal[p].EjectedMass;
00478 Gal[t].MetalsEjectedMass += Gal[p].MetalsEjectedMass;
00479 Gal[t].BlackHoleMass += Gal[p].BlackHoleMass;
00480
00481
00482 #ifdef UseFullSfr
00483   for(snap = 0; snap < MAXSNAPS; snap++)
00484     Gal[t].Sfr[snap] += Gal[p].Sfr[snap];
00485 #else
00486 #ifdef SAVE_MEMORY
00487   Gal[t].Sfr += Gal[p].Sfr;
00488 #else
00489   for(outputbin = 0; outputbin < NOUT; outputbin++)
00490     Gal[t].Sfr[outputbin] += Gal[p].Sfr[outputbin];
00491 #endif
00492 #endif
00493
00494 /* If BulgeFormationInMinorMergersOn=1, the stars of the satellite galaxy
00495 * in a minor merger are transferred into the bulge of the central galaxy.
00496 * Note that this is different from the major merger treatment, where all
00497 * the stars in both galaxies are transferred to the bulge. */
00498 if(BulgeFormationInMinorMergersOn)
00499 {
00500   Gal[t].BulgeMass += Gal[p].StellarMass;
00501   Gal[t].MetalsBulgeMass += Gal[p].MetalsStellarMass;
00502
00503 #ifdef UseFullSfr

```

```

00504     for(snap = 0; snap < MAXSNAPS; snap++)
00505         Gal[t].SfrBulge[snap] += Gal[p].Sfr[snap];
00506 #else
00507 #ifdef SAVE_MEMORY
00508     Gal[t].SfrBulge += Gal[p].Sfr;
00509 #else
00510     for(outputbin = 0; outputbin < NOUT; outputbin++)
00511         Gal[t].SfrBulge[outputbin] += Gal[p].Sfr[outputbin];
00512 #endif
00513 #endif
00514 }
00515
00516 /* Add the luminosities of the satellite and central galaxy */
00517 #ifdef OUTPUT_REST_MAGS
00518     for(outputbin = 0; outputbin < NOUT; outputbin++)
00519     {
00520
00521     for(j = 0; j < NMAG; j++)
00522     {
00523         Gal[t].Lum[j][outputbin] += Gal[p].Lum[j][outputbin];
00524         Gal[t].YLum[j][outputbin] += Gal[p].YLum[j][outputbin];
00525 #ifdef ICL
00526
00527         Gal[t].ICLLum[j][outputbin] += Gal[p].ICLLum[j][outputbin];
00528 #endif
00529     }
00530     if(BulgeFormationInMinorMergersOn)
00531     {
00532         for(j = 0; j < NMAG; j++)
00533         {
00534             Gal[t].LumBulge[j][outputbin] += Gal[p].Lum[j][outputbin];
00535             Gal[t].YLumBulge[j][outputbin] += Gal[p].YLum[j][outputbin];
00536         }
00537     }
00538     else
00539     {
00540         for(j = 0; j < NMAG; j++)
00541         {
00542             Gal[t].LumBulge[j][outputbin] += Gal[p].LumBulge[j][outputbin];
00543             Gal[t].YLumBulge[j][outputbin] += Gal[p].YLumBulge[j][outputbin];
00544         }
00545     }
00546
00547     Gal[t].MassWeightAge[outputbin] += Gal[p].MassWeightAge[outputbin];
00548 }
00549
00550 #endif
00551 #ifdef COMPUTE_OBS_MAGS
00552     for(outputbin = 0; outputbin < NOUT; outputbin++)
00553     {
00554         for(j = 0; j < NMAG; j++)
00555         {
00556             Gal[t].ObsLum[j][outputbin] += Gal[p].ObsLum[j][outputbin];
00557             Gal[t].ObsYLum[j][outputbin] += Gal[p].ObsYLum[j][outputbin];
00558 #ifdef ICL
00559             Gal[t].ObsICL[j][outputbin] += Gal[p].ObsICL[j][outputbin];
00560 #endif
00561
00562 #ifdef OUTPUT_MOMAF_INPUTS
00563             Gal[t].dObsLum[j][outputbin] += Gal[p].dObsLum[j][outputbin];
00564             Gal[t].dObsYLum[j][outputbin] += Gal[p].dObsYLum[j][outputbin];
00565 #endif
00566         }
00567         if(BulgeFormationInMinorMergersOn)
00568         {
00569             for(j = 0; j < NMAG; j++)
00570             {

```

```

00571             Gal[t].ObsLumBulge[j][outputbin] += Gal[p].ObsLum[j][outputbin];
00572             Gal[t].ObsYLumBulge[j][outputbin] += Gal[p].ObsYLum[j][outputbin];

00573 #ifdef OUTPUT_MOMAF_INPUTS
00574             Gal[t].dObsLumBulge[j][outputbin] += Gal[p].dObsLum[j][outputbin];
00575             Gal[t].dObsYLumBulge[j][outputbin] += Gal[p].dObsYLum[j][outputbin]
00576 ;
00577         }
00578     }
00579     else
00580     {
00581         for(j = 0; j < NMAG; j++)
00582         {
00583             Gal[t].ObsLumBulge[j][outputbin] += Gal[p].ObsLumBulge[j][outputb
00584             in];
00585             Gal[t].ObsYLumBulge[j][outputbin] += Gal[p].ObsYLumBulge[j][output
00586             bin];
00587 #ifdef OUTPUT_MOMAF_INPUTS
00588             Gal[t].dObsLumBulge[j][outputbin] += Gal[p].dObsLumBulge[j][output
00589             bin];
00590             Gal[t].dObsYLumBulge[j][outputbin] += Gal[p].dObsYLumBulge[j][output
00591             bin];
00592 #endif
00593         }
00594     }
00595 /*      for(outputbin = 0; outputbin < NOUT; outputbin++) */
00596 /*      { */
00597 /*          for(j = 0; j < NMAG; j++) */
00598 /*          { */
00599 /*              Gal[t].Lum[j][outputbin] += Gal[p].Lum[j][outputbin]; */
00600 /*          } */
00601 /*      if(BulgeFormationInMinorMergersOn) */
00602 /*      { */
00603 /*          for(j = 0; j < NMAG; j++) */
00604 /*          { */
00605 /*              Gal[t].LumBulge[j][outputbin] += Gal[p].Lum[j][outputbin]; */
00606 /*          } */
00607 /*      } */
00608 /*      else */
00609 /*      { */
00610 /*          for(j = 0; j < NMAG; j++) */
00611 /*          { */
00612 /*              Gal[t].LumBulge[j][outputbin] += Gal[p].LumBulge[j][outputbin];
00613 /*          } */
00614 /*      */
00615 /*      } */
00616 /*      */
00617
00618 }
00619
00620
00621
00622 /**@brief Merger burst recipe used before Croton2006 - in a major merger all
00623 *       the cold gas from the central and satellite galaxies is consumed
00624 *       to form a bulge. */
00625
00626 void do_major_merger_starburst(int p, int centralgal, double time, double deltaT,
00627     int halonr)
00628 /** @brief If StarBurstRecipe = 0 (introduced by Kauffmann 1999), the burst

```

```

00629 *      only occurs for major mergers. In this case all the cold gas in
00630 *      the satellite and central galaxies is consumed to form stars and
00631 *      these are transferred to the bulge of the new galaxy formed. There
00632 *      is no proper feedback from star formation, just the gas properties
00633 *      updated. */
00634
00635     double mstars, metallicity;
00636     double dis;
00637     int merger_centralgal;
00638 #ifndef UseFullSfr
00639     int outputbin;
00640 #endif
00641
00642     /* consume all gas in a starburst */
00643     if(StarBurstsInMajorMergersOn)
00644     {
00645         mstars = Gal[p].ColdGas;
00646         metallicity = get_metallicity(Gal[p].ColdGas, Gal[p].MetalsColdGas);
00647
00648         /* Galaxies have already merged at this point, so all the cold gas
00649             in the new galaxy formed is consumed */
00650         Gal[p].StellarMass += Gal[p].ColdGas * (1 - RecycleFraction);
00651         Gal[p].MetalsStellarMass += Gal[p].MetalsColdGas * (1 - RecycleFraction);
00652
00653         Gal[p].ColdGas -= Gal[p].ColdGas * (1 - RecycleFraction);
00654         Gal[p].MetalsColdGas -= Gal[p].MetalsColdGas * (1 - RecycleFraction);
00655         merger_centralgal = Gal[p].CentralGal;
00656
00657         dis=
00658             sqrt(pow(Gal[centralgal].Pos[0] - Gal[merger_centralgal].Pos[0], 2.0) +
00659                 pow(Gal[centralgal].Pos[1] - Gal[merger_centralgal].Pos[1], 2.0) +
00660                 pow(Gal[centralgal].Pos[2] - Gal[merger_centralgal].Pos[2], 2.0))/(1
+ ZZ[Halo[Gal[centralgal].Halonr].SnapNum]);
00661
00662     /* There is no feedback computed for this burst mode, only the yield
00663         * of metals from star formation is added to the different gas
00664         * components. No mass is transferred. */
00665     if(FeedbackRecipe == 0) /* As in De Lucia et al. (2004) */
00666     {
00667         if(EjectionOn == 1)
00668             Gal[centralgal].MetalsEjectedMass += Yield * mstars;
00669         else
00670             Gal[centralgal].MetalsHotGas += Yield * mstars;
00671     }
00672     /*TODO shouldn't this be == 2? that is Guo2010 version that allows gas
00673         * to go into a type 1 if the type 2 is outside R_vir */
00674     /*in bursts the metals go to the hotgas - this is not used.*/
00675     else if(FeedbackRecipe == 1)
00676         if (dis < Gal[centralgal].Rvir)
00677             Gal[centralgal].MetalsHotGas += Yield * mstars;
00678         else
00679             Gal[Gal[p].CentralGal].MetalsHotGas += Yield * mstars;
00680
00681     if(FeedbackRecipe == 2)
00682         Gal[Gal[p].CentralGal].MetalsHotGas += Yield * mstars;
00683
00684     /* update the star formation rate */
00685 #ifdef UseFullSfr
00686     Gal[p].Sfr[Halonr].SnapNum] += mstars / deltaT;
00687 #else
00688 #ifdef SAVE_MEMORY
00689     Gal[p].Sfr += mstars / deltaT;
00690 #else
00691     for(outputbin = 0; outputbin < NOUT; outputbin++)
00692     {
00693         if(Halo[halonr].SnapNum == ListOutputSaps[outputbin])
00694         {

```

```

00695             Gal[p].Sfr[outputbin] += mstars / deltaT;
00696             break;
00697         }
00698     }
00699 #endif
00700 #endif
00701     /* Update the luminosities */
00702     add_to_luminosities(p, mstars, time, metallicity);
00703 }
00704
00705 }
00706
00707
00708 /** @brief In a major merger, both disks are destroyed and all the mass transferred
00709 *      to the bulge. */
00710
00711 void make_bulge_from_burst(int p)
00712 {
00713     int outputbin, j;
00714
00715 #ifdef UseFullSfr
00716     int snap;
00717 #endif
00718
00719     /* generate bulge */
00720     Gal[p].BulgeMass = Gal[p].StellarMass;
00721     Gal[p].MetalsBulgeMass = Gal[p].MetalsStellarMass;
00722
00723     /* update the star formation rate */
00724 #ifdef UseFullSfr
00725     for(snap = 0; snap < MAXSNAPS; snap++)
00726         Gal[p].SfrBulge[snap] += Gal[p].Sfr[snap];
00727 #else
00728 #ifdef SAVE_MEMORY
00729     Gal[p].SfrBulge = Gal[p].Sfr;
00730 #else
00731     for(outputbin = 0; outputbin < NOUT; outputbin++)
00732         Gal[p].SfrBulge[outputbin] += Gal[p].Sfr[outputbin];
00733 #endif
00734 #endif
00735
00736 #ifdef OUTPUT_REST_MAGS
00737     for(outputbin = 0; outputbin < NOUT; outputbin++)
00738     {
00739         for(j = 0; j < NMAG; j++)
00740         {
00741             Gal[p].LumBulge[j][outputbin] = Gal[p].Lum[j][outputbin];
00742             Gal[p].YLumBulge[j][outputbin] = Gal[p].YLum[j][outputbin];
00743         }
00744     }
00745 #endif
00746 #ifdef COMPUTE_OBS_MAGS
00747     for(outputbin = 0; outputbin < NOUT; outputbin++)
00748     {
00749         for(j = 0; j < NMAG; j++)
00750         {
00751             Gal[p].ObsLumBulge[j][outputbin] = Gal[p].ObsLum[j][outputbin];
00752             Gal[p].ObsYLumBulge[j][outputbin] = Gal[p].ObsYLum[j][outputbin];
00753 #ifdef OUTPUT_MOMAF_INPUTS
00754             Gal[p].dObsLumBulge[j][outputbin] = Gal[p].dObsLum[j][outputbin];
00755             Gal[p].dObsYLumBulge[j][outputbin] = Gal[p].dObsYLum[j][outputbin];
00756 #endif
00757         }
00758     }
00759 #endif
00760 }
```

```

00761
00762
00763 /** @brief Merger burst recipe from Somerville 2001 (used after Croton2006) */
00764
00765 double collisional_starburst_recipe(double mass_ratio, int merger_centralgal, int
00766           centralgal,
00767           double time, double deltaT, int halonr)
00768 {
00769     /** @brief If StarBurstRecipe = 1 (since Croton2006), the Somerville 2001
00770     *          model of bursts is used. The burst can happen for both major
00771     *          and minor mergers, with a fraction of the added cold gas from
00772     *          the satellite and central being consumed. SN Feedback from
00773     *          starformation is computed and the sizes of bulge and disk
00774     *          followed (not done for the other burst mode).*/
00775     double mstars, vesc, reheated_mass, ejected_mass, fac, metallicity, metallicity
00776           SF, CentralVvir, eburst,ejected_sat,MergeCentralVvir,dis;
00777     double Ggas,refrac;
00778 #ifndef UseFullSfr
00779     int outputbin;
00780 #endif
00781
00782
00783 /* This is the major and minor merger starburst recipe of Somerville 2001.
00784 * The coefficients in eburst are taken from TJ Cox's PhD thesis and should
00785 * be more accurate than previous. */
00786
00787 Ggas=Gal[merger_centralgal].ColdGas;
00788
00789 /* the bursting fraction given the mass ratio */
00790 /* m_dot = 0.56*(m_sat/m_centeral)^0.7*m_gas */
00791 eburst = 0.56 * pow(mass_ratio, 0.7);
00792 mstars = eburst * Gal[merger_centralgal].ColdGas;
00793 if(mstars < 0.0)
00794     mstars = 0.0;
00795
00796 /* this bursting results in SN feedback on the cold/hot gas
00797 * TODO this is the same code used in star_formation_and_feedback.c
00798 *      and should be merged if possible. */
00799 CentralVvir = Gal[centralgal].Vvir;
00800 MergeCentralVvir = Gal[merger_centralgal].Vvir;
00801
00802 if(FeedbackRecipe == 0)
00803 {
00804     vesc = Gal[merger_centralgal].Vvir;
00805     reheated_mass = (4 / 3.0) * FeedbackEpsilon * mstars * EtaSNcode *
00806         EnergySNcode / (vesc * vesc);
00807 }
00808 else if(FeedbackRecipe == 1)
00809 {
00810     reheated_mass = FeedbackReheatingEpsilon * mstars;
00811 }
00812 else
00813     reheated_mass = 0.0;
00814 if(FeedbackRecipe == 2)
00815 {
00816     if (Gal[merger_centralgal].Type == 0)
00817         reheated_mass = FeedbackReheatingEpsilon * mstars*(.5+1./pow(Gal[merger_c
00818           entralgal].Vmax/ReheatPreVelocity, ReheatSlope));
00819     else
00820         reheated_mass = FeedbackReheatingEpsilon * mstars*(.5+1./pow(Gal[merger_c
00821           entralgal].InfallVmax/ReheatPreVelocity, ReheatSlope));
00822 }
```

```

00822     if(reheated_mass < 0.0)
00823         reheated_mass = 0.0;
00825
00826
00827     /* determin how much of the energy of SN feedback is used to reheat the gas
00828      * compared to the part used to eject gas*/
00829
00830
00831     refrac = Energy_in_Reheat(merger_centralgal);
00832
00833     dis=
00834         sqrt(pow(Gal[centralgal].Pos[0] - Gal[merger_centralgal].Pos[0], 2.0) +
00835             pow(Gal[centralgal].Pos[1] - Gal[merger_centralgal].Pos[1], 2.0) +
00836             pow(Gal[centralgal].Pos[2] - Gal[merger_centralgal].Pos[2], 2.0))/(1 +
00837             ZZ[Halo[Gal[centralgal].HaloNr].SnapNum]);
00838
00839     if(FeedbackRecipe == 2 || dis > Gal[centralgal].Rvir)
00840     {
00841         if (reheated_mass * MergeCentralVvir * MergeCentralVvir > refrac * mstars *
00842             EtaSNcode * EnergySNcode)
00843             reheated_mass = refrac * mstars * EtaSNcode * EnergySNcode / MergeCentral
00844             Vvir / MergeCentralVvir;
00845     }
00846     else
00847     {
00848         if (reheated_mass * CentralVvir * CentralVvir > refrac * mstars *
00849             EtaSNcode * EnergySNcode)
00850             reheated_mass = refrac * mstars * EtaSNcode * EnergySNcode / CentralVvir
00851             / CentralVvir;
00852     }
00853
00854     /* cant use more cold gas than is available! so balance SF and feedback */
00855     if((mstars + reheated_mass) > Gal[merger_centralgal].ColdGas)
00856     {
00857         fac = Gal[merger_centralgal].ColdGas / (mstars + reheated_mass);
00858         mstars *= fac;
00859         reheated_mass *= fac;
00860     }
00861     Gal[merger_centralgal].StarMerge +=mstars*0.57;
00862
00863     /* initialize ejected mass from satellite galaxies*/
00864     ejected_sat=0.0;
00865
00866     /* determine ejection (for FeedbackRecipe 1 we need to know the
00867      * correct stars and reheating) */
00868     if(FeedbackRecipe == 2)
00869     {
00870         if (merger_centralgal == centralgal)
00871             {
00872                 ejected_mass =
00873                     (FeedbackEjectionEfficiency* (EtaSNcode * EnergySNcode) *mstars *min(
00874                         1./FeedbackEjectionEfficiency,.5+1/pow(Gal[centralgal].Vmax/EjectPreVelocity,
00875                         EjectSlope)) -
00876                         reheated_mass*CentralVvir*CentralVvir) /(CentralVvir*CentralVvir);
00877
00878                 ejected_sat=0.0;
00879             }
00880         else
00881             {
00882                 if (dis < Gal[centralgal].Rvir)
00883                 {
00884                     ejected_sat=

```

```

00882          (FeedbackEjectionEfficiency * (EtaSNcode * EnergySNcode) *mstars
00883      * min(1./FeedbackEjectionEfficiency,.5+1./pow(Gal[merger_centralgal].InfallVmax/
00884      EjectPreVelocity,EjectSlope)) -
00885          reheated_mass*MergeCentralVvir*MergeCentralVvir)/(MergeCentralV
00886      vir*MergeCentralVvir);
00887
00888
00889      }
00890      else
00891      {
00892          ejected_sat=
00893          (FeedbackEjectionEfficiency * (EtaSNcode * EnergySNcode) *mstars
00894      * min(1./FeedbackEjectionEfficiency,.5+1./pow(Gal[merger_centralgal].InfallVmax/
00895      EjectPreVelocity,EjectSlope)) -
00896          reheated_mass*MergeCentralVvir*MergeCentralVvir)/(MergeCentralV
00897      vir*MergeCentralVvir);
00898
00899
00900      }
00901
00902      }
00903  else
00904  {
00905      if(FeedbackRecipe == 0 && merger_centralgal == centralgal && EjectionOn ==
1)
00906      {
00907          ejected_mass = reheated_mass;
00908          reheated_mass = 0.0;
00909      }
00910  else
00911      ejected_mass = 0.0;
00912  }
00913
00914
00915  if(FeedbackRecipe == 1)
00916  {
00917      if (dis < Gal[centralgal].Rvir)
00918      {
00919          ejected_mass =
00920          (FeedbackEjectionEfficiency * (EtaSNcode * EnergySNcode) / (CentralVV
00921          ir * CentralVvir) -
00922              FeedbackReheatingEpsilon) * mstars;
00923          if(ejected_mass < 0.0)
00924              ejected_mass = 0.0;
00925      else
00926      {
00927          ejected_mass =
00928          (FeedbackEjectionEfficiency * (EtaSNcode * EnergySNcode) / (MergeCent
00929          ralVvir * MergeCentralVvir) -
00930              FeedbackReheatingEpsilon) * mstars;
00931          if(ejected_mass < 0.0)
00932              ejected_mass = 0.0;
00933      }
00934      ejected_sat = 0.0;
00935  }
00936
00937
00938  if(ejected_mass < 0.0)
00939  {

```

```

00940         ejected_mass = 0.0;
00941     }
00942
00943     if (ejected_sat < 0.0)
00944         ejected_sat=0.0;
00945
00946     /* update the star formation rate */
00947 #ifdef UseFullSfr
00948     Gal[merger_centralgal].Sfr[Halo[halonr].SnapNum] += mstars / deltaT;
00949 #else
00950 #ifdef SAVE_MEMORY
00951     Gal[merger_centralgal].Sfr += mstars / deltaT;
00952 #else
00953     for(outputbin = 0; outputbin < NOUT; outputbin++)
00954     {
00955         if(Halo[halonr].SnapNum == ListOutputSamps[outputbin])
00956         {
00957             Gal[merger_centralgal].Sfr[outputbin] += mstars / deltaT;
00958             break;
00959         }
00960     }
00961 #endif
00962 #endif
00963
00964     /* update for star formation */
00965     metallicity = get_metallicity(Gal[merger_centralgal].ColdGas, Gal[merger_cen-
00966     tralgal].MetalsColdGas);
00967
00968     update_from_star_formation(merger_centralgal, mstars, metallicity);
00969
00970     /* but remember the value of the metallicity when SF occurs */
00971     metallicitySF = metallicity;
00972
00973     /* formation of new metals - instantaneous recycling approximation
00974      * - only SNII also recompute the metallicity of the cold phase */
00975     if (FeedbackRecipe == 2)
00976     {   if(Gal[merger_centralgal].ColdGas > 1e-8 && mass_ratio <
00977         ThreshMajorMerger)
00978         {
00979             Gal[merger_centralgal].MetalsColdGas += Yield * (1.0 - FracZtoHot) * ms-
00980             tars;
00981             Gal[merger_centralgal].MetalsHotGas += Yield * FracZtoHot * mstars;
00982         }
00983     else
00984     {
00985         Gal[merger_centralgal].MetalsHotGas += Yield * mstars;
00986     }
00987     if(Gal[merger_centralgal].ColdGas > 1e-8 && mass_ratio < ThreshMajorMerger)
00988     {
00989         Gal[merger_centralgal].MetalsColdGas += Yield * (1.0 - FracZtoHot) * ms-
00990             tars;
00991         Gal[centralgal].MetalsHotGas += Yield * FracZtoHot * mstars;
00992     }
00993 }
00994
00995     metallicity = get_metallicity(Gal[merger_centralgal].ColdGas, Gal[merger_cen-
00996     tralgal].MetalsColdGas);
00997     /* update for feedback now */
00998     update_from_feedback(merger_centralgal, centralgal, reheated_mass, ejected_mass,
00999     , ejected_sat, metallicity);

```

```

01000
01001 /* update the luminosities due to the stars formed */
01002 add_to_luminosities(merger_centralgal, mstars, time, metallicitySF);
01003
01004
01005 if (Ggas > 0.)
01006     return mstars/Ggas;
01007 else
01008     return 0.0;
01009
01010 }
01011
01012 /** @brief Calculates the bulge size after a merger. */
01013 void bulgesize_from_merger(double mass_ratio, int merger_centralgal, int p, double
01014 Mcstar, double Mcbulge, double Mcgas, double frac)
01015 {
01016     /** @brief For any type of merger calculates the new bulge size using
01017      * Eq. 33 in Guo2010:
01018      *
01019      * 
$$\frac{GM^2}{R_{\text{new,bulge}}} = \frac{GM^2_1}{R_1} + C \frac{GM^2_2}{R_2} + \alpha_{\text{inter}} \frac{GM_1M_2}{R_1 + R_2}$$

01020      *
01021      *
01022      * This implementation assumed that the new bulge occupies the
01023      * same space as the components that formed it. */
01024
01025     double Mc, Rc;
01026     double Mp, Mpgas, Mpstar, Mpbulge, Rp;
01027     double fint, c;
01028
01029     fint=0.5;
01030     c=0.5;
01031
01032     /*p represents the satellite and c the central galaxy*/
01033     Mpgas=Gal[p].ColdGas;
01034     Mpstar=Gal[p].StellarMass;
01035     Mpbulge=Gal[p].BulgeMass;
01036
01037     /* calculate radius for the object that will form the new bulge - Rc and Rp */
01038     /* Minor Merger */
01039     if(mass_ratio < ThreshMajorMerger)
01040     {
01041         /* In a minor merger only the stars of the satellite galaxy are moved
01042          * to the bulge of the central galaxy, therefore only stellar
01043          * components are used to compute radius and masses. */
01044         frac=0.0;
01045         /* in a minor merger only consider the bulge mass of the central galaxy */

01046         Mc=Mcbulge;
01047         Rc=Gal[merger_centralgal].BulgeSize;
01048         /* and stellarmass of satellite*/
01049         Mp=Mpstar;
01050         if (Mp >0.0)
01051             Rp=(Gal[p].StellarDiskRadius/3.*1.68*(Mpstar-Mpbulge)+Gal[p].BulgeSize*Mp
01052             bulge)/Mpstar;
01053             else
01054                 Rp=0.0;
01055     /* Major Merger */
01056     else
01057     {
01058         /* on a major merger both stellar and gas (after a burst) components
01059          * from the final bulge and need to be considered */
01060          /* Mc = bulge mass + burst of central*/
01061         Mc=Mcstar+frac*Mcgas;
01062         if (Mc > 0.0)
01063             Rc=(Gal[merger_centralgal].StellarDiskRadius/3.*1.68*(Mcstar-Mcbulge) +

```

