Land Information System

LIS Reference Manual

January 5, 2008

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, MD 20771
http://lis.gsfc.nasa.gov

Contents

I LIS Overview **25**

II LIS offline program **26**

1 LIS offline program **27**
1.0.1 lisdrv (Source File: lisdrv.F90) . . . 27

III LIS core structures and methods **28**

2 LIS core structures and methods **29**
2.1 Fortran: Module Interface lisdrv_module (Source File: lisdrv_module.F90) . . . 29
2.1.1 Overview . . . 29
2.1.2 lisconfig (Source File: lisdrv_module.F90) . . . 30
2.1.3 lisconfig_offline (Source File: lisdrv_module.F90) . . . 30
2.1.4 lisconfig_coupled1 (Source File: lisdrv_module.F90) . . . 31
2.1.5 lisconfig_coupled2 (Source File: lisdrv_module.F90) . . . 32
2.1.6 LIS_ticktime (Source File: lisdrv_module.F90) . . . 32
2.1.7 LIS_endofrun (Source File: lisdrv_module.F90) . . . 33
2.1.8 LIS_timeToRunNest (Source File: lisdrv_module.F90) . . . 33
2.1.9 LIS_resetclock (Source File: lisdrv_module.F90) . . . 33
2.1.10 LIS_finalize (Source File: lisdrv_module.F90) . . . 34
2.2 Fortran: Module Interface lis_module (Source File: lis_module.F90) . . . 34
2.3 Fortran: Module Interface grid_module (Source File: grid_module.F90) . . . 39
2.3.1 Overview . . . 39
2.4 Fortran: Module Interface listime_mgr (Source File: listime_mgr.F90) . . . 40
2.4.1 isMonthlyAlarmRinging (Source File: listime_mgr.F90) . . . 40
2.4.2 timemgr_init (Source File: listime_mgr.F90) . . . 40
2.4.3 resetclock (Source File: listime_mgr.F90) . . . 41
2.4.4 timemgr_set (Source File: listime_mgr.F90) . . . 41
2.4.5 timemgr_print (Source File: listime_mgr.F90) . . . 42
2.4.6 advance_timestep (Source File: listime_mgr.F90) . . . 42
2.4.7 get_step_size (Source File: listime_mgr.F90) . . . 42
2.4.8 get_nstep (Source File: listime_mgr.F90) . . . 43
2.4.9 get_curr_day (Source File: listime_mgr.F90) . . . 43
2.4.10 get_julhr (Source File: listime_mgr.F90) . . . 44
2.4.11 get_curr_calday (Source File: listime_mgr.F90) . . . 44
2.4.12 is_last_step (Source File: listime_mgr.F90) . . . 45
2.4.13 setMonthlyAlarm (Source File: listime_mgr.F90) . . . 45
2.4.14 isMonthlyAlarmRinging1 (Source File: listime_mgr.F90) . . . 46
2.4.15 isMonthlyAlarmRinging2 (Source File: listime_mgr.F90) . . . 46
2.4.16 setHourlyAlarm (Source File: listime_mgr.F90) . . . 47
2.4.17 isHourlyAlarmRinging (Source File: listime_mgr.F90) . . . 47
2.4.18 computeTimeBookEnds (Source File: listime_mgr.F90) . . . 48
2.4.19 computeMonthlyWeights (Source File: listime_mgr.F90) . . . 48
2.4.20 computeMonthlyWeights (Source File: listime_mgr.F90) . . . 49
2.4.21 date2time (Source File: listime_mgr.F90) . . . 50
2.4.22 time2date (Source File: listime_mgr.F90) . . . 50

2.4.23 tick (Source File: listime_mgr.F90) . . . 51
2.4.24 julhr_date (Source File: listime_mgr.F90) . . . 52
2.4.25 Method . . . 52
2.4.26 tmjul4 (Source File: listime_mgr.F90) . . . 53
2.4.27 Method . . . 53
2.5 Fortran: Module Interface spmdMod (Source File: spmdMod.F90) . . . 53
2.5.1 spmd_init (Source File: spmdMod.F90) . . . 54
2.5.2 spmd_init_offline (Source File: spmdMod.F90) . . . 54
2.5.3 spmd_init_coupled (Source File: spmdMod.F90) . . . 55
2.5.4 spmd_setup (Source File: spmdMod.F90) . . . 55
2.5.5 spmd_finalize (Source File: spmdMod.F90) . . . 55
2.6 Fortran: Module Interface output_metaMod (Source File: output_metaMod.F90) . . . 56
2.7 Fortran: Module Interface domain_module.F90 (Source File: domain_module.F90) . . . 57
2.7.1 Overview . . . 57
2.7.2 LIS_domain_init (Source File: domain_module.F90) . . . 58
2.7.3 LIS_domain_finalize (Source File: domain_module.F90) . . . 58
2.8 Fortran: Module Interface lsm_module (Source File: lsm_module.F90) . . . 58
2.8.1 Overview . . . 58
2.8.2 LIS_lsm_init (Source File: lsm_module.F90) . . . 59
2.8.3 LIS_setuplsm (Source File: lsm_module.F90) . . . 60
2.8.4 LIS_lsm_main (Source File: lsm_module.F90) . . . 60
2.8.5 LIS_perturb_states (Source File: lsm_module.F90) . . . 61
2.8.6 LIS_readrestart (Source File: lsm_module.F90) . . . 61
2.8.7 LIS_lsm_output (Source File: lsm_module.F90) . . . 62
2.8.8 setLSMDynparams (Source File: lsm_module.F90) . . . 62
2.8.9 LIS_force2tile (Source File: lsm_module.F90) . . . 63
2.8.10 LIS_writerestart (Source File: lsm_module.F90) . . . 63
2.8.11 LIS_lsm_setexport (Source File: lsm_module.F90) . . . 64
2.8.12 LIS_lsm_finalize (Source File: lsm_module.F90) . . . 64
2.8.13 applyLSMPert (Source File: lsm_module.F90) . . . 64
2.9 Fortran: Module Interface baseforcing_module (Source File: baseforcing_module.F90) . . . 65
2.9.1 Overview . . . 65
2.9.2 LIS_baseforcing_init (Source File: baseforcing_module.F90) . . . 66
2.9.3 LIS_baseforcing_init_coupled (Source File: baseforcing_module.F90) . . . 67
2.9.4 perturbSetup (Source File: baseforcing_module.F90) . . . 67
2.9.5 LIS_perturb_forcing (Source File: baseforcing_module.F90) . . . 67
2.9.6 LIS_get_base_forcing (Source File: baseforcing_module.F90) . . . 68
2.9.7 LIS_baseforcing_finalize (Source File: baseforcing_module.F90) . . . 68
2.9.8 allocate_forcing_mem (Source File: baseforcing_module.F90) . . . 69
2.9.9 allocate_forcing_mem_coupled (Source File: baseforcing_module.F90) . . . 69
2.10 Fortran: Module Interface suppforcing_module (Source File: suppforcing_module.F90) . . . 69
2.10.1 Overview . . . 70
2.10.2 LIS_suppforcing_init (Source File: suppforcing_module.F90) . . . 70
2.10.3 LIS_get_supp_forcing (Source File: suppforcing_module.F90) . . . 71
2.10.4 LIS_suppforcing_finalize (Source File: suppforcing_module.F90) . . . 71
2.11 Fortran: Module Interface dataassim_module (Source File: dataassim_module.F90) . . . 72
2.11.1 Overview . . . 72
2.11.2 LIS_dataassim_init (Source File: dataassim_module.F90) . . . 72
2.11.3 LIS_dataassim_run (Source File: dataassim_module.F90) . . . 73
2.11.4 LIS_dataassim_output (Source File: dataassim_module.F90) . . . 73
2.11.5 LIS_dataassim_finalize (Source File: dataassim_module.F90) . . . 74

2.12 Fortran: Module Interface observations_module (Source File: observations_module.F90) . . . 74
2.12.1 LIS_initObservations (Source File: observations_module.F90) . . . 74
2.12.2 LIS_readobservations (Source File: observations_module.F90) . . . 75
2.12.3 LIS_perturb_observations (Source File: observations_module.F90) . . . 76
2.13 Fortran: Module Interface perturb_module (Source File: perturb_module.F90) . . . 76
2.13.1 LIS_perturb_init (Source File: perturb_module.F90) . . . 76
2.14 Fortran: Module Interface param_module (Source File: param_module.F90) . . . 77
2.14.1 Overview . . . 77
2.14.2 LIS_param_init (Source File: param_module.F90) . . . 77
2.14.3 LIS_setDynparams (Source File: param_module.F90) . . . 78
2.14.4 LIS_output_params (Source File: param_module.F90) . . . 79
2.14.5 LIS_param_finalize (Source File: param_module.F90) . . . 80
2.14.6 writelandmask (Source File: param_module.F90) . . . 81
2.14.7 writelandcover (Source File: param_module.F90) . . . 81
2.14.8 writetopography (Source File: param_module.F90) . . . 82
2.15 Fortran: Module Interface albedo_module (Source File: albedo_module.F90) . . . 82
2.15.1 Overview . . . 83
2.15.2 albedo_setup (Source File: albedo_module.F90) . . . 83
2.15.3 read_mxsnalb (Source File: albedo_module.F90) . . . 84
2.15.4 read_albedo (Source File: albedo_module.F90) . . . 84
2.15.5 albedo_finalize (Source File: albedo_module.F90) . . . 85
2.15.6 writealbedo (Source File: albedo_module.F90) . . . 85
2.16 Fortran: Module Interface gfrac_module (Source File: gfrac_module.F90) . . . 85
2.16.1 Overview . . . 86
2.16.2 greenness_setup (Source File: gfrac_module.F90) . . . 86
2.16.3 read_greenness (Source File: gfrac_module.F90) . . . 86
2.16.4 greenness_finalize (Source File: gfrac_module.F90) . . . 87
2.16.5 writegfrac (Source File: gfrac_module.F90) . . . 87
2.17 Fortran: Module Interface lai_module (Source File: lai_module.F90) . . . 88
2.17.1 Overview . . . 88
2.17.2 lai_setup (Source File: lai_module.F90) . . . 89
2.17.3 read_lai (Source File: lai_module.F90) . . . 89
2.17.4 lai_finalize (Source File: lai_module.F90) . . . 89
2.17.5 writeLAI (Source File: lai_module.F90) . . . 90
2.18 Fortran: Module Interface sai_module (Source File: sai_module.F90) . . . 90
2.18.1 Overview . . . 90
2.18.2 sai_setup (Source File: sai_module.F90) . . . 91
2.18.3 read_sai (Source File: sai_module.F90) . . . 91
2.18.4 sai_finalize (Source File: sai_module.F90) . . . 92
2.18.5 writeSAI (Source File: sai_module.F90) . . . 92
2.19 Fortran: Module Interface snow_module (Source File: snow_module.F90) . . . 92
2.19.1 snow_setup (Source File: snow_module.F90) . . . 93
2.19.2 read_snow (Source File: snow_module.F90) . . . 93
2.19.3 snow_finalize (Source File: snow_module.F90) . . . 94
2.20 Fortran: Module Interface soils_module (Source File: soils_module.F90) . . . 94
2.20.1 Overview . . . 94
2.20.2 soils_setup (Source File: soils_module.F90) . . . 95
2.20.3 read_soils (Source File: soils_module.F90) . . . 95
2.20.4 soils_finalize (Source File: soils_module.F90) . . . 96
2.20.5 writesoils (Source File: soils_module.F90) . . . 96
2.20.6 check_error (Source File: check_error.F90) . . . 96

2.21 Fortran: Module Interface constantsmod.F90 (Source File: constantsMod.F90) 97
2.22 Fortran: Module Interface drv_output_mod (Source File: drv_output_mod.F90) 98
2.22.1 Overview 98
2.22.2 drv_writevar_bin (Source File: drv_output_mod.F90) 99
2.22.3 drv_writevar_restart (Source File: drv_output_mod.F90) 99
2.22.4 drv_readvar_restart (Source File: drv_output_mod.F90) 100
2.22.5 drv_readvar_gridded (Source File: drv_output_mod.F90) 100
2.22.6 drv_writevar_grib (Source File: drv_output_mod.F90) 100
2.22.7 drv_writevar_netcdf (Source File: drv_output_mod.F90) 101
2.22.8 writevar_bin_withstats_real (Source File: drv_output_mod.F90) 101
2.22.9 writevar_bin_real (Source File: drv_output_mod.F90) 102
2.22.10 writevar_restart_int (Source File: drv_output_mod.F90) 102
2.22.11 writevar_restart_real (Source File: drv_output_mod.F90) 103
2.22.12 writevar_restart_char (Source File: drv_output_mod.F90) 103
2.22.13 readvar_restart_int (Source File: drv_output_mod.F90) 104
2.22.14 readvar_restart_real (Source File: drv_output_mod.F90) 105
2.22.15 readvar_restart_char (Source File: drv_output_mod.F90) 105
2.22.16 readvar_gridded_real (Source File: drv_output_mod.F90) 106
2.22.17 writevar_netcdf_real (Source File: drv_output_mod.F90) 106
2.22.18 writevar_netcdf_real3d (Source File: drv_output_mod.F90) 107
2.22.19 writevar_netcdf_withstats_real (Source File: drv_output_mod.F90) 108
2.22.20 writevar_grib_real (Source File: drv_output_mod.F90) 108
2.22.21 writevar_grib_withstats_real (Source File: drv_output_mod.F90) 109
2.22.22 writevar_grib_withstats_real3d (Source File: drv_output_mod.F90) 111
2.22.23 drv_tile2grid (Source File: drv_output_mod.F90) 112
2.22.24 drv_write_innov (Source File: drv_output_mod.F90) 112
2.22.25 drv_write_dastats (Source File: drv_output_mod.F90) 113
2.22.26 endrun (Source File: endrun.F90) 114
2.23 Fortran: Module Interface gswp_module (Source File: gswp_module.F90) 114
2.23.1 getgswp_monindex (Source File: gswp_module.F90) 115
2.23.2 getgswp_timeindex (Source File: gswp_module.F90) 115
2.24 Fortran: Module Interface grib_module (Source File: grib_module.F90) 115
2.24.1 grib_init (Source File: grib_module.F90) 116
2.24.2 grib1_finalize (Source File: grib_module.F90) 116
2.24.3 grib1_setup (Source File: grib_module.F90) 116
2.24.4 grib2_setup (Source File: grib_module.F90) 117
2.24.5 grib3_setup (Source File: grib_module.F90) 117
2.24.6 gribdat (Source File: grib_module.F90) 118
2.24.7 lapseRateCorrection (Source File: lapseRateCorrection.F90) 118
2.25 Fortran: Module Interface lis_fileIOMod (Source File: lis_fileIOMod.F90) 119
2.25.1 putget (Source File: lis_fileIOMod.F90) 119
2.25.2 create_output_directory (Source File: lis_fileIOMod.F90) 120
2.25.3 create_output_filename (Source File: lis_fileIOMod.F90) 121
2.25.4 create_restart_filename (Source File: lis_fileIOMod.F90) 121
2.25.5 create_stats_filename (Source File: lis_fileIOMod.F90) 122
2.25.6 create_innov_filename (Source File: lis_fileIOMod.F90) 123
2.25.7 create_daspread_filename (Source File: lis_fileIOMod.F90) 123
2.25.8 putget_int (Source File: lis_fileIOMod.F90) 124
2.25.9 putget_real (Source File: lis_fileIOMod.F90) 124
2.25.10 create_gain_filename (Source File: lis_fileIOMod.F90) 125
2.26 Fortran: Module Interface lis_fileUnitsMod (Source File: lis_fileUnitsMod.F90) 126