```

Gal[merger_centralgal].BulgeSize*Mcbulge+Gal[merger_centralgal].GasDiskRadius*frac
c*Mcgas/3.*1.68) / (Mcgas*frac+Mcstar);
01064     else
01065         Rc=0.0;
01066     /* and satellite Mp */
01067     Mp=Mpstar+frac*Mpgas;
01068     if (Mp > 0.0)
01069         Rp=(Gal[p].StellarDiskRadius/3.*1.68*(Mpstar-Mpbulge)+Gal[p].BulgeSize*Mp
bulge+Gal[p].GasDiskRadius*frac*Mpgas/3.*1.68) / (Mpgas*frac+Mpstar);
01070     else
01071         Rp=0.0;
01072
01073 }
01074
01075 /* If both original radius are bigger then 0 then this is Eq. 33 in Guo 2010
01076 * solved for R_new,bulge with all terms divided by G and C. */
01077 if (Rp >1.e-8 && Rc >1.e-8 )
01078     Gal[merger_centralgal].BulgeSize= (Mp+Mc) * (Mp+Mc) / (Mp*Mp/Rp+Mc*Mc/Rc+fint/c*Mc
*Mc/(Rp+Rc));
01079
01080 if (Rc >1.e-8 && Rp <1.e-8)
01081     Gal[merger_centralgal].BulgeSize= (Mp+Mc) * (Mp+Mc) / (Mc*Mc/Rc+fint/c*Mc*Mc/(Rp+R
c));
01082
01083 if(Rc <1.e-8 && Rp <1.e-8)
01084     Gal[merger_centralgal].BulgeSize=0.0;
01085
01086 if (Rc <1.e-8 && Rp > 1.e-8)
01087     Gal[merger_centralgal].BulgeSize= (Mp+Mc) * (Mp+Mc) / (Mp*Mp/Rp+fint/c*Mc*Mc/(Rp+R
c));
01088
01089
01090 if ((Mp+Mc > 0.0 && Gal[merger_centralgal].BulgeSize == 0.0 ) || (Mp+Mc == 0.0 &&
Gal[merger_centralgal].BulgeSize> 0.0))
01091 {
01092
01093     if (Gal[p].ColdGas+Gal[p].StellarMass <1.e-8)
01094         printf("????!\n");
01095     printf("bulgesize wrong in merger. bulgemass %f, bulgesize %f, Rp %f, Rc %f
, Mp %f, Mc %f ,mass ratio %f\n",Gal[merger_centralgal].BulgeMass, Gal[p].
BulgeSize,Rp,Rc,Mp,Mc,mass_ratio,Gal[merger_centralgal].HaloNr,merger_centralgal)
;
01096
01097     printf("p: stellarmass %f, bulgemass %f, bulgesize %f, coldgas %f,gasdisk %
f,stellardisk %f \n",Gal[p].StellarMass, Gal[p].BulgeMass, Gal[p].BulgeSize, Gal[p].
.ColdGas, Gal[p].GasDiskRadius, Gal[p].StellarDiskRadius);
01098     printf("central: stellarmass %f, bulgemass %f, bulgesize %f, coldgas %f,gas
disk %f,stellardisk %f \n",Gal[merger_centralgal].StellarMass, Gal[merger_centralg
al].BulgeMass, Gal[merger_centralgal].BulgeSize, Gal[merger_centralgal].ColdGas,
Gal[merger_centralgal].GasDiskRadius, Gal[merger_centralgal].StellarDiskRadius);
01099     exit(0);
01100 }
01101
01102
01103 }

```

4.33 code/recipe_misc.c File Reference

[recipe_misc.c](#) contains a mix of recipes used to: calculate disk sizes, initiate a galaxy structure, get the metallicity, add luminosities, convert snap to age, convert snap to z, calculate max of two numbers, get virial mass, get virial velocity, get virial radius, luminosity to mass, H2 conversion, update central galaxy, update type 1 and type2.

Functions

- double `get_disk_radius` (int halonr, int p)
This routine is no longer called (after Guo2010).
- void `get_gas_disk_radius` (int p)
Updates the gas disk radius.
- void `get_stellar_disk_radius` (int p)
Updates the stellar disk radius.
- double `get_initial_disk_radius` (int halonr, int p)
Initiates the value of the disk radius.
- void `init_galaxy` (int p, int halonr)
Initializes the Galaxy Structure by setting all its elements to zero.
- double `get_metallicity` (double gas, double metals)
Gets metallicity for a given mass of material and metals.
- void `add_to_luminosities` (int p, double mstars, double time, double metallicity)
Whenever star formation occurs, calculates the luminosity corresponding to the mass of stars formed, considering the metallicity and age of the material.
- double `NumToTime` (int num)
Converts snapnum into age - in Myr.
- double `RedshiftObs` (int num)
Converts snapnum into redshift.
- double `dmax` (double x, double y)
gets the maximum of two numbers.
- double `get_virial_mass` (int halonr, int p)
*Calculates the virial mass: $M_{\text{crit}200}$ for central halos with $M_{\text{crit}200}$ or $\text{Len} * \text{PartMass}$ for central halos without.*
- double `get_virial_velocity` (int halonr, int p)
Calculates the virial velocity from the virial mass and radius.
- double `get_virial_radius` (int halonr, int p)
Calculates virial radius from a critical overdensity.
- double `lum_to_mag` (double lum)
Converts luminosities into magnitudes.
- double `cal_H2` (int p)
Model the formation of molecular gas.
- void `update_centralgal` (int ngal, int halonr)

Updates properties of central galaxies.

- void [update_type_1](#) (int ngal, int halonr, int cenngal, int prog)
Updates properties of type 1 galaxies.
- void [update_type_2](#) (int ngal, int halonr, int prog, int mostmassive)
Updates properties of type 2 galaxies.
- void [update_hotgas](#) (int ngal, int centralgal)
Not Used - update_hot_frac instead.

4.33.1 Detailed Description

Definition in file [recipe_misc.c](#).

4.33.2 Function Documentation

4.33.2.1 void add_to_luminosities (int p, double mstars, double time, double metallicity)

The semi-analytic code uses look up tables produced by Evolutionary Population Synthesis Models to convert the mass formed on every star formation episode into a luminosity. Each of These tables corresponds to a simple stellar population i.e, a population with a single metallicity. For a given IMF, metatilicity and age, the tables give the luminosity for a $10^{11} M_{\odot}$ burst. The default model uses a Chabrier IMF and stellar populations from Bruzual & Charlot 2003 with 6 different metallicities.

The magnitudes are immediately calculated for each output bin, so that we know the age of each population that contributed to a galaxy total population: the age between creation and output. Apart from the different ages of the populations at a given output bin, if the option COMPUTE_OBS_MAGS is turned on, then we also need to know the K-corrections (going the opposite directions as in observations) that will affect each population.

For each metallicity there is a look up table which has the different magnitudes for each age and then this is k-corrected to all the snapshots.

If MetallicityOption = 0 -> only solar metallicity. If MetallicityOption = 1 -> 6 metallicities.

Definition at line [341](#) of file [recipe_misc.c](#).

References [GALAXY::dObsLum](#), [GALAXY::dObsYLum](#), [find_interpolated_lum\(\)](#), [Gal](#), [Hubble_h](#), [ListOutputSnaps](#), [GALAXY::Lum](#), [MagTableZz](#), [GALAXY::MassWeightAge](#), [MetallicityOption](#), [NumToTime\(\)](#), [GALAXY::ObsLum](#), [GALAXY::ObsYLum](#), [RecycleFraction](#), [UnitTime_in_Megayears](#), and [GALAXY::YLum](#).

Referenced by [collisional_starburst_recipe\(\)](#), [do_major_merger_starburst\(\)](#), and [starformation_and_feedback\(\)](#).

4.33.2.2 double cal_H2 (int p)

Created by Qi, but never used. The table is given by Krumholz.

Definition at line [627](#) of file [recipe_misc.c](#).

References [GALAXY::ColdGas](#), [find_interpolate_h2\(\)](#), [Gal](#), [GALAXY::GasDiskRadius](#), [get_metallicity\(\)](#), [H2](#), and [GALAXY::MetalsColdGas](#).

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

4.33.2.3 double dmax (double x, double y)

Definition at line 549 of file [recipe_misc.c](#).

4.33.2.4 double get_disk_radius (int halonr, int p)

Definition at line 20 of file [recipe_misc.c](#).

References [DiskRadiusMethod](#), [Gal](#), [Halo](#), [GALAXY::Rvir](#), and [GALAXY::Vvir](#).

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

4.33.2.5 void get_gas_disk_radius (int p)

The gas disk is assumed to be thin, in centrifugal equilibrium and to have an exponential density profile:

$$\Sigma(R_{\text{gas}}) = \Sigma_{\text{gas}0} e^{-\frac{R_{\text{gas}}}{R_{\text{gas,d}}}},$$

where $R_{\text{gas,d}}$ is the scale length of the gas disk and $\Sigma_{\text{gas}0}$ is the corresponding central surface density.

Assuming a flat circular velocity curve (galaxy with a negligible self-gravity) in an isothermal dark matter halo, the gas disk scale length is given by:

$$R_{\text{gas,d}} = \frac{J_{\text{gas}}/M_{\text{gas}}}{2V_{\text{max}}},$$

assuming conservation of the angular momentum of the cooling gas and that the maximum circular velocity of satellite galaxies does not change after infall (inner dark matter regions are compact and don't change).

Definition at line 61 of file [recipe_misc.c](#).

References [Gal](#), [GALAXY::GasDiskRadius](#), [GALAXY::GasSpin](#), [GALAXY::InfallVmax](#), [GALAXY::Type](#), and [GALAXY::Vmax](#).

Referenced by [cool_gas_onto_galaxy\(\)](#), and [deal_with_galaxy_merger\(\)](#).

4.33.2.6 double get_initial_disk_radius (int halonr, int p)

First determination of radius in Guo2010 (same as in Delucia2007), after this, the disks are updated using [get_gas_disk_radius](#) and [get_stellar_disk_radius](#). Two options are available:

If [DiskRadiusMethod](#) = 0 then $R_{\text{disk}} = \frac{R_{\text{vir}}}{10}$

If [DiskRadiusMethod](#) = 1 or 2 then the Mo, Mao & White (1998) formalism is used with a Bullock style λ :

$$R_d = \frac{1}{\sqrt{2}} \frac{j_d}{m_d} \lambda r_{200}$$

and using the Milky Way as an approximate guide $R_{\text{disk}} = 3R_d$.

Definition at line 121 of file [recipe_misc.c](#).

References [DiskRadiusMethod](#), [Gal](#), [Halo](#), [GALAXY::Rvir](#), and [GALAXY::Vvir](#).

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

4.33.2.7 double get_metallicity (double gas, double metals)

Definition at line 297 of file [recipe_misc.c](#).

Referenced by [cal_H2\(\)](#), [check_disk_instability\(\)](#), [collisional_starburst_recipe\(\)](#), [cool_gas_onto_galaxy\(\)](#), [do_AGN_heating\(\)](#), [do_major_merger_starburst\(\)](#), [grow_black_hole\(\)](#), [reincorporate_gas\(\)](#), [starformation_and_feedback\(\)](#), and [update_from_feedback\(\)](#).

4.33.2.8 void get_stellar_disk_radius (int p)

The stellar disk is assumed to be thin, in centrifugal equilibrium and to have an exponential density profile:

$$\Sigma(R_\star) = \Sigma_{\star 0} e^{-\frac{R_\star}{R_{\star,d}}},$$

where $R_{\star,d}$ is the scale length of the gas disk and $\Sigma_{\star 0}$ is the corresponding central surface density.

Assuming a flat circular velocity curve (galaxy with a negligible self-gravity) in an isothermal dark matter halo, the gas disk scale length is given by:

$$R_{\star,d} = \frac{J_\star/M_\star}{2V_{\max}},$$

assuming that the maximum circular velocity of satellite galaxies does not change after infall (inner dark matter regions are compact and don't change).

Definition at line 93 of file [recipe_misc.c](#).

References [Gal](#), [GALAXY::InfallVmax](#), [GALAXY::StellarDiskRadius](#), [GALAXY::StellarSpin](#), [GALAXY::Type](#), and [GALAXY::Vmax](#).

Referenced by [deal_with_galaxy_merger\(\)](#), [starformation_and_feedback\(\)](#), and [update_from_starFormation\(\)](#).

4.33.2.9 double get_virial_mass (int halonr, int p)

Definition at line 561 of file [recipe_misc.c](#).

References [Halo](#), [halo_data::Len](#), [halo_data::M_Crit200](#), and [PartMass](#).

Referenced by [estimate_merging_time\(\)](#), [get_virial_radius\(\)](#), [get_virial_velocity\(\)](#), [init_galaxy\(\)](#), [prepare_galaxy_for_output\(\)](#), and [update_centralgal\(\)](#).

4.33.2.10 double get_virial_radius (int halonr, int p)

Calculates virial radius using: $M_{\text{vir}} = \frac{4}{3}\pi R_{\text{vir}}^3 \rho_c \Delta_c$.

From which, assuming $\Delta_c = 200$, $* R_{\text{vir}} = \left(\frac{3M_{\text{vir}}}{4\pi 200\rho_c} \right)^{1/3}$

Definition at line 592 of file [recipe_misc.c](#).

References [G](#), [get_virial_mass\(\)](#), [Halo](#), [Hubble](#), [Omega](#), [OmegaLambda](#), [halo_data::SnapNum](#), and [ZZ](#).

Referenced by [estimate_merging_time\(\)](#), [get_virial_velocity\(\)](#), [init_galaxy\(\)](#), and [update_centralgal\(\)](#).

4.33.2.11 double get_virial_velocity (int halonr, int p)

Calculates virial velocity: $V_{\text{vir}} = \frac{GM_{\text{vir}}}{R_{\text{vir}}}$

Definition at line 577 of file [recipe_misc.c](#).

References [G](#), [get_virial_mass\(\)](#), and [get_virial_radius\(\)](#).

Referenced by [estimate_merging_time\(\)](#), [init_galaxy\(\)](#), and [update_centralgal\(\)](#).

4.33.2.12 void init_galaxy (int *p*, int *halonr*)

Definition at line 146 of file [recipe_misc.c](#).

References [GALAXY::BlackHoleMass](#), [GALAXY::BulgeMass](#), [GALAXY::BulgeSize](#), [GALAXY::ColdGas](#), [GALAXY::CoolingRadius](#), [GALAXY::DescendantGal](#), [GALAXY::DisruptOn](#), [GALAXY::dObsLum](#), [GALAXY::dObsLumBulge](#), [GALAXY::dObsLumDust](#), [GALAXY::dObsYLum](#), [GALAXY::dObsYLumBulge](#), [GALAXY::EjectedMass](#), [GALAXY::FirstProgGal](#), [Gal](#), [GALAXY::GasDiskRadius](#), [GALAXY::GasSpin](#), [get_initial_disk_radius\(\)](#), [get_virial_mass\(\)](#), [get_virial_radius\(\)](#), [get_virial_velocity\(\)](#), [Halo](#), [GALAXY::HaloNr](#), [GALAXY::HaloSpin](#), [GALAXY::HotFrac](#), [GALAXY::HotGas](#), [GALAXY::HotRadius](#), [GALAXY::ICLLum](#), [GALAXY::ICM](#), [GALAXY::Inclination](#), [GALAXY::InfallSnap](#), [GALAXY::InfallVmax](#), [halo_data::Len](#), [GALAXY::Len](#), [GALAXY::Lum](#), [GALAXY::LumBulge](#), [GALAXY::LumDust](#), [GALAXY::MassWeightAge](#), [GALAXY::MergCentralPos](#), [GALAXY::MergeSat](#), [GALAXY::MergTime](#), [GALAXY::MetalsBulgeMass](#), [GALAXY::MetalsColdGas](#), [GALAXY::MetalsEjectedMass](#), [GALAXY::MetalsHotGas](#), [GALAXY::MetalsICM](#), [GALAXY::MetalsStellarMass](#), [halo_data::MostBoundID](#), [GALAXY::MostBoundID](#), [GALAXY::Mvir](#), [GALAXY::NextProgGal](#), [GALAXY::ObsICL](#), [GALAXY::ObsLum](#), [GALAXY::ObsLumBulge](#), [GALAXY::ObsLumDust](#), [GALAXY::ObsYLum](#), [GALAXY::ObsYLumBulge](#), [GALAXY::OriMergTime](#), [halo_data::Pos](#), [GALAXY::Pos](#), [random_generator](#), [GALAXY::Rvir](#), [GALAXY::Sfr](#), [GALAXY::SfrBulge](#), [halo_data::SnapNum](#), [GALAXY::SnapNum](#), [halo_data::Spin](#), [GALAXY::StarMerge](#), [GALAXY::StellarDiskRadius](#), [GALAXY::StellarMass](#), [GALAXY::StellarSpin](#), [GALAXY::Type](#), [halo_data::Vel](#), [GALAXY::Vel](#), [halo_data::Vmax](#), [GALAXY::Vmax](#), [GALAXY::Vvir](#), [GALAXY::XrayLum](#), [GALAXY::YLum](#), and [GALAXY::YLumBulge](#).

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

4.33.2.13 double lum_to_mag (double *lum*)

Converts luminosities into magnitudes: $M = -2.5\log_{10}(L)$

Definition at line 614 of file [recipe_misc.c](#).

Referenced by [prepare_galaxy_for_momaf\(\)](#), and [prepare_galaxy_for_output\(\)](#).

4.33.2.14 double NumToTime (int *num*)

Definition at line 535 of file [recipe_misc.c](#).

References [Age](#).

Referenced by [add_to_luminosities\(\)](#), [main\(\)](#), [update_type_1\(\)](#), and [update_type_2\(\)](#).

4.33.2.15 double RedshiftObs (int *num*)

Definition at line 542 of file [recipe_misc.c](#).

References [ZZ](#).

4.33.2.16 void update_centralgal (int *ngal*, int *halonr*)

M_{vir} , R_{vir} and V_{vir} are only updated for type 0's. Once galaxies become satellites these quantities stay unchanged, so will be the values at infall.

If type = 0 then the HotRadius is the Viral Radius, which will be used in the cooling recipe.

Other infall information will not be used for central galaxies so we do not care whether they carry the correct values.

Definition at line 668 of file [recipe_misc.c](#).

References [Gal](#), [get_virial_mass\(\)](#), [get_virial_radius\(\)](#), [get_virial_velocity\(\)](#), [Halo](#), [GALAXY::HaloSpin](#), [GALAXY::HotRadius](#), [GALAXY::InfallSnap](#), [GALAXY::InfallVmax](#), [GALAXY::MergeOn](#), [GALAXY::Mvir](#), [GALAXY::Rvir](#), [halo_data::SnapNum](#), [GALAXY::Type](#), [halo_data::Vmax](#), and [GALAXY::Vvir](#).

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

4.33.2.17 void update_hotgas (int *ngal*, int *centralgal*)

Updates the fraction of hot gas attached on each dark matter particles of the subhalo.

Definition at line 840 of file [recipe_misc.c](#).

References [GALAXY::CentralGal](#), [Gal](#), [Halo](#), [GALAXY::HaloNr](#), [GALAXY::HotFrac](#), [GALAXY::HotGas](#), [GALAXY::Len](#), [PartMass](#), [GALAXY::Pos](#), [GALAXY::Rvir](#), [halo_data::SnapNum](#), [GALAXY::Type](#), and [ZZ](#).

4.33.2.18 void update_type_1 (int *ngal*, int *halonr*, int *cenngal*, int *prog*)

If the galaxy has just become a satellite (its type hasn't yet been reset from 0 to 1), the fraction of hot gas to dark matter halo mass is recorded $HotFrac = \frac{M_{\text{hot}}}{M_{\text{vir}}}$.

If MERGE01 = 1, then a dynamical friction decay time scale is calculated for type 1's (as is done for type 2 - introduced for millennium II where the increased resolution means type 1 always retain some dark matter and orbit around for a long time). This is only calculated when the baryonic mass of the type 1 becomes larger than its dark matter mass. The code finds the type 0 to which this galaxy should merge and then sets up the merging clock.

Definition at line 699 of file [recipe_misc.c](#).

References [GALAXY::ColdGas](#), [halo_data::Descendant](#), [estimate_merging_time\(\)](#), [halo_data::FirstHaloInFOFgroup](#), [Gal](#), [Halo](#), [GALAXY::HotFrac](#), [GALAXY::HotGas](#), [GALAXY::Len](#), [GALAXY::MergeOn](#), [GALAXY::MergTime](#), [GALAXY::Mvir](#), [NumToTime\(\)](#), [GALAXY::OriMergTime](#), [PartMass](#), [GALAXY::SnapNum](#), [GALAXY::StellarMass](#), and [GALAXY::Type](#).

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

4.33.2.19 void update_type_2 (int *ngal*, int *halonr*, int *prog*, int *mostmassive*)

Sets HotFrac and Hot Radius to 0, since all the hot gas has been stripped. Calls estimate_merging_time to get the merging time scale, calculated for the orbital decay due to dynamical friction, since this galaxy has lost its dark matter halo and its position cannot be tracked.

Definition at line 814 of file [recipe_misc.c](#).

References [estimate_merging_time\(\)](#), [Gal](#), [Halo](#), [GALAXY::HotFrac](#), [GALAXY::HotRadius](#),

`GALAXY::MergeOn`, `GALAXY::MergTime`, `NumToTime()`, `GALAXY::OriMergTime`, `GALAXY::SnapNum`, and `GALAXY::Type`.

Referenced by `main()`.

4.34 code/recipe_misc.c

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <string.h>
00004 #include <math.h>
00005 #include <time.h>
00006 #include <gsl/gsl_rng.h>
00007
00008 #include "allvars.h"
00009 #include "proto.h"
00010
00011 /**@file recipe_misc.c
00012 * @brief recipe_misc.c contains a mix of recipes used to: calculate disk
00013 * sizes, initiate a galaxy structure, get the metallicity, add
00014 * luminosities, convert snap to age, convert snap to z, calculate
00015 * max of two numbers, get virial mass, get virial velocity, get virial
00016 * radius, luminosity to mass, H2 conversion, update central galaxy,
00017 * update type 1 and type2. */
00018
00019 /** This routine is no longer called (after Guo2010) */
00020 double get_disk_radius(int halonr, int p)
00021 {
00022     double SpinParameter;
00023
00024
00025     if(DiskRadiusMethod == 1)
00026     {
00027         /* See Mo, Mao & White (1998) eq12, and using a Bullock style lambda. Since
00028            this is the scale length
00029            we take the typical star forming region as 3 times this using the Milky
00030            Way as an approximate guide */
00031         SpinParameter =
00032             sqrt(Halo[halonr].Spin[0] * Halo[halonr].Spin[0] + Halo[halonr].Spin[1] *
00033                 Halo[halonr].Spin[1] +
00034                     Halo[halonr].Spin[2] * Halo[halonr].Spin[2]) / (1.414 * Gal[p].Vvir
00035             * Gal[p].Rvir);
00036         return 3.0 * (SpinParameter / 1.414) * Gal[p].Rvir;
00037     }
00038     else
00039     /* simple prescription */
00040     return Gal[p].Rvir / 10.0;
00041
00042
00043     /* The gas disk is assumed to be thin, in centrifugal equilibrium and to have
00044     * an exponential density profile:
00045     *
00046     * \f$ \Sigma(R_{\rm{gas}}) = \Sigma_{\rm{gas0}} e^{-\frac{R_{\rm{gas}}}{R_{\rm{gas,d}}}}, \f$
00047     *
00048     * where \f$R_{\rm{gas,d}}\f$ is the scale length of the gas disk and
00049     * \f$\Sigma_{\rm{gas0}}\f$ is the corresponding central surface density.
00050     *
00051     * Assuming a flat circular velocity curve (galaxy with a negligible self-gravity)
00052     * in an isothermal dark matter halo, the gas disk scale length is given by:
00053

```

```

00054  *
00055  *    $R_{\rm{gas}} = \frac{J_{\rm{gas}}}{M_{\rm{gas}}} \cdot 2V_{\rm{max}}$ ,
00056  *
00057  *   assuming conservation of the angular momentum of the cooling gas and that the
00058  *   maximum circular velocity of satellite galaxies does not change after infall
00059  *   (inner dark matter regions are compact and don't change). */
00060
00061 void get_gas_disk_radius(int p)
00062 {
00063     double dgas;
00064     if (Gal[p].Type == 0)
00065         dgas = 3.0*sqrt(Gal[p].GasSpin[0]*Gal[p].GasSpin[0]+Gal[p].GasSpin[1]*Gal[p].
00066             GasSpin[1]+Gal[p].GasSpin[2]*Gal[p].GasSpin[2])/ 2.0 / Gal[p].Vmax;
00067     else
00068         dgas = 3.0*sqrt(Gal[p].GasSpin[0]*Gal[p].GasSpin[0]+Gal[p].GasSpin[1]*Gal[p].
00069             GasSpin[1]+Gal[p].GasSpin[2]*Gal[p].GasSpin[2])/ 2.0 / Gal[p].InfallVmax;
00070
00071     Gal[p].GasDiskRadius=dgas;
00072
00073
00074 /** @brief Updates the stellar disk radius.
00075  *
00076  *   The stellar disk is assumed to be thin, in centrifugal equilibrium and to hav
00077  *   e
00078  *   an exponential density profile:
00079  *    $\Sigma(R_{\star}) = \Sigma_{\star} e^{-\frac{R_{\star}}{R_{\rm{star,d}}}}$ ,
00080  *   where  $R_{\rm{star,d}}$  is the scale length of the gas disk and
00081  *    $\Sigma_{\star}$  is the corresponding central surface density.
00082  *
00083  *   Assuming a flat circular velocity curve (galaxy with a negligible self-gravit
00084  *   y)
00085  *   in an isothermal dark matter halo, the gas disk scale length is given by:
00086  *
00087  *    $R_{\rm{star,d}} = \frac{J_{\star}}{M_{\star}} \cdot 2V_{\rm{max}}$ ,
00088  *
00089  *   assuming that the maximum circular velocity of satellite galaxies does not
00090  *   change after infall (inner dark matter regions are compact and don't change).
00091  */
00092
00093 void get_stellar_disk_radius(int p)
00094 {
00095     double dstar;
00096
00097     if (Gal[p].Type == 0)
00098         dstar=3.0*sqrt(Gal[p].StellarSpin[0]*Gal[p].StellarSpin[0]+Gal[p].
00099             StellarSpin[1]*Gal[p].StellarSpin[1]+Gal[p].StellarSpin[2]*Gal[p].StellarSpin[2])
00100             / 2.0 / Gal[p].Vmax;
00101     else
00102         dstar=3.0*sqrt(Gal[p].StellarSpin[0]*Gal[p].StellarSpin[0]+Gal[p].StellarSpin
00103             [1]*Gal[p].StellarSpin[1]+Gal[p].StellarSpin[2]*Gal[p].StellarSpin[2])/ 2.0 /
00104             Gal[p].InfallVmax;
00105
00106 /** @brief Initiates the value of the disk radius.
00107  *
00108  *   First determination of radius in Guo2010 (same as in Delucia2007), after this
00109  *   the disks are updated using get_gas_disk_radius and get_stellar_disk_radius.

```

```

00110 * Two options are available:
00111 *
00112 * If DiskRadiusMethod = 0 then  $R_{\rm disk} = \frac{R_{\rm vir}}{10}$ 
00113 *
00114 * If DiskRadiusMethod = 1 or 2 then the Mo, Mao & White (1998) formalism is
00115 * used with a Bullock style  $\lambda$ :
00116 *
00117 *  $R_d = \frac{1}{\sqrt{2}} \frac{j_d}{m_d} \lambda r_{200}$ 
00118 *
00119 * and using the Milky Way as an approximate guide  $R_{\rm disk} = 3R_d$ . *
00120 /
00121 double get_initial_disk_radius(int halonr, int p)
00122 {
00123     double SpinParameter;
00124
00125
00126     if(DiskRadiusMethod == 1 || DiskRadiusMethod == 2 )
00127     {
00128         /*spin parameter*/
00129         SpinParameter =
00130             sqrt(Halo[halonr].Spin[0] * Halo[halonr].Spin[0] + Halo[halonr].Spin[1] *
00131                 Halo[halonr].Spin[1] +
00132                     Halo[halonr].Spin[2] * Halo[halonr].Spin[2]) / (1.414 * Gal[p].Vvir
00133 * Gal[p].Rvir);
00134         /*Rdisk=3*Rd=3*spin*R_vir/sqrt(2)*/
00135         return 3.0 * (SpinParameter / 1.414) * Gal[p].Rvir;
00136     }
00137     else
00138         /* simpler prescription */
00139         return Gal[p].Rvir / 10.0;
00140 }
00141
00142
00143
00144 /** @brief Initializes the Galaxy Structure by setting all its
00145 * elements to zero. */
00146 void init_galaxy(int p, int halonr)
00147 {
00148     int j, outputbin;
00149
00150 #ifdef UseFullSfr
00151     int snap;
00152 #endif
00153
00154     if(halonr != Halo[halonr].FirstHaloInFOFgroup)
00155     {
00156         printf("Hah?\n");
00157         exit(1);
00158     }
00159
00160     Gal[p].Type = 0;
00161
00162     Gal[p].HaloNr = halonr;
00163     Gal[p].MostBoundID = Halo[halonr].MostBoundID;
00164     Gal[p].SnapNum = Halo[halonr].SnapNum - 1;
00165
00166     for(j = 0; j < 3; j++)
00167     {
00168         Gal[p].Pos[j] = Halo[halonr].Pos[j];
00169         Gal[p].Vel[j] = Halo[halonr].Vel[j];
00170         Gal[p].GasSpin[j] = Halo[halonr].Spin[j];
00171         Gal[p].StellarSpin[j] = Halo[halonr].Spin[j];
00172         Gal[p].HaloSpin[j] = Halo[halonr].Spin[j];
00173         Gal[p].MergCentralPos[j] = Gal[p].Pos[j];

```

```

00174      }
00175
00176     Gal[p].Len = Halo[halonr].Len;
00177     Gal[p].Vmax = Halo[halonr].Vmax;
00178     Gal[p].InfallVmax = Halo[halonr].Vmax;
00179     Gal[p].Vvir = get_virial_velocity(halonr,p);
00180     Gal[p].Mvir = get_virial_mass(halonr,p);
00181     Gal[p].Rvir = get_virial_radius(halonr,p);
00182     Gal[p].MergeSat = 0.0;
00183     Gal[p].InfallSnap = 0;
00184
00185     Gal[p].ColdGas = 0.0;
00186     Gal[p].StellarMass = 0.0;
00187     Gal[p].BulgeMass = 0.0;
00188     Gal[p].HotGas = 0.0;
00189     Gal[p].EjectedMass = 0.0;
00190     Gal[p].BlackHoleMass = 0.0;
00191     /*ram pressure*/
00192     Gal[p].HotFrac=0.0;
00193     Gal[p].HotRadius=Gal[p].Rvir;
00194 #ifdef GALAXYTREE
00195     Gal[p].DisruptOn = 0;
00196 #endif
00197     Gal[p].MetalsColdGas = 0.0;
00198     Gal[p].MetalsStellarMass = 0.0;
00199     Gal[p].MetalsBulgeMass = 0.0;
00200     Gal[p].MetalsHotGas = 0.0;
00201     Gal[p].MetalsEjectedMass = 0.0;
00202
00203 // TODO document this, why the factor of 2.25 iso 2?
00204     Gal[p].Inclination = gsl_rng_uniform(random_generator) * M_PI / 2.25;
00205
00206
00207 #ifdef UseFullSfr
00208     for(snap = 0; snap < MAXSNAPS; snap++)
00209     {
00210         Gal[p].Sfr[snap] = 0.0;
00211         Gal[p].SfrBulge[snap] = 0.0;
00212     }
00213 #else
00214 #ifdef SAVE_MEMORY
00215     Gal[p].Sfr = 0.0;
00216     Gal[p].SfrBulge = 0.0;
00217 #else
00218     for(outputbin = 0; outputbin < NOUT; outputbin++)
00219     {
00220         Gal[p].Sfr[outputbin] = 0.0;
00221         Gal[p].SfrBulge[outputbin] = 0.0;
00222     }
00223 #endif
00224 #endif
00225     Gal[p].StarMerge=0.0;
00226
00227     Gal[p].XrayLum = 0.0;
00228     Gal[p].GasDiskRadius = get_initial_disk_radius(halonr, p);
00229     Gal[p].StellarDiskRadius=Gal[p].GasDiskRadius;
00230     Gal[p].BulgeSize=0.0;
00231
00232     Gal[p].OriMergTime=0.0;
00233
00234     Gal[p].ICM=0.0;
00235     Gal[p].MetalsICM=0.0;
00236
00237
00238     Gal[p].MergTime = 0.0;
00239     Gal[p].CoolingRadius = 0.0;
00240

```

```

00241
00242 #ifdef OUTPUT_REST_MAGS
00243   for(outputbin = 0; outputbin < NOUT; outputbin++)
00244   {
00245     for(j = 0; j < NMAG; j++)
00246     {
00247       Gal[p].Lum[j][outputbin]      = 0.0;
00248       Gal[p].YLum[j][outputbin]    = 0.0;
00249       Gal[p].LumBulge[j][outputbin] = 0.0;
00250       Gal[p].YLumBulge[j][outputbin] = 0.0;
00251       Gal[p].LumDust[j][outputbin] = 0.0;
00252 #ifdef ICL
00253   Gal[p].ICLLum[j][outputbin] = 0.0;
00254 #endif
00255   }
00256   Gal[p].MassWeightAge[outputbin] = 0.0;
00257 }
00258 #endif
00259 #ifdef COMPUTE_OBS_MAGS
00260   for(outputbin = 0; outputbin < NOUT; outputbin++)
00261   {
00262     for(j = 0; j < NMAG; j++)
00263     {
00264       Gal[p].ObsLum[j][outputbin]      = 0.0;
00265       Gal[p].ObsYLum[j][outputbin]    = 0.0;
00266       Gal[p].ObsLumBulge[j][outputbin] = 0.0;
00267       Gal[p].ObsYLumBulge[j][outputbin] = 0.0;
00268       Gal[p].ObsLumDust[j][outputbin] = 0.0;
00269 #ifdef ICL
00270   Gal[p].ObsICL[j][outputbin] = 0.0;
00271 #endif
00272
00273 #ifdef OUTPUT_MOMAF_INPUTS
00274   Gal[p].dObsLum[j][outputbin]      = 0.0;
00275   Gal[p].dObsYLum[j][outputbin]    = 0.0;
00276   Gal[p].dObsLumBulge[j][outputbin] = 0.0;
00277   Gal[p].dObsYLumBulge[j][outputbin] = 0.0;
00278   Gal[p].dObsLumDust[j][outputbin] = 0.0;
00279 #endif
00280   }
00281 }
00282 #endif
00283
00284
00285 #ifdef GALAXYTREE
00286   Gal[p].NextProgGal = -1;
00287   Gal[p].FirstProgGal = -1;
00288   Gal[p].DescendantGal = -1;
00289 #endif
00290
00291 }
00292
00293
00294
00295 //TODO get rid of it
00296 /**@brief Gets metallicity for a given mass of material and metals. */
00297 double get_metallicity(double gas, double metals)
00298 {
00299   double metallicity;
00300
00301
00302   if(gas > 0.0 && metals > 0.0)
00303   {
00304     metallicity = metals / gas;
00305     if(metallicity < 1.0)
00306       return metallicity;
00307     else

```

```

00308         return 1.0;
00309     }
00310     else
00311         return 0.0;
00312
00313 }
00314 /*TODO take away magnitudes and work with luminosities*/
00315
00316 /**@brief Whenever star formation occurs, calculates the luminosity corresponding
00317 *      to the mass of stars formed, considering the metallicity and age of the
00318 *      material.
00319 *
00320 * The semi-analytic code uses look up tables produced by Evolutionary Populations
00321 * Synthesis Models to convert the mass formed on every star formation episode
00322 * into a luminosity. Each of These tables corresponds to a simple stellar
00323 * population i.e, a population with a single metallicity. For a given IMF,
00324 * metallicity and age, the tables give the luminosity for a
00325 * \f$ 10^{11} M_{\odot}\f$ burst. The default model uses a Chabrier IMF and
00326 * stellar populations from Bruzual & Charlot 2003 with 6 different metallicities
00327 .
00328 *
00329 * The magnitudes are immediately calculated for each output bin, so that we know
00330 * the age of each population that contributed to a galaxy total population: the
00331 * age between creation and output. Apart from the different ages of the populations
00332 * at a given output bin, if the option COMPUTE_OBS_MAGS is turned on, then we also
00333 * need to know the K-corrections (going the opposite directions as in observations)
00334 * that will affect each population.
00335 *
00336 * For each metallicity there is a look up table which has the different magnitudes
00337 * for each age and then this is k-corrected to all the snapshots.
00338 *
00339 * If MetallicityOption = 0 -> only solar metallicity.
00340 * If MetallicityOption = 1 -> 6 metallicities.
00341 */
00342 void add_to_luminosities(int p, double mstars, double time, double metallicity)
00343 {
00344     int outputbin, idx, metindex, tabindex, zindex, j;
00345     double f1, f2, fmet1, fmet2, fz1, fz2;
00346     double X1, X2, age;
00347     int a1, a2;
00348     double tbc;
00349
00350     //TODO define elsewhere and maybe make the 10. an input parameter?
00351     /* Time bellow which the luminosities are corrected for extinction due to
00352      * molecular birth clouds. */
00353     /*TODO - the idea here is to convert from 1e7 years into internal units
00354      * of 1e12/h...this division should be avoid and UnitTime_in_Megayears
00355      * is only an approximation, not 1e6!!! look at set_units() at init.c*/
00356     tbc = 10. / UnitTime_in_Megayears * Hubble_h;
00357
00358
00359     /*mstars was in units of 1.e10 Msun - now in units of 1.e11 Msun/h*/
00360     X1 = mstars * 0.1 / Hubble_h;
00361     /*1/2.5*ln10*/
00362     X2 = -0.4 * M_LN10;
00363
00364     /* now we have to change the luminosities accordingly */

```

```

00365  /* note: we already know at which place we have to look up the tables,
00366   * since we know the output times, the current time and the metallicity */
00367 #ifdef OUTPUT_REST_MAGS
00368   if(MetallicityOption == 0) // use only solar metallicity
00369   {
00370     for(outputbin = 0; outputbin < NOUT; outputbin++)
00371     {
00372       /*finds the 2 closest points in the SPS table in terms of age and metallicit
y*/
00373       /*time gives the time_to_present for the current step while
00374        * NumToTime(ListOutputSamps[outputbin]) gives the time of the output sna
p - units 1e12Yrs/h*/
00375       find_interpolated_lum(time, NumToTime(ListOutputSamps[outputbin]), meta
llicity,
00376                               &metindex, &tabindex, &f1, &f2, &fmet1, &fmet2);
00377       metindex = 4; // reset met index to use only solar metallicity
00378       age = time - NumToTime(ListOutputSamps[outputbin]);
00379       /*For rest-frame, there is no K-correction on magnitudes,
00380        * hence the 0 in MagTableZz[j][metindex][0][tabindex] */
00381       for(j = 0; j < NMAG; j++)
00382     {
00383       //interpolation between the points found by find_interpolated_l
um
00384       Gal[p].Lum[j][outputbin] +=
00385         X1 * exp(X2 * (f1 * MagTableZz[j][metindex][0][tabindex] +
00386                         f2 * MagTableZz[j][metindex][0][tabindex + 1]));
00387       /*compute a luminosity to be used for extinction due to young birth
clouds*/
00388       if (age <= tbc)
00389     {
00390       //interpolation between the points found by find_interpolated_l
um
00391       Gal[p].YLum[j][outputbin] +=
00392         X1 * exp(X2 * (f1 * MagTableZz[j][metindex][0][tabindex] +
00393                         f2 * MagTableZz[j][metindex][0][tabindex + 1]));
00394     }
00395   }
00396   //Age in 1e12Yrs/h (multiply by 1000.0 to get Gyr/h)
00397   Gal[p].MassWeightAge[outputbin] += age * mstars * (1.-RecycleFraction);

00398   }
00399 }
00400 else if(MetallicityOption == 1) // use all 6 metallicities
00401 {
00402   for(outputbin = 0; outputbin < NOUT; outputbin++)
00403   {
00404     /*finds the 2 closest points in the SPS table in terms of age and metallicit
y*/
00405     find_interpolated_lum(time, NumToTime(ListOutputSamps[outputbin]), meta
llicity,
00406                               &metindex, &tabindex, &f1, &f2, &fmet1, &fmet2);
00407     age = time - NumToTime(ListOutputSamps[outputbin]);
00408     /*For rest-frame, there is no K-correction on magnitudes,
00409      * hence the 0 in MagTableZz[j][metindex][0][tabindex] */
00410     for(j = 0; j < NMAG; j++)
00411   {
00412     //interpolation between the points found by find_interpolated_l
um
00413     Gal[p].Lum[j][outputbin] +=
00414       X1 * exp(X2 * (fmet1 * (f1 * MagTableZz[j][metindex][0][tabindex]
+
00415                               f2 * MagTableZz[j][metindex][0][tabindex
+ 1]) +
00416                               fmet2 * (f1 * MagTableZz[j][metindex + 1][0][tabin
dex] +
00417                               f2 * MagTableZz[j][metindex + 1][0][tabin

```

```

        dex + 1)));
00418     /*compute a luminosity to be used for extinction due to young birth
   clouds*/
00419     if (age <= tbc)
00420     {
00421         //interpolation between the points found by find_interpolated_l
   um
00422         Gal[p].YLum[j][outputbin] +=
00423             X1 * exp(X2 * (fmet1 * (f1 * MagTableZz[j][metindex][0][tabin
   dex] +
00424                 f2 * MagTableZz[j][metindex][0][tabin
   dex + 1]) +
00425                 abindex) +
00426                 f2 * MagTableZz[j][metindex + 1][0][t
   abindex + 1]));
00427     }
00428 }
00429 //Age in 1e12Yrs/h (multiply by 1000.0 to get Gyr/h)
00430 Gal[p].MassWeightAge[outputbin] += age * mstars * (1.-RecycleFraction);

00431 }
00432 }
00433 #endif
00434
00435
00436 #ifdef COMPUTE_OBS_MAGS
00437     if(MetallicityOption == 0) // use only solar metallicity
00438     {
00439         for(outputbin = 0; outputbin < NOUT; outputbin++)
00440         {
00441             /*finds the 2 closest points in the SPS table in terms of age and metallicity
   */
00442             find_interpolated_lum(time, NumToTime(ListOutputSamps[outputbin]), meta
   llicity,
00443                             &metindex, &tabindex, &f1, &f2, &fmet1, &fmet2);
00444             metindex = 4; // reset met index to use only solar metallicity
00445             //TODO - For MRII the interpolation should be done on redshift as well-
   find_interpolated_obsLum
00446             zindex = 63 - ListOutputSamps[outputbin];
00447             age = time - NumToTime(ListOutputSamps[outputbin]);
00448             /* Note the zindex in MagTableZz[j][metindex][zindex][tabindex] meaning
   the
00449                 * magnitudes are now "inversely k-corrected to get observed frame at o
   utput bins" */
00450             for(j = 0; j < NMAG; j++)
00451             {
00452                 //interpolation between the points found by find_interpolated_l
   um
00453                 Gal[p].ObsLum[j][outputbin] +=
00454                     X1 * exp(X2 * (f1 * MagTableZz[j][metindex][zindex][tabindex] +
00455                         f2 * MagTableZz[j][metindex][zindex][tabindex + 1]
   ));
00456 #ifdef OUTPUT_MOMAF_INPUTS
00457             //interpolation between the points found by find_interpolated_lum
00458             Gal[p].dObsLum[j][outputbin] +=
00459                 X1 * exp(X2 * (f1 * MagTableZz[j][metindex][zindex+1][tab
   index] +
00460                               f2 * MagTableZz[j][metindex][zindex+1][ta
   bindex + 1]));
00461 #endif
00462             /*compute a luminosity to be used for extinction due to young birth
   clouds*/
00463             if (age <= tbc)
00464             {
00465                 //interpolation between the points found by find_interpolated_l
   um

```

```

00466             Gal[p].ObsYLum[j][outputbin] +=
00467             X1 * exp(X2 * (f1 * MagTableZz[j][metindex][zindex][tabindex]
00468             +
00469             f2 * MagTableZz[j][metindex][zindex][tabindex
00470             + 1)));
00471 #ifdef OUTPUT_MOMAF_INPUTS
00472             //interpolation between the points found by find_interpolated_l
00473             um
00474             Gal[p].dObsYLum[j][outputbin] +=
00475             X1 * exp(X2 * (f1 * MagTableZz[j][metindex][zindex+1][tabinde
00476             x] +
00477             f2 * MagTableZz[j][metindex][zindex+1][tabinde
00478             x + 1)));
00479 #endif
00480         }
00481     }
00482     }
00483     else if(MetallicityOption == 1) // use all 6 metallicities
00484     {
00485         for(outputbin = 0; outputbin < NOUT; outputbin++)
00486         {
00487             /*finds the 2 closest points in the SPS table in terms of age and metallicity
00488             */
00489             find_interpolated_lum(time, NumToTime(ListOutputSamps[outputbin]), meta
00490             lllicity,
00491             &metindex, &tabindex, &f1, &f2, &fmet1, &fmet2);
00492             //TODO - For MRII the interpolation should be done on redshift as well-
00493             find_interpolated_obsLum
00494             zindex = 63 - ListOutputSamps[outputbin];
00495             age = time - NumToTime(ListOutputSamps[outputbin]);
00496             /* Note the zindex in MagTableZz[j][metindex][zindex][tabindex] meaning
00497             the
00498                 * magnitudes are now "inversely k-corrected to get observed frame at o
00499                 utput bins" */
00500             for(j = 0; j < NMAG; j++)
00501             {
00502                 //interpolation between the points found by find_interpolated_l
00503                 um
00504                 Gal[p].ObsLum[j][outputbin] +=
00505                 X1 * exp(X2 * (fmet1 * (f1 * MagTableZz[j][metindex][zindex][tabi
00506                 ndex] +
00507                 f2 * MagTableZz[j][metindex][zindex][tabi
00508                 ndex + 1)) +
00509                 fmet2 * (f1 * MagTableZz[j][metindex + 1][zindex][
00510                 tabindex] +
00511                 f2 * MagTableZz[j][metindex + 1][zindex][
00512                 tabindex + 1])));
00513 #ifdef OUTPUT_MOMAF_INPUTS
00514             //interpolation between the points found by find_interpolated_lum
00515             Gal[p].dObsLum[j][outputbin] +=
00516             X1 * exp(X2 * (fmet1 * (f1 * MagTableZz[j][metindex][zind
00517             ex+1][tabindex] +
00518             f2 * MagTableZz[j][metindex][zindex+1][ta
00519             bindex + 1]) +
00520             fmet2 * (f1 * MagTableZz[j][metindex + 1]
00521             [zindex+1][tabindex] +
00522             f2 * MagTableZz[j][metind
00523             ex + 1][zindex+1][tabindex + 1])));
00524 #endif
00525             /*compute a luminosity to be used for extinction due to young birth
00526             clouds*/
00527             if (age <= tbc)
00528             {
00529                 //interpolation between the points found by find_interpolated_l
00530                 um
00531                 Gal[p].ObsYLum[j][outputbin] +=

```

```

00512             X1 * exp(X2 * (fmet1 * (f1 * MagTableZz[j][metindex][zindex] [
00513                 tabindex] +
00514                 tabindex + 1)) +
00515                 ex][tabindex] +
00516                 fmet2 * (f1 * MagTableZz[j][metindex + 1][zind
00517                     ex][tabindex + 1]));
00518 #ifdef OUTPUT_MOMAF_INPUTS
00519             //interpolation between the points found by find_interpolated_l
00520             um
00521             Gal[p].dObsYLum[j][outputbin] +=
00522                 X1 * exp(X2 * (fmet1 * (f1 * MagTableZz[j][metindex][zindex+1
00523                     ] [tabindex] +
00524                     f2 * MagTableZz[j][metindex][zindex+1
00525                     ] [tabindex + 1]) +
00526                     ex+1][tabindex] +
00527                     fmet2 * (f1 * MagTableZz[j][metindex + 1][zind
00528                     ex+1][tabindex + 1]));
00529             #endif
00530         }
00531
00532
00533
00534 /**@brief Converts snapnum into age - in Myr*/
00535 double NumToTime(int num)
00536 {
00537
00538     return Age[num];
00539 }
00540
00541 /**@brief Converts snapnum into redshift*/
00542 double RedshiftObs(int num)
00543 {
00544     return ZZ[num];
00545 }
00546
00547
00548 /**@brief gets the maximum of two numbers. */
00549 double dmax(double x, double y)
00550 {
00551     if(x > y)
00552         return x;
00553     else
00554         return y;
00555 }
00556
00557
00558 /**@brief Calculates the virial mass: \f$M_{\rm crit200}\f$ for central halos
00559     *      with \f$M_{\rm crit200}\f$ or Len*PartMass for central halos without.
00560     */
00561 double get_virial_mass(int halonr, int p)
00562 {
00563     if(halonr == Halo[halonr].FirstHaloInFOFgroup && Halo[halonr].M_Crit200)
00564         return Halo[halonr].M_Crit200; /* take spherical overdensity mass estima
00565     te */
00566     else
00567         return Halo[halonr].Len * PartMass;
00568

```

```

00568 }
00569
00570
00571
00572 /**@brief Calculates the virial velocity from the virial mass and radius.
00573 *
00574 * Calculates virial velocity:
00575 *    $V_{\rm vir} = \sqrt{\frac{GM}{R_{\rm vir}}}$ 
00576
00577 double get_virial_velocity(int halonr,int p)
00578 {
00579     return sqrt(G * get_virial_mass(halonr,p) / get_virial_radius(halonr,p));
00580 }
00581
00582
00583
00584 /**@brief Calculates virial radius from a critical overdensity
00585 *
00586 * Calculates virial radius using:
00587 *    $R_{\rm vir} = \left(\frac{3M}{4\pi}\right)^{1/3} \frac{\rho_c}{\Delta_c}$ .
00588 *
00589 * From which, assuming  $\Delta_c=200$ ,
00590 *    $R_{\rm vir} = \left(\frac{3M}{4\pi \cdot 200 \cdot \rho_c}\right)^{1/3}$ 
00591 */
00592 double get_virial_radius(int halonr, int p)
00593 {
00594     double zplus1, hubble_of_z, rhocrit, fac;
00595
00596
00597     zplus1 = 1 + ZZ[Halo[halonr].SnapNum];
00598     /*get H for current z*/
00599     hubble_of_z =
00600         Hubble * sqrt(Omega * zplus1 * zplus1 * zplus1 + (1 - Omega - OmegaLambda) *
00601             zplus1 * zplus1 +
00602                 OmegaLambda);
00603
00604     rhocrit = 3 * hubble_of_z * hubble_of_z / (8 * M_PI * G);
00605     fac = 1 / (200 * 4 * M_PI / 3.0 * rhocrit);
00606     return pow(get_virial_mass(halonr,p) * fac, 1.0 / 3);
00607
00608
00609
00610 /**@brief Converts luminosities into magnitudes
00611 *
00612 * Converts luminosities into magnitudes:
00613 *    $M = -2.5 \log_{10}(L)$ 
00614 double lum_to_mag(double lum)
00615 {
00616     if(lum > 0)
00617         return -2.5 * log10(lum);
00618     else
00619         return 99.0;
00620 }
00621
00622 #ifdef H2FORMATION
00623
00624 /** @brief Model the formation of molecular gas. Created by Qi,
00625 *          but never used. The table is given by Krumholz. */
00626
00627 double cal_H2(int p)
00628
00629 {
00630     double h2mass,metallicity,rho;
00631     double f1,f2,fmet1,fmet2;
00632     int tabindex,metindex;
00633

```

```

00634     metalicity=get_metallicity(Gal[p].ColdGas, Gal[p].MetalsColdGas);
00635     if (metalicity <1.e-8)
00636         metalicity=-9.;
00637     else
00638         metalicity=log10(metalicity/0.02);
00639     if (Gal[p].ColdGas < 1.e-8)
00640         rho=-99.;
00641     else
00642         rho=log10(Gal[p].ColdGas/(Gal[p].GasDiskRadius*Gal[p].GasDiskRadius)/M_PI);
00643     rho=rho -2 +log10(0.73); /* in the unit of log(Msun/pc^-2)*/
00644
00645     find_interpolate_h2(metalicity,rho,&tabindex,&metindex,&f1,&f2,&fmet1,&fmet2);
00646
00647     h2mass = f1*(fmet1*H2[tabindex][metindex]+fmet2*H2[tabindex][metindex+1])+f2*(f
00648     met1*H2[tabindex+1][metindex]+fmet2*H2[tabindex+1][metindex+1]);
00649     h2mass *=Gal[p].ColdGas;
00650
00651 } 
00652
00653 #endif
00654
00655
00656
00657 /**@brief Updates properties of central galaxies.
00658 *
00659 * \f$M_{\rm vir}\f$, \f$R_{\rm vir}\f$ and \f$V_{\rm vir}\f$ are only
00660 * updated for type 0's. Once galaxies become satellites these quantities
00661 * stay unchanged, so will be the values at infall.
00662 *
00663 * If type = 0 then the HotRadius is the Viral Radius, which will be used in
00664 * the cooling recipe.
00665 *
00666 * Other infall information will not be used for central galaxies so we do not
00667 * care whether they carry the correct values. */
00668 void update_centralgal(int ngal,int halonr)
00669 {
00670     int j;
00671     Gal[ngal].Type = 0;
00672
00673     Gal[ngal].InfallVmax = Halo[halonr].Vmax;
00674     Gal[ngal].Rvir = get_virial_radius(halonr,ngal);
00675     Gal[ngal].Vvir = get_virial_velocity(halonr,ngal);
00676     Gal[ngal].Mvir = get_virial_mass(halonr,ngal);
00677     Gal[ngal].InfallSnap = Halo[halonr].SnapNum;
00678
00679     /* if type =0 then hotradius =viral radius, this will be used in the cooling re
00680     cipe; */
00681     Gal[ngal].HotRadius = Gal[ngal].Rvir;
00682     Gal[ngal].MergeOn= 0;
00683     for (j=0;j<3;j++)
00684         Gal[ngal].HaloSpin[j] = Halo[halonr].Spin[j];
00685
00686 /**@brief Updates properties of type 1 galaxies.
00687 *
00688 * If the galaxy has just become a satellite (its type hasn't yet been reset
00689 * from 0 to 1), the fraction of hot gas to dark matter halo mass is recorded
00690 * \f$HotFrac=\frac{M_{\rm hot}}{(M_{\rm vir})}\f$.
00691 *
00692 * If MERGE01 = 1, then a dynamical friction decay time scale is calculated
00693 * for type 1's (as is done for type 2 - introduced for millennium II where the
00694 * increased resolution means type 1 always retain some dark matter and orbit
00695 * around for a long time). This is only calculated when the baryonic mass of
00696 * the type 1 becomes larger than its dark matter mass. The code finds the type
00697 * 0 to which this galaxy should merge and then sets up the merging clock.
00698 */

```

```

00699 void update_type_1(int ngal, int halonr, int cenngal,int prog)
00700 {
00701     int current,descendant,firstdes;
00702     int j;
00703
00704     if (Gal[ngal].Type ==0)
00705     {
00706
00707         if (Gal[ngal].HotGas >0.0)
00708             Gal[ngal].HotFrac = Gal[ngal].HotGas/Gal[ngal].Mvir;
00709         else
00710             Gal[ngal].HotFrac = 0.0;
00711     }
00712     Gal[ngal].Type = 1;
00713
00714 #ifdef MERGE01
00715
00716     if (Gal[ngal].MergeOn == 0)
00717     {
00718         /*If baryonic mass > dark matter mass*/
00719         if (Gal[ngal].ColdGas+Gal[ngal].StellarMass > Halo[halonr].Len*PartMass)
00720         {
00721
00722             current= halonr;
00723             descendant = Halo[halonr].Descendant;
00724             firstdes = Halo[Halo[halonr].FirstHaloInFOFgroup].Descendant;
00725
00726
00727             /* In case this is the last snapnum (firstdes == -1), it means that we
00728             tracked all
00729             * the way down to redshift =0 and mergeon should be trun o
00730             n. Otherwise, it is the
00731             * case that the current halo and the corresponding fof cen
00732             tral subhalo are
00733             * "mysteriously" lost in the dark matter simulation at an
00734             intermediate redshift
00735             * and this galaxy would not be treated further anyway furt
00736             her. Thus the mergeon
00737             * value is irrelevant. Here mergeon is set to 1. */
00738             if (descendant == -1)
00739                 Gal[ngal].MergeOn = 1;
00740
00741             /* checks when the galaxy "disappears" (when it merges with the type 0)
00742             in order to get
00743             * the type 0 ID into which this type 1 will be merged. */
00744             while(descendant >= 0)
00745             {
00746
00747                 if (firstdes != Halo[firstdes].FirstHaloInFOFgroup)
00748                     break;
00749
00750                 if (Halo[descendant].FirstHaloInFOFgroup != Halo[firstdes].FirstHal
00751                 oInFOFgroup)
00752                     break;
00753
00754                 if (descendant != Halo[descendant].FirstHaloInFOFgroup && current =
00755                 = Halo[descendant].FirstProgenitor)
00756                     if (Gal[ngal].ColdGas+Gal[ngal].StellarMass < Halo[descendant].
00757 Len * PartMass)
00758                         break;
00759
00760                 if (descendant == Halo[descendant].FirstHaloInFOFgroup && current =
00761                 = Halo[descendant].FirstProgenitor)
00762                     break;
00763
00764                 if (descendant == Halo[descendant].FirstHaloInFOFgroup && current

```

```

        != Halo[descendant].FirstProgenitor)
00756            {
00757                Gal[ngal].MergeOn = 1;
00758                break;
00759            }
00760
00761            if (descendant != Halo[descendant].FirstHaloInFOFgroup && current !=
00762            = Halo[descendant].FirstProgenitor)
00763                break;
00764
00765            current=descendant;
00766            firstdes = Halo[firstdes].Descendant;
00767            descendant=Halo[descendant].Descendant;
00768
00769            /* In case this is the last snapnum (firstdes == -1), it means that
00770            we tracked all
00771            * the way down to redshift =0 and mergeon should be trun on. Other
00772            wise, it is the
00773            * case that the current halo and the corresponding fo central sub
00774            halo are
00775            * "mysteriously" lost in the dark matter simulation at an intermed
00776            iate redshift
00777            * and this galaxy would not be treated further anyway further. Thu
00778            s the mergeon
00779            * value is irrelevant. Here mergeon is set to 1. */
00780            if(firstdes == -1)
00781            {
00782                if (descendant == -1)
00783                    Gal[ngal].MergeOn = 1;
00784                break;
00785            }
00786
00787            /*test*/
00788            if(descendant < 0 && Gal[ngal].MergeOn != 1)
00789            {
00790                printf("something strange here \n");
00791                printf("haloNr %d , firstdes %d, current %d, descendant %d \n",halo
00792                nr, firstdes, current, descendant);
00793                // exit(0);
00794            }
00795            /*Sets up the dynamical friction decay merging clock as for type 2 gala
00796            xies. */
00797            if (descendant < 0 || Gal[ngal].MergeOn == 1)
00798            {
00799                Gal[ngal].MergeOn = 1;
00800                Gal[ngal].MergTime = estimate_merging_time(prog,Halo[Halo[halonr].F
00801                irstHaloInFOFgroup].FirstProgenitor,ngal);
00802                // Gal[ngal].MergTime = estimate_merging_time(halonr,Halo[halonr
00803                ].FirstHaloInFOFgroup,ngal);
00804                Gal[ngal].MergTime -= NumToTime(Halo[halonr].SnapNum) - NumToTime(
00805                Halo[prog].SnapNum);
00806                /* it is for calculating the position of type 2 */
00807                Gal[ngal].OriMergTime=Gal[ngal].MergTime;
00808            }
00809        }
00810    #endif
00811
00812    /*Mvir, Rvir and Vvir keep their value fixed after infall*/
00813
00814
00815
00816
00817
00818 /**@brief Updates properties of type 2 galaxies.
00819 *
00820 * Sets HotFrac and Hot Radius to 0, since all the hot gas has been stripped.

```

```

00811 * Calls estimate_merging_time to get the merging time scale, calculated for
00812 * the orbital decay due to dynamical friction, since this galaxy has lost its
00813 * dark matter halo and its position cannot be tracked. */
00814 void update_type_2(int ngal,int halonr, int prog,int mostmassive)
00815 {
00816     if (Gal[ngal].Type == 0)
00817     {
00818         /* all hot gas is stripped */
00819         Gal[ngal].HotFrac = 0.0;
00820         Gal[ngal].HotRadius = 0.0;
00821     }
00822     /* Estimate remaining merging timescale, and make the galaxy a type=2 object. */
00823     /
00824     Gal[ngal].Type = 2;
00825     if (Gal[ngal].MergeOn == 0)
00826     {
00827         Gal[ngal].MergTime = estimate_merging_time(prog,mostmassive,ngal);
00828         Gal[ngal].MergTime -= NumToTime(Halo[halonr].SnapNum) - NumToTime(Halo[prog]
00829             ].SnapNum);
00830         /* it is for calculating the position of type 2 */
00831         Gal[ngal].OriMergTime=Gal[ngal].MergTime;
00832     }
00833
00834
00835 /*TODO this function is never called.*/
00836 /**@brief Not Used - update_hot_frac instead
00837 *
00838 * Updates the fraction of hot gas attached on each dark matter particles
00839 * of the subhalo. */
00840 void update_hotgas(int ngal,int centralgal)
00841 {
00842     int p;
00843     double d;
00844
00845     for (p = 0; p < ngal; p++)
00846     {
00847         d = sqrt(pow(Gal[centralgal].Pos[0] - Gal[Gal[p].CentralGal].Pos[0], 2.0) +
00848                 pow(Gal[centralgal].Pos[1] - Gal[Gal[p].CentralGal].Pos[1], 2.0) +
00849                 pow(Gal[centralgal].Pos[2] - Gal[Gal[p].CentralGal].Pos[2], 2.0))/
00850                 (1 + ZZ[Halo[Gal[centralgal].HaloNr].SnapNum]);
00851
00852         if (Gal[p].Type == 1 && d < Gal[centralgal].Rvir)
00853         {
00854             if (Gal[p].HotGas < 1.e-8)
00855                 Gal[p].HotFrac=0.0;
00856             else
00857                 Gal[p].HotFrac=Gal[p].HotGas/(Gal[p].Len*PartMass);
00858         }
00859     }
00860
00861

```

4.35 code/recipe_reincorporation.c File Reference

[recipe_reincorporation.c](#) calculates the fraction of ejected gas that gets reincorporated into the hot fraction per timestep.

Functions

- void `reincorporate_gas` (int centralgal, int p, double dt)
reincorporates ejected gas back into the central galaxy hot halo

4.35.1 Detailed Description

2 options are available to reincorporate the gas from the external reservoir:

- $\dot{M}_{\text{eject}} = -\gamma \left(\frac{M_{\text{ejected}}}{t_{\text{dyn},h}} \right)$ (Eq. 3 Delucia2004) (`ReIncorporationRecipe == 1`);
- $\dot{M}_{\text{eject}} = -\gamma \left(\frac{V_{\text{vir}}}{220 \text{km/s}} \right) \left(\frac{M_{\text{ejected}}}{t_{\text{dyn},h}} \right)$ (Eq. 23 Guo2010) (`ReIncorporationRecipe == 2`)

Note that `ReIncorporationRecipe == 1` doesn't correspond to Delucia2007 anymore as Qi changed the place where the reincorporated gas ends. In delucia2007 only central galaxies reincorporated. Now satellites outside central `Rvir` can also do so.

Definition in file `recipe_reincorporation.c`.

4.35.2 Function Documentation

4.35.2.1 void reincorporate_gas (int centralgal, int p, double dt)

Definition at line 34 of file `recipe_reincorporation.c`.

References `GALAXY::EjectedMass`, `Gal`, `get_metallicity()`, `Halo`, `GALAXY::HaloNr`, `GALAXY::HotGas`, `GALAXY::MetalsEjectedMass`, `GALAXY::MetalsHotGas`, `GALAXY::Pos`, `ReIncorporationFactor`, `ReIncorporationRecipe`, `GALAXY::Rvir`, `halo_data::SnapNum`, `GALAXY::Type`, `update_hot_frac()`, `GALAXY::Vvir`, and `ZZ`.

Referenced by `main()`.

4.36 code/recipe_reincorporation.c

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <string.h>
00004 #include <math.h>
00005 #include <time.h>
00006
00007 #include "allvars.h"
00008 #include "proto.h"
00009
00010 // TODO this could be in recipe_starformation_and_feedback
00011
00012 /** @file recipe_reincorporation.c
00013 *  @brief recipe_reincorporation.c calculates the fraction of ejected gas that
00014 *         gets reincorporated into the hot fraction per timestep.
00015 *
00016 *         2 options are available to reincorporate the gas from the external
00017 *         reservoir:
00018 *             -\f$ \dot{M}_{\text{eject}} = -\gamma \left( \frac{M_{\text{ejected}}}{t_{\text{dyn},h}} \right) \f$
00019 */

```