2.26.1 getNextUnitNumber (Source File: lis_fileUnitsMod.F90) . . . 126
2.26.2 releaseUnitNumber (Source File: lis_fileUnitsMod.F90) . . . 126
2.27 Fortran: Module Interface lis_logmod (Source File: lis_logmod.F90) . . . 127
2.27.1 lis_log_init (Source File: lis_logmod.F90) . . . 127
2.27.2 lis_log_msg (Source File: lis_logmod.F90) . . . 127
2.27.3 lis_abort (Source File: lis_logmod.F90) . . . 128
2.27.4 lis_alert (Source File: lis_logmod.F90) . . . 128
2.27.5 lis_flush (Source File: lis_logmod.F90) . . . 129
2.27.6 makepdsn (Source File: makepdsn.F90) . . . 129
2.27.7 microMetCorrection (Source File: microMetCorrection.F90) . . . 130
2.28 Fortran: Module Interface mpishorthand.F90 (Source File: mpishorthand.F90) . . . 130
2.29 Fortran: Module Interface precision (Source File: precision.F90) . . . 131
2.29.1 readcard (Source File: readcard.F90) . . . 131
2.29.2 readkpds (Source File: readkpds.F90) . . . 132
2.29.3 soiltype (Source File: soiltype.F90) . . . 133
2.29.4 stats (Source File: stats.F90) . . . 134
2.29.5 stats_da (Source File: stats_da.F90) . . . 134
2.29.6 zterp (Source File: zterp.F90) . . . 134
2.29.7 coszenith (Source File: zterp.F90) . . . 136
2.29.8 LIS_localtime (Source File: zterp.F90) . . . 137
2.29.9 gasdev (Source File: gasdev.F90) . . . 137
2.29.10 ran2 (Source File: gasdev.F90) . . . 137
2.30 Fortran: Module Interface infnan (Source File: infnan.F90) . . . 137
2.31 Fortran: Module Interface optimization_module (Source File: optimization_module.F90) . . . 138
2.31.1 LIS_optimization_init (Source File: optimization_module.F90) . . . 138
2.31.2 readObsAttributes (Source File: readObsAttributes.F90) . . . 138
2.31.3 readObsPertAttributes (Source File: readObsPertAttributes.F90) . . . 139
2.31.4 tile_module (Source File: tile_module.F90) . . . 140
2.31.5 Overview . . . 140
2.31.6 verify (Source File: verify.F90) . . . 140
2.31.7 ft_check_index (Source File: FTable.c) . . . 141
2.32 Fortran: Module Interface LIS_alb_FTable (Source File: LIS_alb_FTable.c) . . . 141
2.32.1 registerreadmxsnoalb (Source File: LIS_alb_FTable.c) . . . 141
2.32.2 readmxsnoalb (Source File: LIS_alb_FTable.c) . . . 142
2.32.3 registerreadalbedo (Source File: LIS_alb_FTable.c) . . . 142
2.32.4 readalbedo (Source File: LIS_alb_FTable.c) . . . 142
2.33 Fortran: Module Interface LIS_dataassim_FTable (Source File: LIS_dataassim_FTable.c) . . . 143
2.33.1 registerdaobssetup (Source File: LIS_dataassim_FTable.c) . . . 143
2.33.2 readobsdataconfig (Source File: LIS_dataassim_FTable.c) . . . 143
2.33.3 registerdaobs (Source File: LIS_dataassim_FTable.c) . . . 143
2.33.4 readdaobs (Source File: LIS_dataassim_FTable.c) . . . 144
2.33.5 registergetnso (Source File: LIS_dataassim_FTable.c) . . . 144
2.33.6 getselectedobsnumber (Source File: LIS_dataassim_FTable.c) . . . 144
2.33.7 registerdainit (Source File: LIS_dataassim_FTable.c) . . . 145
2.33.8 dataassiminit (Source File: LIS_dataassim_FTable.c) . . . 145
2.33.9 registerapplyincrements (Source File: LIS_dataassim_FTable.c) . . . 145
2.33.10 applyincrements (Source File: LIS_dataassim_FTable.c) . . . 145
2.33.11 registercomputeincrements (Source File: LIS_dataassim_FTable.c) . . . 146
2.33.12 computeincrements (Source File: LIS_dataassim_FTable.c) . . . 146
2.33.13 registerdafinalize (Source File: LIS_dataassim_FTable.c) . . . 146
2.33.14 dafinalize (Source File: LIS_dataassim_FTable.c) . . . 146

2.33.15 registerdaoutput (Source File: LIS_dataassim_FTable.c) . . . 147
2.33.16 daoutput (Source File: LIS_dataassim_FTable.c) . . . 147
2.34 Fortran: Module Interface LIS_domain_FTable (Source File: LIS_domain_FTable.c) . . . 147
2.34.1 registerdomain (Source File: LIS_domain_FTable.c) . . . 147
2.34.2 makedomain (Source File: LIS_domain_FTable.c) . . . 148
2.34.3 registerinput (Source File: LIS_domain_FTable.c) . . . 148
2.34.4 readinput (Source File: LIS_domain_FTable.c) . . . 148
2.35 Fortran: Module Interface LIS_forcing_FTable (Source File: LIS_forcing_FTable.c) . . . 148
2.35.1 registerget (Source File: LIS_forcing_FTable.c) . . . 149
2.35.2 retrieveforcing (Source File: LIS_forcing_FTable.c) . . . 149
2.35.3 registertimeinterp (Source File: LIS_forcing_FTable.c) . . . 149
2.35.4 timeinterp (Source File: LIS_forcing_FTable.c) . . . 149
2.35.5 registerdefinenative (Source File: LIS_forcing_FTable.c) . . . 150
2.35.6 definenative (Source File: LIS_forcing_FTable.c) . . . 150
2.35.7 registerforcingfinal (Source File: LIS_forcing_FTable.c) . . . 150
2.35.8 forcingfinalize (Source File: LIS_forcing_FTable.c) . . . 151
2.35.9 registerreadforcingelev (Source File: LIS_forcing_FTable.c) . . . 151
2.35.10 readforcingelev (Source File: LIS_forcing_FTable.c) . . . 151
2.36 Fortran: Module Interface LIS_gfrac_FTable (Source File: LIS_gfrac_FTable.c) . . . 152
2.36.1 registerreadgfrac (Source File: LIS_gfrac_FTable.c) . . . 152
2.36.2 readgfrac (Source File: LIS_gfrac_FTable.c) . . . 152
2.37 Fortran: Module Interface LIS_laisai_FTable (Source File: LIS_laisai_FTable.c) . . . 152
2.37.1 registerreadlai (Source File: LIS_laisai_FTable.c) . . . 153
2.37.2 readlai (Source File: LIS_laisai_FTable.c) . . . 153
2.37.3 registerreadsai (Source File: LIS_laisai_FTable.c) . . . 153
2.37.4 readsai (Source File: LIS_laisai_FTable.c) . . . 154
2.38 Fortran: Module Interface LIS_landcover_FTable (Source File: LIS_landcover_FTable.c) . . . 154
2.38.1 registerrreadlc (Source File: LIS_landcover_FTable.c) . . . 154
2.38.2 readlandcover (Source File: LIS_landcover_FTable.c) . . . 154
2.38.3 registerrreadmask (Source File: LIS_landcover_FTable.c) . . . 155
2.38.4 readlandmask (Source File: LIS_landcover_FTable.c) . . . 155
2.39 Fortran: Module Interface LIS_lsm_FTable (Source File: LIS_lsm_FTable.c) . . . 155
2.39.1 registerlsmini (Source File: LIS_lsm_FTable.c) . . . 155
2.39.2 lsmini (Source File: LIS_lsm_FTable.c) . . . 156
2.39.3 registerlsmrun (Source File: LIS_lsm_FTable.c) . . . 156
2.39.4 lsmrun (Source File: LIS_lsm_FTable.c) . . . 156
2.39.5 registerlsmfinalize (Source File: LIS_lsm_FTable.c) . . . 156
2.39.6 lsmfinalize (Source File: LIS_lsm_FTable.c) . . . 157
2.39.7 registerlsmsetup (Source File: LIS_lsm_FTable.c) . . . 157
2.39.8 lsmsetup (Source File: LIS_lsm_FTable.c) . . . 157
2.39.9 registerlsmrestart (Source File: LIS_lsm_FTable.c) . . . 157
2.39.10 lsmrestart (Source File: LIS_lsm_FTable.c) . . . 158
2.39.11 registerlsmoutput (Source File: LIS_lsm_FTable.c) . . . 158
2.39.12 lsmoutput (Source File: LIS_lsm_FTable.c) . . . 158
2.39.13 registerlsmf2t (Source File: LIS_lsm_FTable.c) . . . 158
2.39.14 lsmf2t (Source File: LIS_lsm_FTable.c) . . . 159
2.39.15 registerlsmwrst (Source File: LIS_lsm_FTable.c) . . . 159
2.39.16 lsmwrst (Source File: LIS_lsm_FTable.c) . . . 159
2.39.17 registergetstatevar (Source File: LIS_lsm_FTable.c) . . . 160
2.39.18 getlsmstatevar (Source File: LIS_lsm_FTable.c) . . . 160
2.39.19 registersetstatevar (Source File: LIS_lsm_FTable.c) . . . 160

2.39.20 setlsmstatevar (Source File: LIS_lsm_FTable.c) . . . 161
2.39.21 registerobstransform (Source File: LIS_lsm_FTable.c) . . . 161
2.39.22 obstransform (Source File: LIS_lsm_FTable.c) . . . 161
2.39.23 registergetobspred (Source File: LIS_lsm_FTable.c) . . . 162
2.39.24 getobspred (Source File: LIS_lsm_FTable.c) . . . 162
2.39.25 registermapobstolsm (Source File: LIS_lsm_FTable.c) . . . 162
2.39.26 mapobstolsm (Source File: LIS_lsm_FTable.c) . . . 163
2.39.27 registerlsmexport (Source File: LIS_lsm_FTable.c) . . . 163
2.39.28 lsmsetexport (Source File: LIS_lsm_FTable.c) . . . 163
2.39.29 registerlsmdynsetup (Source File: LIS_lsm_FTable.c) . . . 164
2.39.30 lsmdynsetup (Source File: LIS_lsm_FTable.c) . . . 164
2.39.31 registerqclsmstate (Source File: LIS_lsm_FTable.c) . . . 164
2.39.32 qclsmstate (Source File: LIS_lsm_FTable.c) . . . 164
2.39.33 registerscalelsmstatevar (Source File: LIS_lsm_FTable.c) . . . 165
2.39.34 scalelsmstatevar (Source File: LIS_lsm_FTable.c) . . . 165
2.39.35 registerdescalelsmstatevar (Source File: LIS_lsm_FTable.c) . . . 166
2.39.36 descalelsmstatevar (Source File: LIS_lsm_FTable.c) . . . 166
2.39.37 registerupdatelsmstate (Source File: LIS_lsm_FTable.c) . . . 166
2.39.38 updatelsmstate (Source File: LIS_lsm_FTable.c) . . . 167
2.40 Fortran: Module Interface LIS_perturb_FTable (Source File: LIS_perturb_FTable.c) . . . 167
2.40.1 registerperturbinit (Source File: LIS_perturb_FTable.c) . . . 167
2.40.2 perturbinit (Source File: LIS_perturb_FTable.c) . . . 167
2.40.3 registerperturbmethod (Source File: LIS_perturb_FTable.c) . . . 168
2.40.4 perturbmethod (Source File: LIS_perturb_FTable.c) . . . 168
2.41 Fortran: Module Interface LIS_runmode_FTable (Source File: LIS_runmode_FTable.c) . . . 168
2.41.1 registerlisinit (Source File: LIS_runmode_FTable.c) . . . 169
2.41.2 lisinit (Source File: LIS_runmode_FTable.c) . . . 169
2.41.3 registerlisrun (Source File: LIS_runmode_FTable.c) . . . 169
2.41.4 lisrun (Source File: LIS_runmode_FTable.c) . . . 169
2.41.5 registerlisfinalize (Source File: LIS_runmode_FTable.c) . . . 170
2.41.6 lisfinalize (Source File: LIS_runmode_FTable.c) . . . 170
2.42 Fortran: Module Interface LIS_soils_FTable (Source File: LIS_soils_FTable.c) . . . 170
2.42.1 registerreadsand (Source File: LIS_soils_FTable.c) . . . 170
2.42.2 readsand (Source File: LIS_soils_FTable.c) . . . 171
2.42.3 registerreadclay (Source File: LIS_soils_FTable.c) . . . 171
2.42.4 readclay (Source File: LIS_soils_FTable.c) . . . 171
2.42.5 registerreadsilt (Source File: LIS_soils_FTable.c) . . . 172
2.42.6 readsilt (Source File: LIS_soils_FTable.c) . . . 172
2.42.7 registerreadtexture (Source File: LIS_soils_FTable.c) . . . 172
2.42.8 readsoiltexture (Source File: LIS_soils_FTable.c) . . . 173
2.42.9 registerreadporosity (Source File: LIS_soils_FTable.c) . . . 173
2.42.10 readporosity (Source File: LIS_soils_FTable.c) . . . 173
2.42.11 registerreadpsisat (Source File: LIS_soils_FTable.c) . . . 174
2.42.12 readpsisat (Source File: LIS_soils_FTable.c) . . . 174
2.42.13 registerreadksat (Source File: LIS_soils_FTable.c) . . . 174
2.42.14 readksat (Source File: LIS_soils_FTable.c) . . . 175
2.42.15 registerreadbexp (Source File: LIS_soils_FTable.c) . . . 175
2.42.16 readbexp (Source File: LIS_soils_FTable.c) . . . 175
2.42.17 registerreadquartz (Source File: LIS_soils_FTable.c) . . . 176
2.42.18 readquartz (Source File: LIS_soils_FTable.c) . . . 176
2.42.19 registerreadcolor (Source File: LIS_soils_FTable.c) . . . 176