```

00020 *      (Eq. 3 Delucia2004) (ReIncorporationRecipe == 1);
00021 *      -\f$\dot{M}_{ej} = -\gamma \left( \frac{V_{vir}}{\rm 220 km/s} \right) \left( \frac{M_{ejected}}{t_{dyn,h}} \right) \f$ 
00022 *
00023 *
00024 *      (Eq. 23 Guo2010) (ReIncorporationRecipe == 2)
00025 *
00026 *      Note that ReIncorporationRecipe == 1 doesn't correspond to Delucia2007
00027 *
00028 *      anymore as Qi changed the place where the reincorporated gas ends.
00029 *      In delucia2007 only central galaxies reincorporated. Now satellites
00030 *      outside central Rvir can also do so.
00031 */
00032 /** @brief reincorporates ejected gas back into the central galaxy hot halo */
00033
00034 void reincorporate_gas(int centralgal, int p, double dt) {
00035     double reincorporated, metallicity, dis;
00036     double hottmp;
00037
00038     /* calculate distance of satellite to central galaxy, used to check if satellite is inside
00039     * central Rvir */
00040     dis = sqrt(pow(Gal[centralgal].Pos[0] - Gal[p].Pos[0], 2.0) + pow(
00041                 Gal[centralgal].Pos[1] - Gal[p].Pos[1], 2.0) + pow(Gal[ce
00042                 ntrgal].Pos[2]
00043                 - Gal[p].Pos[2], 2.0)) / (1 + ZZ[Halo[Gal[centralgal].
00044                     HaloNr].SnapNum]);
00045
00046     /* Delucia2007 -> mdot_eject=-gama_ej * m_ejected/tdyn */
00047     if (ReIncorporationRecipe == 1) // TODO avoid using magic numbers, define
00048         global constants instead
00049     {
00050         reincorporated =
00051             ReIncorporationFactor * Gal[centralgal].EjectedMass / (Gal[centralgal]
00052                 .Rvir / Gal[centralgal].Vvir) * dt;
00053
00054     }
00055
00056     /* Guo2010 -> mdot_eject=-gama_ej * m_ejected/tdyn * Vvir/220 */
00057     if (ReIncorporationRecipe == 2) // TODO avoid using magic numbers, define
00058         global constants instead
00059     {
00060         reincorporated =
00061             ReIncorporationFactor * Gal[p].EjectedMass / (Gal[p].Rvir / Gal[p].
00062                 Vvir) * dt * pow(Gal[p].Vvir/220.,1.);
00063
00064     if (reincorporated > Gal[p].EjectedMass)
00065         reincorporated = Gal[p].EjectedMass;
00066
00067     /*Option for Guo2010 to keep track of the hot gas in type 1's*/
00068     if (Gal[p].Type == 1)
00069         update_hot_frac(p, reincorporated, Gal[p].HotGas);
00070
00071     /*Update ejected and hot gas contents*/
00072     metallicity = get_metallicity(Gal[p].EjectedMass, Gal[p].
00073         MetalsEjectedMass);
00074     Gal[p].EjectedMass -= reincorporated;
00075     Gal[p].MetalsEjectedMass -= metallicity * reincorporated;
00076     Gal[p].HotGas += reincorporated;
00077     Gal[p].MetalsHotGas += metallicity * reincorporated;
00078 }

```

00078

4.37 code/recipe_starformation_and_feedback.c File Reference

[recipe_starformation_and_feedback.c](#) computes the amount of stars formed from the cold gas, the amount of gas reheated from cold to hot and the amount of gas ejected from hot to external.

Functions

- void [starformation_and_feedback](#) (int p, int centralgal, double time, double dt, int halonr)
Main recipe, calculates the fraction of cold gas turned into stars due to star formation; the fraction of mass instantaneously recycled and returned to the cold gas; the fraction of gas reheated from cold to hot, ejected from hot to external and returned from ejected to hot due to SN feedback.
- void [update_from_star_formation](#) (int p, double stars, double metallicity)
Updates the different components due to star formation: mass and metals in stars and cold gas and stellar spin.
- void [update_from_feedback](#) (int p, int centralgal, double reheated_mass, double ejected_mass, double ejected_sat, double metallicity)
Updates cold, hot and external gas components due to SN reheating and ejection.
- void [check_disk_instability](#) (int p)
Checks for disk stability using the Mo, Mao & White (1998) criteria.
- void [cal_gas_recycle](#) (int ngal)

4.37.1 Detailed Description

The routine is divided in two parts, star formation and SN feedback, with a number of different implementations controlled by input parameters.

There are 4 options for the **Star Formation Recipe**: Kauffmann 1993 (0), Kauffmann 1996c (1) - used in Delucia2007,

$$0 - \dot{M}_\star = \alpha_{SF,old} \left(\frac{V_{vir}}{1000} \right)^0 \frac{M_{cold}}{t_{dyn,h}} = \alpha_{SF,old} \frac{M_{cold}}{t_{dyn,h}} \quad (\text{Eq. 3 Kauffmann1993}) \quad (\text{StarFormationRecipe} = 0)$$

$$1 - \dot{M}_\star = \alpha_{SF} \frac{(M_{cold} - M_{crit})}{t_{dyn,disk}} \quad (\text{Eq.7 \& 8 Kauffmann1996c}) \quad (\text{StarFormationRecipe} = 1)$$

- where $M_{crit} = 3.8 \times 10^9 \left(\frac{V_{vir}}{200 \text{ kms}^{-1}} \right) \left(\frac{r_{disk}}{10 \text{ kpc}} \right) M_\odot$ is derived from a critical gas surface density:
 $\Sigma_{crit}(R) = 120 \left(\frac{V_{vir}}{200 \text{ kms}^{-1}} \right) \left(\frac{R}{\text{kpc}} \right)^{-1} M_\odot \text{ pc}^{-2}$.

$$2 - M_{crit} = 3.8 \times 10^9 \left(\frac{V_{max}}{200 \text{ kms}^{-1}} \right) \left(\frac{r_{disk}}{10 \text{ kpc}} \right) M_\odot \quad (\text{Eq. 16 Guo2010}) \quad (\text{StarFormationRecipe} = 2),$$

- same as 1 but using V_{max} or $V_{max,infall}$ instead of V_{vir} and allowing SF in satellites.

$$3 - \dot{M}_\star = \alpha_{SF} \frac{M_{H_2}}{t_{dyn,h}}$$

- unpublished version by Qi based on Krumholz (StarFormationRecipe =3).

There are 3 options for the **SN Feedback Recipe**:

0 - Delucia 2004 retention or ejection scheme (0),

$$1 - \Delta m_{\text{reheated}} = \epsilon_{\text{disk}} \Delta m_{\star}$$

The amount of energy released by supernova during the formation of Δm_{\star} stars is $\Delta E_{\text{SN}} = 0.5 \epsilon_{\text{halo}} \Delta m_{\star} V_{\text{SN}}^2$, $V_{\text{SN}} = 630 \text{ km s}^{-1}$. Any excess energy left over from reheating the cold gas is used to eject a mass of gas from the galaxy:

$$\Delta m_{\text{ejected}} = \left(\epsilon_{\text{halo}} \frac{V_{\text{SN}}^2}{V_{\text{vir}}^2} - \epsilon_{\text{disk}} \right) \Delta m_{\star},$$

(Eqs. 17, 18, 19 & 20 Croton2006)(FeedbackRecipe = 1) based on moderate Delucia 2004 ejection scheme.

$$2 - \epsilon_{\text{disk}} = \epsilon \left[0.5 + \left(\frac{V_{\text{max}}}{70 \text{ km/s}} \right)^{-\beta_1} \right], \quad \epsilon_{\text{halo}} = \eta \left[0.5 + \left(\frac{V_{\text{max}}}{70 \text{ km/s}} \right)^{-\beta_2} \right] \quad (\text{Eqs. } 19 \text{ & } 21 \text{ Guo2010})(\text{FeedbackRecipe} = 2) \text{ same as FeedbackRecipe} = 1 * V_{\text{max}} \text{ dependence.}$$

Also, Guo2010 allowed for type 1 satellite to have gas cycles and receive gas from their own satellites when these are outside Rvir of the type 0.

Definition in file [recipe_starformation_and_feedback.c](#).

4.37.2 Function Documentation

4.37.2.1 void cal_gas_recycle (int *ngal*)

4.37.2.2 void check_disk_instability (int *p*)

Calculates the stability of the stellar disk as discussed in Mo, Mao & White (1998). For unstable stars, the required amount is transferred to the bulge to make the disk stable again. Mass, metals and luminosities updated. After Guo2010 the bulge size is followed and needs to be updated. Eq 34 & 35 in Guo2010 are used.

Definition at line 654 of file [recipe_starformation_and_feedback.c](#).

References [GALAXY::BulgeMass](#), [GALAXY::BulgeSize](#), [GALAXY::dObsLum](#), [GALAXY::dObsLumBulge](#), [GALAXY::dObsYLum](#), [GALAXY::dObsYLumBulge](#), [G](#), [Gal](#), [get_metallicity\(\)](#), [GALAXY::InfallVmax](#), [GALAXY::Lum](#), [GALAXY::LumBulge](#), [GALAXY::MetalsBulgeMass](#), [GALAXY::MetalsStellarMass](#), [GALAXY::ObsLum](#), [GALAXY::ObsLumBulge](#), [GALAXY::ObsYLum](#), [GALAXY::ObsYLumBulge](#), [GALAXY::StellarDiskRadius](#), [GALAXY::StellarMass](#), [GALAXY::Type](#), [update_bulge_from_disk\(\)](#), [GALAXY::Vmax](#), [GALAXY::YLum](#), and [GALAXY::YLumBulge](#).

Referenced by [deal_with_galaxy_merger\(\)](#), and [starformation_and_feedback\(\)](#).

4.37.2.3 void starformation_and_feedback (int *p*, int *centralgal*, double *time*, double *dt*, int *halonr*)

Variables: reff=Rdisk, tdyn=Rdisk/Vmax, strdot=Mstar_dot, stars=strdot*dt

Definition at line 83 of file [recipe_starformation_and_feedback.c](#).

References [add_to_luminosities\(\)](#), [cal_H2\(\)](#), [GALAXY::CentralGal](#), [check_disk_instability\(\)](#), [GALAXY::ColdGas](#), [DiskRadiusMethod](#), [EjectionOn](#), [EjectPreVelocity](#), [EjectSlope](#), [Energy_in_Reheat\(\)](#), [EnergySNcode](#), [EtaSNcode](#), [FeedbackEjectionEfficiency](#), [FeedbackEpsilon](#), [FeedbackRecipe](#), [FeedbackReheatingEpsilon](#), [FracZtoHot](#), [Gal](#), [GALAXY::GasDiskRadius](#), [get_metallicity\(\)](#),

`get_stellar_disk_radius()`, `Halo`, `GALAXY::HaloNr`, `GALAXY::InfallVmax`, `ListOutputSnaps`, `GALAXY::MetalsColdGas`, `GALAXY::MetalsHotGas`, `GALAXY::Pos`, `ReheatPreVelocity`, `ReheatSlope`, `GALAXY::Rvir`, `GALAXY::Sfr`, `SfrAlpha`, `SfrEfficiency`, `SfrLawPivotVelocity`, `SfrLawSlope`, `GALAXY::SnapNum`, `halo_data::SnapNum`, `StarFormationRecipe`, `GALAXY::StellarMass`, `TrackDiskInstability`, `GALAXY::Type`, `update_from_feedback()`, `update_from_star_formation()`, `GALAXY::Vmax`, `GALAXY::Vvir`, `Yield`, and `ZZ`.

Referenced by `main()`.

4.37.2.4 void update_from_feedback(int *p*, int *centralgal*, double *reheated_mass*, double *ejected_mass*, double *ejected_sat*, double *metallicity*)

Definition at line 445 of file `recipe_starformation_and_feedback.c`.

References `GALAXY::CentralGal`, `GALAXY::ColdGas`, `GALAXY::EjectedMass`, `EjectionOn`, `EjectionRecipe`, `FeedbackRecipe`, `Gal`, `get_metallicity()`, `Halo`, `GALAXY::HaloNr`, `GALAXY::HotGas`, `GALAXY::HotRadius`, `GALAXY::MetalsColdGas`, `GALAXY::MetalsEjectedMass`, `GALAXY::MetalsHotGas`, `GALAXY::Pos`, `GALAXY::Rvir`, `halo_data::SnapNum`, `GALAXY::Type`, `update_hot_frac()`, and `ZZ`.

Referenced by `collisional_starburst_recipe()`, and `starformation_and_feedback()`.

4.37.2.5 void update_from_star_formation(int *p*, double *stars*, double *metallicity*)

Definition at line 422 of file `recipe_starformation_and_feedback.c`.

References `GALAXY::BulgeMass`, `GALAXY::ColdGas`, `DiskRadiusMethod`, `Gal`, `GALAXY::GasSpin`, `get_stellar_disk_radius()`, `GALAXY::MetalsColdGas`, `GALAXY::MetalsStellarMass`, `RecycleFraction`, `GALAXY::StellarMass`, and `GALAXY::StellarSpin`.

Referenced by `collisional_starburst_recipe()`, and `starformation_and_feedback()`.

4.38 code/recipe_starformation_and_feedback.c

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <string.h>
00004 #include <math.h>
00005 #include <time.h>
00006
00007 #include "allvars.h"
00008 #include "proto.h"
00009
00010 /** @file recipe_starformation_and_feedback.c
00011 *  @brief recipe_starformation_and_feedback.c computes the amount of stars
00012 *         formed from the cold gas, the amount of gas reheated from cold to hot
00013 *         and the amount of gas ejected from hot to external.
00014 *
00015 * The routine is divided in two parts, star formation and SN feedback, with a
00016 * number of different implementations controlled by input parameters.
00017 *
00018 * There are 4 options for the <B>Star Formation Recipe</B>: Kauffmann 1993 (0),
00019 * Kauffmann 1996c (1) - used in Delucia2007,
00020 *
00021 * 0 -\f$ \dot{M}_{\rm star} = \alpha_{\rm SF,old}
00022 *   \left(\frac{V_{\rm vir}}{1000}\right)^0\frac{M_{\rm cold}}{t_{\rm dyn,h}}\f$
00023 *   =\alpha_{\rm SF,old}\frac{M_{\rm cold}}{t_{\rm dyn,h}}\f$
```

```

00024 *      (Eq. 3 Kauffmann1993) (StarFormationRecipe = 0)
00025 *
00026 *      1 -\f$\dot{M}_{\star} = \alpha_{\rm SF}
00027 *          \frac{(M_{\rm cold}-M_{\rm crit})}{t_{\rm dyn,disk}}\f$
00028 *          (Eq. 7 & 8 Kauffmann1996c) (StarFormationRecipe = 1)
00029 *          - where \f$M_{\rm crit}=3.8\times 10^9
00030 *              \left(\frac{V_{\rm vir}}{200}, \rm km\,s}^{-1}\right)
00031 *              \left(\frac{r_{\rm disk}}{10}, \rm kpc\right) M_{\odot}\f$
00032 *          is derived from a critical gas surface density:
00033 *          \f$\Sigma_{\rm crit}(R)=
00034 *              120\left(\frac{V_{\rm vir}}{200}, \rm km\,s}^{-1}\right)
00035 *              \left(\frac{R}{\rm kpc}\right)^{-1} M_{\odot} \rm pc}^{-2}\f$.
00036 *
00037 *      2 -\f$M_{\rm crit}=3.8\times 10^9
00038 *          \left(\frac{V_{\rm max}}{200}, \rm km\,s}^{-1}\right)
00039 *          \left(\frac{r_{\rm disk}}{10}, \rm kpc\right) M_{\odot}\f$
00040 *          (Eq. 16 Guo2010) (StarFormationRecipe = 2), \n
00041 *          - same as 1 but using \f$V_{\rm max}\f$ or \f$V_{\rm max,infall}\f$ instead of \f$V_{\rm vir}\f$ and allowing SF in satellites.
00042 *
00043 *
00044 *      3 -\f$\dot{M}_{\star} = \alpha_{\rm SF}\frac{M_{\rm H_2}}{t_{\rm dyn,h}}\f$ 
00045 *          - unpublished version by Qi based on Krumholz (StarFormationRecipe = 3).

00046 *
00047 * There are 3 options for the <B>SN Feedback Recipe</B>:
00048 *
00049 * 0 - Delucia 2004 retention
00050 * or ejection scheme (0),
00051 *
00052 * 1 - \f$\Delta m_{\rm reheat}=\epsilon_{\rm disk}\Delta m_{\star}\f$\n
00053 * The amount of energy released by supernova during the formation of
00054 * \f$\Delta m_{\star}\f$ stars is
00055 * \f$\Delta E_{\rm SN}=0.5\epsilon_{\rm halo}\Delta m_{\star} V_{\rm SN}^2\f$,
00056 * \f$V_{\rm SN}=630\rm km\,s}^{-1}\f$. Any excess energy left over from
00057 * reheating the cold gas is used to eject a mass of gas from the galaxy:\n
00058 * \f$\Delta m_{\rm ejected}=\left(\epsilon_{\rm halo}\frac{V_{\rm vir}}{SN}^2(V_{\rm vir}^2-\epsilon_{\rm disk})\right)\Delta m_{\star}\f$, \n
00059 * (Eqs. 17, 18, 19 & 20 Croton2006) (FeedbackRecipe = 1) based on moderate
00060 * Delucia 2004 ejection scheme.
00061 *
00062 *
00063 * 2 - \f$\epsilon_{\rm disk}=\epsilon_{\rm halo}\bigg[0.5+\left(\frac{V_{\rm max}}{70\rm km/s}\right)^{-\beta_1}\bigg]\f$,
00064 * \f$\epsilon_{\rm halo}=\eta\bigg[0.5+\left(\frac{V_{\rm max}}{70\rm km/s}\right)^{-\beta_2}\bigg]\f$,
00065 * (Eqs. 19 & 21 Guo2010) (FeedbackRecipe = 2)
00066 * same as FeedbackRecipe = 1 * Vmax dependence.
00067 *
00068 *
00069 *
00070 * Also, Guo2010 allowed for type 1 satellite to have gas cycles and receive
00071 * gas from their own satellites when these are outside Rvir of the type 0.
00072 * */
00073
00074 /*TO DO - eliminate duplications by merging CentralVvir and MergeCentralVvir
00075 * and centralgal and Gal[p].centralgal into single variables with value
00076 * defined by a conditions on dis */
00077
00078 /** @brief Main recipe, calculates the fraction of cold gas turned into stars due
00079 * to star formation; the fraction of mass instantaneously recycled and
00080 * returned to the cold gas; the fraction of gas reheated from cold to hot
00081 * ejected from hot to external and returned from ejected to hot due to
00082 * SN feedback. */
00083 void starformation_and_feedback(int p, int centralgal, double time, double dt, int
halonr)

```

```

00084 {
00085     /*! Variables: reff=Rdisk, tdyn=Rdisk/Vmax, strdot=Mstar_dot, stars=strdot
00086     t*dt*/
00086     double reff, tdyn, strdot, stars, reheated_mass, ejected_mass,fac, vesc, cold_c
00087     rit, metallicity,H2Mass,
00087     metallicitySF, CentralVvir,MergeCentralVvir,d,HeatSatToCen;
00088     float tmp,refrac,dis;
00089     double ejected_sat;
00090 #ifndef UseFullSfr
00091     int outputbin;
00092 #endif
00093
00094 /*STAR FORMATION RECIPES */
00095
00096     /* In Guo2010 tdyn is given by Vmax for central galaxies or
00097     * Vmax at infall for satellites*/
00098     if(StarFormationRecipe == 2)
00099     {
00100         reff = Gal[p].GasDiskRadius;
00101         if (Gal[p].Type == 0)
00102             tdyn = Gal[p].GasDiskRadius / Gal[p].Vmax;
00103         else
00104             tdyn = Gal[p].GasDiskRadius / Gal[p].InfallVmax;
00105     }
00106     else //delucia2007
00107     {
00108         reff = Gal[p].GasDiskRadius;
00109         tdyn = Gal[p].GasDiskRadius / Gal[p].Vvir;
00110     }
00111
00112     /* Original kennicut type recipe - Kauffmann (1993)
00113     * Since SFRlaw_slope=0 => tmp=1 and strdot=SfrAlpha*M_cold/tdyn */
00114     if(StarFormationRecipe == 0)
00115     {
00116         tmp = SfrAlpha * pow(Gal[p].Vvir / SfrLawPivotVelocity, SfrLawSlope);
00117         if(tmp <= 1.)
00118             strdot = tmp * Gal[p].ColdGas / tdyn;
00119         else
00120             strdot = Gal[p].ColdGas / tdyn;
00121     }
00122
00123     /* Use critical gas surface density - Kauffmann (1996c)
00124     * cold_crit from eq7 x piR^2, (Vvir in km/s, reff in Mpc/h)
00125     * in units of 10^10Msun/h */
00126     else if(StarFormationRecipe == 1)
00127     {
00128         cold_crit = 0.19 * Gal[p].Vvir * reff;
00129
00130         if(Gal[p].ColdGas > cold_crit)
00131         {
00132             strdot = SfrEfficiency * (Gal[p].ColdGas - cold_crit) / tdyn;
00133         }
00134         else
00135             strdot = 0.0;
00136     }
00137
00138
00139     /* Same as 1 but using Vmax or InfallVmax instead of Vvir */
00140     if(StarFormationRecipe == 2)
00141     {
00142         if (Gal[p].Type == 0)
00143             cold_crit = 0.19 * Gal[p].Vmax * reff;
00144         else
00145             cold_crit = 0.19 * Gal[p].InfallVmax * reff;
00146         //remove the threshold -> cold_crit *= 0.5;
00147         if(Gal[p].ColdGas > cold_crit)
00148         {

```

```

00149         strdot = SfrEfficiency * (Gal[p].ColdGas - cold_crit) / tdyn * pow(Gal[
00150             p].Vvir / SfrLawPivotVelocity, SfrLawSlope);
00151     }
00152     else
00153     {
00154 }
00155 #ifdef H2FORMATION
00156
00157 /* use H2 formation model from Krumholz */
00158 if(StarFormationRecipe == 3)
00159 {
00160
00161     H2Mass=cal_H2(p);
00162     strdot = SfrEfficiency * H2Mass / tdyn;
00163 }
00164
00165 #endif
00166 /*TODO - Note that Units of dynamical time are Mpc/Km/s - dt units are 1e12Yr
s/h
00167     *      lucky enough these are very similar....still the conversion should
00168     *      be mentioned 3.06e19 to 3.15e19 */
00169 stars = strdot * dt;
00170 if(stars < 0.0)
00171 stars = 0.0;
00172
00173
00174 /* SN FEEDBACK RECIPES */
00175
00176 /* In Guo2010 type 1s can eject, reincorporate gas and get gas from their
00177 * own satellites (is not sent to the type 0 galaxy as in Delucia2007),
00178 * for gas flow computations:
00179 * If satellite is inside Rvir of main halo, Vvir of main halo used
00180 * If it is outside, the Vvir of its central subhalo is used. */
00181
00182 CentralVvir = Gal[centralgal].Vvir; // main halo Vvir
00183 MergeCentralVvir = Gal[Gal[p].CentralGal].Vvir; //central subhalo Vvir
00184
00185 /*initialize ejected mass from satellite galaxies*/
00186 ejected_sat=0.0;
00187
00188 /* De Lucia et al. (2004): Retention (EjectionOn=0), Ejection (EjectionOn=1)
00189 * NOTE that the 'wind scheme' is not implemented here */
00190 if(FeedbackRecipe == 0)
00191 {
00192     vesc = Gal[p].Vvir;
00193     reheated_mass = (4 / 3.0) * FeedbackEpsilon * stars * EtaSNcode *
EnergySNcode / (vesc * vesc);
00194 }
00195
00196 /* DeLucia2007: cold-> hot-> ejected (see Martin 1999) */
00197 if(FeedbackRecipe == 1)
00198 {
00199     reheated_mass = FeedbackReheatingEpsilon * stars;
00200 }
00201
00202
00203 /* Feedback depends on the circular velocity of the host halo
00204 * Guo2010 - eq 18 & 19*/
00205 if(FeedbackRecipe == 2)
00206 {
00207     if (Gal[Gal[p].CentralGal].Type == 0)
00208     {
00209         reheated_mass = FeedbackReheatingEpsilon * stars*(.5+1./pow(Gal[Gal[p].
CentralGal].Vmax/ReheatPreVelocity, ReheatSlope));
00210     }
00211 else

```

```

00212         {
00213             reheated_mass = FeedbackReheatingEpsilon * stars*.5+1./pow(Gal[Gal[p].
00214             CentralGal].InfallVmax/ReheatPreVelocity,ReheatSlope));
00215         }
00216
00217
00218     if(reheated_mass < 0.0)
00219     {
00220         printf("Something strange here (SF1)....\n");
00221         exit(32);
00222         reheated_mass = 0.0;
00223     }
00224
00225
00226     dis=
00227     sqrt(pow(Gal[centralgal].Pos[0] - Gal[Gal[p].CentralGal].Pos[0], 2.0) +
00228         pow(Gal[centralgal].Pos[1] - Gal[Gal[p].CentralGal].Pos[1], 2.0) +
00229         pow(Gal[centralgal].Pos[2] - Gal[Gal[p].CentralGal].Pos[2], 2.0))/(1 +
00230         ZZ[Halo[Gal[centralgal].HaloNr].SnapNum]);
00231
/* Determine how much of the energy of SN feedback is used to reheat the
00232 * gas compared to the amount used to eject gas
00233 * Since refrac=1 => reheat mass < stars * V_SN / (Central_Vvir^2)
00234 * assuring that the reheating energy is not bigger than the available E_SN */
00235     refrac = Energy_in_Reheat(p);
00236
00237     if(FeedbackRecipe == 2 || dis > Gal[centralgal].Rvir)
00238     {
00239         if (reheated_mass * MergeCentralVvir * MergeCentralVvir > refrac * stars *
00240             (EtaSNcode * EnergySNcode))
00241             reheated_mass = refrac * stars * (EtaSNcode * EnergySNcode) / (MergeCentr
00242             alVvir * MergeCentralVvir);
00243         else
00244         {
00245             if (reheated_mass * CentralVvir * CentralVvir > refrac * stars * (
00246                 EtaSNcode * EnergySNcode))
00247                 reheated_mass = refrac * stars * (EtaSNcode * EnergySNcode) / (CentralVvi
00248                 r * CentralVvir);
00249
00250             /* cant use more cold gas than is available! so balance SF and feedback */
00251             if((stars + reheated_mass) > Gal[p].ColdGas)
00252             {
00253                 fac = Gal[p].ColdGas / (stars + reheated_mass);
00254
00255                 stars *= fac;
00256                 reheated_mass *= fac;
00257             }
00258
00259             /* Determine ejection (for FeedbackRecipe 2 we have the dependence on Vmax)
00260             * Guo2010 - eq 22
00261             * Note that satellites can now retain gas and have their own gas cycle*/
00262             if(FeedbackRecipe == 2)
00263             {
00264                 //galaxy is not orbiting a subhalo
00265                 if (Gal[Gal[p].CentralGal].Type == 0)
00266                 {
00267                     ejected_mass =
00268                         (FeedbackEjectionEfficiency* (EtaSNcode * EnergySNcode) *stars *min(1
00269                         ./FeedbackEjectionEfficiency,.5+1/pow(Gal[centralgal].Vmax/EjectPreVelocity,
00270                         EjectSlope)) -
00271                         reheated_mass*CentralVvir*CentralVvir) /(CentralVvir*CentralVvir);
00270

```

```

00271         ejected_sat=0.0;
00272     }
00273     else
00274     {
00275
00276         if (dis < Gal[centralgal].Rvir)
00277         {
00278             //no matter where type 1 is, always keeps its gas
00279             ejected_sat=
00280                 (FeedbackEjectionEfficiency * (EtaSNcode * EnergySNcode) *stars *
00281                 min(1./FeedbackEjectionEfficiency,.5+1./pow(Gal[Gal[p].CentralGal].InfallVmax/
00282                     EjectPreVelocity,EjectSlope)) -
00283                     reheated_mass*MergeCentralVvir*MergeCentralVvir)/(MergeCentralVv
00284                     ir*MergeCentralVvir);
00285
00286             ejected_mass = 0.0;
00287
00288         }
00289         else
00290         {
00291             /* Amount that will go into a type 1 instead of the type 0 */
00292             ejected_sat=
00293                 (FeedbackEjectionEfficiency * (EtaSNcode * EnergySNcode) *stars *
00294                 min(1./FeedbackEjectionEfficiency,.5+1./pow(Gal[Gal[p].CentralGal].InfallVmax/
00295                     EjectPreVelocity,EjectSlope)) -
00296                     reheated_mass*MergeCentralVvir*MergeCentralVvir)/(MergeCentralVv
00297                     ir*MergeCentralVvir);
00298
00299             ejected_mass=0.0;
00300
00301         }
00302
00303 /* In De Lucia et al. (2004) only the central galaxy can eject outside the halo
00304
00305 * For the other galaxies it is assumed that the gas reheated by supernovae
00306 * explosions to the virial temperature of the (sub)halo is stripped out
00307 * and immediately added to the hot component of the FOF halo */
00308 if(FeedbackRecipe == 0 && p == centralgal && EjectionOn == 1)
00309 {
00310     ejected_mass = reheated_mass;
00311     reheated_mass = 0.0;
00312 }
00313
00314 /* Ejection in Delucia2007 with the two different Vvir cases added by Guo 2010
00315 * the satellites eject the mass of their central halo/subhalo */
00316 if(FeedbackRecipe == 1)
00317 {
00318     if (dis < Gal[centralgal].Rvir)
00319     {
00320         ejected_mass =
00321             (FeedbackEjectionEfficiency * (EtaSNcode * EnergySNcode) / (CentralVv
00322                 ir * CentralVvir) -
00323                     FeedbackReheatingEpsilon) * stars;
00324         if(ejected_mass < 0.0)
00325             ejected_mass = 0.0;
00326     }
00327     else
00328         ejected_mass =
00329             (FeedbackEjectionEfficiency * (EtaSNcode * EnergySNcode) / (MergeCent

```

```

00329             FeedbackReheatingEpsilon) * stars;
00330         if(ejected_mass < 0.0)
00331             ejected_mass = 0.0;
00332
00333     }
00334     ejected_sat = 0.0;
00335 }
00336
00337 if(ejected_sat < 0.0)
00338 {
00339     ejected_sat = 0.0;
00340 }
00341
00342 if (ejected_mass <0.0)
00343     ejected_mass =0.0;
00344
00345
00346
00347 /* update the star formation rate */
00348 #ifdef UseFullSfr
00349     Gal[p].Sfr[Halo[halonr].SnapNum] += stars / (dt * STEPS);
00350 #else
00351 #ifdef SAVE_MEMORY
00352     /*Sfr=stars/(dt*steps)=strdot*dt/(dt*steps)=strdot/steps -> average over the ST
EPS*/
00353     Gal[p].Sfr += stars / (dt * STEPS);
00354 #else
00355     for(outputbin = 0; outputbin < NOUT; outputbin++)
00356     {
00357         if(Halo[halonr].SnapNum == ListOutputSamps[outputbin])
00358         {
00359             Gal[p].Sfr[outputbin] += stars / (dt * STEPS);
00360             break;
00361         }
00362     }
00363 #endif
00364 #endif
00365
00366 /* update for star formation
00367 * updates Mcold, StellarMass, MetalsMcold and MetalsStellarMass
00368 * in Guo2010 case updates the stellar spin -> hardwired, not an option */
00369 metallicity = get_metallicity(Gal[p].ColdGas, Gal[p].MetalsColdGas);
00370 update_from_starFormation(p, stars, metallicity);
00371
00372 /* Store the value of the metallicity of the cold phase when SF occurs */
00373 metallicitySF = metallicity;
00374
00375 /* Formation of new metals - instantaneous recycling approximation - only SNII
00376 also recompute the metallicity of the cold phase */
00377 // TODO - update metals PRODUCED from star formation - should be inside update-
from star formation
00378 if (FeedbackRecipe == 2)
00379 {
00380     if(Gal[p].ColdGas > 1.0e-8)
00381     {
00382         /*if metals could be sent into the cold phase they could only go to typ
e 1 */
00383         Gal[p].MetalsColdGas += Yield * (1.0 - FracZtoHot) * stars;
00384         Gal[Gal[p].CentralGal].MetalsHotGas += Yield * FracZtoHot * stars;// =0
not used
00385     }
00386     else
00387         Gal[Gal[p].CentralGal].MetalsHotGas += Yield * stars;
00388 }
00389 else
00390 {
    /*if metals could be sent into the cold phase they could only go to typ

```

```

e 0 */
00392     if(Gal[p].ColdGas > 1.0e-8)
00393     {
00394         Gal[p].MetalsColdGas += Yield * (1.0 - FracZtoHot) * stars;
00395         Gal[centralgal].MetalsHotGas += Yield * FracZtoHot * stars; // =0 not u
00396     sed
00397     }
00398     else
00399     {
00400         Gal[centralgal].MetalsHotGas += Yield * stars;
00401     }
00402
00403     metallicity = get_metallicity(Gal[p].ColdGas, Gal[p].MetalsColdGas);
00404
00405     /* Update For Feedback */
00406     /* update cold, hot, ejected gas fractions and respective metallicities
00407      * there are a number of changes introduced by Guo2010 concerning where
00408      * the gas ends up */
00409     update_from_feedback(p, centralgal, reheated_mass, ejected_mass, ejected_sat,me
tallicity);
00410
00411     /* Update the luminosities due to the stars formed */
00412     add_to_luminosities(p, stars, time, metallicitySF);
00413
00414     if(TrackDiskInstability)
00415     {
00416         if(Gal[p].StellarMass > 0.0)
00417             check_disk_instability(p);
00418         if (DiskRadiusMethod == 2)
00419             get_stellar_disk_radius(p);
00420     }
00421
00422 /** @brief Updates the different components due to star formation: mass
00423      *          and metals in stars and cold gas and stellar spin. */
00424 void update_from_starFormation(int p, double stars, double metallicity)
00425 {
00426     int i;
00427
00428     /* Update the Stellar Spin when forming stars */
00429     if (Gal[p].StellarMass-Gal[p].BulgeMass+(1 - RecycleFraction) *stars > 1.e-8)
00430     {
00431         for (i = 0; i < 3; i++)
00432             Gal[p].StellarSpin[i]=((Gal[p].StellarSpin[i])*(Gal[p].StellarMass-Gal[p].
BulgeMass)+(1-RecycleFraction)*stars*Gal[p].GasSpin[i])/(Gal[p].StellarMass+(1 -
RecycleFraction) *stars-Gal[p].BulgeMass);
00433
00434     /* Update Gas and Metals from star formation */
00435
00436     Gal[p].ColdGas -= (1 - RecycleFraction) * stars;
00437     Gal[p].MetalsColdGas -= metallicity * (1 - RecycleFraction) * stars;
00438     Gal[p].StellarMass += (1 - RecycleFraction) * stars;
00439     Gal[p].MetalsStellarMass += metallicity * (1 - RecycleFraction) * stars;
00440
00441     if (DiskRadiusMethod == 2)
00442         get_stellar_disk_radius(p);
00443
00444 /** @brief Updates cold, hot and external gas components due to SN
00445      *          reheating and ejection. */
00446 void update_from_feedback(int p, int centralgal, double reheated_mass, double eje
cted_mass, double ejected_sat,
00447                                     double metallicity)
00448 {
00449     double metallicityHot,d;
00450     double massremain;
00451     /*ejected_sat=ejected_mass;*/
00452

```

```

00453 /* check first just to be sure - Nothing changed reheated mass since last check
00454     * TODO - this cannot be needed! */
00455     if(reheated_mass > Gal[p].ColdGas+1.0e-8 && reheated_mass > 1.0e-8)
00456     {
00457         printf("Something strange here (SF2)....%e\t%e\n", reheated_mass, Gal[p].
00458             ColdGas);
00459         exit(19);
00460         reheated_mass = Gal[p].ColdGas;
00461     }
00462 /* Delucia2004 SN feedback scheme
00463     * if EjectionOn == 1 cold gas is ejected and lost
00464     * if EjectionOn == 0 cold gas is only reheatd*/
00465     if(FeedbackRecipe == 0)
00466     {
00467         if(EjectionOn == 1 && p == centralgal)
00468         {
00469             Gal[p].ColdGas -= ejected_mass;
00470             Gal[p].MetalsColdGas -= metallicity * ejected_mass;
00471             Gal[p].EjectedMass += ejected_mass;
00472             Gal[p].MetalsEjectedMass += metallicity * ejected_mass;
00473         }
00474     else
00475     {
00476         Gal[p].ColdGas -= reheated_mass;
00477         Gal[p].MetalsColdGas -= metallicity * reheated_mass;
00478         Gal[p].HotGas += reheated_mass;
00479         Gal[p].MetalsHotGas += metallicity * reheated_mass;
00480     }
00481 }
00482
00483 /* Delucia2007 with Guo2010 modifications to account for satellite gas cycles */
00484
00485     if(FeedbackRecipe == 1)
00486     {
00487         d = sqrt(pow(Gal[centralgal].Pos[0] - Gal[Gal[p].CentralGal].Pos[0], 2.0) +
00488                 pow(Gal[centralgal].Pos[1] - Gal[Gal[p].CentralGal].Pos[1], 2.0) +
00489                 pow(Gal[centralgal].Pos[2] - Gal[Gal[p].CentralGal].Pos[2], 2.0))/
00490                 (1 + ZZ[Halo[Gal[centralgal].HaloNr].SnapNum]);
00491
00492         //remove from cold gas
00493         Gal[p].ColdGas -= reheated_mass;
00494         Gal[p].MetalsColdGas -= metallicity * reheated_mass;
00495
00496         //add to hot gas
00497         /*If galaxy is a satellite (its central galaxy its a type 1) and its inside
00498             * Rvir of the main halo, the gas will be transferred to the central galaxy.*/
00499     */
00500     if (d<Gal[centralgal].Rvir && Gal[Gal[p].CentralGal].Type == 1)
00501     {
00502         Gal[centralgal].HotGas +=reheated_mass;
00503         Gal[centralgal].MetalsHotGas += reheated_mass * metallicity;
00504     }
00505     else
00506     {
00507         //otherwise gas can go to the type 0 or type 1 centralgalaxy
00508         Gal[Gal[p].CentralGal].HotGas += reheated_mass;
00509         Gal[Gal[p].CentralGal].MetalsHotGas += metallicity * reheated_mass;
00510     }
00511
00512     //ejected mass taken from the central galaxy

```

```

00511     if(ejected_mass > Gal[centralgal].HotGas)
00512         ejected_mass = Gal[centralgal].HotGas;
00513
00514     // remove from hot gas
00515     if (d<Gal[centralgal].Rvir && Gal[Gal[p].CentralGal].Type == 1)
00516     {
00517         metallicityHot = get_metallicity(Gal[centralgal].HotGas, Gal[centralgal]
00518             ].MetalsHotGas);
00519         Gal[centralgal].HotGas -= ejected_mass;
00520         Gal[centralgal].MetalsHotGas -= metallicityHot * ejected_mass;
00521         Gal[centralgal].EjectedMass += ejected_mass;
00522         Gal[centralgal].MetalsEjectedMass += metallicityHot * ejected_mass;
00523     }
00524     else
00525     {
00526         metallicityHot = get_metallicity(Gal[Gal[p].CentralGal].HotGas, Gal[
00527             Gal[p].CentralGal].MetalsHotGas);
00528         Gal[Gal[p].CentralGal].HotGas -= ejected_mass;
00529         Gal[Gal[p].CentralGal].MetalsHotGas -= metallicityHot * ejected_mass;
00530         Gal[Gal[p].CentralGal].EjectedMass += ejected_mass;
00531         Gal[Gal[p].CentralGal].MetalsEjectedMass += metallicityHot * ejected_ma
00532             ss;
00533     }
00534 }
00535
00536
00537     if(FeedbackRecipe == 2 )
00538     {
00539         if (Gal[p].CentralGal == 0 && Gal[p].CentralGal != centralgal)
00540         {
00541             printf("wrong in updating feedback \n");
00542             exit(0);
00543         }
00544
00545         d = sqrt(pow(Gal[centralgal].Pos[0] - Gal[Gal[p].CentralGal].Pos[0], 2.0) +
00546                 pow(Gal[centralgal].Pos[1] - Gal[Gal[p].CentralGal].Pos[1], 2.0) +
00547                     pow(Gal[centralgal].Pos[2] - Gal[Gal[p].CentralGal].Pos[2], 2.0))/(
00548                         1 + ZZ[Halos[Gal[centralgal].HaloNr].SnapNum]);
00549
00550 //REHEAT
00551     Gal[p].ColdGas -= reheated_mass;
00552     Gal[p].MetalsColdGas -= metallicity * reheated_mass;
00553
00554
00555     if (d<Gal[centralgal].Rvir && Gal[Gal[p].CentralGal].Type == 1)
00556     {
00557         // mass that remains on satellite (the rest goes to type 0)
00558         // for reheat - massremain, for eject - ejected mass
00559         massremain=reheated_mass*Gal[p].HotRadius/Gal[p].Rvir;
00560         ejected_sat = ejected_sat*Gal[p].HotRadius/Gal[p].Rvir;
00561     }
00562     else
00563     {
00564         massremain=reheated_mass;
00565     }
00566
00567     if(massremain > reheated_mass)
00568         massremain = reheated_mass;
00569
00570     if (Gal[Gal[p].CentralGal].Type == 1)
00571         update_hot_frac(Gal[p].CentralGal, massremain, Gal[Gal[p].CentralGal].