2.42.20 readcolor (Source File: LIS_soils_FTable.c) . . . 177
2.43 Fortran: Module Interface LIS_suppforcing_FTable (Source File: LIS_suppforcing_FTable.c) . 177
2.43.1 registerdefinenativesupp (Source File: LIS_suppforcing_FTable.c) . . . 177
2.43.2 definenativesupp (Source File: LIS_suppforcing_FTable.c) . . . 177
2.43.3 registerreadsupp (Source File: LIS_suppforcing_FTable.c) . . . 178
2.43.4 getsuppforc (Source File: LIS_suppforcing_FTable.c) . . . 178
2.43.5 registersuppti (Source File: LIS_suppforcing_FTable.c) . . . 178
2.43.6 timeinterpsupp (Source File: LIS_suppforcing_FTable.c) . . . 178
2.43.7 registersuppforcingfinal (Source File: LIS_suppforcing_FTable.c) . . . 179
2.43.8 suppforcingfinalize (Source File: LIS_suppforcing_FTable.c) . . . 179
2.44 Fortran: Module Interface LIS_tbot_FTable (Source File: LIS_tbot_FTable.c) . . . 179
2.44.1 registerreadtbot (Source File: LIS_tbot_FTable.c) . . . 179
2.44.2 readtbot (Source File: LIS_tbot_FTable.c) . . . 180
2.45 Fortran: Module Interface LIS_topo_FTable (Source File: LIS_topo_FTable.c) . . . 180
2.45.1 registerreadelev (Source File: LIS_topo_FTable.c) . . . 180
2.45.2 readelev (Source File: LIS_topo_FTable.c) . . . 181
2.45.3 registerreadslope (Source File: LIS_topo_FTable.c) . . . 181
2.45.4 readslope (Source File: LIS_topo_FTable.c) . . . 181
2.45.5 registerreadaspect (Source File: LIS_topo_FTable.c) . . . 182
2.45.6 readaspect (Source File: LIS_topo_FTable.c) . . . 182
2.45.7 registerreadcurv (Source File: LIS_topo_FTable.c) . . . 182
2.45.8 readcurv (Source File: LIS_topo_FTable.c) . . . 183
2.46 Fortran: Module Interface LIS_snowd_FTable (Source File: LIS_snowd_FTable.c) . . . 183
2.46.1 registerreadsnowdepth (Source File: LIS_snowd_FTable.c) . . . 183
2.46.2 readsnowdepth (Source File: LIS_snowd_FTable.c) . . . 183
2.46.3 lis_log_msgC.c (Source File: lis_log_msgC.c) . . . 184
2.46.4 lis_log_msgC (Source File: lis_log_msgC.c) . . . 184
2.46.5 lis_memory_managementC (Source File: lis_memory_managementC.c) . . . 184
2.46.6 lis_calloc (Source File: lis_memory_managementC.c) . . . 184
2.46.7 lis_malloc (Source File: lis_memory_managementC.c) . . . 184
2.46.8 listask_for_point (Source File: listask_for_point.c) . . . 184

IV Plugin Interfaces in LIS 185

3 Plugin interfaces in LIS 186
3.1 Fortran: Module Interface runmode_pluginMod (Source File: runmode_pluginMod.F90) . . . 186
3.1.1 runmode_plugin (Source File: runmode_pluginMod.F90) . . . 186
3.2 Fortran: Module Interface domain_pluginMod (Source File: domain_pluginMod.F90) . . . 187
3.2.1 domain_plugin (Source File: domain_pluginMod.F90) . . . 187
3.3 Fortran: Module Interface baseforcing_pluginMod (Source File: baseforcing_pluginMod.F90) . 187
3.3.1 baseforcing_plugin (Source File: baseforcing_pluginMod.F90) . . . 188
3.4 Fortran: Module Interface suppforcing_pluginMod (Source File: suppforcing_pluginMod.F90) 189
3.4.1 suppforcing_plugin (Source File: suppforcing_pluginMod.F90) . . . 189
3.5 Fortran: Module Interface lsm_pluginMod (Source File: lsm_pluginMod.F90) . . . 190
3.5.1 lsm_plugin (Source File: lsm_pluginMod.F90) . . . 190
3.6 Fortran: Module Interface param_pluginMod (Source File: param_pluginMod.F90) . . . 191
3.6.1 topo_plugin (Source File: param_pluginMod.F90) . . . 191
3.6.2 soils_plugin (Source File: param_pluginMod.F90) . . . 192
3.6.3 laisai_plugin (Source File: param_pluginMod.F90) . . . 193
3.6.4 landcover_plugin (Source File: param_pluginMod.F90) . . . 194

3.6.5 tbot_plugin (Source File: param_pluginMod.F90) . . . 195
3.6.6 gfrac_plugin (Source File: param_pluginMod.F90) . . . 195
3.6.7 alb_plugin (Source File: param_pluginMod.F90) . . . 196
3.6.8 snow_plugin (Source File: param_pluginMod.F90) . . . 196
3.7 Fortran: Module Interface dataassim_pluginMod.F90 (Source File: dataassim_pluginMod.F90) 197
3.7.1 dataassim_plugin (Source File: dataassim_pluginMod.F90) . . . 197
3.8 Fortran: Module Interface dataobs_pluginMod (Source File: dataobs_pluginMod.F90) . . . 198
3.8.1 dataobs_plugin (Source File: dataobs_pluginMod.F90) . . . 198
3.9 Fortran: Module Interface perturb_pluginMod (Source File: perturb_pluginMod.F90) . . . 199
3.9.1 perturb_plugin (Source File: perturb_pluginMod.F90) . . . 199
3.10 Fortran: Module Interface lsmda_pluginMod (Source File: lsmda_pluginMod.F90) . . . 200
3.10.1 lsmda_plugin (Source File: lsmda_pluginMod.F90) . . . 200
3.11 Fortran: Module Interface lsmcpl_pluginMod (Source File: lsmcpl_pluginMod.F90) . . . 201
3.11.1 lsmcpl_plugin (Source File: lsmcpl_pluginMod.F90) . . . 201
3.12 Fortran: Module Interface pluginIndices (Source File: pluginIndices.F90) . . . 202

V Interpolation Tools in LIS 206

4 Interpolation tools in LIS 207
4.0.1 bilinear_interp_input (Source File: bilinear_interp_input.F90) . . . 207
4.0.2 bilinear_interp (Source File: bilinear_interp.F90) . . . 208
4.0.3 conserv_interp_input (Source File: conserv_interp_input.F90) . . . 209
4.0.4 conserv_interp (Source File: conserv_interp.F90) . . . 210
4.0.5 neighbor_interp_input (Source File: neighbor_interp_input.F90) . . . 212
4.0.6 neighbor_interp (Source File: neighbor_interp.F90) . . . 213
4.0.7 compute_earth_coord (Source File: compute_earth_coord.F90) . . . 214
4.0.8 compute_earth_coord_latlon (Source File: compute_earth_coord_latlon.F90) . . . 216
4.0.9 compute_earth_coord_merc (Source File: compute_earth_coord_merc.F90) . . . 216
4.0.10 compute_earth_coord_lambert (Source File: compute_earth_coord_lambert.F90) . . . 217
4.0.11 compute_earth_coord_gauss (Source File: compute_earth_coord_gauss.F90) . . . 218
4.0.12 compute_earth_coord_ease (Source File: compute_earth_coord_ease.F90) . . . 219
4.0.13 compute_earth_coord_polar (Source File: compute_earth_coord_polar.F90) . . . 220
4.0.14 compute_grid_coord (Source File: compute_grid_coord.F90) . . . 221
4.0.15 compute_grid_coord_latlon (Source File: compute_grid_coord_latlon.F90) . . . 222
4.0.16 compute_grid_coord_merc (Source File: compute_grid_coord_merc.F90) . . . 223
4.0.17 compute_grid_coord_lambert (Source File: compute_grid_coord_lambert.F90) . . . 223
4.0.18 compute_grid_coord_gauss (Source File: compute_grid_coord_gauss.F90) . . . 224
4.0.19 compute_grid_coord_ease (Source File: compute_grid_coord_ease.F90) . . . 225
4.0.20 compute_grid_coord_polar (Source File: compute_grid_coord_polar.F90) . . . 226
4.0.21 get_field_pos (Source File: get_fieldpos.F90) . . . 227
4.0.22 polfixs (Source File: polfixs.F90) . . . 228
4.0.23 latlonTopolar (Source File: latlonTopolar.F90) . . . 229
4.0.24 polarToLatLon (Source File: polarToLatLon.F90) . . . 230
4.0.25 lltops (Source File: lltops.F90) . . . 230
4.0.26 pstoll (Source File: pstoll.F90) . . . 231
4.1 Fortran: Module Interface map_utils (Source File: map_utils.F90) . . . 232
4.1.1 Supported Projections . . . 232
4.1.2 Remarks . . . 232
4.1.3 Assumptions . . . 233
4.1.4 Data Definitions: . . . 233

4.1.5 Usage . . . 233
4.1.6 map_init (Source File: map_utils.F90) . . . 235
4.1.7 map_set (Source File: map_utils.F90) . . . 235
4.1.8 latlon_to_ij (Source File: map_utils.F90) . . . 236
4.1.9 ij_to_latlon (Source File: map_utils.F90) . . . 237
4.1.10 compute_stnwts (Source File: compute_stnwts.F90) . . . 237
4.1.11 interp_stndata (Source File: interp_stndata.F90) . . . 238
4.1.12 normalize_stnwts (Source File: normalize_stnwts.F90) . . . 239
4.1.13 ezlh_convert (Source File: ezlh_convert.F90) . . . 239
4.1.14 ezlh_inverse (Source File: ezlh_inverse.F90) . . . 240

VI User-defined extensible components in LIS 242

5 User-defined extensible components in LIS 243

VII Running Modes in LIS 244

6 Running Modes in LIS 245

7 Retrospective Running Mode 245
7.1 Fortran: Module Interface retrospective_runMod (Source File: retrospective_runMod.F90) . . 245
7.1.1 lis_init_retrospective (Source File: retrospective_runMod.F90) . . . 245
7.1.2 lis_run_retrospective (Source File: retrospective_runMod.F90) . . . 246
7.1.3 lis_final_retrospective (Source File: retrospective_runMod.F90) . . . 247

VIII Domain implementations in LIS 249

8 Domain implementations in LIS 250

9 Lat/Lon domain 250
9.0.4 createtiles_latlon (Source File: createtiles_latlon.F90) . . . 250
9.0.5 calculate_domveg (Source File: calculate_domveg.F90) . . . 251
9.0.6 create_vegtilespace_latlon (Source File: create_vegtilespace_latlon.F90) . . . 251
9.0.7 readinput_latlon (Source File: readinput_latlon.F90) . . . 252

10 Mercator domain 253
10.0.8 createtiles_merc (Source File: createtiles_merc.F90) . . . 253
10.0.9 create_vegtilespace_merc (Source File: create_vegtilespace_merc.F90) . . . 254
10.0.10 readinput_merc (Source File: readinput_merc.F90) . . . 254

11 Lambert Conformal domain 255
11.0.11 createtiles_lambert (Source File: createtiles_lambert.F90) . . . 255
11.0.12 create_vegtilespace_lambert (Source File: create_vegtilespace_lambert.F90) . . . 256
11.0.13 readinput_lambert (Source File: readinput_lambert.F90) . . . 257

12 Polar stereographic domain 257
12.0.14 createtiles_polar (Source File: createtiles_polar.F90) . . . 257
12.0.15 create_vegtilespace_polar (Source File: create_vegtilespace_polar.F90) . . . 258
12.0.16 readinput_polar (Source File: readinput_polar.F90) . . . 259

13 GSWP domain **259**
13.0.17 createtiles_gswp (Source File: createtiles_gswp.F90) . . . 259
13.0.18 readinput_gswp (Source File: readinput_gswp.F90) . . . 260