```

```

        HotGas);
00572     if (d<Gal[centralgal].Rvir && Gal[Gal[p].CentralGal].Type == 1)
00573     {
00574         /* If galaxy is a type 2 orbiting type 1 but inside rvir of
00575            the type 0, part of the reheated gas goes to each */
00576         Gal[Gal[p].CentralGal].HotGas += massremain;
00577         Gal[Gal[p].CentralGal].MetalsHotGas += metallicity * massremain;
00578         Gal[centralgal].HotGas += reheated_mass - massremain;
00579         Gal[centralgal].MetalsHotGas +=(reheated_mass - massremain) * metallicity;
00580     }
00581     else
00582     {
00583         // Otherwise there is no sharing everything goes to central (0 or 1)
00584         Gal[Gal[p].CentralGal].HotGas += massremain;
00585         Gal[Gal[p].CentralGal].MetalsHotGas += metallicity * massremain;
00586     }
00587
00588
00589     //EJECTION
00590     metallicityHot = get_metallicity(Gal[Gal[p].CentralGal].HotGas, Gal[Gal[p].CentralGal].MetalsHotGas);
00591
00592     if (Gal[Gal[p].CentralGal].Type == 1)
00593     {
00594         if (ejected_sat > Gal[Gal[p].CentralGal].HotGas)
00595             ejected_sat = Gal[Gal[p].CentralGal].HotGas;
00596
00597         //TO DO - not needed
00598         if (Gal[Gal[p].CentralGal].Type == 1)
00599             update_hot_frac(Gal[p].CentralGal, -ejected_sat, Gal[Gal[p].CentralGal].HotGas);
00600
00601         if (d < Gal[centralgal].Rvir)
00602         {
00603             //TO DO - this could all be done outside this if statement and
00604             // avoid repetition of 1 and 2
00605
00606             /* If type 1, or type 2 orbiting type 1 near type 0
00607                * ejected gas is taken away from the type 1 */
00608             Gal[Gal[p].CentralGal].HotGas -= ejected_sat;//1.1
00609             Gal[Gal[p].CentralGal].MetalsHotGas -= metallicityHot * ejected_sat
00610             ;//1.2
00611
00612             if (EjectionRecipe == 1)
00613             {
00614                 // In Delucia2007 gas ejected from satellites goes to hot of type 0
00615
00616                 Gal[centralgal].HotGas += ejected_sat;
00617                 Gal[centralgal].MetalsHotGas += metallicityHot * ejected_sat;
00618             }
00619             else if (EjectionRecipe == 2)
00620             {
00621                 // In Guo2010 ejected gas goes to ejected of type 1
00622                 Gal[Gal[p].CentralGal].EjectedMass += ejected_sat;//1.3
00623                 Gal[Gal[p].CentralGal].MetalsEjectedMass += metallicityHot * ejected_sat;
00624             }
00625         }
00626     }
00627     else
00628     {
00629         // If it is far way from type 0 goes to type 1 in both implementations
00630         Gal[Gal[p].CentralGal].HotGas -= ejected_sat;//2.1
00631         Gal[Gal[p].CentralGal].MetalsHotGas -= metallicityHot * ejected_sat
00632         ;//2.2
00633         Gal[Gal[p].CentralGal].EjectedMass += ejected_sat;//2.3

```

```

00629             Gal[Gal[p].CentralGal].MetalsEjectedMass += metallicityHot * ejected
00630             d_sat; //2.4
00631         }
00632     else // If central galaxy is a type 0 - it takes everything
00633     {
00634         if (ejected_mass > Gal[Gal[p].CentralGal].HotGas)
00635             ejected_mass = Gal[Gal[p].CentralGal].HotGas;
00636
00637         Gal[centralgal].HotGas -= ejected_mass;
00638         Gal[centralgal].MetalsHotGas -= ejected_mass*metallicityHot;
00639         Gal[centralgal].EjectedMass += ejected_mass;
00640         Gal[centralgal].MetalsEjectedMass += ejected_mass*metallicityHot;
00641     }
00642 }
00643
00644
00645
00646
00647
00648 }
00649
00650
00651 /** @brief Checks for disk stability using the
00652 *          Mo, Mao & White (1998) criteria */
00653
00654 void check_disk_instability(int p)
00655 {
00656
00657     double Mcrit, metallicity, fraction, stars, diskmass;
00658     double Lumdisk;
00659     int outputbin, j;
00660
00661 /** @brief Calculates the stability of the stellar disk as discussed
00662 *          in Mo, Mao & White (1998). For unstable stars, the required
00663 *          amount is transferred to the bulge to make the disk stable again.
00664 *          Mass, metals and luminosities updated. After Guo2010 the bulge
00665 *          size is followed and needs to be updated.
00666 *          Eq 34 & 35 in Guo2010 are used. */
00667
00668
00669 /* check stellar disk -> eq 34 Guo2010*/
00670 if (Gal[p].Type != 0)
00671     Mcrit = Gal[p].InfallVmax * Gal[p].InfallVmax * Gal[p].StellarDiskRadius / G;
00672
00673 else
00674     Mcrit = Gal[p].Vmax * Gal[p].Vmax * Gal[p].StellarDiskRadius / G;
00675 diskmass = Gal[p].StellarMass - Gal[p].BulgeMass;
00676 stars = diskmass - Mcrit;
00677
00678 /* add excess stars to the bulge */
00679 if(stars > 0.0)
00680 {
00681     /* to caculate the bulge size */
00682     update_bulge_from_disk(p,stars);
00683
00684     metallicity = get_metallicity(Gal[p].StellarMass, Gal[p].MetalsStellarMass);
00685     Gal[p].BulgeMass += stars;
00686     Gal[p].MetalsBulgeMass += metallicity * stars;
00687
00688     fraction = stars / diskmass;
00689
00690 #ifdef OUTPUT_REST_MAGS
00691     for(outputbin = 0; outputbin < NOUT; outputbin++)
00692     {
00693         for(j = 0; j < NMAG; j++)

```

```

00693         {
00694             Lumdisk = Gal[p].Lum[j][outputbin]-Gal[p].LumBulge[j][outputbin];
00695             Gal[p].LumBulge[j][outputbin] += fraction * Lumdisk;
00696             Lumdisk = Gal[p].YLum[j][outputbin]-Gal[p].YLumBulge[j][outputbin];
00697             Gal[p].YLumBulge[j][outputbin] += fraction * Lumdisk;
00698         }
00699     }
00700 #endif
00701 #ifdef COMPUTE_OBS_MAGS
00702     for(outputbin = 0; outputbin < NOUT; outputbin++)
00703     {
00704         for(j = 0; j < NMAG; j++)
00705         {
00706             Lumdisk = Gal[p].ObsLum[j][outputbin]-Gal[p].ObsLumBulge[j][outputb
00707             in];
00708             Gal[p].ObsLumBulge[j][outputbin] += fraction * Lumdisk;
00709             Lumdisk = Gal[p].ObsYLum[j][outputbin]-Gal[p].ObsYLumBulge[j][outpu
00710             tbin];
00711             Gal[p].ObsYLumBulge[j][outputbin] += fraction * Lumdisk;
00712 #ifdef OUTPUT_MOMAF_INPUTS
00713             Lumdisk = Gal[p].dObsLum[j][outputbin]-Gal[p].dObsLumBulge[j][outpu
00714             tbin];
00715             Gal[p].dObsLumBulge[j][outputbin] += fraction * Lumdisk;
00716             Lumdisk = Gal[p].dObsYLum[j][outputbin]-Gal[p].dObsYLumBulge[j][out
00717             putbin];
00718             Gal[p].dObsYLumBulge[j][outputbin] += fraction * Lumdisk;
00719 #endif
00720         }
00721     }
00722     if ((Gal[p].BulgeMass > 1e-8 && Gal[p].BulgeSize == 0.0) || (Gal[p].
00723     BulgeMass == 0.0 && Gal[p].BulgeSize >1e-8))
00724     {
00725         printf("bulgesize wrong in diskinstability.c \n");
00726         exit(0);
00727     }
00728 }
00729
00730 #ifdef GASRECYCLE
00731
00732 #ifdef GALAXYTREE
00733 void cal_gas_recycle(int ngal);
00734 {
00735     int ind, p, q;
00736     double
00737     ind=0;
00738     for(p=0;p<ngal;p++)
00739     {
00740         if (Gal[p].Type != 3)
00741             ind=ind+1;
00742         for (p = 0; p < ind; p++)
00743         {
00744             RecGas = 0.0;
00745             metalcold=0.0;
00746             q = p + NumGals -ind ;
00747             update_from_recycle(q,ptime,previoustime);
00748             HaloGal[q].MetalsColdGas = HaloGal[q].MetalsColdGas+ metalcold;
00749             HaloGal[q].MetalsStellarMass = HaloGal[q].MetalsStellarMass -metalcold;
00750             HaloGal[q].ColdGas = HaloGal[q].ColdGas + RecGas;
00751             starmass=HaloGal[q].StellarMass;
00752             HaloGal[q].StellarMass = HaloGal[q].StellarMass - RecGas;
00753             if (HaloGal[q].StellarMass >= 0.)

```

```

00754      {
00755          if (starmass >0. )
00756          {
00757              HaloGal[q].BulgeMass=HaloGal[q].BulgeMass-HaloGal[q].BulgeMass/star
00758              mass*RecGas;
00759              HaloGal[q].MetalsBulgeMass=HaloGal[q].MetalsBulgeMass-HaloGal[q].
00760              BulgeMass/starmass*metalcold;
00761          }
00762      else
00763          printf("wrong in gas recycle . Stellarmass %f, RecGAs %f\n",starmass,
00764          RecGas);
00765          exit(0);
00766      }
00767  }
00768 }
00769 #endif
00770
00771 #ifdef GALAXYTREE
00772 double update_from_recycle(int p, double time, double previoustime)
00773 {
00774     int q,metindex, tabindex,outputbin;
00775     double f1, f2, fmet1, fmet2, T, ptime, pretime;
00776     float mass;
00777     ptime=time;
00778     pretime=previoustime;
00779
00780     q=HaloGal[p].FirstProgGal;
00781
00782     while (q >= 0 )
00783     {
00784         update_from_recycle(q, ptime,pretime);
00785         q=HaloGal[q].NextProgGal;
00786     }
00787
00788     find_interpolation_point(NumToTime(HaloGal[p].SnapNum),previoustime, &tabindex
00789
00790
00791         &f1, &f2 );
00792
00793     if(tabindex < 0)
00794     {
00795         printf("Something strange here (recyletable)....%t%d snap %d\n",
00796             NumToTime(HaloGal[p].SnapNum)-time, tabindex ,HaloGal[p].SnapNum);
00797         exit(19);
00798     }
00799     T = NumToTime(HaloGal[p].SnapNum-1) - NumToTime(HaloGal[p].SnapNum);
00800
00801 #ifdef SAVE_MEMORY
00802     mass=HaloGal[p].Sfr*T*(Frac[tabindex] * f1+Frac[tabindex+1] * f2);
00803
00804
00805
00806     find_interpolation_point(NumToTime(HaloGal[p].SnapNum),time, &tabindex,
00807
00808         &f1, &f2 );
00809 #ifdef SAVE_MEMORY
00810     mass=HaloGal[p].Sfr*T * (Frac[tabindex] * f1+Frac[tabindex+1] * f2)-mass;
00811
00812 #endif
00813

```

```

00814     metalcold=metalcold+mass*get_metallicity(HaloGal[p].ColdGas, HaloGal[p] .
00815         MetalsColdGas);
00816     if (HaloGal[p].FirstProgGal == -1 && mass > HaloGal[p].StellarMass)
00817         printf("somthing is wrong here(recycle)");
00818     RecGas=RecGas+mass;
00819
00820
00821 }
00822 #endif
00823 #endif
00824
00825 /** @brief Introduced in Guo2010 to track the change in size of bulges
00826 *           after their growth due to disk instabilities. */
00827
00828 void update_bulge_from_disk(int p, double stars)
00829 {
00830     double sigma0,bulgesize;
00831     double fraction, diskmass,cb,cd,fint,disksize,disksizefinal,massfrac;
00832     double Aa,Bb,Cc,Dd,Eint ;
00833     double bulgemass,delta;
00834     double orisize;
00835     int j;
00836
00837 /** @brief Updates bulge from disk instability -> stars represents the mass
00838 *           transferred to the bulge, which occupies a size in the bulge equal
00839 *           to the occupied in the disk. */
00840
00841
00842     orisize=Gal[p].BulgeSize; //remove, not used
00843     diskmass=(Gal[p].StellarMass-Gal[p].BulgeMass);
00844
00845     /* alpha_inter=2.0/C=0.5 (alpha larger than in mergers since
00846      * the existing and newly formed bulges are concentric)*/
00847     fint=4.0;
00848
00849     /* update the stellardisk spin due to the angular momentum transfer
00850      * from disk to bulge changing the specific angular momentum for disk stars.
00851      * This should be done on the main routine, as this is update bulge.*/
00852     massfrac=stars/diskmass;
00853     for (j = 0; j <3 ; j++)
00854         Gal[p].StellarSpin[j]=Gal[p].StellarSpin[j]/(1-massfrac);
00855
00856     /* update disksize done, disk mass is automatically given by total-bulge*/
00857
00858 //GET BULGE SIZE - Eq. 35 in Guo2010
00859 /* if previous Bulge Mass = 0
00860    -> bulge size is given directly from newly formed bulge */
00861 if(Gal[p].BulgeMass <1.e-9)
00862 {
00863     /* size of newly formed bulge, which consists of the stellar mass
00864      * transferred from the disk. This is calculated using bulge_from_disk
00865      * which receives Delta_M/DiskMass and returns Rb/Rd. From eq 35 and
00866      * since DiskMass=2PISigma(Rd)^2 we see that Delta_M/DiskMass=1-(1+Rb/Rd
00867      )*exp(-Rb/Rd),
00868      * so function bulge_from_disk avoids calculating the slow "ln" function
00869      */
00870     bulgesize=bulge_from_disk(stars/diskmass)*Gal[p].StellarDiskRadius/3.;
00871 }
00872 else
00873 {
00874     bulgesize=bulge_from_disk(stars/diskmass)*Gal[p].StellarDiskRadius/3. ;
00875     /* combine the old with newly formed bulge and calculate the
00876      * bulge size assuming energy conservation as for mergers but
00877      * using alpha=2. - eq 33 */

```

```

00878     Gal[p].BulgeSize=(Gal[p].BulgeMass+stars)*(Gal[p].BulgeMass+stars)/
00879             (Gal[p].BulgeMass*Gal[p].BulgeMass/Gal[p].BulgeSize+stars*stars
00880             /bulgesize+fint*Gal[p].BulgeMass*stars/(Gal[p].BulgeSize+bulgesize));
00881 }
00882 // TODO - check why we need it
00883 if ((Gal[p].BulgeMass+stars > 1.e-8 && Gal[p].BulgeSize == 0.0) || (Gal[p].
00884 BulgeMass+stars == 0 && Gal[p].BulgeSize >1.e-8))
00885 {
00886     printf("bulgesize wrong in disk instability. stellarmass %f, bulgemass %f,
00887     bulgesize %f, coldgas %f,gasdisk %f,stellardisk %f masstransfer %f trassize %f,
00888     oribulgesize %f\n",Gal[p].StellarMass, Gal[p].BulgeMass, Gal[p].BulgeSize, Gal[p].
00889     ColdGas, Gal[p].GasDiskRadius, Gal[p].StellarDiskRadius, stars,bulgesize,orisize);
00890 }
00891 }
00892
00893 /** @brief Calculates the size of the disk that contains the
00894 *          mass transferred to the bulge. */
00895 double bulge_from_disk(double frac)
00896 {
00897     double x1,x2,x0,value;
00898 /** @brief Calculates the size of the disk that contains the
00899 *          mass transferred to the bulge. The bulge is assumed
00900 *          to form with the same size. avoid doing "ln" from eq 35*/
00901     x1=0.0;
00902     x2=1.;
00903     while (func_size(x2,frac)*func_size(x1,frac)>0)
00904     {
00905         x1=x2;
00906         x2=x2*2;
00907     }
00908     x0=x1+(x2-x1)/2.;
00909     value=func_size(x0,frac);
00910     if (value < 0)
00911         value = -value;
00912
00913     while(value>0.00001)
00914     {
00915         if(func_size(x0,frac)*func_size(x2,frac)>0)
00916
00917             x2=x0;
00918
00919         else
00920             x1=x0;
00921         x0=x1+(x2-x1)/2.;
00922         value=func_size(x0,frac);
00923         if (value < 0)
00924             value = -value;
00925     }
00926
00927     return x0;
00928 }
00929
00930
00931 double func_size(double x, double a)
00932 {
00933     return exp(-x)*(1+x)-(1-a);
00934 }
00935
00936 double Energy_in_Reheat(int p)
00937 {
00938     double refrac;
00939     if (SNinReheat >0.1 )

```

```

00940      {
00941          refrac=SNinReheat*(1-(0.002/Gal[p].GasDiskRadius)+0.5);
00942          if (refrac < 0.5)
00943              refrac=0.5;
00944          if (refrac >1)
00945              refrac=1.;
00946      }
00947  else
00948      refrac=1.;
00949  return refrac;
00950 }
```

4.39 code/save.c File Reference

Copies the relevant properties in Galaxy structure into Galaxy_Output structure and saves them into the output files (SA_z**_**) - redshift/filenr.

Functions

- void [save_galaxies](#) (int filenr, int tree)

Saves the Galaxy_Output structure for all the galaxies in the current tree into the current output file (one for each input dark matter file) for the chosen snapshots.

- void [finalize_galaxy_file](#) (int filenr)

Writes an header in the output files.

- void [finalize_momaf_file](#) (int filenr)

- void [get_coordinates](#) (float *pos, float *vel, long long ID, int tree, int halonr, int snapnum)

get_coordinates receives the positions and velocities of a type 2 galaxy and updates them with the values from the most bound particle identified at disruption time.

- void [prepare_galaxy_for_output](#) (int n, int filenr, int tree, struct **GALAXY** *g, struct **GALAXY_OUTPUT** *o)

Copies all the relevant properties from the Galaxy structure into the Galaxy output structure, some units are corrected.

- void [fix_units_for_output](#) (struct **GALAXY_OUTPUT** *o)

Removes h from units of galaxy properties.

- void [prepare_galaxy_for_momaf](#) (int n, int filenr, int tree, struct **GALAXY** *g, struct **MOMAF_INPUTS** *o)

- void [save_galaxy_tree](#) (int filenr, int tree)

Saves galaxies if GALAXY_TREE=1.

- int [walk](#) (int nr)

Walks up the main leaf of a tree TODO - is that what it actually does?

4.39.1 Detailed Description

There are two distinct procedures to write the output depending on whether GALAXY_TREE option is turned on or off. If it is on the full galaxy tree is written using save_galaxy_tree. If it is off, the output is only written for the chosen output snap numbers using save_galaxies.

If USE_MEMORY_TO_MINIMIZE_IO ON, these routines copy the galaxy data from the working structures into pointers until that has been done for all the tree in a given file.

After all the galaxy trees are written finalize_galaxy_file is called in [main.c](#) to include an header in the output files. If GALAXY_TREE=1 three numbers are written: 1 (int); size_of_struct(Galaxy_Output) (int); and TotGalCount(int). If GALAXY_TREE=0 then the header is: Ntrees (int); total number of galaxies for the snapshot corresponding to the current file -> TotGalaxies[n] (int); and the number of galaxies on each tree on the current snapshot -> TreeNgals[n] (int*Ntrees).

If USE_MEMORY_TO_MINIMIZE_IO ON, finalize_galaxy_file also copies the galaxy data stored in pointers into the output files, so that it is all done in one go for a given file. This is done using either write_galaxy_data_snap (for SNAP output), write_all_galaxy_data (for GALAXYTREE option) or write_galaxy_for_momaf (for MOMAF option).

If UPDATETYPE2 is defined, the positions of type 2 galaxies (satellites without a dark matter halo) will be updated before output to contain the subsequent positions of the most bound dark matter particle at disruption time (using get_coordinates).

Definition in file [save.c](#).

4.39.2 Function Documentation

4.39.2.1 void finalize_galaxy_file (int filenr)

If GALAXYTREE=1 three numbers are written: 1 (int); size_of_struct(Galaxy_Output) (int); and TotGalCount(int). If GALAXYTREE=0 then the header is: Ntrees (int); total number of galaxies for the snapshot corresponding to the current file -> TotGalaxies[n] (int); and the number of galaxies on each tree on the current snapshot -> TreeNgals[n] (int*Ntrees).

If USE_MEMORY_TO_MINIMIZE_IO ON, finalize_galaxy_file also copies the galaxy data stored in pointers into the output files, so that it is all done in one go for a given file. This is done using either write_galaxy_data_snap (for SNAP output), write_all_galaxy_data (for GALAXYTREE option) or write_galaxy_for_momaf (for MOMAF option).

Definition at line 211 of file [save.c](#).

References [FileNameGalaxies](#), [ListOutputSamps](#), [myfwrite\(\)](#), [Ntrees](#), [offset_galaxydata](#), [offset_galsnapdata](#), [open_outputtree_file\(\)](#), [output_file](#), [OutputDir](#), [TotGalaxies](#), [TotGalCount](#), [TreeNgals](#), [write_all_galaxy_data\(\)](#), [write_galaxy_data_snap\(\)](#), and [ZZ](#).

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

4.39.2.2 void finalize_momaf_file (int filenr)

Definition at line 287 of file [save.c](#).

References [FileNameGalaxies](#), [ListOutputSamps](#), [myfwrite\(\)](#), [offset_momafdata](#), [OutputDir](#), [TotGalaxies](#), and [write_galaxy_for_momaf\(\)](#).

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

4.39.2.3 void fix_units_for_output(struct GALAXY_OUTPUT * o)

If desired (makefile option FIX_OUTPUT_UNITS is set), the output properties of the galaxies can be scaled to physical units excluding any factors of $h = \text{Hubble}/100 \text{ km/s/Mpc}$.

Definition at line 703 of file [save.c](#).

References [GALAXY_OUTPUT::BlackHoleMass](#), [GALAXY_OUTPUT::BulgeMass](#), [GALAXY_OUTPUT::BulgeSize](#), [GALAXY_OUTPUT::CentralMvir](#), [GALAXY_OUTPUT::ColdGas](#), [GALAXY_OUTPUT::CoolingRadius](#), [GALAXY_OUTPUT::EjectedMass](#), [GALAXY_OUTPUT::GasDiskRadius](#), [GALAXY_OUTPUT::GasSpin](#), [GALAXY_OUTPUT::HotGas](#), [GALAXY_OUTPUT::HotRadius](#), [Hubble_h](#), [GALAXY_OUTPUT::ICM](#), [GALAXY_OUTPUT::MetalsBulgeMass](#), [GALAXY_OUTPUT::MetalsColdGas](#), [GALAXY_OUTPUT::MetalsEjectedMass](#), [GALAXY_OUTPUT::MetalsHotGas](#), [GALAXY_OUTPUT::MetalsICM](#), [GALAXY_OUTPUT::MetalsStellarMass](#), [GALAXY_OUTPUT::Mvir](#), [GALAXY_OUTPUT::Pos](#), [GALAXY_OUTPUT::Rvir](#), [GALAXY_OUTPUT::StellarDiskRadius](#), [GALAXY_OUTPUT::StellarMass](#), and [GALAXY_OUTPUT::StellarSpin](#).

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

4.39.2.4 void get_coordinates(float * pos, float * vel, long long ID, int tree, int halonr, int snapnum)

Pos and Vel for the type 2 galaxy are updated with PosList and VelList, containing the corresponding properties for the most bound dark matter particle identified at disruption time. The velocity is converted into a peculiar velocity: $v_p = \sqrt{AAv}$.

Definition at line 331 of file [save.c](#).

References [AA](#), [CountIDs_halo](#), [IdList](#), [OffsetIDs](#), [OffsetIDs_halo](#), [PosList](#), [TreeFirstHalo](#), and [VelList](#).

Referenced by [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.39.2.5 void prepare_galaxy_for_momaf(int n, int filenr, int tree, struct GALAXY * g, struct MOMAF_INPUTS * o)

Definition at line 741 of file [save.c](#).

References [GALAXY::dObsLumBulge](#), [GALAXY::dObsLumDust](#), [MOMAF_INPUTS::dObsMagBulge](#), [MOMAF_INPUTS::dObsMagDust](#), [GALAXY::GalID](#), [MOMAF_INPUTS::GalID](#), [halo_ids_data::HaloID](#), [MOMAF_INPUTS::HaloID](#), [Haloid](#)s, [GALAXY::HaloNr](#), [lum_to_mag\(\)](#), [GALAXY::MergCentralPos](#), [GALAXY::MergTime](#), [GALAXY::ObsLumBulge](#), [GALAXY::ObsLumDust](#), [MOMAF_INPUTS::ObsMagBulge](#), [MOMAF_INPUTS::ObsMagDust](#), [GALAXY::OriMergTime](#), [GALAXY::Pos](#), [MOMAF_INPUTS::Pos](#), [GALAXY::SnapNum](#), [MOMAF_INPUTS::SnapNum](#), [GALAXY::Type](#), [GALAXY::Vel](#), [MOMAF_INPUTS::Vel](#), and [ZZ](#).

Referenced by [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.39.2.6 void prepare_galaxy_for_output(int n, int filenr, int tree, struct GALAXY * g, struct GALAXY_OUTPUT * o)

Definition at line 382 of file [save.c](#).

References [GALAXY::BlackHoleMass](#), [GALAXY_OUTPUT::BlackHoleMass](#), [BoxSize](#), [GALAXY::BulgeMass](#), [GALAXY_OUTPUT::BulgeMass](#), [GALAXY::BulgeSize](#), [GALAXY_OUTPUT::BulgeSize](#), [GALAXY::CentralMvir](#), [GALAXY::ColdGas](#), [GALAXY_OUTPUT::ColdGas](#), [GALAXY::CoolingRadius](#), [GALAXY_OUTPUT::CoolingRadius](#),

GALAXY::DescendantGal, GALAXY_OUTPUT::DescendantGal, GALAXY::DisruptOn, GALAXY_-
 OUTPUT::DisruptOn, GALAXY::dObsLum, GALAXY::dObsLumBulge, GALAXY::dObsLumDust,
 GALAXY_OUTPUT::dObsMag, GALAXY_OUTPUT::dObsMagBulge, GALAXY_-
 OUTPUT::dObsMagDust, GALAXY::EjectedMass, GALAXY_OUTPUT::EjectedMass,
 halo_data::FileNr, GALAXY_OUTPUT::FileTreeNr, halo_aux_data::FirstGalaxy, halo_-
 data::FirstHaloInFOFgroup, GALAXY::FirstProgGal, GALAXY_OUTPUT::FirstProgGal, fix_units_-
 for_ouput(), GALAXY_OUTPUT::FOCentralGal, GALAXY::GalID, GALAXY_OUTPUT::GalID,
 GALAXY::GasDiskRadius, GALAXY_OUTPUT::GasDiskRadius, GALAXY::GasSpin, GALAXY_-
 OUTPUT::GasSpin, get_virial_mass(), Halo, HaloAux, HaloGal, halo_ids_data::Haloid,
 GALAXY_OUTPUT::HaloID, HaloIDs, GALAXY_OUTPUT::HaloIndex, GALAXY::HaloNr,
 Hashbits, GALAXY::HotGas, GALAXY_OUTPUT::HotGas, GALAXY::HotRadius, GALAXY_-
 OUTPUT::HotRadius, Hubble_h, GALAXY::ICLLum, GALAXY::ICM, GALAXY_OUTPUT::ICM,
 GALAXY::InfallSnap, GALAXY_OUTPUT::InfallSnap, GALAXY::InfallVmax, GALAXY_-
 OUTPUT::InfallVmax, GALAXY::LastProgGal, GALAXY_OUTPUT::LastProgGal, halo_data::Len,
 GALAXY::Len, GALAXY_OUTPUT::Len, GALAXY::Lum, lum_to_mag(), GALAXY::LumBulge,
 GALAXY::LumDust, GALAXY_OUTPUT::Mag, GALAXY_OUTPUT::MagBulge, GALAXY_-
 OUTPUT::MagDust, GALAXY_OUTPUT::MagICL, GALAXY::MainLeaf, GALAXY_-
 OUTPUT::MainLeafId, GALAXY::MassWeightAge, GALAXY_OUTPUT::MassWeightAge,
 GALAXY::MergCentralPos, GALAXY_OUTPUT::MergeOn, GALAXY::MergeOn, GALAXY_-
 OUTPUT::MergTime, GALAXY::MergTime, GALAXY::MetalsBulgeMass, GALAXY_-
 OUTPUT::MetalsBulgeMass, GALAXY::MetalsColdGas, GALAXY_OUTPUT::MetalsColdGas,
 GALAXY::MetalsEjectedMass, GALAXY_OUTPUT::MetalsEjectedMass, GALAXY::MetalsHotGas,
 GALAXY_OUTPUT::MetalsHotGas, GALAXY::MetalsICM, GALAXY_OUTPUT::MetalsICM,
 GALAXY::MetalsStellarMass, GALAXY_OUTPUT::MetalsStellarMass, GALAXY_-
 OUTPUT::MMSubID, GALAXY::Mvir, GALAXY_OUTPUT::Mvir, halo_data::NextHaloInFOFgroup,
 GALAXY::NextProgGal, GALAXY_OUTPUT::NextProgGal, halo_aux_data::NGalaxies,
 GALAXY::ObsICL, GALAXY::ObsLum, GALAXY::ObsLumBulge, GALAXY::ObsLumDust,
 GALAXY_OUTPUT::ObsMag, GALAXY_OUTPUT::ObsMagBulge, GALAXY_-
 OUTPUT::ObsMagDust, GALAXY_OUTPUT::ObsMagICL, GALAXY_OUTPUT::OriMergTime,
 GALAXY::OriMergTime, peano_hilbert_key(), GALAXY_OUTPUT::PeanoKey, GALAXY::Pos,
 GALAXY_OUTPUT::Pos, GALAXY_OUTPUT::Redshift, GALAXY::Rvir, GALAXY_OUTPUT::Rvir,
 GALAXY::Sfr, GALAXY_OUTPUT::Sfr, GALAXY::SfrBulge, GALAXY_OUTPUT::SfrBulge,
 GALAXY::SnapNum, GALAXY_OUTPUT::SnapNum, GALAXY::StellarDiskRadius, GALAXY_-
 OUTPUT::StellarDiskRadius, GALAXY::StellarMass, GALAXY_OUTPUT::StellarMass,
 GALAXY::StellarSpin, GALAXY_OUTPUT::StellarSpin, halo_data::SubhaloIndex, GALAXY_-
 OUTPUT::SubID, GALAXY::TreeRoot, GALAXY_OUTPUT::TreeRootId, GALAXY_OUTPUT::Type,
 GALAXY::Type, UnitMass_in_g, UnitTime_in_Megayears, UnitTime_in_s, GALAXY::Vel, GALAXY_-
 OUTPUT::Vel, GALAXY::Vmax, GALAXY_OUTPUT::Vmax, GALAXY::Vvir, GALAXY_-
 OUTPUT::Vvir, GALAXY::XrayLum, GALAXY_OUTPUT::XrayLum, and ZZ.