IX Land Surface Parameters in LIS **261**

14 Land Surface Parameters in LIS **262**

15 Landmask/Landcover Data **262**
15.0.19 read_latlon_umdmask (Source File: read_latlon_umdmask.F90) . . . 262
15.0.20 read_latlon_umdlc (Source File: read_latlon_umdlc.F90) . . . 262
15.0.21 read_latlon_usgsmask (Source File: read_latlon_usgsmask.F90) . . . 263
15.0.22 read_latlon_usgslc (Source File: read_latlon_usgslc.F90) . . . 264
15.0.23 read_latlon_igbplc (Source File: read_latlon_igbplc.F90) . . . 264
15.0.24 read_lambert_umdmask (Source File: read_lambert_umdmask.F90) . . . 265
15.0.25 read_lambert_umdlc (Source File: read_lambert_umdlc.F90) . . . 265
15.0.26 read_lambert_usgsmask (Source File: read_lambert_usgsmask.F90) . . . 266
15.0.27 read_lambert_usgslc (Source File: read_lambert_usgslc.F90) . . . 267
15.0.28 read_merc_umdmask (Source File: read_merc_umdmask.F90) . . . 268
15.0.29 read_merc_umdlc (Source File: read_merc_umdlc.F90) . . . 268
15.0.30 read_merc_usgsmask (Source File: read_merc_usgsmask.F90) . . . 269
15.0.31 read_merc_usgslc (Source File: read_merc_usgslc.F90) . . . 270
15.0.32 read_polar_umdmask (Source File: read_polar_umdmask.F90) . . . 270
15.0.33 read_polar_umdlc (Source File: read_polar_umdlc.F90) . . . 271
15.0.34 read_polar_usgsmask (Source File: read_polar_usgsmask.F90) . . . 272
15.0.35 read_polar_usgslc (Source File: read_polar_usgslc.F90) . . . 272
15.0.36 read_gaussian_gfs_lc (Source File: read_gaussian_gfs_lc.F90) . . . 273
15.0.37 read_gaussian_gfs_mask (Source File: read_gaussian_gfs_mask.F90) . . . 274

16 Albedo Data **274**
16.0.38 read_gaussian_albedo (Source File: read_gaussian_albedo.F90) . . . 274
16.0.39 read_gaussian_mxsnoalb (Source File: read_gaussian_mxsnoalb.F90) . . . 275
16.0.40 read_gswpalbedo (Source File: read_gswpalbedo.F90) . . . 275
16.0.41 read_gswpmxsnoalb (Source File: read_gswpmxsnoalb.F90) . . . 276
16.0.42 read_lcalbedo (Source File: read_lcalbedo.F90) . . . 277
16.0.43 read_lcmxsnoalb (Source File: read_lcmxsnoalb.F90) . . . 277
16.0.44 read_llalbedo (Source File: read_llalbedo.F90) . . . 278
16.0.45 read_llmxsnoalb (Source File: read_llmxsnoalb.F90) . . . 279
16.0.46 read_mercalbedo (Source File: read_mercalbedo.F90) . . . 279
16.0.47 read_mercmxsnoalb (Source File: read_mercmxsnoalb.F90) . . . 280
16.0.48 read_psalbedo (Source File: read_psalbedo.F90) . . . 280
16.0.49 read_psmxsnoalb (Source File: read_psmxsnoalb.F90) . . . 281

17 Greenness Fraction Data **282**
17.0.50 read_gswpgfrac (Source File: read_gswpgfrac.F90) . . . 282
17.0.51 read_lcgfrac (Source File: read_lcgfrac.F90) . . . 282
17.0.52 read_llgfrac (Source File: read_llgfrac.F90) . . . 283
17.0.53 read_mercgfrac (Source File: read_mercgfrac.F90) . . . 284
17.0.54 read_psgfrac (Source File: read_psgfrac.F90) . . . 285
17.0.55 read_catchgfrac (Source File: read_catchgfrac.F90) . . . 285
17.0.56 read_gaussian_frac (Source File: read_gaussian_gfrac.F90) . . . 286

18 Leaf Area Index Data **286**
18.0.57 read_catchlai (Source File: read_catchlai.F90) . . . 287
18.0.58 read_gswplai (Source File: read_gswplai.F90) . . . 287
18.0.59 read_gswpsai (Source File: read_gswpsai.F90) . . . 288
18.0.60 read_lambert_avhrrlai (Source File: read_lambert_avhrrlai.F90) . . . 288
18.0.61 read_lambert_avhrrsai (Source File: read_lambert_avhrrsai.F90) . . . 289
18.0.62 read_lambert_modislai (Source File: read_lambert_modislai.F90) . . . 290
18.0.63 read_lambert_modissai (Source File: read_lambert_modissai.F90) . . . 290
18.0.64 read_ll_avhrrlai (Source File: read_ll_avhrrlai.F90) . . . 291
18.0.65 read_ll_avhrrsai (Source File: read_ll_avhrrsai.F90) . . . 292
18.0.66 read_ll_modislai (Source File: read_ll_modislai.F90) . . . 293
18.0.67 read_ll_modissai (Source File: read_ll_modissai.F90) . . . 293
18.0.68 read_merc_avhrrlai (Source File: read_merc_avhrrlai.F90) . . . 294
18.0.69 read_merc_avhrrsai (Source File: read_merc_avhrrsai.F90) . . . 295
18.0.70 read_merc_modislai (Source File: read_merc_modislai.F90) . . . 295
18.0.71 read_merc_modissai (Source File: read_merc_modissai.F90) . . . 296
18.0.72 read_ps_avhrrlai (Source File: read_ps_avhrrlai.F90) . . . 297
18.0.73 read_ps_avhrrsai (Source File: read_ps_avhrrsai.F90) . . . 297
18.0.74 read_ps_modislai (Source File: read_ps_modislai.F90) . . . 298
18.0.75 read_ps_modissai (Source File: read_ps_modissai.F90) . . . 299

19 Soils **300**
19.0.76 read_gswp_bexp (Source File: read_gswp_bexp.F90) . . . 300
19.0.77 read_gswp_ksat (Source File: read_gswp_ksat.F90) . . . 300
19.0.78 read_gswp_psisat (Source File: read_gswp_psisat.F90) . . . 301
19.0.79 read_gswp_soilclass (Source File: read_gswp_soilclass.F90) . . . 301
19.0.80 read_gswpclay (Source File: read_gswpclay.F90) . . . 302
19.0.81 read_gswpsand (Source File: read_gswpsand.F90) . . . 303
19.0.82 read_gswpsilt (Source File: read_gswpsilt.F90) . . . 303
19.0.83 read_lambert_faoclay (Source File: read_lambert_faoclay.F90) . . . 304
19.0.84 read_lambert_faocolor (Source File: read_lambert_faocolor.F90) . . . 305
19.0.85 read_lambert_faosand (Source File: read_lambert_faosand.F90) . . . 305
19.0.86 read_lambert_faosilt (Source File: read_lambert_faosilt.F90) . . . 306
19.0.87 read_lambert_statsgoclay (Source File: read_lambert_statsgoclay.F90) . . . 307
19.0.88 read_lambert_statsgosand (Source File: read_lambert_statsgosand.F90) . . . 307
19.0.89 read_lambert_statsgosilt (Source File: read_lambert_statsgosilt.F90) . . . 308
19.0.90 read_lambert_statsgotexture (Source File: read_lambert_statsgotexture.F90) . . . 309
19.0.91 read_faoclay (Source File: read_faoclay.F90) . . . 309
19.0.92 read_faocolor (Source File: read_faocolor.F90) . . . 310
19.0.93 read_faosand (Source File: read_faosand.F90) . . . 310
19.0.94 read_faosilt (Source File: read_faosilt.F90) . . . 311
19.0.95 read_statsgoclay (Source File: read_statsgoclay.F90) . . . 312
19.0.96 read_statsgocolor (Source File: read_statsgocolor.F90) . . . 312
19.0.97 read_statsgososand (Source File: read_statsgosand.F90) . . . 313
19.0.98 read_statsgosilt (Source File: read_statsgosilt.F90) . . . 313
19.0.99 read_statsgotexture (Source File: read_statsgotexture.F90) . . . 314
19.0.100 read_merc_faoclay (Source File: read_merc_faoclay.F90) . . . 315
19.0.101 read_merc_faocolor (Source File: read_merc_faocolor.F90) . . . 315
19.0.102 read_merc_faosand (Source File: read_merc_faosand.F90) . . . 316
19.0.103 read_merc_faosilt (Source File: read_merc_faosilt.F90) . . . 317
19.0.104 read_merc_statsgoclay (Source File: read_merc_statsgoclay.F90) . . . 317

19.0.105 read_merc_statsgosand (Source File: read_merc_statsgosand.F90) 318
19.0.106 read_merc_statsgosilt (Source File: read_merc_statsgosilt.F90) 319
19.0.107 read_merc_statsgotexture (Source File: read_merc_statsgotexture.F90) 319
19.0.108 read_polar_faoclay (Source File: read_polar_faoclay.F90) 320
19.0.109 read_polar_faocolor (Source File: read_polar_faocolor.F90) 321
19.0.110 read_polar_faosand (Source File: read_polar_faosand.F90) 321
19.0.111 read_polar_faosilt (Source File: read_polar_faosilt.F90) 322
19.0.112 read_polar_statsgoclay (Source File: read_polar_statsgoclay.F90) 323
19.0.113 read_polar_statsgosand (Source File: read_polar_statsgosand.F90) 323
19.0.114 read_polar_statsgosilt (Source File: read_polar_statsgosilt.F90) 324
19.0.115 read_polar_statsgotexture (Source File: read_polar_statsgotexture.F90) 325
19.0.116 read_gfs_texture (Source File: read_gfs_texture.F90) 325

20 Snow **326**
20.0.117 read_llsnowdepth (Source File: read_llsnowdepth.F90) 326
20.0.118 read_lcsnowdepth (Source File: read_lcsnowdepth.F90) 326
20.0.119 read_mercsnowdepth (Source File: read_mercsnowdepth.F90) 327
20.0.120 read_polarsnowdepth (Source File: read_polarsnowdepth.F90) 327

21 Topography **328**
21.0.121 read_lambert_elev_gtopo30 (Source File: read_lambert_elev_gtopo30.F90) 328
21.0.122 read_elev_gtopo30 (Source File: read_elev_gtopo30.F90) 328
21.0.123 read_slope_gtopo30 (Source File: read_slope_gtopo30.F90) 329
21.0.124 read_aspect_gtopo30 (Source File: read_aspect_gtopo30.F90) 330
21.0.125 read_curv_gtopo30 (Source File: read_curv_gtopo30.F90) 330
21.0.126 read_merc_elev_gtopo30 (Source File: read_merc_elev_gtopo30.F90) 331
21.0.127 read_polar_elev_gtopo30 (Source File: read_polar_elev_gtopo30.F90) 331
21.0.128 read_elev_gaussian (Source File: read_elev_gaussian.F90) 332

22 Bottom Temperature **332**
22.0.129 read_gswptbot (Source File: read_gswptbot.F90) 333
22.0.130 read_lambert_statictbot (Source File: read_lambert_statictbot.F90) 333
22.0.131 read_statictbot (Source File: read_statictbot.F90) 334
22.0.132 read_merc_statictbot (Source File: read_merc_statictbot.F90) 334
22.0.133 read_polar_statictbot (Source File: read_polar_statictbot.F90) 335
22.0.134 read_gaussian_tbot (Source File: read_gaussian_tbot.F90) 336

X Base forcing analyses in LIS **337**

23 Overview **338**

24 ECMWF Reanalysis **338**
24.1 Fortran: Module Interface ecmwfreanaldomain_module (Source File: ecmwfreanaldomain_module.F90) 338
24.1.1 defineNativeECMWFREANAL (Source File: ecmwfreanaldomain_module.F90) 339
24.1.2 readecmwfreanalcrd (Source File: readecmwfreanalcrd.F90) 340
24.1.3 read_ecmwfreanal_elev (Source File: read_ecmwfreanal_elev.F90) 340
24.1.4 readecmwfreanalmask (Source File: readecmwfreanalmask.F90) 341
24.1.5 getecmwfreanal (Source File: getecmwfreanal.F90) 341
24.1.6 retecmwfreanal (Source File: retecmwfreanal.F90) 342
24.1.7 ecmwfreanalgrid_2_lisgrid (Source File: retecmwfreanal.F90) 343
24.1.8 fillgaps (Source File: retecmwfreanal.F90) 344

24.1.9 time_interp_ecmwfreanal (Source File: time_interp_ecmwfreanal.F90) 344
24.1.10 geogfill2 (Source File: geogfill2.F90) 345
24.1.11 ecmwfreanalforcing_finalize (Source File: ecmwfreanalforcing_finalize.F90) 346

25 ECMWF **346**
25.1 Fortran: Module Interface ecmwfdomain_module (Source File: ecmwfdomain_module.F90) . . 346
25.1.1 defineNativeECMWF (Source File: ecmwfdomain_module.F90) 348
25.1.2 readecmwfcrd (Source File: readecmwfcrd.F90) 348
25.1.3 read_ecmwf_elev (Source File: read_ecmwf_elev.F90) 349
25.1.4 getecmwf (Source File: getecmwf.F90) 349
25.1.5 retecmwf (Source File: retecmwf.F90) 350
25.1.6 ret_inst3 (Source File: retecmwf.F90) 351
25.1.7 ret_accum (Source File: retecmwf.F90) 352
25.1.8 interp_iv (Source File: retecmwf.F90) 353
25.1.9 time_interp_ecmwf (Source File: time_interp_ecmwf.F90) 354
25.1.10 ecmwfforcing_finalize (Source File: ecmwfforcing_finalize.F90) 355

26 GDAS **355**
26.1 Fortran: Module Interface gdasdomain_module (Source File: gdasdomain_module.F90) 356
26.1.1 defineNativeGDAS (Source File: gdasdomain_module.F90) 357
26.1.2 readgdascrd (Source File: readgdascrd.F90) 358
26.1.3 read_gdas_elev (Source File: read_gdas_elev.F90) 358
26.1.4 getgdas (Source File: getgdas.F90) 359
26.1.5 gdasfile (Source File: getgdas.F90) 360
26.1.6 gdasfilef06 (Source File: getgdas.F90) 360
26.1.7 retgdas (Source File: retgdas.F90) 361
26.1.8 interp_gdas (Source File: retgdas.F90) 362
26.1.9 time_interp_gdas (Source File: time_interp_gdas.F90) 363
26.2 Fortran: Module Interface gdasforcing_finalize (Source File: gdasforcing_finalize.F90) 364

27 GEOS **364**
27.1 Fortran: Module Interface geosdomain_module (Source File: geosdomain_module.F90) 364
27.1.1 defineNativeGEOS (Source File: geosdomain_module.F90) 366
27.1.2 readgeoscrd (Source File: readgeoscrd.F90) 366
27.1.3 getgeos (Source File: getgeos.F90) 367
27.1.4 geosfile (Source File: getgeos.F90) 368
27.1.5 readgeos (Source File: readgeos.F90) 368
27.1.6 time_interp_geos (Source File: time_interp_geos.F90) 369
27.1.7 geosforcing_finalize (Source File: geosforcing_finalize.F90) 370

28 GSWP **371**
28.1 Fortran: Module Interface gswpdomain_module (Source File: gswpdomain_module.F90) . . . 371
28.1.1 defineNativeGSWP (Source File: gswpdomain_module.F90) 372
28.1.2 readgswpcrd (Source File: readgswpcrd.F90) 372
28.1.3 getgswp (Source File: getgswp.F90) 373
28.1.4 readgswp (Source File: readgswp.F90) 373
28.1.5 gswp_process_forcing (Source File: readgswp.F90) 374
28.1.6 interp_gswp (Source File: readgswp.F90) 375
28.1.7 gswp_expand_to_grid (Source File: readgswp.F90) 376
28.1.8 gswp_2d_to_1d (Source File: readgswp.F90) 376
28.1.9 gswp_mask (Source File: readgswp.F90) 377
28.1.10 time_interp_gswp (Source File: time_interp_gswp.F90) 377

28.1.11 finterp (Source File: finterp.F90) . . . 378
28.1.12 gswp_fillgaps (Source File: gswp_fillgaps.F90) . . . 379
28.2 Fortran: Module Interface gswpforcing_finalize (Source File: gswpforcing_finalize.F90) . . . 380

29 GSWP1 **380**
29.1 Fortran: Module Interface gswp1domain_module (Source File: gswp1domain_module.F90) . . . 380
29.1.1 defineNativeGSWP1 (Source File: gswp1domain_module.F90) . . . 381
29.1.2 readgswp1crd (Source File: readgswp1crd.F90) . . . 381
29.1.3 getgswp1 (Source File: getgswp1.F90) . . . 382
29.1.4 gswp1file (Source File: getgswp1.F90) . . . 382
29.1.5 readgswp1 (Source File: readgswp1.F90) . . . 383
29.2 Core Functions of readgswp1 . . . 383
29.2.1 check_error_gswp1 (Source File: readgswp1.F90) . . . 383
29.2.2 time_interp_gswp1 (Source File: time_interp_gswp1.F90) . . . 384
29.3 Fortran: Module Interface gswp1forcing_finalize (Source File: gswp1forcing_finalize.F90) . . . 384

30 NLDAS **385**
30.1 Fortran: Module Interface nldasdomain_module (Source File: nldasdomain_module.F90) . . . 385
30.1.1 defineNativeNLDAS (Source File: nldasdomain_module.F90) . . . 386
30.1.2 readnldascrd (Source File: readnldascrd.F90) . . . 387
30.1.3 read_edas_elev (Source File: read_edas_elev.F90) . . . 388
30.1.4 read_edas_elev_latlon (Source File: read_edas_elev_latlon.F90) . . . 388
30.1.5 orig_elevdiff (Source File: orig_elevdiff.F90) . . . 389
30.1.6 nldas_ec_removal (Source File: nldas_ec_removal.F90) . . . 390
30.1.7 getnldas (Source File: getnldas.F90) . . . 390
30.1.8 nldasfile (Source File: getnldas.F90) . . . 391
30.1.9 retnldas (Source File: retnldas.F90) . . . 392
30.1.10 interp_nldas (Source File: interp_nldas.F90) . . . 393
30.1.11 time_interp_nldas (Source File: time_interp_nldas.F90) . . . 394
30.2 Fortran: Module Interface nldasforcing_finalize (Source File: nldasforcing_finalize.F90) . . . 395

31 NLDAS2 **395**
31.1 Fortran: Module Interface nldas2domain_module (Source File: nldas2domain_module.F90) . . . 395
31.1.1 defineNativeNLDAS2 (Source File: nldas2domain_module.F90) . . . 397
31.1.2 readnldas2crd (Source File: readnldas2crd.F90) . . . 397
31.1.3 read_nldas2_elev (Source File: read_nldas2_elev.F90) . . . 398
31.1.4 getnldas2 (Source File: getnldas2.F90) . . . 398
31.1.5 nldas2filea (Source File: getnldas2.F90) . . . 399
31.1.6 nldas2fileb (Source File: getnldas2.F90) . . . 400
31.1.7 retnldas2 (Source File: retnldas2.F90) . . . 401
31.1.8 interp_nldas2 (Source File: retnldas2.F90) . . . 402
31.1.9 time_interp_nldas2 (Source File: time_interp_nldas2.F90) . . . 403
31.2 Fortran: Module Interface nldas2forcing_finalize (Source File: nldas2forcing_finalize.F90) . . . 404

32 Princeton **404**
32.1 Fortran: Module Interface princetondomain_module (Source File: princetondomain_module.F90) 404
32.1.1 defineNativePRINCETON (Source File: princetondomain_module.F90) . . . 405
32.1.2 readprincetoncrd (Source File: readprincetoncrd.F90) . . . 406
32.1.3 getprinceton (Source File: getprinceton.F90) . . . 406
32.1.4 retprinceton (Source File: retprinceton.F90) . . . 407
32.1.5 princetongrid_2_lisgrid (Source File: retprinceton.F90) . . . 408
32.1.6 time_interp_princeton (Source File: time_interp_princeton.F90) . . . 409

32.1.7 read_princeton_elev (Source File: read_princeton_elev.F90) 410
32.1.8 read_princeton_elev_latlon (Source File: read_princeton_elev_latlon.F90) 410
32.1.9 princetonforcing_finalize (Source File: princetonforcing_finalize.F90) 411

33 Rhone-AGG **411**
33.1 Fortran: Module Interface rhonedomain_module (Source File: rhonedomain_module.F90) . . . 411
33.1.1 defineNativeRHONE (Source File: rhonedomain_module.F90) 412
33.1.2 readrhonecrd (Source File: readrhonecrd.F90) 412
33.1.3 getrhone (Source File: getrhone.F90) 412
33.1.4 rhonefile (Source File: getrhone.F90) 413
33.1.5 readrhone (Source File: readrhone.F90) 414
33.2 Core Functions of readrhone 414
33.2.1 check_error_rhone (Source File: readrhone.F90) 414
33.2.2 time_interp_rhone (Source File: time_interp_rhone.F90) 414
33.3 Core Functions of time_interp_rhone 415
33.3.1 rhoneforcing_finalize (Source File: rhoneforcing_finalize.F90) 415

34 Template **415**
34.1 Fortran: Module Interface templatedomain_module (Source File: templatedomain_module.F90) 416
34.1.1 defineNativetemplate (Source File: templatedomain_module.F90) 416
34.1.2 time_interp_template (Source File: time_interp_template.F90) 417

XI Supplemental forcing analyses in LIS **419**

35 Overview **420**

36 CMAP **421**
36.1 Fortran: Module Interface cmapdomain_module (Source File: cmapdomain_module.F90) . . . 421
36.1.1 defineNativeCMAP (Source File: cmapdomain_module.F90) 422
36.1.2 readcmapcrd (Source File: readcmapcrd.F90) 422
36.1.3 getcmap (Source File: getcmap.F90) 423
36.1.4 cmapfile (Source File: getcmap.F90) 423
36.1.5 cmapfile_old (Source File: getcmap.F90) 424
36.1.6 glbprecip_cmap (Source File: glbprecip_cmap.F90) 425
36.1.7 interp_cmap (Source File: interp_cmap.F90) 426
36.1.8 time_interp_cmap (Source File: time_interp_cmap.F90) 427

37 TRMM 3B42RT **427**
37.1 Fortran: Module Interface TRMM3B42RTdomain_module (Source File: TRMM3B42RTdomain_module.F90) 427
37.1.1 defineNativeTRMM3B42RT (Source File: TRMM3B42RTdomain_module.F90) 428
37.1.2 readTRMM3B42RTcrd (Source File: readTRMM3B42RTcrd.F90) 429
37.1.3 getTRMM3B42RT (Source File: getTRMM3B42RT.F90) 429
37.1.4 TRMM3B42RTfile (Source File: getTRMM3B42RT.F90) 430
37.1.5 glbprecip_TRMM3B42RT (Source File: glbprecip_TRMM3B42RT.F90) 431
37.1.6 interp_TRMM3B42RT (Source File: interp_TRMM3B42RT.F90) 431
37.1.7 time_interp_TRMM3B42RT (Source File: time_interp_3B42RT.F90) 432

38 TRMM 3B42V6 **433**
38.0.8 defineNativeTRMM3B42V6 (Source File: TRMM3B42V6domain_module.F90) 433
38.0.9 readTRMM3B42V6crd (Source File: readTRMM3B42V6crd.F90) 433
38.0.10 getTRMM3B42V6 (Source File: getTRMM3B42V6.F90) 434
38.0.11 TRMM3B42V6file (Source File: getTRMM3B42V6.F90) 437
38.0.12 glbprecip_TRMM3B42V6.F90 (Source File: glbprecip_TRMM3B42V6.F90) 437
38.0.13 interp_TRMM3B42V6.F90 (Source File: interp_TRMM3B42V6.F90) 440

39 CMORPH **441**
39.1 Fortran: Module Interface cmordomain_module (Source File: cmordomain_module.F90) . . . 441
39.1.1 defineNativeCMORPH (Source File: cmordomain_module.F90) 442
39.1.2 readcmorcrd (Source File: readcmorcrd.F90) 442
39.1.3 getcmor (Source File: getcmor.F90) 443
39.1.4 cmorfile (Source File: getcmor.F90) 444
39.1.5 glbprecip_cmor (Source File: glbprecip_cmor.F90) 444
39.1.6 interp_cmor (Source File: interp_cmor.F90) 445
39.1.7 time_interp_cmor (Source File: time_interp_cmor.F90) 446

40 STAGEII **446**
40.1 Fortran: Module Interface stg2domain_module (Source File: stg2domain_module.F90) 446
40.1.1 defineNativeSTG2 (Source File: stg2domain_module.F90) 447
40.1.2 readstg2crd (Source File: readstg2crd.F90) 448
40.1.3 getstg2 (Source File: getstg2.F90) 448
40.2 Fortran: Module Interface stg2_finalize (Source File: stg2_finalize.F90) 449
40.2.1 interp_stg2 (Source File: interp_stg2.F90) 450
40.2.2 time_interp_stg2 (Source File: time_interp_stg2.F90) 451
40.2.3 precip_stg2 (Source File: precip_stg2.F90) 451
40.2.4 stg2file (Source File: stg2file.F90) 452

41 STAGEIV **452**
41.1 Fortran: Module Interface stg4domain_module (Source File: stg4domain_module.F90) 453
41.1.1 defineNativeSTG4 (Source File: stg4domain_module.F90) 454
41.1.2 readstg4crd (Source File: readstg4crd.F90) 454
41.1.3 getstg4 (Source File: getstg4.F90) 455
41.1.4 precip_stg4 (Source File: precip_stg4.F90) 456
41.1.5 interp_stg4 (Source File: interp_stg4.F90) 456
41.1.6 time_interp_stg4 (Source File: time_interp_stg4.F90) 457
41.1.7 stg4file (Source File: stg4file.F90) 458
41.2 Fortran: Module Interface stg4_finalize (Source File: stg4_finalize.F90) 458

42 AGRMET radiation **459**
42.1 Fortran: Module Interface agrrad_module (Source File: agrrad_module.F90) 459
42.1.1 defineNativeAGRRAD (Source File: agrrad_module.F90) 460
42.1.2 readagrradcrd (Source File: readagrradcrd.F90) 460
42.1.3 getagrrad (Source File: getagrrad.F90) 461
42.1.4 readagrrad (Source File: readagrrad.F90) 462
42.1.5 agrrad_filename (Source File: readagrrad.F90) 463
42.1.6 agrrad_fillgaps (Source File: readagrrad.F90) 463
42.1.7 time_interp_agrrad (Source File: time_interp_agrrad.F90) 464
42.1.8 agrrad_finalize (Source File: agrrad_finalize.F90) 464

43 CEOP forcing **465**
43.1 Fortran: Module Interface ceopdomain_module (Source File: ceopdomain_module.F90) 465
43.1.1 defineNativeCEOP (Source File: ceopdomain_module.F90) 465
43.1.2 readceopcrd (Source File: readceopcrd.F90) 466
43.1.3 getceop (Source File: getceop.F90) 466
43.1.4 readceop (Source File: readceop.F90) 467
43.1.5 readbon (Source File: readbon.F90) 468
43.1.6 readfpk (Source File: readfpk.F90) 469
43.1.7 readsgp (Source File: readsgp.F90) 470
43.1.8 time_interp_ceop (Source File: time_interp_ceop.F90) 471

44 SCAN forcing **472**
44.1 Fortran: Module Interface scandomain_module (Source File: scandomain_module.F90) 472
44.1.1 defineNativeSCAN (Source File: scandomain_module.F90) 472
44.1.2 readscancrd (Source File: readscancrd.F90) 473
44.1.3 getscan (Source File: getscan.F90) 473
44.1.4 readscan (Source File: readscan.F90) 474
44.1.5 time_interp_scan (Source File: time_interp_scan.F90) 475