Referenced by [save_galaxies\(\)](#), and [save_galaxy_tree\(\)](#).

4.39.2.7 void save_galaxies (int filenr, int tree)

If UPDATETYPETWO=1 then the positions and velocities of type 2 galaxies are updated from the most bound dark matter particle. After that the [GALAXY_OUTPUT](#) structure is created and written. input: int file number (current file where the output is being written), int tree number (tree being currently treated).

If USE_MEMORY_TO_MINIMIZE_IO ON, this write statements in this routine copy the galaxy data from the working structures into pointers until that has been done for all the tree in a given file.

Definition at line 58 of file [save.c](#).

References [CountIDs_snaptree](#), [FileNameGalaxies](#), [get_coordinates\(\)](#), [HaloGal](#), [IdList](#), [LastSnapShotNr](#), [ListOutputSnaps](#), [myfread\(\)](#), [myfree\(\)](#), [myfseek\(\)](#), [myfwrite\(\)](#), [mymalloc\(\)](#), [Nids](#), [NtotHalos](#), [Ntrees](#),

NumGals, offset_auxdata, offset_galsnapdata, offset_momafdata, OffsetIDs, OffsetIDs_snaptree, OutputDir, MOMAF_INPUTS::Pos, PosList, prepare_galaxy_for_momaf(), prepare_galaxy_for_output(), SimulationDir, MOMAF_INPUTS::SnapNum, TotGalaxies, TotIds, TotSnaps, TreeNgals, MOMAF_INPUTS::Vel, VelList, and ZZ.

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

4.39.2.8 void save_galaxy_tree (int filenr, int tree)

This routine is similar to save_galaxies except for the initial part where galaxies are ordered.

If USE_MEMORY_TO_MINIMIZE_IO ON, this write statements in this routine copy the galaxy data from the working structures into pointers until that has been done for all the tree in a given file.

Definition at line 812 of file [save.c](#).

References CountIDs_snaptree, GALAXY::DescendantGal, GALAXY::Done, FileNameGalaxies, GALAXY::FirstProgGal, GalaxiesInOrder, GALAXY::GalID, get_coordinates(), HaloGal, IdList, GALAXY::LastProgGal, LastSnapShotNr, ListOutputSnaps, GALAXY::MainLeaf, myread(), myfree(), myfseek(), myfwrite(), mymalloc(), GALAXY::NextProgGal, Nids, NtotHalos, Ntrees, NumGals, offset_auxdata, offset_galaxydata, offset_momafdata, OffsetIDs, OffsetIDs_snaptree, open_outputtree_file(), open_treaux_file(), output_file, OutputDir, MOMAF_INPUTS::Pos, PosList, prepare_galaxy_for_momaf(), prepare_galaxy_for_output(), MOMAF_INPUTS::SnapNum, TotGalaxies, TotGalCount, TotIds, TotSnaps, treeaux_file, GALAXY::TreeRoot, MOMAF_INPUTS::Vel, VelList, and walk().

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

4.39.2.9 int walk (int nr)

Definition at line 993 of file [save.c](#).

References GALAXY::Done, GALAXY::FirstProgGal, GalaxiesInOrder, GalCount, GALAXY::GalID, HaloGal, GALAXY::LastProgGal, GALAXY::MainLeaf, GALAXY::NextProgGal, GALAXY::TreeRoot, and [walk\(\)](#).

Referenced by [save_galaxy_tree\(\)](#), and [walk\(\)](#).

4.40 code/save.c

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <string.h>
00004 #include <math.h>
00005 #include <time.h>
00006 #include "allvars.h"
00007 #include "proto.h"
00008
00009 /**@file save.c
00010 * @brief Copies the relevant properties in Galaxy structure into
00011 * Galaxy_Output structure and saves them into the output
00012 * files (SA_z**_**) - redshift/filenr.
00013 *
00014 * There are two distinct procedures to write the output depending
00015 * on whether GALAXY_TREE option is turned on or off. If it is on
00016 * the full galaxy tree is written using save_galaxy_tree. If it
00017 * is off, the output is only written for the chosen output snap
00018 * numbers using save_galaxies.
00019 *
```

```

00020  *      If USE_MEMORY_TO_MINIMIZE_IO ON, these routines copy the galaxy
00021  *      data from the working structures into pointers until that has
00022  *      been done for all the tree in a given file.
00023  *
00024  *      After all the galaxy trees are written finalize_galaxy_file is
00025  *      called in main.c to include an header in the output files. If
00026  *      GALAXY_TREE=1 three numbers are written: 1 (int);
00027  *      size_of_struct(Galaxy_Output) (int); and TotGalCount(int). If
00028  *      GALAXY_TREE=0 then the header is: Ntrees (int); total number of
00029  *      galaxies for the snapshot corresponding to the current file ->
00030  *      TotGalaxies[n] (int); and the number of galaxies on each tree
00031  *      on the current snapshot -> TreeNgals[n] (int*Ntrees).
00032  *
00033  *      If USE_MEMORY_TO_MINIMIZE_IO ON, finalize_galaxy_file also copies
00034  *      the galaxy data stored in pointers into the output files, so that
00035  *      it is all done in one go for a given file. This is done using either
00036  *      write_galaxy_data_snap (for SNAP output), write_all_galaxy_data
00037  *      (for GALAXYTREE option) or write_galaxy_for_momaf (for MOMAF option).
00038  *
00039  *      If UPDATETYPE2 is defined, the positions of type 2 galaxies
00040  *      (satellites without a dark matter halo) will be updated before
00041  *      output to contain the subsequent positions of the most bound dark
00042  *      matter particle at disruption time (using get_coordinates).
00043  */
00044
00045 /**@brief Saves the Galaxy_Output structure for all the galaxies in
00046 *      the current tree into the current output file (one for each
00047 *      input dark matter file) for the chosen snapshots.
00048 *
00049 *      If UPDATETYPE2=1 then the positions and velocities of type 2
00050 *      galaxies are updated from the most bound dark matter particle.
00051 *      After that the GALAXY_OUTPUT structure is created and written.
00052 *      input: int file number (current file where the output is
00053 *      being written), int tree number (tree being currently treated).
00054 *
00055 *      If USE_MEMORY_TO_MINIMIZE_IO ON, this write statements in this
00056 *      routine copy the galaxy data from the working structures into
00057 *      pointers until that has been done for all the tree in a given file.*/
00058 void save_galaxies(int filenr, int tree)
00059 {
00060 #ifndef USE_MEMORY_TO_MINIMIZE_IO
00061     char buf[1000];
00062 #endif
00063     FILE *fd;
00064     int i, n;
00065     struct GALAXY_OUTPUT galaxy_output;
00066 #ifdef OUTPUT_MOMAF_INPUTS
00067     FILE *fd_momaf;
00068     struct MOMAF_INPUTS momaf_inputs;
00069 #endif
00070
00071     /*The output will be written in a different file for each of the
00072      * chosen output snap shots*/
00073     for(n = 0; n < NOUT; n++)
00074     {
00075         /* If UPDATETYPE2 is ON, the positions of type 2 satellites (don't have
00076          an halo)
00077          * will be updated before the output following the positions of the mos
00078          * t bound particle
00079          * identified just before disruption. This information is available at
00080          * the treeaux
00081          * files. */
00082 #ifdef UPDATETYPE2
00083 #ifdef USE_MEMORY_TO_MINIMIZE_IO
00084         fd = (FILE *) 2;
00085         offset_auxdata = 0;
00086     #else

```

```

00084 #ifdef MRII
00085     sprintf(buf, "%s/treedata/treeaux_sf1_%03d.%d", SimulationDir,
00086             LastSnapShotNr, filenr);
00087 #else
00088     sprintf(buf, "%s/treedata/treeaux_%03d.%d", SimulationDir, LastSnapShotNr,
00089             filenr);
00090 #endif
00091     if (!(fd = fopen(buf, "r")))
00092     {
00093         printf("Can't open file '%s'\n", buf);
00094         fflush(stdout);
00095         exit(1);
00096     }
00097 /* The properties for the most bound particles will be read from the treeau
x
00098 * file into the following variables, defined at allvars.c. These include
00099 * Nids, OffsetIDs, IdList, PosList, VelList. */
00100 myfseek(fd, 4 * sizeof(int) + 2 * TotSnaps * sizeof(int)
00101         + 2 * TotSnaps * Ntrees * sizeof(int) + 2 * sizeof(int) *
00102         NtotHalos, SEEK_CUR);
00103 Nids = CountIDs_snaptree[ListOutputSamps[n] * Ntrees + tree];
00104 OffsetIDs = OffsetIDs_snaptree[ListOutputSamps[n] * Ntrees + tree];
00105 IdList = mymalloc(sizeof(long long) * Nids, "idlist");
00106 PosList = mymalloc(3 * sizeof(float) * Nids, "poslist");
00107 VelList = mymalloc(3 * sizeof(float) * Nids, "vellist");
00108 myfseek(fd, OffsetIDs * sizeof(long long), SEEK_CUR);
00109 myfread(IdList, sizeof(long long), Nids, fd);
00110 myfseek(fd,
00111         (TotIDs - Nids) * sizeof(long long) + OffsetIDs * (3 * sizeof(float
00112         ) -
00113                     sizeof(long long
00114         )), SEEK_CUR);
00115 myfread(PosList, 3 * sizeof(float), Nids, fd);
00116 myfseek(fd, (TotIDs - Nids) * sizeof(float) * 3, SEEK_CUR);
00117 myfread(VelList, 3 * sizeof(float), Nids, fd);
00118#endif USE_MEMORY_TO_MINIMIZE_IO
00119 fclose(fd);
00120#endif UPDATE_TYPE_TWO
00121#endif USE_MEMORY_TO_MINIMIZE_IO
00122 fd = (FILE *) (10 + n);
00123 offset_galsnapdata[n] = 0;
00124#endif OUTPUT_MOMAF_INPUTS
00125 fd_momaf = (FILE *) (10000 + n);
00126 offset_momafdata[n] = 0;
00127#endif
00128#else
00129     sprintf(buf, "%s/%s_z%1.2f_%d", OutputDir, FileNameGalaxies, ZZ[ListOutputs
00130         naps[n]], filenr);
00131     if (!(fd = fopen(buf, "r+")))
00132     {
00133         printf("can't open file '%s'\n", buf);
00134         exit(1);
00135     }
00136#endif OUTPUT_MOMAF_INPUTS
00137 sprintf(buf, "%s/%s_%d.%d", OutputDir, FileNameGalaxies, filenr, ListOutputSna
00138 ps[n]);
00139 if (!(fd_momaf = fopen(buf, "r+")))
00140 {
00141     printf("can't open file '%s'\n", buf);
00142     exit(1);
00143 }
00144#endif
00145#endif

```

```

00143      /* As each tree is completed it is written separately into the output file,
00144      so it is
00145      * necessary to jump into the end of the current output file (after the pro
00146      eprties
00147      * already written) */
00148      /* Jump file header (number of trees (1) + total number of galaxies (1) + n
00149      umber of
00150      * galaxies on each tree (Ntrees)) */
00151      myfseek(fd, (2 + Ntrees) * sizeof(int), SEEK_CUR);
00152      //Jump all the galaxy structures already written
00153      myfseek(fd, TotGalaxies[n] * sizeof(struct GALAXY_OUTPUT), SEEK_CUR);
00154 #ifdef OUTPUT_MOMAF_INPUTS
00155      myfseek(fd_momaf, sizeof(int), SEEK_CUR);
00156      myfseek(fd_momaf, TotGalaxies[n] * sizeof(struct MOMAF_INPUTS), SEEK_CUR);
00157 #endif
00158      /*Go through all the galaxies in the current tree*/
00159      for(i = 0; i < NumGals; i++)
00160      {
00161          /*
00162          /*write galaxy if it's snapnum corresponds to a desired output snapshot
00163          * (current being written = n)*/
00164          if(HaloGal[i].SnapNum == ListOutputSamps[n])
00165          {
00166 #ifdef UPDATETYPE TWO
00167              /* Update positions of type 2's */
00168              if(HaloGal[i].Type==2)
00169                  get_coordinates(HaloGal[i].Pos, HaloGal[i].Vel, HaloGal[i]
00170 ] .MostBoundID, tree, HaloGal[i].HaloNr, HaloGal[i].SnapNum);
00171 #endif
00172              prepare_galaxy_for_output(n, filenr, tree, &HaloGal[i], &galaxy_out
00173 put);
00174              //WRITES THE OUTPUT
00175              myfwrite(&galaxy_output, sizeof(struct GALAXY_OUTPUT), 1, fd);
00176 #ifdef OUTPUT_MOMAF_INPUTS
00177              prepare_galaxy_for_momaf(n, filenr, tree, &HaloGal[i], &momaf_inputs);
00178              //WRITES THE OUTPUT for MOMAF
00179              myfwrite(&momaf_inputs, sizeof(struct MOMAF_INPUTS), 1, fd_momaf);
00180 #endif
00181          }
00182      }
00183      TotGalaxies[n]++; //this will be written later
00184      TreeNgals[n][tree]++; //this will be written later (Number of galaxi
00185      es in each tree)
00186      }
00187      }
00188 #ifndef USE_MEMORY_TO_MINIMIZE_IO
00189     fclose(fd);
00190 #endif
00191 #endif
00192 #endif
00193     }
00194
00195 }
00196
00197 /** @brief Writes an header in the output files.
00198 *
00199 * If GALAXYTREE=1 three numbers are written: 1 (int);
00200 * size_of_struct(Galaxy_Output) (int); and TotGalCount(int). If
00201 * GALAXYTREE=0 then the header is: Ntrees (int); total number of
00202 * galaxies for the snapshot corresponding to the current file ->
00203 * TotGalaxies[n] (int); and the number of galaxies on each tree

```

```

00204 * on the current snapshot -> TreeNgals[n] (int*Ntrees).
00205 *
00206 * If USE_MEMORY_TO_MINIMIZE_IO ON, finalize_galaxy_file also copies
00207 * the galaxy data stored in pointers into the output files, so that
00208 * it is all done in one go for a given file. This is done using either
00209 * write_galaxy_data_snap (for SNAP output), write_all_galaxy_data
00210 * (for GALAXYTREE option) or write_galaxy_for_momaf (for MOMAF option).*/
00211 void finalize_galaxy_file(int filenr)
00212 {
00213     /*Properties to be written if GALAXY_TREE=1*/
00214     int one, size_of_struct;
00215     one = 1;
00216     size_of_struct = sizeof(struct GALAXY_OUTPUT);
00217
00218 #ifndef USE_MEMORY_TO_MINIMIZE_IO
00219     char buf[1000];
00220 #endif
00221     FILE *fd;
00222
00223 #ifndef GALAXYTREE
00224     int n;
00225
00226     for(n = 0; n < NOUT; n++)
00227     {
00228 #ifdef USE_MEMORY_TO_MINIMIZE_IO
00229         fd = (FILE *) (10 + n);
00230         offset_galsnapdata[n] = 0;
00231     #else
00232         sprintf(buf, "%s/%s_z%1.2f_%d", OutputDir, FileNameGalaxies, ZZ[
00233             ListOutputSnaps[n]], filenr);
00234         if(!(fd = fopen(buf, "r+")))
00235         {
00236             printf("can't open file '%s'\n", buf);
00237             exit(1);
00238         }
00239     #endif
00240     /*IF GALAXY_TREE=0 -> snapshots output, write:*/
00241     myfwrite(&Ntrees, sizeof(int), 1, fd); //Number of trees
00242     myfwrite(&TotGalaxies[n], sizeof(int), 1, fd); // total number of galaxies
00243     myfwrite(TreeNgals[n], sizeof(int), Ntrees, fd); // Number of galaxies in ea
00244     ch tree
00245 /* If USE_MEMORY_TO_MINIMIZE_IO ON all the galaxy data for all the trees
00246 * in a file is kept in memory. Then write_all_galaxy_data_snap() writes
00247 * everything in the desired output snaps into the output file in one go. */
00248 #ifndef USE_MEMORY_TO_MINIMIZE_IO
00249     fclose(fd);
00250 #else
00251     write_galaxy_data_snap(n, filenr);
00252 #endif
00253 }
00254 #else //If defined GALAXYTREE
00255
00256 #ifdef USE_MEMORY_TO_MINIMIZE_IO
00257     fd = (FILE *) 4;
00258     offset_galaxydata = 0;
00259 #else //USE_MEMORY_TO_MINIMIZE_IO
00260 #ifdef NEW_IO
00261     fd = output_file;
00262     rewind(fd);
00263 #else
00264     fd = open_outputtree_file(filenr, "r+");
00265 #endif //NEW_IO
00266 #endif //USE_MEMORY_TO_MINIMIZE_IO
00267
00268 /* IF GALAXY_TREE ON and for DB compatible output, write: */

```

```

00269     myfwrite(&one,sizeof(int),1,fd); // write 1
00270     myfwrite(&size_of_struct,sizeof(int),1,fd); // size of an output structure (Galaxy_Output)
00271     myfwrite(&TotGalCount, sizeof(int), 1, fd); // the total number of galaxies
00272
00273 /* If USE_MEMORY_TO_MINIMIZE_IO ON all the galaxy data for all the trees
00274 * in a file is kept in memory. Then write_all_galaxy_data() writes everything
00275 * into the output file in one go. */
00276 #ifndef USE_MEMORY_TO_MINIMIZE_IO
00277 #ifndef NEW_IO
00278     fclose(fd);
00279 #endif
00280 #else
00281     write_all_galaxy_data(filenr);
00282 #endif
00283 #endif
00284 }
00285
00286 #ifdef OUTPUT_MOMAF_INPUTS
00287 void finalize_momaf_file(int filenr)
00288 {
00289 #ifndef USE_MEMORY_TO_MINIMIZE_IO
00290     char buf[1000];
00291 #endif
00292     FILE *fd;
00293     int n;
00294
00295     for(n = 0; n < NOUT; n++)
00296     {
00297 #ifdef USE_MEMORY_TO_MINIMIZE_IO
00298         fd = (FILE *) (10000+n);
00299         offset_momafdata[n] = 0;
00300     #else
00301         sprintf(buf, "%s/%s_%d.%d", OutputDir, FileNameGalaxies, filenr,
00302             ListOutputSamps[n]);
00303         if(!(fd = fopen(buf, "r+")))
00304         {
00305             printf("can't open file '%s'\n", buf);
00306             exit(1);
00307         }
00308         myfwrite(&TotGalaxies[n], sizeof(int), 1, fd);
00309 #ifndef USE_MEMORY_TO_MINIMIZE_IO
00310         fclose(fd);
00311     #else
00312         write_galaxy_for_momaf(n, filenr);
00313     #endif
00314     }
00315 }
00316 #endif
00317
00318
00319
00320
00321 /**@brief get_coordinates receives the positions and velocities of a
00322 *      type 2 galaxy and updates them with the values from the
00323 *      most bound particle identified at disruption time.
00324 *
00325 *      Pos and Vel for the type 2 galaxy are updated with PosList
00326 *      and VelList, containing the corresponding properties for the
00327 *      most bound dark matter particle identified at disruption time.
00328 *      The velocity is converted into a peculiar velocity:
00329 *      \f$ v_{\rm rm{p}}=\sqrt{AA}v \f$. */
00330 #ifdef UPDATETYPE TWO
00331 void get_coordinates(float *pos, float *vel, long long ID, int tree, int halonr,
00332     int snapnum)
00333 {

```

```

00333     int m, k, start, nids;
00334
00335     start = OffsetIDs_halo[TreeFirstHalo[tree] + halonr] - OffsetIDs;
00336     nids = CountIDs_halo[TreeFirstHalo[tree] + halonr];
00337
00338     while(nids > 0)
00339     {
00340         m = nids / 2;
00341         if(IdList[start + m] == ID)
00342         {
00343             for(k = 0; k < 3; k++)
00344             {
00345                 pos[k] = PosList[3 * (start + m) + k];
00346                 vel[k] = sqrt(AA[snapnum]) * Vellist[3 * (start + m) + k]; /* to c
onvert to peculiar velocity */
00347             }
00348
00349             if(pos[0] == 0 && pos[1] == 0 && pos[2] == 0)
00350             {
00351                 printf
00352                     ("This treeaux-files does not (yet) contain the coordinates\n for
the desired output time!\n");
00353                 fflush(stdout);
00354                 exit(0);
00355             }
00356
00357             return;
00358         }
00359
00360         if(IdList[start + m] < ID)
00361         {
00362             nids -= m;
00363             start += m;
00364         }
00365         else
00366         {
00367             nids = m;
00368         }
00369     }
00370
00371     printf("ID not found! - What's going on? ID=%d\n", (int) ID);
00372     fflush(stdout);
00373     exit(0);
00374     return;
00375 }
00376 #endif
00377
00378
00379
00380 /**@brief Copies all the relevant properties from the Galaxy structure
00381 * into the Galaxy output structure, some units are corrected.*/
00382 void prepare_galaxy_for_output(int n, int filenr, int tree, struct GALAXY *g, str
uct GALAXY_OUTPUT *o)
00383 {
00384     int j;
00385 #ifdef GALAXYTREE
00386     long long big;
00387     double scalefac;
00388     int i, k, lenmax, tmpfirst, next;
00389     /*Used to calculate the peano-hilbert key*/
00390     double xx=1.11, yy=1.23, zz=1.54;
00391 #endif
00392
00393 #ifndef NO_PROPS_OUTPUTS
00394     o->InfallVmax = g->InfallVmax;
00395
00396     o->InfallSnap = g->InfallSnap;

```

```

00397     o->HotRadius = g->HotRadius;
00398
00399
00400 #ifdef UPDATETYPE TWO
00401     if(g->Type == 2)
00402     {
00403
00404         for(j = 0; j < 3; j++)
00405         {
00406             o->Pos[j]=g->MergCentralPos[j] + (-g->MergCentralPos[j] + g->Pos[j])*sq
00407             rt(g->MergTime/g->OriMergTime);
00408             if (o->Pos[j] < 0 || o->Pos[j] > 500)
00409             {
00410                 printf("wrong inpos of type 2\n");
00411                 exit(0);
00412             }
00413         }
00414     }
00415 #endif
00416     if(g->Type == 2 || (g->Type == 1 && g->MergeOn == 1))
00417     {
00418         o->OriMergTime=g->OriMergTime;
00419         o->MergTime = g->MergTime;
00420     }
00421 else
00422 {
00423     o->OriMergTime=0.0;
00424     o->MergTime = 0.0;
00425 }
00426
00427     o->Type = g->Type;
00428
00429 #ifndef GALAXYTREE
00430     o->HaloIndex = g->HaloNr;
00431 #endif
00432     o->SnapNum = g->SnapNum;
00433 /*new model*/
00434     o->CentralMvir = get_virial_mass(Halo[g->HaloNr].FirstHaloInFOFgroup,1);
00435 /*de lucia*/
00436 // o->CentralMvir = get_virial_mass(Halo[g->HaloNr].FirstHaloInFOFgroup);
00437     for(j = 0; j < 3; j++)
00438     {
00439         if (g->Type !=2)
00440             o->Pos[j] = g->Pos[j];
00441         o->Vel[j] = g->Vel[j];
00442         o->GasSpin[j] = g->GasSpin[j];
00443         o->StellarSpin[j] = g->StellarSpin[j];
00444     }
00445 //o->Haloj=g->Haloj;
00446     o->Len = g->Len;
00447     o->Mvir = g->Mvir;
00448     o->Rvir = g->Rvir;
00449     o->Vvir = g->Vvir;
00450     o->Vmax = g->Vmax;
00451 /*ram pressure*/
00452 /*if (g->InfallGasFrac <1.e-6)*/
00453 /* o->retainfac=0;*/
00454 /*else*/
00455     /* o->retainfac=g->HotGas/g->InfallMass/g->InfallGasFrac;*/
00456
00457 // o->ReheatedMass = g->ReheatedMass;
00458 // o->MetalsReheatedMass = g->MetalsReheatedMass;
00459
00460 // o->CoolingGas = g->CoolingGas;
00461 // o->HotStart = g->HotStart;
00462 // o->HotStripping = g->HotStripping;

```

```

00463 // o->MetalsHotStart = g->MetalsHotStart;
00464 // o->MetalsHotStripping = g->MetalsHotStripping;
00465 // o->HotICM = g->HotICM;
00466 // o->MetalsHotICM = g->MetalsHotICM;
00467
00468 #ifdef GALAXYTREE
00469 // -o->HotRadius = g->HotRadius;
00470#endif
00471 o->ColdGas = g->ColdGas;
00472 o->BulgeSize=g->BulgeSize;
00473 o->StellarMass = g->StellarMass;
00474 o->BulgeMass = g->BulgeMass;
00475 o->HotGas = g->HotGas;
00476 o->EjectedMass = g->EjectedMass;
00477 o->BlackHoleMass = g->BlackHoleMass;
00478 // - o->MergeSat = g->MergeSat;
00479
00480 #ifdef GALAXYTREE
00481 o->DisruptOn = g->DisruptOn;
00482 o->MergeOn = g->MergeOn;
00483#endif
00484 o->MetalsColdGas = g->MetalsColdGas;
00485 o->MetalsStellarMass = g->MetalsStellarMass;
00486 o->MetalsBulgeMass = g->MetalsBulgeMass;
00487 o->MetalsHotGas = g->MetalsHotGas;
00488 o->MetalsEjectedMass = g->MetalsEjectedMass;
00489
00490
00491 /*The starformation rate at each snapshot is an average of the quiescent SF
00492 * that happened on each time step and just the sum for the one that happened
00493 * in bursts.
00494 *
00495 * SFR is in units of 1e10Msun/1e12Yrs -> converted -> g/s -> converted -> Msu
n/Yr
00496 * look at set_units() at init.c to see why UnitTime_in_s shouldn't be used. */
00497
00498 #ifdef UseFullSfr
00499 for(j = 0; j < MAXSNAPS; j++)
00500 {
00501     o->Sfr[j] = g->Sfr[j] * UnitMass_in_g / UnitTime_in_s * SEC_PER_YEAR / SOLAR
R_MASS;
00502     o->SfrBulge[j] = g->SfrBulge[j] * UnitMass_in_g / UnitTime_in_s * SEC_PER_Y
EAR / SOLAR_MASS;
00503 }
00504 #ifdef SAVE_MEMORY
00505     o->Sfr = g->Sfr* UnitMass_in_g / UnitTime_in_s * SEC_PER_YEAR / SOLAR_MASS;
00506     o->SfrBulge = g->SfrBulge * UnitMass_in_g / UnitTime_in_s * SEC_PER_YEAR / SOL
AR_MASS;
00507 #else
00508     o->Sfr = g->Sfr[n] * UnitMass_in_g / UnitTime_in_s * SEC_PER_YEAR / SOLAR_MASS;
00509     o->SfrBulge = g->SfrBulge[n] * UnitMass_in_g / UnitTime_in_s * SEC_PER_YEAR / SOLAR
_MASS;
00510 #endif
00511#endif
00512 // - o->StarMerge=g->StarMerge;
00513 o->XrayLum = g->XrayLum;
00514 o->GasDiskRadius = g->GasDiskRadius;
00515 o->StellarDiskRadius = g->StellarDiskRadius;
00516 // o->halfradius = g->halfradius;
00517 // o->periradius = g->periradius;
00518 o->CoolingRadius = g->CoolingRadius;
00519 o->ICM = g->ICM;
00520 o->MetalsICM = g->MetalsICM;
00521
00522#endif

```

```

00523
00524
00525 #ifdef OUTPUT_REST_MAGS
00526 /* Luminosities are converted into Mags in various bands */
00527   for(j = 0; j < NMAG; j++)
00528   {
00529     o->Mag[j]      = lum_to_mag(g->Lum[j][n]);
00530     o->MagBulge[j] = lum_to_mag(g->LumBulge[j][n]);
00531     o->MagDust[j]  = lum_to_mag(g->LumDust[j][n]);
00532 #ifdef ICL
00533     o->MagICL[j]   = lum_to_mag(g->ICLLum[j][n]);
00534 #endif
00535   }
00536   if(g->StellarMass > 0.0)
00537   {
00538     o->MassWeightAge = g->MassWeightAge[n] / g->StellarMass ;
00539     /*TODO -this is something...code units are in 1e12/h, so all we need to
00540      * do is multiply by 1000. to get Gyr. Instead we /1000. and then * by
00541      * something close to 1e6. look at set_units() at init.c to see why
00542      * UnitTime_in_Megayears is not exact. */
00543     o->MassWeightAge = o->MassWeightAge / 1000. *UnitTime_in_Megayears /
00544     Hubble_h; //Age in Gyr
00545   }
00546   else
00547   {
00548     o->MassWeightAge = 0.;
00549   }
00550 #endif
00551 #ifdef OUTPUT_OBS_MAGS
00552 #ifdef COMPUTE_OBS_MAGS
00553 /* Luminosities in various bands */
00554   for(j = 0; j < NMAG; j++)
00555   {
00556     o->ObsMag[j]      = lum_to_mag(g->ObsLum[j][n]);
00557     o->ObsMagBulge[j] = lum_to_mag(g->ObsLumBulge[j][n]);
00558     o->ObsMagDust[j]  = lum_to_mag(g->ObsLumDust[j][n]);
00559 #ifdef ICL
00560     o->ObsMagICL[j]   = lum_to_mag(g->ObsICL[j][n]);
00561 #endif
00562 #ifdef OUTPUT_MOMAF_INPUTS
00563     o->dObsMag[j]      = lum_to_mag(g->dObsLum[j][n]);
00564     o->dObsMagBulge[j] = lum_to_mag(g->dObsLumBulge[j][n]);
00565     o->dObsMagDust[j]  = lum_to_mag(g->dObsLumDust[j][n]);
00566 #endif
00567   }
00568 #endif
00569 #endif
00570
00571 #ifndef NO_PROPS_OUTPUTS
00572 #ifdef GALAXYTREE
00573   o->GalID = g->GalID;
00574   o->FirstProgGal = g->FirstProgGal;
00575   o->NextProgGal = g->NextProgGal;
00576   o->LastProgGal = g->LastProgGal;
00577   o->MainLeafId = g->MainLeaf;
00578   o->TreeRootId = g->TreeRoot;
00579   o->HaloID = HaloIDs[g->HaloNr].HaloID;
00580
00581   o->DescendantGal = g->DescendantGal;
00582   o->Redshift = ZZ[g->SnapNum];
00583
00584 // TODO check how to treat next setting of multiplier prefixes properly
00585 // Use make/input parameters?
00586 #ifdef MRII
00587   // TODO assumes < 10^9 galaxies per tree, is that always correct?
00588   big = (((filenr * (long long) 1000000) + tree) * (long long) 1000000000);

```

```

00589 #else
00590     big = (((filenr * (long long) 1000000) + tree) * (long long) 1000000);
00591 #endif
00592
00593     o->FileTreeNr = big;
00594
00595     o->GalID += big;
00596
00597     if(o->FirstProgGal >= 0)
00598         o->FirstProgGal += big;
00599
00600     if(o->LastProgGal >= 0)
00601         o->LastProgGal += big;
00602     else
00603         o->LastProgGal = o->GalID;
00604
00605     if(o->MainLeafId >= 0)
00606         o->MainLeafId += big;
00607     else
00608         o->MainLeafId = o->GalID;
00609
00610     if(o->TreeRootId >= 0)
00611         o->TreeRootId += big;
00612     else // TODO this should not occur, throw an error if it does?
00613         o->TreeRootId = -1;
00614
00615     if(o->NextProgGal >= 0)
00616         o->NextProgGal += big;
00617
00618     if(o->DescendantGal >= 0)
00619         o->DescendantGal += big;
00620 // Set pointer ("foreign key") to central galaxy of FOF group this galaxy is in
00621 .
00622 // TODO next seems to work on millimil.
00623 // Assumes that firstgalaxy in halo is its central galaxy.
00624 // This is not generally true (see comments in main.c::join_galaxies_of_progenit
00625 // ors).
00626 // Therefore an extra test is included and
00627 o->FOFCentralGal = HaloGal[HaloAux[Halo[g->HaloNr].FirstHaloInFOFgroup].
00628 FirstGalaxy].GalID+big;
00629 // test whether the FOF central galaxy actually is a type-0
00630 // if not, search for the type-0 galaxy
00631 if(HaloGal[HaloAux[Halo[g->HaloNr].FirstHaloInFOFgroup].FirstGalaxy].Type != 0)
00632 {
00633     printf("a FOFCentralGal, galaxyId=%lld, has been assigned that is no type
00634 0!\n",o->FOFCentralGal);
00635     // TBD can we do something else ?
00636     // For example store FOFCentralGalaxy on HaloAux, to be set in update_ce
00637     // ntralgal ?
00638     // That introduces memory overhead for what should be very rare occurrenc
00639     // e.
00640     // TODO check this code is correct.
00641     for(i = 0; i < HaloAux[Halo[g->HaloNr].FirstHaloInFOFgroup].NGalaxies; i+
00642     )
00643         if(HaloGal[HaloAux[Halo[g->HaloNr].FirstHaloInFOFgroup].
00644 FirstGalaxy+i].Type == 0)
00645             {
00646                 o->FOFCentralGal = HaloGal[HaloAux[Halo[g->HaloNr].
00647 FirstHaloInFOFgroup].FirstGalaxy+i].GalID+big;
00648                 break;
00649             }
00650     }
00651
00652
00653
00654     scalefac = 1.0 / BoxSize;
00655

```