XII Data Assimilation in LIS **476**

A Data Assimilation Algorithms in LIS **477**

B Direct Insertion **477**
B.1 Fortran: Module Interface directInsertion_module (Source File: directInsertion_module.F90) . 477
B.1.1 di_init (Source File: directInsertion_module.F90) 477
B.1.2 di_increments (Source File: directInsertion_module.F90) 477
B.1.3 di_update (Source File: directInsertion_module.F90) 478
B.1.4 di_diagnostics (Source File: directInsertion_module.F90) 479
B.1.5 di_final (Source File: directInsertion_module.F90) 479

C GMAO Ensemble Kalman Filter **479**
C.1 Fortran: Module Interface gmaoenkf_module (Source File: gmaoenkf_module.F90) 479
C.1.1 gmaoenkf_init (Source File: gmaoenkf_module.F90) 480
C.1.2 gmaoenkf_increments (Source File: gmaoenkf_module.F90) 480
C.1.3 gmaoenkf_update (Source File: gmaoenkf_module.F90) 482
C.1.4 gmaoenkf_diagnostics (Source File: gmaoenkf_module.F90) 482
C.1.5 pruneVarname (Source File: gmaoenkf_module.F90) 483
C.1.6 assemble_obs_cov (Source File: gmaoenkf_module.F90) 483
C.1.7 gmaoenkf_final (Source File: gmaoenkf_module.F90) 484
C.1.8 getLSMvarnames (Source File: gmaoenkf_module.F90) 484
C.1.9 getObsPert (Source File: gmaoenkf_module.F90) 484
C.1.10 generateObservations (Source File: gmaoenkf_module.F90) 485
C.1.11 generateObsparam (Source File: gmaoenkf_module.F90) 485
C.1.12 getLSMData (Source File: gmaoenkf_module.F90) 486
C.2 Fortran: Module Interface enkf_general (Source File: enkf_general.F90) 486
C.2.1 enkf_increments (Source File: enkf_general.F90) 486
C.2.2 hadamard_for_repr_matrix (Source File: enkf_general.F90) 487
C.2.3 gaspari_cohn (Source File: enkf_general.F90) 488
C.2.4 get_gaspari_cohn (Source File: enkf_general.F90) 488
C.3 Fortran: Module Interface enkf_types (Source File: enkf_types.F90) 488

C.4 Fortran: Module Interface my_matrix_functions (Source File: my_matrix_functions.F90) . . . 488
C.4.1 (Source File: my_matrix_functions.F90) . . . 489
C.4.2 row_variance (Source File: my_matrix_functions.F90) . . . 489
C.4.3 row_third_moment (Source File: my_matrix_functions.F90) . . . 489
C.4.4 five_number_summary (Source File: my_matrix_functions.F90) . . . 489
C.4.5 adjust_mean (Source File: my_matrix_functions.F90) . . . 490
C.4.6 adjust_std (Source File: my_matrix_functions.F90) . . . 490
C.4.7 matrix_std (Source File: my_matrix_functions.F90) . . . 491
C.4.8 ludcmp (Source File: my_lu_decomp.F90) . . . 491
C.4.9 lubksb (Source File: my_lu_decomp.F90) . . . 491
C.4.10 nr_sort (Source File: nr_sort.F90) . . . 492

D Perturbation Algorithms in LIS 492

E Static Perturbation Algorithm 492
E.1 Fortran: Module Interface staticPerturbMod (Source File: staticPerturbMod.F90) . . . 492
E.1.1 staticPerturb_init (Source File: staticPerturbMod.F90) . . . 492
E.1.2 staticPerturburb (Source File: staticPerturbMod.F90) . . . 493

F GMAO Perturbation Algorithm 493
F.1 Fortran: Module Interface gmaopert_Mod (Source File: gmaopert_Mod.F90) . . . 493
F.1.1 gmaopert_init (Source File: gmaopert_Mod.F90) . . . 493
F.1.2 gmaoperturb (Source File: gmaopert_Mod.F90) . . . 494
F.2 Fortran: Module Interface landpert_module.F90 (Source File: landpert_module.F90) . . . 494
F.2.1 allocate_pert_param (Source File: landpert_module.F90) . . . 494
F.3 Fortran: Module Interface landpert_routines.F90 (Source File: landpert_routines.F90) . . . 495
F.3.1 assemble_pert_param (Source File: landpert_routines.F90) . . . 495
F.3.2 get_sqrt_corr_matrix (Source File: landpert_routines.F90) . . . 495
F.3.3 get_init_Pert_rseed (Source File: landpert_routines.F90) . . . 496
F.3.4 get_pert (Source File: landpert_routines.F90) . . . 496
F.3.5 propagate_pert (Source File: landpert_routines.F90) . . . 498
F.3.6 truncate_std_normal (Source File: landpert_routines.F90) . . . 499
F.3.7 adjust_mean (Source File: landpert_routines.F90) . . . 499
F.4 Fortran: Module Interface nr_jacobi.F90 (Source File: nr_jacobi.F90) . . . 500
F.4.1 jacobi (Source File: nr_jacobi.F90) . . . 500
F.5 Fortran: Module Interface nr_ran2_gasdev.F90 (Source File: nr_ran2_gasdev.F90) . . . 500
F.6 Fortran: Module Interface random_fields.F90 (Source File: random_fields.F90) . . . 501
F.6.1 sqrt_gauss_spectrum_2d (Source File: random_fields.F90) . . . 501
F.6.2 rfg2d_fft (Source File: random_fields.F90) . . . 501
F.6.3 get_fft_grid (Source File: random_fields.F90) . . . 502
F.6.4 generate_white_field (Source File: random_fields.F90) . . . 503

XIII Observation data handling for assimilation in LIS 504

G Observation data handling 505

H Synthetic Soil Moisture from Noah 505
H.1 Fortran: Module Interface syntheticsm1obs_module (Source File: syntheticsm1obs_module.F90) 505
H.1.1 syntheticsm1obs_setup (Source File: syntheticsm1obs_module.F90) . . . 505
H.1.2 read_syntheticsm1obs (Source File: read_syntheticsm1obs.F90) . . . 506
H.1.3 getNSO_synsm1.F90 (Source File: getNSO_synsm1.F90) . . . 506

I Synthetic Soil Moisture from Catchment **507**
I.1 Fortran: Module Interface syntheticsm2obs_module (Source File: syntheticsm2obs_module.F90) 507
I.1.1 syntheticsm2obs_setup (Source File: syntheticsm2obs_module.F90) 507
I.1.2 read_syntheticsm2obs (Source File: read_syntheticsm2obs.F90) 507
I.1.3 getNSO_synsm2.F90 (Source File: getNSO_synsm2.F90) 508

J Synthetic Snow Cover Area **509**
J.1 Fortran: Module Interface syntheticscaobs_module (Source File: syntheticscaobs_module.F90) 509
J.1.1 syntheticscaobs_setup (Source File: syntheticscaobs_module.F90) 509
J.1.2 read_syntheticscaobs (Source File: read_syntheticscaobs.F90) 509
J.1.3 getNSO_synsca (Source File: getNSO_synsca.F90) 510

K Synthetic Snow Water Equivalent **511**
K.1 Fortran: Module Interface syntheticswe1obs_module (Source File: syntheticswe1obs_module.F90) 511
K.1.1 syntheticswe1obs_setup (Source File: syntheticswe1obs_module.F90) 511
K.1.2 read_syntheticswe1obs (Source File: read_syntheticswe1obs.F90) 511
K.1.3 getNSO_synswe1 (Source File: getNSO_synswe1.F90) 512

XIV Land Surface Models in LIS **513**

L Land Surface Models in LIS **514**

M Noah land surface model version 2.7.1 **514**
M.1 Fortran: Module Interface noah_module (Source File: noah_module.F90) 514
M.2 Fortran: Module Interface noah_varder (Source File: noah_varder.F90) 517
M.2.1 noah_varder_ini (Source File: noah_varder.F90) 518
M.2.2 readnoahcrd (Source File: readnoahcrd.F90) 518
M.2.3 noah_setvegparms (Source File: noah_setvegparms.F90) 519
M.2.4 noah_setup (Source File: noah_setup.F90) 519
M.2.5 noah_setsoils (Source File: noah_setsoils.F90) 520
M.2.6 noah_settbot (Source File: noah_settbot.F90) 520
M.2.7 noah_coldstart (Source File: noah_coldstart.F90) 521
M.2.8 noah_dynsetup (Source File: noah_dynsetup.F90) 521
M.2.9 noah_f2t (Source File: noah_f2t.F90) 522
M.2.10 noah_main (Source File: noah_main.F90) 522
M.2.11 noah_output (Source File: noah_output.F90) 523
M.2.12 noah_binout (Source File: noah_binout.F90) 524
M.2.13 noah_gribout (Source File: noah_gribout.F90) 524
M.2.14 noah_netcdfout (Source File: noah_netcdfout.F90) 525
M.2.15 noah_totinit (Source File: noah_totinit.F90) 526
M.2.16 noah_writerst (Source File: noah_writerst.F90) 527
M.2.17 noah_dump_restart (Source File: noah_writerst.F90) 527
M.2.18 noahrst (Source File: noahrst.F90) 528
M.2.19 noah_transform_synsca (Source File: noah_transform_synsca.F90) 529
M.2.20 noah_getscavars (Source File: noah_getscavars.F90) 530
M.2.21 noah_map_sca (Source File: noah_map_sca.F90) 530
M.2.22 noah_setscavars (Source File: noah_setscavars.F90) 531
M.2.23 noah_updatescavars (Source File: noah_updatescavars.F90) 531
M.2.24 noah_getsoilm (Source File: noah_getsoilm.F90) 532
M.2.25 noah_getsynsmpred (Source File: noah_getsynsmpred.F90) 533
M.2.26 noah_setsoilm (Source File: noah_setsoilm.F90) 533

M.2.27 noah_descale_soilm (Source File: noah_descale_soilm.F90) . . . 534
M.2.28 noah_qcsoilm (Source File: noah_qcsoilm.F90) . . . 534
M.2.29 noah_scale_soilm (Source File: noah_scale_soilm.F90) . . . 535
M.2.30 noah_updatesoilm (Source File: noah_updatesoilm.F90) . . . 536
M.2.31 noah_updatesnowvars (Source File: noah_updatesnowvars.F90) . . . 536
M.2.32 noah_getsnowvars (Source File: noah_getsnowvars.F90) . . . 537
M.2.33 noah_getsynswepred (Source File: noah_getsynswepred.F90) . . . 537
M.2.34 noah_descale_snow (Source File: noah_descale_snow.F90) . . . 539
M.2.35 noah_qcsnow (Source File: noah_qcsnow.F90) . . . 540
M.2.36 noah_scale_snow (Source File: noah_scale_snow.F90) . . . 540
M.2.37 noah_setwrfexport.F90 (Source File: noah_setwrfexport.F90) . . . 542
M.2.38 noah_wrf_f2t (Source File: noah_wrf_f2t.F90) . . . 542
M.2.39 sh2o_init.F90 (Source File: sh2o_init.F90) . . . 543
M.2.40 noah_finalize (Source File: noah_finalize.F90) . . . 544

N Community Land Model version 2.0 544
N.1 Fortran: Module Interface clmtype (Source File: clmtype.F90) . . . 545
N.2 Fortran: Module Interface clm_varder (Source File: clm_varder.F90) . . . 550
N.2.1 clm_varder_ini (Source File: clm_varder.F90) . . . 550
N.2.2 clm_varder_init (Source File: clm_varder.F90) . . . 551
N.3 Fortran: Module Interface atmdrvMod (Source File: atmdrvMod.F90) . . . 551
N.3.1 atmdrv (Source File: atmdrvMod.F90) . . . 552
N.3.2 clm_setup (Source File: clm_setup.F90) . . . 552
N.3.3 iniTimeConst (Source File: iniTimeConst.F90) . . . 553
N.3.4 iniTimeVar (Source File: iniTimeVar.F90) . . . 554
N.3.5 clm_dynsetup (Source File: clm_dynsetup.F90) . . . 555
N.3.6 clm_main (Source File: clm_main.F90) . . . 555
N.3.7 clm_output (Source File: clm_output.F90) . . . 557
N.3.8 clm_binout (Source File: clm_binout.F90) . . . 557
N.3.9 clm_gribout (Source File: clm_gribout.F90) . . . 558
N.3.10 clm_totinit (Source File: clm_totinit.F90) . . . 559
N.3.11 clm_writerestart (Source File: clm_writerestart.F90) . . . 559
N.3.12 clm_dump_restart (Source File: clm_writerestart.F90) . . . 560
N.3.13 clm_readrestart (Source File: clm_readrestart.F90) . . . 560
N.3.14 clm_finalize (Source File: clm_finalize.F90) . . . 561
N.3.15 clm_setwrfexport.F90 (Source File: clm_setwrfexport.F90) . . . 561
N.3.16 clm2_wrf_f2t (Source File: clm2_wrf_f2t.F90) . . . 562
N.3.17 clmlairead.F90: (Source File: clmlairead.F90) . . . 563

O Mosaic Land Surface Model 563
O.1 Fortran: Module Interface mos_module (Source File: mos_module.F90) . . . 563
O.1.1 mos_varder (Source File: mos_varder.F90) . . . 565
O.1.2 mos_varder_ini (Source File: mos_varder.F90) . . . 566
O.1.3 mos_setup (Source File: mos_setup.F90) . . . 566
O.1.4 mos_f2t (Source File: mos_f2t.F90) . . . 567
O.1.5 mos_readrestart (Source File: mos_readrestart.F90) . . . 567
O.1.6 mos_main (Source File: mos_main.F90) . . . 568
O.1.7 mos_binout (Source File: mos_binout.F90) . . . 569
O.1.8 mapsib2umd (Source File: mapsib2umd.F90) . . . 570
O.1.9 mos_coldstart (Source File: mos_coldstart.F90) . . . 570
O.1.10 mosdynp (Source File: mosdynp.F90) . . . 571

O.1.11 mos_gribout (Source File: mos_gribout.F90) . . . 571
O.1.12 mos_output (Source File: mos_output.F90) . . . 572
O.1.13 mos_totinit (Source File: mos_totinit.F90) . . . 573
O.1.14 mos_writerst (Source File: mos_writerst.F90) . . . 574
O.1.15 mos_dump_restart (Source File: mos_writerst.F90) . . . 574
O.1.16 paramb (Source File: paramb.F90) . . . 575
O.1.17 readmoscrd (Source File: readmoscrd.F90) . . . 576
O.1.18 setmosp (Source File: setmosp.F90) . . . 576
O.2 Fortran: Module Interface sibalb_module (Source File: sibalb_module.F90) . . . 577
O.2.1 texture (Source File: texture.F90) . . . 577
O.2.2 mos_finalize (Source File: mos_finalize.F90) . . . 577

P HySSIB Land Surface Model 578
P.1 Fortran: Module Interface hyssib_module (Source File: hyssib_module.F90) . . . 578
P.2 Fortran: Module Interface hyssib_varder (Source File: hyssib_varder.F90) . . . 581
P.2.1 hyssib_varder_ini (Source File: hyssib_varder.F90) . . . 582
P.3 Fortran: Module Interface hyssibalb_module (Source File: hyssibalb_module.F90) . . . 583
P.3.1 hyssibalb_ini (Source File: hyssibalb_module.F90) . . . 583
P.4 Fortran: Module Interface hyssibveg_module (Source File: hyssibveg_module.F90) . . . 584
P.4.1 hyssibveg_ini (Source File: hyssibveg_module.F90) . . . 585
P.4.2 hyssib_setup (Source File: hyssib_setup.F90) . . . 585
P.4.3 hyssib_dynsetup (Source File: hyssib_dynsetup.F90) . . . 586
P.4.4 hyssib_f2t (Source File: hyssib_f2t.F90) . . . 586
P.4.5 hyssib_output (Source File: hyssib_output.F90) . . . 587
P.4.6 hyssib_binout (Source File: hyssib_binout.F90) . . . 588
P.4.7 hyssib_totinit (Source File: hyssib_totinit.F90) . . . 588
P.4.8 hyssib_writerst (Source File: hyssib_writerst.F90) . . . 589
P.4.9 hyssib_dump_restart (Source File: hyssib_writerst.F90) . . . 590
P.4.10 hyssib_readrst (Source File: hyssib_readrst.F90) . . . 591
P.4.11 hyssib_readcrd (Source File: hyssib_readcrd.F90) . . . 591
P.4.12 hyssib_coldstart (Source File: hyssib_coldstart.F90) . . . 592
P.4.13 hyssib_gfrac.F90 (Source File: hyssib_gfrac.F90) . . . 592
P.4.14 hyssib_main (Source File: hyssib_main.F90) . . . 593
P.4.15 hyssib_mapvegc.F90 (Source File: hyssib_mapvegc.F90) . . . 594
P.4.16 hyssib_settbot (Source File: hyssib_settbot.F90) . . . 594
P.4.17 hyssib_settopostd (Source File: hyssib_settopostd.F90) . . . 595
P.4.18 hyssib_setvegparms (Source File: hyssib_setvegparms.F90) . . . 595
P.4.19 hyssib_finalize (Source File: hyssib_finalize.F90) . . . 596

Q Catchment Land Surface Model 596
Q.0.20 cat_varder.F90 (Source File: cat_varder.F90) . . . 596
Q.0.21 cat_varalloc (Source File: cat_varder.F90) . . . 597
Q.1 Fortran: Module Interface catch_types (Source File: catch_types.F90) . . . 598
Q.2 Fortran: Module Interface cat_diagn_routines (Source File: catch_diagn_routines.F90) . . . 598
Q.2.1 calc_tsurf (Source File: catch_diagn_routines.F90) . . . 598
Q.2.2 calc_arX (Source File: catch_diagn_routines.F90) . . . 598
Q.2.3 calc_asnow (Source File: catch_diagn_routines.F90) . . . 598
Q.2.4 cat_setup (Source File: cat_setup.F90) . . . 599
Q.2.5 cat_dynsetup.F90 (Source File: cat_dynsetup.F90) . . . 600
Q.2.6 cat_main (Source File: cat_main.F90) . . . 600
Q.2.7 cat_output (Source File: cat_output.F90) . . . 601

Q.2.8 cat_binout (Source File: cat_binout.F90) . . . 601
Q.2.9 cat_readrestart (Source File: cat_readrestart.F90) . . . 602
Q.2.10 cat_writerestart (Source File: cat_writerestart.F90) . . . 603
Q.2.11 cat_dump_restart (Source File: cat_writerestart.F90) . . . 603
Q.2.12 cat_totinit (Source File: cat_totinit.F90) . . . 604
Q.2.13 cat_f2t (Source File: cat_f2t.F90) . . . 604
Q.2.14 read_land_parameters1 (Source File: read_land_parameters.F90) . . . 605
Q.2.15 read_land_parameters2 (Source File: read_land_parameters.F90) . . . 606
Q.2.16 read_land_parameters3 (Source File: read_land_parameters.F90) . . . 606
Q.2.17 d2g (Source File: read_land_parameters.F90) . . . 607
Q.3 Fortran: Module Interface clsm_para_mapping (Source File: clsm_para_mapping.F90) . . . 607
Q.3.1 read_catchment_parameters (Source File: clsm_para_mapping.F90) . . . 608
Q.3.2 read_catchment_parameters_finalize (Source File: clsm_para_mapping.F90) . . . 608
Q.3.3 read_rst (Source File: clsm_para_mapping.F90) . . . 608
Q.3.4 get_tile_index (Source File: clsm_para_mapping.F90) . . . 608
Q.3.5 get_param (Source File: clsm_para_mapping.F90) . . . 609
Q.3.6 cat_finalize (Source File: cat_finalize.F90) . . . 609
Q.3.7 cat_getsoilm (Source File: cat_getsoilm.F90) . . . 610
Q.3.8 cat_setsoilm (Source File: cat_setsoilm.F90) . . . 610
Q.3.9 cat_update_soilm (Source File: cat_update_soilm.F90) . . . 611
Q.3.10 cat_qcsoilm (Source File: cat_qcsoilm.F90) . . . 611
Q.3.11 cat_scale_soilm (Source File: cat_scale_soilm.F90) . . . 612
Q.3.12 cat_descale_soilm (Source File: cat_descale_soilm.F90) . . . 613
Q.3.13 cat_getsynsmpred (Source File: cat_getsynsmpred.F90) . . . 613

R SiB2 Land Surface Model 614
R.1 Fortran: Module Interface sib2_module (Source File: sib2_module.F90) . . . 614
R.2 Fortran: Module Interface sib2_varder (Source File: sib2_varder.F90) . . . 614
R.2.1 sib2_varder_ini (Source File: sib2_varder.F90) . . . 615
R.2.2 sib2_setup (Source File: sib2_setup.F90) . . . 615
R.3 Fortran: Module Interface sib2alb_module (Source File: sib2alb_module.F90) . . . 615
R.3.1 sib2alb_ini (Source File: sib2alb_module.F90) . . . 616
R.3.2 readcrdsib2 (Source File: readcrdsib2.F90) . . . 616
R.3.3 sib2_output (Source File: sib2_output.F90) . . . 617
R.3.4 sib2_binout (Source File: sib2_binout.F90) . . . 618
R.3.5 sib2_coldstart (Source File: sib2_coldstart.F90) . . . 618
R.3.6 sib2_writerst (Source File: sib2_writerst.F90) . . . 619
R.3.7 sib2_dump_restart (Source File: sib2_writerst.F90) . . . 620
R.3.8 sib2_readrst (Source File: sib2_readrst.F90) . . . 620
R.3.9 sib2_totinit (Source File: sib2_totinit.F90) . . . 621
R.3.10 sib2_dynsetup (Source File: sib2_dynsetup.F90) . . . 621
R.3.11 sib2_f2t (Source File: sib2_f2t.F90) . . . 622
R.3.12 sib2_main (Source File: sib2_main.F90) . . . 622
R.3.13 sib2_mapvegc (Source File: sib2_mapvegc.F90) . . . 623
R.3.14 sib2_monthly (Source File: sib2_monthly.F90) . . . 623
R.3.15 sib2_setvegparms (Source File: sib2_setvegparms.F90) . . . 624
R.3.16 sib2_finalize (Source File: sib2_finalize.F90) . . . 624

S Template **624**
S.1 Fortran: Module Interface template_varder (Source File: template_varder.F90) 624
S.1.1 template_varder_ini (Source File: template_varder.F90) 625
S.1.2 readtemplatecrd (Source File: readtemplatecrd.F90) 625
S.1.3 template_setup (Source File: template_setup.F90) 626
S.2 Fortran: Module Interface template_module.F90 (Source File: template_module.F90) 626
S.2.1 template_output (Source File: template_output.F90) 627
S.2.2 template_binout (Source File: template_binout.F90) 628
S.2.3 template_dynsetup (Source File: template_dynsetup.F90) 628
S.2.4 template_f2t (Source File: template_f2t.F90) 629
S.2.5 template_main (Source File: template_main.F90) 629
S.2.6 template_main.F90 (Source File: template_main.F90) 629
S.2.7 template_writerst (Source File: template_writerst.F90) 630
S.2.8 templaterst (Source File: templaterst.F90) 630
S.2.9 template_finalize (Source File: template_finalize.F90) 630

Part I
LIS Overview

Part II
LIS offline program

1 LIS offline program

1.0.1 lisdrv (Source File: lisdrv.F90)

Main program for LIS in an offline simulation
Main driver program for LIS. It performs four main functions

LIS_config calls the routines to read the runtime configurations

LIS_Init calls the initializations based on the runmode

LIS_run calls the run routines based on the runmode

LIS_finalize calls the cleanup routines

The routines invoked are :

lisconfig (2.1.2)
call to initialize configuration tool and read model independent options

lisinit (2.41.2)
call to initialize lis based on the runmode

lisrun (2.41.4)
call to run lis based on the runmode

lisfinalize (2.41.6)
call to cleanup lis structures based on the runmode

REVISION HISTORY:

```
    14Nov02    Sujay Kumar  Initial Specification
    21Oct05    Sujay Kumar  Modified to include the runmodes. Switched
                            to a init,run,finalize mode
 program lisdrv
```

USES:

```
   use lisdrv_module

  call lisconfig
  call lisinit(lis%runmode)
  call lisrun(lis%runmode)
  call lisfinalize(lis%runmode)
```

Part III
LIS core structures and methods

2 LIS core structures and methods

LIS core, the central part of the software, is the infrastructure that integrates the use of different components in LIS. The functions performed by LIS core include operations related to the overall control, runtime statistics, inter-language support, error logging and dynamic memory management functions. Routines to manage domain decomposition, load balancing, fault tolerance, etc. are also encapsulated as generic routines in the high performance computing and communications (HPCC) component. The time management tools in LIS provides useful functions for time and data calculations and higher level functions to control model timestepping and alarms. Another tool implemented in the LIS core structure is the generic I/O tool, which provides capabilities to read the input data locally, handling different data formats. also provide support for distributed data output and multiple formats. Other miscellaneous tools incorporated in the LIS core include methods to perform spatial and temporal interpolation, reprojection, domain subsetting etc. The abstractions providing representations for the behavior of LSMs, domains, and data, runmodes, and data assimilation are also incorporated in the LIS core.

2.1 Fortran: Module Interface lisdrv_module (Source File: lisdrv_module.F90)

The code in this file contains the basic datastructures and control routines for the operation of LIS

2.1.1 Overview

This module contains the defintion and specification of the basic datastructures that define a LIS instance. It consists of:

lis datastructure containg generic LIS variables

lisdom datastructure containing grid, tile spaces, grid projection, domain decomposition information.

metadata_output datastructure defining the metadata for model output.

config_lis instance of the configuration class.

lisvm instance of the virtual machine controlling the LIS resources.

REVISION HISTORY:

```
 14 Nov 2002    Sujay Kumar  Initial Specification
```

USES:

```
use ESMF_Mod
use lis_module
use grid_module
use tile_module
use listime_mgr
use spmdMod
use output_metaMod
use map_utils

implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
 ---------------------------------------------------------------------------
  public :: lisconfig
  public :: LIS_ticktime
  public :: LIS_endofrun
  public :: LIS_TimeToRunNest
  public :: LIS_resetclock
  public :: LIS_finalize

 ---------------------------------------------------------------------------
```

PUBLIC TYPES:

```
 ---------------------------------------------------------------------------
  public :: lis
  public :: lisdom
  public :: metadata_output
  public :: config_lis
  public :: initialized
  public :: lisvm
```

2.1.2 lisconfig (Source File: lisdrv_module.F90)

INTERFACE:

```
interface lisconfig
```

PRIVATE MEMBER FUNCTIONS:

```
    module procedure lisconfig_offline
    module procedure lisconfig_coupled1
    module procedure lisconfig_coupled2
```

DESCRIPTION:

This interface provides routines for the setup of LIS configuration management. It initializes the LIS Config utility, reads the model independent specifications, and intializes the SPMD parallel processing mode. This routine also initializes the LIS time manager, and finally allocates memory for the LIS generic datastructures.