```

00646     xx = g->Pos[0] * scalefac + 1.0;
00647     yy = g->Pos[1] * scalefac + 1.0;
00648     zz = g->Pos[2] * scalefac + 1.0;
00649
00650
00651     i = DOUBLE_to_HASHBITS(xx);
00652     j = DOUBLE_to_HASHBITS(yy);
00653     k = DOUBLE_to_HASHBITS(zz);
00654
00655
00656     o->PeanoKey = peano_hilbert_key(i, j, k, Hashbits);
00657     //printf("i %d, j %d, k %d o->PeanoKey %d \n",i,j,k,o->PeanoKey);
00658     // add info for Gerard
00659 #ifdef MRII
00660     big = g->SnapNum* (long long) 100000000000 + Halo[g->HaloNr].FileNr* (long long)
00661           1000000 + Halo[g->HaloNr].SubhaloIndex,
00662 #else
00663     big = g->SnapNum* (long long) 1000000000000 + Halo[g->HaloNr].FileNr* (long lon
00664           g) 100000000 + Halo[g->HaloNr].SubhaloIndex;
00665 #endif
00666
00667     o->SubID = big;
00668
00669     tmpfirst = Halo[g->HaloNr].FirstHaloInFOFgroup;
00670     lenmax = 0;
00671     next = tmpfirst;
00672     while(next != -1)
00673     {
00674         if(Halo[next].Len > lenmax)
00675         {
00676             lenmax = Halo[next].Len;
00677             tmpfirst = next;
00678         }
00679     // TODO check next relation for both Millennium and Millennium-II !
00680 #ifdef MRII
00681     big = g->SnapNum* (long long) 100000000000 + Halo[tmpfirst].FileNr* (long long)
00682           1000000 + Halo[tmpfirst].SubhaloIndex;
00683 #else
00684     big = g->SnapNum* (long long) 1000000000000 + Halo[tmpfirst].FileNr* (long long)
00685           ) 100000000 + Halo[tmpfirst].SubhaloIndex;
00686 #endif
00687     o->MMSubID = big;
00688
00689 #endif
00690 #endifdef NO_PROPS_OUTPUTS
00691 #ifdef FIX_OUTPUT_UNITS
00692     fix_units_for_ouput(o);
00693 #endif
00694 #endif
00695
00696 }
00697
00698
00699
00700 /**@brief Removes h from units of galaxy properties. If desired (makefile option
00701 * FIX_OUTPUT_UNITS is set), the output properties of the galaxies can be scaled
00702 * to physical units excluding any factors of h == Hubble/100 km/s/Mpc. */
00703 void fix_units_for_ouput(struct GALAXY_OUTPUT *o)
00704 {
00705     o->Pos[0] /= Hubble_h;
00706     o->Pos[1] /= Hubble_h;
00707     o->Pos[2] /= Hubble_h;
00708     o->CentralMvir /= Hubble_h;

```

```

00709     o->Mvir /= Hubble_h;
00710     o->Rvir /= Hubble_h;
00711
00712     o->GasSpin[0] /= Hubble_h;
00713     o->GasSpin[1] /= Hubble_h;
00714     o->GasSpin[2] /= Hubble_h;
00715     o->StellarSpin[0] /= Hubble_h;
00716     o->StellarSpin[1] /= Hubble_h;
00717     o->StellarSpin[2] /= Hubble_h;
00718     o->HotRadius /= Hubble_h;
00719     o->ColdGas /= Hubble_h;
00720     o->StellarMass /= Hubble_h;
00721     o->BulgeMass /= Hubble_h;
00722     o->HotGas /= Hubble_h;
00723     o->EjectedMass /= Hubble_h;
00724     o->BlackHoleMass /= Hubble_h;
00725     o->ICM /= Hubble_h;
00726     o->MetalsColdGas /= Hubble_h;
00727     o->MetalsStellarMass /= Hubble_h;
00728     o->MetalsBulgeMass /= Hubble_h;
00729     o->MetalsHotGas /= Hubble_h;
00730     o->MetalsEjectedMass /= Hubble_h;
00731     o->MetalsICM /= Hubble_h;
00732     o->BulgeSize /= Hubble_h;
00733     o->StellarDiskRadius /= Hubble_h;
00734     o->GasDiskRadius /= Hubble_h;
00735     o->CoolingRadius /= sqrt(Hubble_h);
00736 }
00737
00738
00739
00740 #ifdef OUTPUT_MOMAF_INPUTS
00741 void prepare_galaxy_for_momaf(int n, int filenr, int tree, struct GALAXY *g, stru
ct MOMAF_INPUTS *o)
00742 {
00743     int j;
00744     float dm,dmb,dz;
00745 #ifdef GALAXYTREE
00746     long long big;
00747 #endif
00748
00749 #ifdef UPDATETYPE TWO
00750     if(g->Type == 2)
00751     {
00752 //         get_coordinates(g->Pos, g->Vel, g->MostBoundID, tree, g->HaloNr, g->SnapN
um);
00753
00754         for(j = 0; j < 3; j++)
00755         {
00756             o->Pos[j]=g->MergCentralPos[j] + (-g->MergCentralPos[j] + g->Pos[j])*sq
rt(g->MergTime/g->OriMergTime);
00757             if (o->Pos[j] < 0 || o->Pos[j] > 500)
00758             {
00759                 printf("wrong inpos of type 2\n");
00760                 exit(0);
00761             }
00762         }
00763     }
00764
00765 }
00766 #endif
00767
00768     o->SnapNum = g->SnapNum;
00769     for(j = 0; j < 3; j++)
00770     {
00771         if (g->Type != 2)
00772             o->Pos[j] = g->Pos[j];

```

```

00773     o->Vel[j] = g->Vel[j];
00774 }
00775 #ifdef COMPUTE_OBS_MAGS
00776   for(j = 0; j < NMAG; j++)
00777   {
00778     o->ObsMagBulge[j] = lum_to_mag(g->ObsLumBulge[j][n]);
00779     o->ObsMagDust[j] = lum_to_mag(g->ObsLumDust[j][n]);
00780     dmb = lum_to_mag(g->dObsLumBulge[j][n]);
00781     dm = lum_to_mag(g->dObsLumDust[j][n]);
00782     dz = ZZ[g->SnapNum-1] - ZZ[g->SnapNum];
00783     o->dObsMagBulge[j] = (dmb - o->ObsMagBulge[j])/dz;
00784     o->dObsMagDust[j] = (dm - o->ObsMagDust[j])/dz;
00785   }
00786 #endif
00787
00788 #ifdef GALAXYTREE
00789   o->GalID = g->GalID;
00790   o->HaloID = HaloIDs[g->HaloNr].HaloID;
00791 // TODO check how to treat next, currently ok for Millennium-II
00792 //   big = (((filenr * (long long) 1000000) + tree) * (long long) 1000000);
00793   big = (((filenr * (long long) 1000000) + tree) * (long long) 1000000000);
00794   o->GalID += big;
00795 #else
00796   o->GalID = (long long) (2);
00797   o->HaloID = (long long) (2);
00798 #endif
00799 }
00800 #endif
00801
00802
00803 #ifdef GALAXYTREE
00804 /**@brief Saves galaxies if GALAXY_TREE=1
00805 *
00806 * This routine is similar to save_galaxies except for the intial
00807 * part where galaxies are ordered.
00808 *
00809 * If USE_MEMORY_TO_MINIMIZE_IO ON, this write statements in this
00810 * routine copy the galaxy data from the working structures into
00811 * pointers until that has been done for all the tree in a given file.*/
00812 void save_galaxy_tree(int filenr, int tree) {
00813 #ifndef USE_MEMORY_TO_MINIMIZE_IO
00814   char buf[1000];
00815 #endif
00816   FILE *fd;
00817   int i, j, n, p, num;
00818   struct GALAXY_OUTPUT galaxy_output;
00819 #ifdef OUTPUT_MOMAF_INPUTS
00820   FILE *fd_momaf;
00821   struct MOMAF_INPUTS momaf_inputs;
00822 #endif
00823
00824   for (i = 0; i < NumGals; i++) {
00825     HaloGal[i].Done = 0;
00826     HaloGal[i].LastProgGal = -1;
00827     HaloGal[i].MainLeaf = -1;
00828     HaloGal[i].TreeRoot = -1;
00829   }
00830   GalaxiesInOrder = (int *) mymalloc(NumGals * sizeof(int), "GalaxiesInOrder");
00831   for (num = LastSnapShotNr; num >= 0; num--) {
00832     for (i = 0; i < NumGals; i++) {
00833       if (HaloGal[i].SnapNum == num)
00834         if (HaloGal[i].Done == 0) {
00835           walk(i);
00836         }
00837     }
00838   }

```

```

00839     for (i = 0; i < NumGals; i++) {
00840         p = HaloGal[i].FirstProgGal;
00841         while (p >= 0) {
00842             HaloGal[p].DescendantGal = i;
00843             p = HaloGal[p].NextProgGal;
00844         }
00845     }
00846     for (i = 0; i < NumGals; i++) {
00847         if (HaloGal[i].FirstProgGal >= 0)
00848             HaloGal[i].FirstProgGal = HaloGal[HaloGal[i].
FirstProgGal].GalID;
00849
00850         if (HaloGal[i].LastProgGal >= 0)
00851             HaloGal[i].LastProgGal = HaloGal[HaloGal[i].LastProgGal].
GalID;
00852
00853         if (HaloGal[i].MainLeaf >= 0)
00854             HaloGal[i].MainLeaf = HaloGal[HaloGal[i].MainLeaf].GalID;
00855
00856         if (HaloGal[i].TreeRoot >= 0)
00857             HaloGal[i].TreeRoot = HaloGal[HaloGal[i].TreeRoot].GalID;
00858
00859         if (HaloGal[i].NextProgGal >= 0)
00860             HaloGal[i].NextProgGal = HaloGal[HaloGal[i].NextProgGal].
GalID;
00861
00862         if (HaloGal[i].DescendantGal >= 0)
00863             HaloGal[i].DescendantGal = HaloGal[HaloGal[i].
DescendantGal].GalID;
00864     }
00865 /* This loop, where type 2 positions and velocities are obtained
00866 * from the dark matter particles and the MOMAF output is written
00867 * is identical to what is on save_galaxies*/
00868
00869     for (n = 0; n < NOUT; n++) {
00870 #ifdef UPDATETYPE TWO
00871 #ifdef USE_MEMORY_TO_MINIMIZE_IO
00872         fd = (FILE *) 2;
00873         offset_auxdata = 0;
00874 #else //USE_MEMORY_TO_MINIMIZE_IO
00875 #ifdef NEW_IO
00876         fd = treeaux_file;
00877         rewind(fd);
00878 #else //NEW_IO
00879         fd = open_treeaux_file(filenr);
00880 #endif //NEW_IO
00881 #endif //USE_MEMORY_TO_MINIMIZE_IO
00882         /*As in save_galaxies, read in the properties for the dark matter
00883          * particles that will be used to update the positions of type 2's*/
00884         myfseek(fd, 4 * sizeof(int) + 2 * TotSamps * sizeof(int) + 2 *
TotSamps
00885             * Ntrees * sizeof(int) + 2 * sizeof(int) * NtotHalos, SEEK_CU
R);
00886         Nids = CountIDs_snaptree[ListOutputSamps[n] * Ntrees + tree];
00887         OffsetIDs = OffsetIDs_snaptree[ListOutputSamps[n] * Ntrees + tree
];
00888         IdList = mymalloc(sizeof(long long) * Nids, "idlist");
00889         PosList = mymalloc(3 * sizeof(float) * Nids, "poslist");
00890         Vellist = mymalloc(3 * sizeof(float) * Nids, "vellist");
00891         myfseek(fd, OffsetIDs * sizeof(long long), SEEK_CUR);
00892         myfread(IdList, sizeof(long long), Nids, fd);
00893         myfseek(fd, (TotIds - Nids) * sizeof(long long) + OffsetIDs * (3
* sizeof(float) - sizeof(long long)), SEEK_CUR);
00894         myfread(PosList, 3 * sizeof(float), Nids, fd);
00895         myfseek(fd, (TotIds - Nids) * sizeof(float) * 3, SEEK_CUR);

```

```

00897             myfread(VelList, 3 * sizeof(float), Nids, fd);
00898 #ifndef USE_MEMORY_TO_MINIMIZE_IO
00899 #ifndef NEW_IO
00900         fclose(fd);
00901 #endif //NEW_IO
00902 #endif //USE_MEMORY_TO_MINIMIZE_IO
00903 #endif //UPDATETYPE TWO
00904
00905         // PosList etc have now been read for current output snapnum
00906         // use these to update the HaloGal positions etc for type 2-s
00907         // write MOMAF output at the same time (if desired)
00908 #ifdef OUTPUT_MOMAF_INPUTS
00909 #ifdef USE_MEMORY_TO_MINIMIZE_IO
00910         fd_momaf = (FILE *) (10000 + n);
00911         offset_momafdata[n] = 0;
00912 #else //USE_MEMORY_TO_MINIMIZE_IO
00913         sprintf(buf, "%s/%s_%d.%d", OutputDir, FileNameGalaxies, filenr, L
    istOutputSnaps[n]);
00914         if (!(fd_momaf = fopen(buf, "r+")))
00915         {
00916             printf("can't open file '%s'\n", buf);
00917             exit(1);
00918         }
00919 #endif //USE_MEMORY_TO_MINIMIZE_IO
00920         myfseek(fd_momaf, sizeof(int), SEEK_CUR);
00921         myfseek(fd_momaf, TotGalaxies[n] * sizeof(struct MOMAF_INPUTS), S
    EEK_CUR);
00922 #endif //OUTPUT_MOMAF_INPUTS
00923
00924         for (i = 0; i < NumGals; i++) {
00925             if (HaloGal[i].SnapNum == ListOutputSnaps[n])
00926             {
00927 #ifdef UPDATETYPE TWO
00928                 if(HaloGal[i].Type == 2)
00929                     get_coordinates(HaloGal[i].Pos, HaloGal[i]
                ].Vel, HaloGal[i].MostBoundID, tree, HaloGal[i].HaloNr, HaloGal[i].SnapNum);
00930 #endif
00931
00932 #ifdef OUTPUT_MOMAF_INPUTS
00933     HaloGal[i], &momaf_inputs);
00934     momaf_inputs, 1, fd_momaf);
00935 #endif //OUTPUT_MOMAF_INPUTS
00936         }
00937     }
00938 }
00939 #ifdef OUTPUT_MOMAF_INPUTS
00940     fclose(fd_momaf);
00941 #endif //OUTPUT_MOMAF_INPUTS
00942
00943 #ifdef UPDATETYPE TWO
00944     myfree(Vellist);
00945     myfree(PosList);
00946     myfree(IdList);
00947 #endif
00948 } // for(n = 0; n < NOUT; n++)
00949
00950
00951 // WRITE GALAXY TREE
00952 #ifdef USE_MEMORY_TO_MINIMIZE_IO
00953     fd = (FILE *) 4;
00954     offset_galaxydata = 0;
00955 #else //USE_MEMORY_TO_MINIMIZE_IO
00956 #ifdef NEW_IO
00957     fd = output_file;
00958 #else //NEW_IO

```

```

00959         fd = open_outputtree_file(filenr,"r+");
00960 #endif //NEW_IO
00961 #endif //USE_MEMORY_TO_MINIMIZE_IO
00962         /* Before, the header was a simple integer for number of galaxies. So, th
e
00963             code had to jump over an int (used to store the number of galaxies) an
d
00964             and over all the galaxies written so far */
00965             // myfseek(fd, sizeof(int) + TotGalCount * sizeof(struct GALAXY_OUTPUT),
SEEK_CUR);
00966
00967             // for DB compatible output, pad the first line with the size of a struct
.
00968 #ifndef NEW_IO
00969     myfseek(fd, (1+TotGalCount) * sizeof(struct GALAXY_OUTPUT), SEEK_CUR);
00970 #endif //NEW_IO
00971             //Loop over all the galaxies
00972             for (j = 0; j < NumGals; j++) {
00973                 i = GalaxiesInOrder[j];
00974                 prepare_galaxy_for_output(n, filenr, tree, &HaloGal[i], &galaxy_o
utput);
00975                 //Write Galaxy Tree
00976                 myfwrite(&galaxy_output, sizeof(struct GALAXY_OUTPUT), 1, fd);
00977                 TotGalCount++;
00978             } //for(i = 0; i < NumGals; i++)
00979
00980 #ifndef USE_MEMORY_TO_MINIMIZE_IO
00981 #ifndef NEW_IO
00982             fclose(fd);
00983 #endif
00984 #endif
00985
00986     myfree(GalaxiesInOrder);
00987     printf("TotGalCount= %d    at tree %d\n", TotGalCount, tree);
00988 }
00989
00990
00991 /**@brief Walks up the main leaf of a tree
00992 * TODO - is that what it actually does?*/
00993 int walk(int nr) {
00994     int last;
00995
00996     last = nr;
00997
00998     if (HaloGal[nr].Done == 0) {
00999         HaloGal[nr].Done = 1;
01000         HaloGal[nr].GalID = GalCount;
01001         GalaxiesInOrder[GalCount++] = nr;
01002         // set treeroot.
01003         if (HaloGal[nr].TreeRoot == -1)
01004             HaloGal[nr].TreeRoot = nr;
01005
01006         if (HaloGal[nr].FirstProgGal >= 0) {
01007             HaloGal[HaloGal[nr].FirstProgGal].TreeRoot = HaloGal[nr].
TreeRoot;
01008             last = walk(HaloGal[nr].FirstProgGal);
01009             HaloGal[nr].MainLeaf = HaloGal[HaloGal[nr].FirstProgGal].
MainLeaf;
01010         } else
01011             HaloGal[nr].MainLeaf = nr;
01012
01013         HaloGal[nr].LastProgGal = last;
01014
01015         if (HaloGal[nr].NextProgGal >= 0) {
01016             HaloGal[HaloGal[nr].NextProgGal].TreeRoot = HaloGal[nr].
TreeRoot;
01017             last = walk(HaloGal[nr].NextProgGal);
}

```

```
01018             }
01019         }
01020     return last;
01021 }
01022
01023
01024 #endif
```

Index

a0
 allvars.c, 39
 allvars.h, 64

AA
 allvars.c, 39
 allvars.h, 64

add_galaxies_together
 proto.h, 166
 recipe_mergers.c, 239

add_infall_to_hot
 proto.h, 167
 recipe_infall.c, 221

add_to_luminosities
 proto.h, 167
 recipe_misc.c, 261

Age
 allvars.c, 40
 allvars.h, 65

age.c
 integrand_time_to_present, 35
 time_to_present, 35

AgeTab
 allvars.c, 40
 allvars.h, 65

AgnEfficiency
 allvars.c, 40
 allvars.h, 65

AGNrecipeOn
 allvars.c, 40
 allvars.h, 65

allvars.c
 a0, 39
 AA, 39
 Age, 40
 AgeTab, 40
 AgnEfficiency, 40
 AGNrecipeOn, 40
 ar, 40
 BaryonFrac, 40
 BlackHoleGrowthRate, 40
 BoxSize, 40
 BulgeFormationInMinorMergersOn, 41
 CoolFunctionsDir, 41
 CoolingCutoff, 41
 CoolingVelocityCutOff, 41

 Corrections, 41
 CountIDs_halo, 41
 CountIDs_snaptree, 41
 DeltaM, 41
 DiskRadiusMethod, 41
 EjectionOn, 42
 EjectionRecipe, 42
 EjectPreVelocity, 42
 EjectSlope, 42
 EnergySN, 42
 EnergySNcode, 42
 EtaSN, 42
 EtaSNcode, 42
 FeedbackEjectionEfficiency, 43
 FeedbackEpsilon, 43
 FeedbackRecipe, 43
 FeedbackReheatingEpsilon, 43
 FileNameGalaxies, 43
 FileNrDir, 43
 FilesPerSnapshot, 43
 FileWithOutputSnaps, 43
 FileWithSnapList, 44
 filled_galaxydata, 44
 filled_galsnapdata, 44
 filled_momafdata, 44
 FirstFile, 44
 FirstHaloInSnap, 44
 FoF_MaxGals, 44
 Frac, 44
 FracZtoHot, 44
 G, 45
 Gal, 45
 GalaxiesInOrder, 45
 GalCount, 45
 H2, 45
 Halo, 45
 HaloAux, 46
 HaloGal, 46
 HaloIDs, 46
 Hashbits, 46
 Hubble, 46
 Hubble_h, 46
 IdList, 46
 LastFile, 46
 LastSnapShotNr, 47

ListInputFilrNr, 47
 ListOutputSnaps, 47
 MagTableZz, 47
 MaxGals, 47
 maxstorage_galaxydata, 47
 maxstorage_galsnapdata, 47
 maxstorage_momafdata, 47
 metalcold, 48
 MetallicityOption, 48
 mu_seed, 48
 Nids, 48
 NTask, 48
 NtotHalos, 48
 Ntrees, 48
 NumGals, 48
 NumMergers, 48
 offset_auxdata, 49
 offset_dbids, 49
 offset_galaxydata, 49
 offset_galsnapdata, 49
 offset_momafdata, 49
 offset_treedata, 49
 OffsetIDs, 49
 OffsetIDs_halo, 49
 OffsetIDs_snaptree, 50
 Omega, 50
 OmegaLambda, 50
 output_file, 50
 OutputDir, 50
 PartMass, 50
 PhotDir, 50
 PhotPrefix, 50
 PosList, 51
 ptr_auxdata, 51
 ptr_dbids, 51
 ptr_galaxydata, 51
 ptr_galsnapdata, 51
 ptr_momafdata, 51
 ptr_treedata, 51
 random_generator, 51
 RecGas, 51
 RecycleFraction, 52
 RedshiftTab, 52
 ReheatPreVelocity, 52
 ReheatSlope, 52
 ReIncorporationFactor, 52
 ReIncorporationRecipe, 52
 Reion_Mc, 52
 Reion_z, 52
 Reionization_z0, 52
 Reionization_zr, 53
 ReionizationOn, 53
 Rho, 53
 RhoCrit, 53
 SatelliteRecipe, 53
 SfrAlpha, 53
 SfrEfficiency, 53
 SfrLawPivotVelocity, 53
 SfrLawSlope, 53
 SimulationDir, 54
 Snaplistlen, 54
 SNinReheat, 54
 StarBurstRecipe, 54
 StarBurstsInMajorMergersOn, 54
 StarFormationRecipe, 54
 ThisTask, 54
 ThreshMajorMerger, 54
 TotGalaxies, 55
 TotGalCount, 55
 TotHalos, 55
 TotIds, 55
 TotSnaps, 55
 TrackDiskInstability, 55
 tree_file, 55
 treeaux_file, 55
 treedbids_file, 56
 TreeFirstHalo, 56
 TreeNgals, 56
 TreeNHalos, 56
 UnitCoolingRate_in_cgs, 56
 UnitDensity_in_cgs, 56
 UnitEnergy_in_cgs, 56
 UnitLength_in_cm, 56
 UnitMass_in_g, 56
 UnitPressure_in_cgs, 57
 UnitTime_in_Megayears, 57
 UnitTime_in_s, 57
 UnitVelocity_in_cm_per_s, 57
 VelList, 57
 Yield, 57
 ZZ, 57
 allvars.h
 a0, 64
 AA, 64
 Age, 65
 AgeTab, 65
 AgnEfficiency, 65
 AGNrecipeOn, 65
 ar, 65
 BaryonFrac, 65
 BlackHoleGrowthRate, 65
 BoxSize, 65
 BulgeFormationInMinorMergersOn, 66
 CoolFunctionsDir, 66
 CoolingCutoff, 66
 CoolingVelocityCutOff, 66
 Corrections, 66
 CountIDs_halo, 66

CountIDs_snaptree, 66
DeltaM, 66
DiskRadiusMethod, 66
EjectionOn, 67
EjectionRecipe, 67
EjectPreVelocity, 67
EjectSlope, 67
EnergySN, 67
EnergySNcode, 67
EtaSN, 67
EtaSNcode, 67
FeedbackEjectionEfficiency, 68
FeedbackEpsilon, 68
FeedbackRecipe, 68
FeedbackReheatingEpsilon, 68
FileNameGalaxies, 68
FileNrDir, 68
FilesPerSnapshot, 68
FileWithOutputSnaps, 68
FileWithSnapList, 69
filled_galaxydata, 69
filled_galsnapdata, 69
filled_momafdata, 69
FirstFile, 69
FirstHaloInSnap, 69
FoF_MaxGals, 69
Frac, 69
FracZtoHot, 69
G, 70
Gal, 70
GalaxiesInOrder, 70
GalCount, 70
H2, 70
Halo, 70
HaloAux, 70
HaloGal, 70
HaloIDs, 70
Hashbits, 70
Hubble, 70
Hubble_h, 70
IdList, 71
IdxTable, 71
IdxType, 71
Idxw5, 71
Idxw6, 71
LastFile, 71
LastSnapShotNr, 71
ListInputFilrNr, 71
ListOutputSnaps, 71
MagTableZz, 71
MaxGals, 72
maxstorage_galaxydata, 72
maxstorage_galsnapdata, 72
maxstorage_momafdata, 72
metalcold, 72
MetallicityOption, 72
mu_seed, 72
Nids, 72
NTask, 72
NtotHalos, 73
Ntrees, 73
NumGals, 73
NumMergers, 73
offset_auxdata, 73
offset_dbids, 73
offset_galaxydata, 73
offset_galsnapdata, 73
offset_momafdata, 74
offset_treedata, 74
OffsetIDs, 74
OffsetIDs_halo, 74
OffsetIDs_snaptree, 74
Omega, 74
OmegaLambda, 74
output_file, 74
OutputDir, 75
PartMass, 75
PhotDir, 75
PhotPrefix, 75
PosList, 75
ptr_auxdata, 75
ptr_dbids, 75
ptr_galaxydata, 75
ptr_galsnapdata, 76
ptr_momafdata, 76
ptr_treedata, 76
random_generator, 76
RecGas, 76
RecycleFraction, 76
RedshiftTab, 76
ReheatPreVelocity, 76
ReheatSlope, 76
ReIncorporationFactor, 77
ReIncorporationRecipe, 77
Reion_Mc, 77
Reion_z, 77
Reionization_z0, 77
Reionization_zr, 77
ReionizationOn, 77
Rho, 77
RhoCrit, 77
SatelliteRecipe, 78
SfrAlpha, 78
SfrEfficiency, 78
SfrLawPivotVelocity, 78
SfrLawSlope, 78
SimulationDir, 78
Snaplistlen, 78

SNinReheat, 78
 StarBurstRecipe, 79
 StarBurstsInMajorMergersOn, 79
 StarFormationRecipe, 79
 StelliteRecipe, 79
 ThisTask, 79
 ThreshMajorMerger, 79
 TotGalaxies, 79
 TotGalCount, 79
 TotHalos, 79
 TotIds, 79
 TotSnaps, 80
 TrackDiskInstability, 80
 tree_file, 80
 treeaux_file, 80
 treedbids_file, 80
 TreeFirstHalo, 80
 TreeNgals, 80
 TreeNHalos, 80
 UnitCoolingRate_in_cgs, 80
 UnitDensity_in_cgs, 81
 UnitEnergy_in_cgs, 81
 UnitLength_in_cm, 81
 UnitMass_in_g, 81
 UnitPressure_in_cgs, 81
 UnitTime_in_Megayears, 81
 UnitTime_in_s, 81
 UnitVelocity_in_cm_per_s, 81
 VelList, 82
 Yield, 82
 ZZ, 82
 ar
 allvars.c, 40
 allvars.h, 65
 BaryonFrac
 allvars.c, 40
 allvars.h, 65
 BlackHoleGrowthRate
 allvars.c, 40
 allvars.h, 65
 BlackHoleMass
 GALAXY, 7
 GALAXY_OUTPUT, 19
 BoxSize
 allvars.c, 40
 allvars.h, 65
 bulge_from_disk
 proto.h, 167
 BulgeFormationInMinorMergersOn
 allvars.c, 41
 allvars.h, 66
 BulgeMass
 GALAXY, 7
 GALAXY_OUTPUT, 19
 bulgemass_r
 proto.h, 167
 recipe_disrupt.c, 209
 BulgeSize
 GALAXY, 7
 GALAXY_OUTPUT, 19
 bulgesize_from_merger
 proto.h, 168
 recipe_mergers.c, 239
 cal_gas_recycle
 proto.h, 168
 recipe_starformation_and_feedback.c, 284
 cal_H2
 proto.h, 168
 recipe_misc.c, 261
 CentralGal
 GALAXY, 7
 CentralMvir
 GALAXY, 7
 GALAXY_OUTPUT, 19
 check_disk_instability
 proto.h, 168
 recipe_starformation_and_feedback.c, 284
 check_options
 proto.h, 168
 checkbulgesize_main
 proto.h, 169
 closeallfiles
 io_tree.c, 122
 proto.h, 169
 code/age.c, 35
 code/allvars.c, 36
 code/allvars.h, 61
 code/cool_func.c, 92
 code/init.c, 96
 code/io_tree.c, 120
 code/main.c, 141
 code/mymalloc.c, 154
 code/peano.c, 157
 code/proto.h, 160
 code/read_parameters.c, 195
 code/recipe_cooling.c, 202
 code/recipe_disrupt.c, 209
 code/recipe_dust.c, 215
 code/recipe_infall.c, 220
 code/recipe_mergers.c, 237
 code/recipe_misc.c, 259
 code/recipe_reincorporation.c, 280
 code/recipe_starformation_and_feedback.c, 283
 code/save.c, 301
 ColdGas
 GALAXY, 7

GALAXY_OUTPUT, 19
collisional_starburst_recipe
 proto.h, 169
 recipe_mergers.c, 239
construct_galaxies
 proto.h, 169
cool_func.c
 CoolRate, 93
 dmin, 92
 get_metaldependent_cooling_rate, 92
 get_rate, 92
 metallicities, 93
 name, 93
 read_cooling_functions, 92
 test, 92
cool_gas_onto_galaxy
 proto.h, 169
 recipe_cooling.c, 203
CoolFunctionsDir
 allvars.c, 41
 allvars.h, 66
cooling_recipe
 proto.h, 169
 recipe_cooling.c, 203
CoolingCutoff
 allvars.c, 41
 allvars.h, 66
CoolingGas
 GALAXY, 7
CoolingRadius
 GALAXY, 8
 GALAXY_OUTPUT, 19
CoolingVelocityCutOff
 allvars.c, 41
 allvars.h, 66
CoolRate
 cool_func.c, 93
Corrections
 allvars.c, 41
 allvars.h, 66
CountIDs_halo
 allvars.c, 41
 allvars.h, 66
CountIDs_snaptree
 allvars.c, 41
 allvars.h, 66
deal_with_galaxy_merger
 proto.h, 170
 recipe_mergers.c, 240
DeltaM
 allvars.c, 41
 allvars.h, 66
Descendant
 halo_data, 28
 halo_ids_data, 30
DescendantGal
 GALAXY, 8
 GALAXY_OUTPUT, 19
diskmass_r
 proto.h, 170
 recipe_disrupt.c, 209
DiskRadiusMethod
 allvars.c, 41
 allvars.h, 66
disrupt
 proto.h, 170
 recipe_disrupt.c, 210
DisruptOn
 GALAXY, 8
 GALAXY_OUTPUT, 19
dmax
 proto.h, 170
 recipe_misc.c, 262
dmin
 cool_func.c, 92
 proto.h, 170
do_AGN_heating
 proto.h, 171
 recipe_cooling.c, 203
do_major_merger_starburst
 proto.h, 171
 recipe_mergers.c, 240
do_reionization
 proto.h, 171
 recipe_infall.c, 221
dObsLum
 GALAXY, 8
dObsLumBulge
 GALAXY, 8
dObsLumDust
 GALAXY, 8
dObsMag
 GALAXY_OUTPUT, 19
dObsMagBulge
 GALAXY_OUTPUT, 20
 MOMAF_INPUTS, 32
dObsMagDust
 GALAXY_OUTPUT, 20
 MOMAF_INPUTS, 32
dObsMagICL
 GALAXY_OUTPUT, 20
dObsYLum
 GALAXY, 8
dObsYLumBulge
 GALAXY, 8
Done
 GALAXY, 9

DoneFlag
 halo_aux_data, 27
 dummy
 halo_ids_data, 30
 EjectedMass
 GALAXY, 9
 GALAXY_OUTPUT, 20
 EjectionOn
 allvars.c, 42
 allvars.h, 67
 EjectionRecipe
 allvars.c, 42
 allvars.h, 67
 EjectPreVelocity
 allvars.c, 42
 allvars.h, 67
 EjectSlope
 allvars.c, 42
 allvars.h, 67
 Energy_in_Reheat
 proto.h, 171
 EnergySN
 allvars.c, 42
 allvars.h, 67
 EnergySNcode
 allvars.c, 42
 allvars.h, 67
 estimate_merging_time
 proto.h, 171
 recipe_mergers.c, 240
 EtaSN
 allvars.c, 42
 allvars.h, 67
 EtaSNcode
 allvars.c, 42
 allvars.h, 67
 evolve_galaxies
 proto.h, 172
 FeedbackEjectionEfficiency
 allvars.c, 43
 allvars.h, 68
 FeedbackEpsilon
 allvars.c, 43
 allvars.h, 68
 FeedbackRecipe
 allvars.c, 43
 allvars.h, 68
 FeedbackReheatingEpsilon
 allvars.c, 43
 allvars.h, 68
 FileNameGalaxies
 allvars.c, 43
 allvars.h, 68
 FileNr
 halo_data, 28
 FileNrDir
 allvars.c, 43
 allvars.h, 68
 FilesPerSnapshot
 allvars.c, 43
 allvars.h, 68
 FileTreeNr
 GALAXY_OUTPUT, 20
 halo_ids_data, 31
 FileWithOutputSnaps
 allvars.c, 43
 allvars.h, 68
 FileWithSnapList
 allvars.c, 44
 allvars.h, 69
 filled_galaxydata
 allvars.c, 44
 allvars.h, 69
 filled_galsnapdata
 allvars.c, 44
 allvars.h, 69
 filled_momafdata
 allvars.c, 44
 allvars.h, 69
 finalize_galaxy_file
 proto.h, 172
 save.c, 302
 finalize_momaf_file
 proto.h, 172
 save.c, 302
 find_index
 init.c, 98
 find_interpolate_h2
 init.c, 98
 proto.h, 172
 find_interpolate_reionization
 init.c, 98
 find_interpolated_lum
 init.c, 98
 proto.h, 172
 find_interpolated_obs_lum
 init.c, 99
 proto.h, 173
 find_interpolation_point
 init.c, 99
 proto.h, 173
 FirstFile
 allvars.c, 44
 allvars.h, 69
 FirstGalaxy
 halo_aux_data, 27

FirstHaloInFOFgroup
 halo_data, 28
 halo_ids_data, 31
FirstHaloInSnap
 allvars.c, 44
 allvars.h, 69
FirstProgenitor
 halo_data, 28
 halo_ids_data, 31
FirstProgGal
 GALAXY, 9
 GALAXY_OUTPUT, 20
fix_units_for_ouput
 proto.h, 173
 save.c, 302
FoF_MaxGals
 allvars.c, 44
 allvars.h, 69
FOFCentralGal
 GALAXY_OUTPUT, 20
Frac
 allvars.c, 44
 allvars.h, 69
FracZtoHot
 allvars.c, 44
 allvars.h, 69
free_galaxies_and_tree
 io_tree.c, 122
 proto.h, 173
free_tree_table
 io_tree.c, 122
 proto.h, 173
func_size
 proto.h, 174

G
 allvars.c, 45
 allvars.h, 70

Gal
 allvars.c, 45
 allvars.h, 70

GalaxiesInOrder
 allvars.c, 45
 allvars.h, 70

GALAXY, 5
 BlackHoleMass, 7
 BulgeMass, 7
 BulgeSize, 7
 CentralGal, 7
 CentralMvir, 7
 ColdGas, 7
 CoolingGas, 7
 CoolingRadius, 8
 DescendantGal, 8

DisruptOn, 8
dObsLum, 8
dObsLumBulge, 8
dObsLumDust, 8
dObsYLum, 8
dObsYLumBulge, 8
Done, 9
EjectedMass, 9
FirstProgGal, 9
GalID, 9
GasDiskRadius, 9
GasSpin, 9
HaloNr, 9
HaloSpin, 10
HotFrac, 10
HotGas, 10
HotRadius, 10
ICLLum, 10
ICM, 10
Inclination, 10
InfallSnap, 10
InfallVmax, 11
LastProgGal, 11
Len, 11
Lum, 11
LumBulge, 11
LumDust, 11
MainLeaf, 11
MassWeightAge, 11
MergCentralPos, 12
MergeOn, 12
MergeSat, 12
MergTime, 12
MetalsBulgeMass, 12
MetalsColdGas, 12
MetalsEjectedMass, 12
MetalsHotGas, 13
MetalsICM, 13
MetalsStellarMass, 13
MostBoundID, 13
Mvir, 13
NextProgGal, 13
ObsICL, 13
ObsLum, 13
ObsLumBulge, 14
ObsLumDust, 14
ObsYLum, 14
ObsYLumBulge, 14
OriMergTime, 14
Pos, 14
Rvir, 14
Sfr, 15
SfrBulge, 15
SnapNum, 15

StarMerge, 15
 StellarDiskRadius, 15
 StellarMass, 15
 StellarSpin, 15
 TreeRoot, 16
 Type, 16
 Vel, 16
 Vmax, 16
 Vvir, 16
 XrayLum, 16
 YLum, 16
 YLumBulge, 17
GALAXY_OUTPUT, 17
 BlackHoleMass, 19
 BulgeMass, 19
 BulgeSize, 19
 CentralMvir, 19
 ColdGas, 19
 CoolingRadius, 19
 DescendantGal, 19
 DisruptOn, 19
 dObsMag, 19
 dObsMagBulge, 20
 dObsMagDust, 20
 dObsMagICL, 20
 EjectedMass, 20
 FileTreeNr, 20
 FirstProgGal, 20
 FOFCentralGal, 20
 GalID, 20
 GasDiskRadius, 20
 GasSpin, 21
 HaloID, 21
 HaloIndex, 21
 HotGas, 21
 HotRadius, 21
 ICM, 21
 InfallSnap, 21
 InfallVmax, 21
 LastProgGal, 21
 Len, 22
 Mag, 22
 MagBulge, 22
 MagDust, 22
 MagICL, 22
 MainLeafId, 22
 MassWeightAge, 22
 MergeOn, 22
 MergTime, 22
 MetalsBulgeMass, 23
 MetalsColdGas, 23
 MetalsEjectedMass, 23
 MetalsHotGas, 23
 MetalsICM, 23
 MetalsStellarMass, 23
 MMSubID, 23
 Mvir, 23
 NextProgGal, 23
 ObsMag, 24
 ObsMagBulge, 24
 ObsMagDust, 24
 ObsMagICL, 24
 OriMergTime, 24
 PeanoKey, 24
 Pos, 24
 Redshift, 24
 Rvir, 24
 Sfr, 25
 SfrBulge, 25
 SnapNum, 25
 StellarDiskRadius, 25
 StellarMass, 25
 StellarSpin, 25
 SubID, 25
 TreeRootId, 25
 Type, 25
 Vel, 26
 Vmax, 26
 Vvir, 26
 XrayLum, 26
GalCount
 allvars.c, 45
 allvars.h, 70
GalID
 GALAXY, 9
 GALAXY_OUTPUT, 20
 MOMAF_INPUTS, 32
gasdev
 recipe_dust.c, 215
GasDiskRadius
 GALAXY, 9
 GALAXY_OUTPUT, 20
GasSpin
 GALAXY, 9
 GALAXY_OUTPUT, 21
get_coordinates
 proto.h, 174
 save.c, 303
get_disk_radius
 proto.h, 174
 recipe_misc.c, 262
get_gas_disk_radius
 proto.h, 174
 recipe_misc.c, 262
get_initial_disk_radius
 proto.h, 174
 recipe_misc.c, 262
get_jump_index

init.c, 99
proto.h, 175
get_metaldependent_cooling_rate
 cool_func.c, 92
 proto.h, 175
get_metallicity
 proto.h, 175
 recipe_misc.c, 262
get_rate
 cool_func.c, 92
 proto.h, 175
get_stellar_disk_radius
 proto.h, 175
 recipe_misc.c, 263
get_virial_mass
 proto.h, 176
 recipe_misc.c, 263
get_virial_radius
 proto.h, 176
 recipe_misc.c, 263
get_virial_velocity
 proto.h, 176
 recipe_misc.c, 263
grow_black_hole
 proto.h, 176
 recipe_mergers.c, 241

H2
 allvars.c, 45
 allvars.h, 70

Halo
 allvars.c, 45
 allvars.h, 70

halo_aux_data, 26
 DoneFlag, 27
 FirstGalaxy, 27
 HaloFlag, 27
 NGalaxies, 27

halo_data, 27
 Descendant, 28
 FileNr, 28
 FirstHaloInFOFgroup, 28
 FirstProgenitor, 28
 Len, 28
 M_Crit200, 28
 M_Mean200, 28
 M_TopHat, 28
 MostBoundID, 29
 NextHaloInFOFgroup, 29
 NextProgenitor, 29
 Pos, 29
 SnapNum, 29
 Spin, 29
 SubHalfMass, 29

SubhaloIndex, 29
Vel, 29
VelDisp, 30
Vmax, 30
halo_ids_data, 30
 Descendant, 30
 dummy, 30
 FileTreeNr, 31
 FirstHaloInFOFgroup, 31
 FirstProgenitor, 31
 HaloID, 31
 LastProgenitor, 31
 MainLeafID, 31
 NextHaloInFOFgroup, 31
 NextProgenitor, 31
 PeanoKey, 31
 Redshift, 31

HaloAux
 allvars.c, 46
 allvars.h, 70

HaloFlag
 halo_aux_data, 27

HaloGal
 allvars.c, 46
 allvars.h, 70

HaloID
 GALAXY_OUTPUT, 21
 halo_ids_data, 31
 MOMAF_INPUTS, 32

HaloIDs
 allvars.c, 46
 allvars.h, 70

HaloIndex
 GALAXY_OUTPUT, 21

HaloNr
 GALAXY, 9

HaloSpin
 GALAXY, 10

Hashbits
 allvars.c, 46
 allvars.h, 70

HighMarkMem
 mymalloc.c, 155

hot_retain_sat
 proto.h, 176
 recipe_infall.c, 221

HotFrac
 GALAXY, 10

HotGas
 GALAXY, 10
 GALAXY_OUTPUT, 21

HotRadius
 GALAXY, 10
 GALAXY_OUTPUT, 21

Hubble
 allvars.c, 46
 allvars.h, 70

Hubble_h
 allvars.c, 46
 allvars.h, 70

ICLLum
 GALAXY, 10

ICM
 GALAXY, 10
 GALAXY_OUTPUT, 21

IdList
 allvars.c, 46
 allvars.h, 71

IdxTable
 allvars.h, 71

IdxType
 allvars.h, 71

Idxw5
 allvars.h, 71

Idxw6
 allvars.h, 71

Inclination
 GALAXY, 10

infall_recipe
 proto.h, 177
 recipe_infall.c, 222

InfallSnap
 GALAXY, 10
 GALAXY_OUTPUT, 21

InfallVmax
 GALAXY, 11
 GALAXY_OUTPUT, 21

init
 init.c, 99
 proto.h, 177

init.c
 find_index, 98
 find_interpolate_h2, 98
 find_interpolate_reionization, 98
 find_interpolated_lum, 98
 find_interpolated_obs_lum, 99
 find_interpolation_point, 99
 get_jump_index, 99
 init, 99
 init_jump_index, 99
 jumpfac, 102
 jumptab, 102
 read_file_nrs, 99
 read_output_snaps, 100
 read_recgas, 100
 read_reionization, 100
 read_sfrz, 100

 read_snap_list, 100
 read_tsud_tables, 100
 set_units, 101
 setup_spectrophotometric_model, 102

init_galaxy
 proto.h, 178
 recipe_misc.c, 264

init_jump_index
 init.c, 99
 proto.h, 178

integrand_time_to_present
 age.c, 35
 proto.h, 178

io_tree.c
 closeallfiles, 122
 free_galaxies_and_tree, 122
 free_tree_table, 122
 load_all_auxdata, 122
 load_all_dbids, 122
 load_all_treedata, 122
 load_tree, 123
 load_tree_table, 123
 myfread, 124
 myfseek, 124
 myfwrite, 124
 open_outputtree_file, 125
 open_tree_file, 125
 open_treeaux_file, 125
 open_treedbids_file, 125
 openallfiles, 125
 write_all_galaxy_data, 125
 write_galaxy_data_snap, 125
 write_galaxy_for_momaf, 126

join_galaxies_of_progenitors
 proto.h, 178

jumpfac
 init.c, 102

jumptab
 init.c, 102

LastFile
 allvars.c, 46
 allvars.h, 71

LastProgenitor
 halo_ids_data, 31

LastProgGal
 GALAXY, 11
 GALAXY_OUTPUT, 21

LastSnapShotNr
 allvars.c, 47
 allvars.h, 71

Len
 GALAXY, 11

GALAXY_OUTPUT, 22
halo_data, 28

ListInputFilrNr
 allvars.c, 47
 allvars.h, 71

ListOutputSnaps
 allvars.c, 47
 allvars.h, 71

load_all_auxdata
 io_tree.c, 122
 proto.h, 178

load_all_dbids
 io_tree.c, 122
 proto.h, 179

load_all_treedata
 io_tree.c, 122
 proto.h, 179

load_tree
 io_tree.c, 123
 proto.h, 179

load_tree_table
 io_tree.c, 123
 proto.h, 179

Lum
 GALAXY, 11

lum_to_mag
 proto.h, 180
 recipe_misc.c, 264

LumBulge
 GALAXY, 11

LumDust
 GALAXY, 11

M_Crit200
 halo_data, 28

M_Mean200
 halo_data, 28

M_TopHat
 halo_data, 28

Mag
 GALAXY_OUTPUT, 22

MagBulge
 GALAXY_OUTPUT, 22

MagDust
 GALAXY_OUTPUT, 22

MagICL
 GALAXY_OUTPUT, 22

MagTableZz
 allvars.c, 47
 allvars.h, 71

main
 main.c, 142

main.c
 main, 142

MainLeaf
 GALAXY, 11

MainLeafID
 halo_ids_data, 31

MainLeafId
 GALAXY_OUTPUT, 22

make_bulge_from_burst
 proto.h, 180
 recipe_mergers.c, 241

MassWeightAge
 GALAXY, 11
 GALAXY_OUTPUT, 22

MaxGals
 allvars.c, 47
 allvars.h, 72

maxstorage_galaxydata
 allvars.c, 47
 allvars.h, 72

maxstorage_galsnapdata
 allvars.c, 47
 allvars.h, 72

maxstorage_momafdata
 allvars.c, 47
 allvars.h, 72

MergCentralPos
 GALAXY, 12

MergeOn
 GALAXY, 12
 GALAXY_OUTPUT, 22

MergeSat
 GALAXY, 12

MergTime
 GALAXY, 12
 GALAXY_OUTPUT, 22

metalcold
 allvars.c, 48
 allvars.h, 72

metallicities
 cool_func.c, 93

MetallicityOption
 allvars.c, 48
 allvars.h, 72

MetalsBulgeMass
 GALAXY, 12
 GALAXY_OUTPUT, 23

MetalsColdGas
 GALAXY, 12
 GALAXY_OUTPUT, 23

MetalsEjectedMass
 GALAXY, 12
 GALAXY_OUTPUT, 23

MetalsHotGas
 GALAXY, 13
 GALAXY_OUTPUT, 23

MetalsICM
 GALAXY, 13
 GALAXY_OUTPUT, 23
 MetalsStellarMass
 GALAXY, 13
 GALAXY_OUTPUT, 23
 MMSubID
 GALAXY_OUTPUT, 23
 momaf_fwrite
 proto.h, 180
 MOMAF_INPUTS, 32
 dObsMagBulge, 32
 dObsMagDust, 32
 GalID, 32
 HaloID, 32
 ObsMagBulge, 32
 ObsMagDust, 33
 Pos, 33
 SnapNum, 33
 Vel, 33
 MostBoundID
 GALAXY, 13
 halo_data, 29
 mu_seed
 allvars.c, 48
 allvars.h, 72
 Mvir
 GALAXY, 13
 GALAXY_OUTPUT, 23
 myfread
 io_tree.c, 124
 proto.h, 180
 myfree
 mymalloc.c, 155
 proto.h, 181
 myfseek
 io_tree.c, 124
 proto.h, 181
 myfwrite
 io_tree.c, 124
 proto.h, 181
 mymalloc
 mymalloc.c, 155
 proto.h, 181
 mymalloc.c
 HighMarkMem, 155
 myfree, 155
 mymalloc, 155
 Nblocks, 155
 OldPrintedHighMark, 155
 print_allocated, 155
 SizeTable, 155
 Table, 155
 TotMem, 156
 name
 cool_func.c, 93
 Nblocks
 mymalloc.c, 155
 NextHaloInFOFgroup
 halo_data, 29
 halo_ids_data, 31
 NextProgenitor
 halo_data, 29
 halo_ids_data, 31
 NextProgGal
 GALAXY, 13
 GALAXY_OUTPUT, 23
 NGalaxies
 halo_aux_data, 27
 Nids
 allvars.c, 48
 allvars.h, 72
 NTask
 allvars.c, 48
 allvars.h, 72
 NtotHalos
 allvars.c, 48
 allvars.h, 73
 Ntrees
 allvars.c, 48
 allvars.h, 73
 NumGals
 allvars.c, 48
 allvars.h, 73
 NumMergers
 allvars.c, 48
 allvars.h, 73
 NumToTime
 proto.h, 181
 recipe_misc.c, 264
 ObsICL
 GALAXY, 13
 ObsLum
 GALAXY, 13
 ObsLumBulge
 GALAXY, 14
 ObsLumDust
 GALAXY, 14
 ObsMag
 GALAXY_OUTPUT, 24
 ObsMagBulge
 GALAXY_OUTPUT, 24
 MOMAF_INPUTS, 32
 ObsMagDust
 GALAXY_OUTPUT, 24
 MOMAF_INPUTS, 33
 ObsMagICL

proto.h
 add_galaxies_together, 166
 add_infall_to_hot, 167
 add_to_luminosities, 167
 bulge_from_disk, 167
 bulgemass_r, 167
 bulgesize_from_merger, 168
 cal_gas_recycle, 168
 cal_H2, 168
 check_disk_instability, 168
 check_options, 168
 checkbulgesize_main, 169
 closeallfiles, 169
 collisional_starburst_recipe, 169
 construct_galaxies, 169
 cool_gas_onto_galaxy, 169
 cooling_recipe, 169
 deal_with_galaxy_merger, 170
 diskmass_r, 170
 disrupt, 170
 dmax, 170
 dmin, 170
 do_AGN_heating, 171
 do_major_merger_starburst, 171
 do_reionization, 171
 Energy_in_Reheat, 171
 estimate_merging_time, 171
 evolve_galaxies, 172
 finalize_galaxy_file, 172
 finalize_momaf_file, 172
 find_interpolate_h2, 172
 find_interpolated_lum, 172
 find_interpolated_obs_lum, 173
 find_interpolation_point, 173
 fix_units_for_output, 173
 free_galaxies_and_tree, 173
 free_tree_table, 173
 func_size, 174
 get_coordinates, 174
 get_disk_radius, 174
 get_gas_disk_radius, 174
 get_initial_disk_radius, 174
 get_jump_index, 175
 get_metaldependent_cooling_rate, 175
 get_metallicity, 175
 get_rate, 175
 get_stellar_disk_radius, 175
 get_virial_mass, 176
 get_virial_radius, 176
 get_virial_velocity, 176
 grow_black_hole, 176
 hot_retain_sat, 176
 infall_recipe, 177
 init, 177

init_galaxy, 178
 init_jump_index, 178
 integrand_time_to_present, 178
 join_galaxies_of_progenitors, 178
 load_all_auxdata, 178
 load_all_dbids, 179
 load_all_treedata, 179
 load_tree, 179
 load_tree_table, 179
 lum_to_mag, 180
 make_bulge_from_burst, 180
 momaf_fwrite, 180
 myfread, 180
 myfree, 181
 myfseek, 181
 myfwrite, 181
 mymalloc, 181
 NumToTime, 181
 open_outputtree_file, 182
 open_tree_file, 182
 open_treeaux_file, 182
 open_treedbids_file, 182
 openallfiles, 182
 peano_hilbert_key, 182
 peri_radius, 182
 prepare_galaxy_for_momaf, 183
 prepare_galaxy_for_output, 183
 print_allocated, 184
 read_cooling_functions, 184
 read_file_nrs, 184
 read_output_snaps, 184
 read_parameter_file, 184
 read_recgas, 185
 read_sfrz, 185
 read_snap_list, 185
 read_tsud_tables, 185
 RedshiftObs, 186
 reincorporate_gas, 186
 sat_radius, 186
 save_galaxies, 186
 save_galaxy_tree, 187
 set_merger_center, 187
 set_units, 187
 setup_spectrophotometric_model, 188
 slab_model, 189
 starformation_and_feedback, 189
 time_to_present, 189
 tstudy, 189
 update_bulge_from_disk, 189
 update_centralgal, 189
 update_from_feedback, 190
 update_from_recycle, 190
 update_from_starFormation, 190
 update_hot_frac, 190

update_hotgas, 190
update_ICL, 191
update_type_1, 191
update_type_2, 191
walk, 191
write_all_galaxy_data, 191
write_galaxy_data_snap, 192
write_galaxy_for_momaf, 192

ptr_auxdata
 allvars.c, 51
 allvars.h, 75

ptr_dbids
 allvars.c, 51
 allvars.h, 75

ptr_galaxydata
 allvars.c, 51
 allvars.h, 75

ptr_galsnapdata
 allvars.c, 51
 allvars.h, 76

ptr_momafdata
 allvars.c, 51
 allvars.h, 76

ptr_treedata
 allvars.c, 51
 allvars.h, 76

quadrants
 peano.c, 158

ran1
 recipe_dust.c, 215

random_generator
 allvars.c, 51
 allvars.h, 76

read_cooling_functions
 cool_func.c, 92
 proto.h, 184

read_file_nrs
 init.c, 99
 proto.h, 184

read_output_snaps
 init.c, 100
 proto.h, 184

read_parameter_file
 proto.h, 184
 read_parameters.c, 196

read_parameters.c
 read_parameter_file, 196

read_recgas
 init.c, 100
 proto.h, 185

read_reionization
 init.c, 100

 read_sfrz
 init.c, 100
 proto.h, 185

 read_snap_list
 init.c, 100
 proto.h, 185

 read_tsud_tables
 init.c, 100
 proto.h, 185

 RecGas
 allvars.c, 51
 allvars.h, 76

 recipe_cooling.c
 cool_gas_onto_galaxy, 203
 cooling_recipe, 203
 do_AGN_heating, 203

 recipe_disrupt.c
 bulgemass_r, 209
 diskmass_r, 209
 disrupt, 210
 peri_radius, 210
 sat_radius, 210

 recipe_dust.c
 gasdev, 215
 ran1, 215
 tsudy, 216

 recipe_infall.c
 add_infall_to_hot, 221
 do_reionization, 221
 hot_retain_sat, 221
 infall_recipe, 222
 update_hot_frac, 222
 update_ICL, 222

 recipe_mergers.c
 add_galaxies_together, 239
 bulgesize_from_merger, 239
 collisional_starburst_recipe, 239
 deal_with_galaxy_merger, 240
 do_major_merger_starburst, 240
 estimate_merging_time, 240
 grow_black_hole, 241
 make_bulge_from_burst, 241
 set_merger_center, 241

 recipe_misc.c
 add_to_luminosities, 261
 cal_H2, 261
 dmax, 262
 get_disk_radius, 262
 get_gas_disk_radius, 262
 get_initial_disk_radius, 262
 get_metallicity, 262
 get_stellar_disk_radius, 263
 get_virial_mass, 263
 get_virial_radius, 263

get_virial_velocity, 263
 init_galaxy, 264
 lum_to_mag, 264
 NumToTime, 264
 RedshiftObs, 264
 update_centralgal, 264
 update_hotgas, 265
 update_type_1, 265
 update_type_2, 265
 recipe_reincorporation.c
 reincorporate_gas, 281
 recipe_starformation_and_feedback.c
 cal_gas_recycle, 284
 check_disk_instability, 284
 starformation_and_feedback, 284
 update_from_feedback, 285
 update_from_star_formation, 285
 RecycleFraction
 allvars.c, 52
 allvars.h, 76
 Redshift
 GALAXY_OUTPUT, 24
 halo_ids_data, 31
 RedshiftObs
 proto.h, 186
 recipe_misc.c, 264
 RedshiftTab
 allvars.c, 52
 allvars.h, 76
 ReheatPreVelocity
 allvars.c, 52
 allvars.h, 76
 ReheatSlope
 allvars.c, 52
 allvars.h, 76
 reincorporate_gas
 proto.h, 186
 recipe_reincorporation.c, 281
 ReIncorporationFactor
 allvars.c, 52
 allvars.h, 77
 ReIncorporationRecipe
 allvars.c, 52
 allvars.h, 77
 Reion_Mc
 allvars.c, 52
 allvars.h, 77
 Reion_z
 allvars.c, 52
 allvars.h, 77
 Reionization_z0
 allvars.c, 52
 allvars.h, 77
 Reionization_zr

allvars.c, 53
 allvars.h, 77
 ReionizationOn
 allvars.c, 53
 allvars.h, 77
 Rho
 allvars.c, 53
 allvars.h, 77
 RhoCrit
 allvars.c, 53
 allvars.h, 77
 rotx_table
 peano.c, 158
 rotxmap_table
 peano.c, 158
 roty_table
 peano.c, 158
 rotymap_table
 peano.c, 158
 Rvir
 GALAXY, 14
 GALAXY_OUTPUT, 24
 sat_radius
 proto.h, 186
 recipe_disrupt.c, 210
 SatelliteRecipe
 allvars.c, 53
 allvars.h, 78
 save.c
 finalize_galaxy_file, 302
 finalize_momaf_file, 302
 fix_units_for_ouput, 302
 get_coordinates, 303
 prepare_galaxy_for_momaf, 303
 prepare_galaxy_for_output, 303
 save_galaxies, 304
 save_galaxy_tree, 305
 walk, 305
 save_galaxies
 proto.h, 186
 save.c, 304
 save_galaxy_tree
 proto.h, 187
 save.c, 305
 sense_table
 peano.c, 158
 set_merger_center
 proto.h, 187
 recipe_mergers.c, 241
 set_units
 init.c, 101
 proto.h, 187
 setup_spectrophotometric_model

init.c, 102
proto.h, 188

Sfr
 GALAXY, 15
 GALAXY_OUTPUT, 25

SfrAlpha
 allvars.c, 53
 allvars.h, 78

SfrBulge
 GALAXY, 15
 GALAXY_OUTPUT, 25

SfrEfficiency
 allvars.c, 53
 allvars.h, 78

SfrLawPivotVelocity
 allvars.c, 53
 allvars.h, 78

SfrLawSlope
 allvars.c, 53
 allvars.h, 78

SimulationDir
 allvars.c, 54
 allvars.h, 78

SizeTable
 mymalloc.c, 155

slab_model
 proto.h, 189

Snaplistlen
 allvars.c, 54
 allvars.h, 78

SnapNum
 GALAXY, 15
 GALAXY_OUTPUT, 25
 halo_data, 29
 MOMAF_INPUTS, 33

SNinReheat
 allvars.c, 54
 allvars.h, 78

Spin
 halo_data, 29

StarBurstRecipe
 allvars.c, 54
 allvars.h, 79

StarBurstsInMajorMergersOn
 allvars.c, 54
 allvars.h, 79

starformation_and_feedback
 proto.h, 189
 recipe_starformation_and_feedback.c, 284

StarFormationRecipe
 allvars.c, 54
 allvars.h, 79

StarMerge
 GALAXY, 15

StellarDiskRadius
 GALAXY, 15
 GALAXY_OUTPUT, 25

StellarMass
 GALAXY, 15
 GALAXY_OUTPUT, 25

StellarSpin
 GALAXY, 15
 GALAXY_OUTPUT, 25

StelliteRecipe
 allvars.h, 79

SubHalfMass
 halo_data, 29

SubhaloIndex
 halo_data, 29

SubID
 GALAXY_OUTPUT, 25

Table
 mymalloc.c, 155

test
 cool_func.c, 92

ThisTask
 allvars.c, 54
 allvars.h, 79

ThreshMajorMerger
 allvars.c, 54
 allvars.h, 79

time_to_present
 age.c, 35
 proto.h, 189

TotGalaxies
 allvars.c, 55
 allvars.h, 79

TotGalCount
 allvars.c, 55
 allvars.h, 79

TotHalos
 allvars.c, 55
 allvars.h, 79

TotIds
 allvars.c, 55
 allvars.h, 79

TotMem
 mymalloc.c, 156

TotSnaps
 allvars.c, 55
 allvars.h, 80

TrackDiskInstability
 allvars.c, 55
 allvars.h, 80

tree_file
 allvars.c, 55
 allvars.h, 80

treeaux_file
 allvars.c, 55
 allvars.h, 80

treedbids_file
 allvars.c, 56
 allvars.h, 80

TreeFirstHalo
 allvars.c, 56
 allvars.h, 80

TreeNgals
 allvars.c, 56
 allvars.h, 80

TreeNHalos
 allvars.c, 56
 allvars.h, 80

TreeRoot
 GALAXY, 16

TreeRootId
 GALAXY_OUTPUT, 25

tsudy
 proto.h, 189
 recipe_dust.c, 216

Type
 GALAXY, 16
 GALAXY_OUTPUT, 25

UnitCoolingRate_in_cgs
 allvars.c, 56
 allvars.h, 80

UnitDensity_in_cgs
 allvars.c, 56
 allvars.h, 81

UnitEnergy_in_cgs
 allvars.c, 56
 allvars.h, 81

UnitLength_in_cm
 allvars.c, 56
 allvars.h, 81

UnitMass_in_g
 allvars.c, 56
 allvars.h, 81

UnitPressure_in_cgs
 allvars.c, 57
 allvars.h, 81

UnitTime_in_Megayears
 allvars.c, 57
 allvars.h, 81

UnitTime_in_s
 allvars.c, 57
 allvars.h, 81

UnitVelocity_in_cm_per_s
 allvars.c, 57
 allvars.h, 81

update_bulge_from_disk

proto.h, 189
 update_centralgal
 proto.h, 189
 recipe_misc.c, 264

update_from_feedback
 proto.h, 190
 recipe_starformation_and_feedback.c, 285

update_from_recycle
 proto.h, 190

update_from_star_formation
 proto.h, 190
 recipe_starformation_and_feedback.c, 285

update_hot_frac
 proto.h, 190
 recipe_infall.c, 222

update_hotgas
 proto.h, 190
 recipe_misc.c, 265

update_ICL
 proto.h, 191
 recipe_infall.c, 222

update_type_1
 proto.h, 191
 recipe_misc.c, 265

update_type_2
 proto.h, 191
 recipe_misc.c, 265

Vel
 GALAXY, 16
 GALAXY_OUTPUT, 26
 halo_data, 29
 MOMAF_INPUTS, 33

VelDisp
 halo_data, 30

VelList
 allvars.c, 57
 allvars.h, 82

Vmax
 GALAXY, 16
 GALAXY_OUTPUT, 26
 halo_data, 30

Vvir
 GALAXY, 16
 GALAXY_OUTPUT, 26

walk
 proto.h, 191
 save.c, 305

write_all_galaxy_data
 io_tree.c, 125
 proto.h, 191

write_galaxy_data_snap
 io_tree.c, 125

proto.h, [192](#)
write_galaxy_for_momaf
 io_tree.c, [126](#)
 proto.h, [192](#)

XrayLum
 GALAXY, [16](#)
 GALAXY_OUTPUT, [26](#)

Yield
 allvars.c, [57](#)
 allvars.h, [82](#)

YLum
 GALAXY, [16](#)
YLumBulge
 GALAXY, [17](#)

ZZ
 allvars.c, [57](#)
 allvars.h, [82](#)