2.1.3 lisconfig_offline (Source File: lisdrv_module.F90)

INTERFACE:

```
Private name: call using lisconfig()
subroutine lisconfig_offline()
```

USES:

```
   use lis_fileUnitsMod, only : getNextUnitNumber
   use lis_logmod,       only : lis_log_init
```

DESCRIPTION:

Performs the initialization of LIS runtime configuration for an offline (uncoupled simulation).
The Calling sequence is :

spmd_init (2.5.1)
performs SPMD initializations

readcard (2.29.1)
reads the model independent options

spmd_setup (2.5.4)
allocates memory for SPMD variables

timemgr_init (2.4.2)
initializes the time manager

2.1.4 lisconfig_coupled1 (Source File: lisdrv_module.F90)

INTERFACE:

```
Private name: call using lisconfig()
subroutine lisconfig_coupled1(nx,ny)
```

USES:

```
use lis_fileUnitsMod, only : getNextUnitNumber
use lis_logmod,       only : lis_log_init

implicit none
```

ARGUMENTS:

```
integer,intent(in) :: nx, ny
```

DESCRIPTION:

Performs the initialization of LIS runtime configuration for an coupled simulation. The processor layout is obtained from a parent component.
The arguments are:

nx Number of processors in the East-West direction in a processor layout

ny Number of processors in the North-South direction in a processor layout

The Calling sequence is :

spmd_init (2.5.1)
performs SPMD initializations

readcard (2.29.1)
reads the model independent options

spmd_setup (2.5.4)
allocates memory for SPMD variables

timemgr_init (2.4.2)
initializes the time manager

2.1.5 lisconfig_coupled2 (Source File: lisdrv_module.F90)

INTERFACE:

```
Private name: call using lisconfig()
subroutine lisconfig_coupled2(vm)
```

USES:

```
use lis_fileUnitsMod, only : getNextUnitNumber
use lis_logmod,       only : lis_log_init

implicit none
```

ARGUMENTS:

```
type(ESMF_VM)       :: vm
```

DESCRIPTION:

Performs the initialization of LIS runtime configuration for an coupled simulation. The virtual machine is obtained from a different component or an application driver.
The Calling sequence is :

spmd_init (2.5.1)
performs SPMD initializations

readcard (2.29.1)
reads the model independent options

spmd_setup (2.5.4)
allocates memory for SPMD variables

timemgr_init (2.4.2)
initializes the time manager

2.1.6 LIS_ticktime (Source File: lisdrv_module.F90)

INTERFACE:

```
subroutine LIS_ticktime()
```

DESCRIPTION:

This routine calls the time manager to increment the runtime clock by the model timestep.
The Calling sequence is :

advance_timestep (2.4.6)
advances the clock

2.1.7 LIS_endofrun (Source File: lisdrv_module.F90)

INTERFACE:

```
function LIS_endofrun() result(finish)
```

ARGUMENTS:

```
logical :: finish
```

DESCRIPTION:

This function checks to see if the runtime clock has reached the specified stop time of the simulation. The arguments are:

finish boolean value indicating if the end of simulation is reached.

The calling sequence is:

is_last_step (2.4.12)
check if the clock has reached the stop time

2.1.8 LIS_timeToRunNest (Source File: lisdrv_module.F90)

INTERFACE:

```
function LIS_timeToRunNest(n) result(check)
```

ARGUMENTS:

```
integer, intent(in) :: n
logical :: check
```

DESCRIPTION:

Check to see if it is time to run the nest based on the model timestep for that particular domain

n index of the nest or domain

check boolean value indicating if the time to run the nest has reached or not

---

2.1.9 LIS_resetclock (Source File: lisdrv_module.F90)

INTERFACE:

```
subroutine LIS_resetclock
```

USES:

```
use listime_mgr, only : resetclock
```

DESCRIPTION:

The calling sequence is:

resetclock (2.4.3)

2.1.10 LIS_finalize (Source File: lisdrv_module.F90)

INTERFACE:

```
subroutine LIS_finalize
```

USES:

```
use spmdMod, only : spmd_finalize
```

DESCRIPTION:

This routine issues the invocation to deallocate and cleanup any allocated data structures.
The calling sequence is:

spmd_finalize (2.5.5)
cleanup SPMD structures

2.2 Fortran: Module Interface lis_module (Source File: lis_module.F90)

Module for specifying model independent variables in LIS. The module does not contain any variables that are specific to the extensible components in LIS. The module specifies variables for overall run control, options for the choice of parameter datasets, data assimilation choices, and the variables controlling the time management in LIS.
The variables specified in this module include:

runmode choice of running mode in LIS (1-retrospective)

nnest number of subdomains or subnests (0 or higher)

nch This array stores the size of tilespace of the running domain, for each processor, for each nest

glbnch This array stores the size of the tilespace of the overall running domain for each nest

ngrid This array stores the size of gridspace of the running domain, for each processor, for each nest

glbngrid This array stores the size of the gridspace of the overall running domain for each nest

gnc Array containing the East-West grid dimension of the overall running grid for each nest

gnr Array containing the North-South grid dimension of the overall running grid for each nest

lnc Array containing the East-West grid dimension of the running grid for each processor, for each nest

lnr Array containing the North-South grid dimension of the running grid for each processor, for each nest

pnc Array containing the East-West grid dimension of the parameter space for each nest, from which data is subsetted.

pnr Array containing the North-South grid dimension of the parameter space for each nest, from which data is subsetted.

domain Choice of "LIS domain" (Please refer to `domain_module` for the definition of a domain in LIS.

lsm Choice of the land surface model in LIS.

texturemap boolean value to denote if soil texture map should be used (0- do not use, 1-use). If this value is set to 1, the sand, clay and silt fraction maps will not be read. Otherwise the texture data is read the fraction data will not be used.

vegsrc Choice of the source of landcover data source

soilsrc Choice of the source of soil parameter data source

colorsrc Choice of the soil color data source(0-do not use)

elevsrc Choice of the elevation data source (0-do not use)

slopesrc Choice of the slope data source (0-do not use)

aspectsrc Choice of the aspect data source (0-do not use)

curvsrc Choice of the curvature data source (0-do not use)

albedosrc Choice of the albedo data source (0-do not use)

gfracsrc Choice of the greenness data source (0-do not use)

porositysrc Choice of the porosity data source (0-do not use)

psisatsrc Choice of the saturated matric potential data source (0-do not use)

ksatsrc Choice of the saturated hydraulic conductivity data source (0-do not use)

bexpsrc Choice of the b parmeter data source (0-do not use)

quartzsrc Choice of the quartz data source (0-do not use)

laisrc Choice of the LAI data source (0-do not use)

snowsrc Choice of the snowdepth data source (0-do not use)

maxt Maximum number of tiles per grid to be considered for subgrid tiling

mina Minumum cutoff percentage of vegetation distribution for subgrid tiling

npesx Number of processors in the East-West dimensions for processor layout

npesy Number of processors in the North-South dimensions for processor layout

udef Value to be used as undefined variable

gridDesc Array describing the running grid, for each nest

soil_gridDesc Array describing the soils data grid, for each nest

topo_gridDesc Array describing the topography data grid, for each nest

lc_gridDesc Array describing the landcover data grid, for each nest

force Choice of forcing data

ecor Choice of topographical downscaling method

use_prism option to specify the use of PRISM downscaling (0-do not use, 1-use)

nf Number of base forcing variables

nsf Number of supplemental forcing variables

rstflag boolean array denoting if the current simulation is a restart or not, for each nest

gridchange boolean array denoting if the native grid for the meteorological forcing need to be changed.

interp Spatial interpolation option (1-bilinear, 2-conservative, 3-neighbor)

tinterp Temporal interpolation option (1-next, 2-uber-next)

shortflag Shortwave radiation source flag (1-instantaneous, 2-time averaged)

longflag Longwave radiation source flag (1-instantaneous, 2-time averaged)

findtime1, findtime2 boolean flags to indicate which time is to be read for temporal interpolation, for each nest

F00_flag,F06_flag flags used to control the forecast forcing read intervals

suppforc Choice of supplemental forcing

perturb_forcing Choice of forcing perturbation algorithm (0-do not use)

pertforcInterval Forcing perturbation frequency (in seconds)

perturb_obs Choice of observation perturbation algorithm (0-do not use)

pertobsInterval Observation perturbation frequency (in seconds)

perturb_state Choice of state perturbation algorithm (0-do not use)

pertstateInterval State perturbation frequency (in seconds)

nt Number of vegetation classes in the landcover dataset

bareclass Index of bare class in the landcover dataset

urbanclass Index of urban class in the landcover dataset

snowclass Index of snow class in the landcover dataset

waterclass Index of water class in the landcover dataset

laiflag flag to control the LAI data read

saiflag flag to contro the SAI data read

mfile Name of the landmask file, for each nest

vfile Name of the landcover file, for each nest

safile Name of the sand fraction data file, for each nest

clfile Name of the clay fraction data file, for each nest

sifile Name of the silt fraction data file, for each nest

txtfile Name of the soil texture data file, for each nest

pofile Name of the porosity data file, for each nest

psisatfile Name of the saturated matric potential data file, for each nest

ksatfile Name of the saturated hydraulic conductivity data file, for each nest

bexpfile Name of the b parameter data file, for each nest

qzfile Name of the quartz data file, for each nest

iscfile Name of the soil color data file, for each nest

elevfile Name of the elevation data file, for each nest

slfile Name of the slope data file, for each nest

aspfile Name of the aspect data file, for each nest

curvfile Name of the curvature data file, for each nest

laifile Name of the LAI data source, for each nest

saifile Name of the SAI data source, for each nest

snowdir Name of the snow depth data source, for each nest

gfracfile Name of the greenness data file, for each nest

albfile Name of the climatology albedo data file, for each nest

mxsnal Name of the max snow albedo data file, for each nest

tbotfile Name of the bottom temperature data file, for each nest

tile_coord_file Name of the catchment-based tile coordinate file, for each nest

tile_veg_file Name of the catchment-based vegetation type file, for each nest

albInterval Frequency of albedo climatology in months

gfracInterval Frequency of greenness climatology in months

laiInterval Frequency of LAI climatology in months

laitime Variable to keep track of LAI data interval

saitime Variable to keep track of SAI data interval

wfor Option to output forcing data

wopt Output methodology (0=no output, 1-tiled, 2-gridded)

wout Output data format (1-binary, 2-grib, 3-netcdf)

wparam Output parameters (0-do not write, 1-write)

wsingle Option to write each variable to a separate file

expcode Three character experiment code

startcode Start mode (1-restart, 2-coldstart)

plevel Logging level

odir Output directory

dfile Diagnostic output file

sdoy Starting julian day

sss Starting second

smn Starting minute

shr Starting hour

sda Starting day

smo Starting month

syr Starting year

endcode End mode (1-specific date)

edoy Ending julian day

ess Ending second

emn Ending minute

ehr Ending hour

eda Ending day

emo Ending month

eyr Ending year

endtime Flag to indicate the end of simulation (1-end of simulation)

etime Ending time

egmt Ending time in GMT

nts Array containing the model timestep for each nest

ts Timestep for the clock (minimum timestep of different nests)

tscount Timestep count for each nest

doy Current julian day

ss Current second

mn Current minute

hr Current hour

da Current day

mo Current month

yr Current year

time Current time

gmt Current time in GMT

daalg Choice of data assimilation algorithm

daset Choice of Assimilation set: (variable to be assimilated + observation dataset)

nstvars Number of state variables to be assimilated

nobtypes Number of observation types to be assimilated

nensem Number of ensembles per grid

forcvarlistFile name of forcing variable list file

forcattribFile name of forcing attributes file

forcpertattribFile name of forcing perturbation attributes file

progpertattribFile name of prognostic variable attributes file

progpertattribFile name of prognostic variable perturbation attributes file

obsattribFile name of observation attributes file

obspertattribFile name of observation perturbation attributes file

nforcepert Number of perturbation

daoutInteval output interval for DA diagnostics

wensems flag to specify if ensemble members are to be output

wobs flag to specify if observations are to be output

winnov flag to specify if normalized innovations are to be output

ncatg Total number of land tiles in the catchment-based parameter files per nest.

REVISION HISTORY:

```
12 Apr 2001: Urszula Jambor; Added domain,lsm,& force namefile paramters
30 Jul 2001: Matt Rodell; Add new soil parameter variables
14 Nov 2002; Sujay Kumar; Optimized version for LIS
14 Oct 2003; Sujay Kumar; Removed LSM specific variables.
19 Jan 2007; Chuck Alonge; Added Flag to output parameters
```

2.3 Fortran: Module Interface grid_module (Source File: grid_module.F90)

The code in this file provides a description of the grid data structure in LIS

2.3.1 Overview

This module contains the grid data structure used in LIS. The data structure contains the variables specified for a single grid cell. It includes:

lat latitude of the grid cell

lon longitude of the grid cell

elev Topological elevation of the grid cell

slope Topological slope of the grid cell

aspect Topological aspect of the grid cell

curv Topological curvature of the grid cell

REVISION HISTORY:

```
14 Nov 2002: Sujay Kumar; Optimized version of grid_module
```

2.4 Fortran: Module Interface listime_mgr (Source File: listime_mgr.F90)

This module contains routines for time managment.The module provides routines for clock initialization, model timestepping, some basic alarm functions, and other useful time managment utilities.

USES:

```
use lis_module
use precision
use spmdMod,only : masterproc
```

2.4.1 isMonthlyAlarmRinging (Source File: listime_mgr.F90)

INTERFACE:

```
interface isMonthlyAlarmRinging
```

PRIVATE MEMBER FUNCTIONS:

```
module procedure isMonthlyAlarmRinging1
module procedure isMonthlyAlarmRinging2
```

DESCRIPTION:

checks if the monthly alarm is ringing. The private functions have different arguments based on the input options specified.

2.4.2 timemgr_init (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine timemgr_init(lis)
  implicit none
```

ARGUMENTS:

```
type(lisdec) :: lis
```

DESCRIPTION:

Initialize the LIS time manager.
NOTE - This assumes that the LIS time specific variables pertaining to start and end times have been set before this routine is called.

lis instance of the `lis_module`

The calling sequence is:

date2time (2.4.21)
to convert current date to a floating point format

timemgr_print display the contents of the time manager

2.4.3 resetclock (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine resetclock(lis)
  implicit none
```

ARGUMENTS:

```
type(lisdec) :: lis
```

DESCRIPTION:

Resets the LIS time manager.
NOTE - This assumes that the LIS time specific variables pertaining to start and end times have been set before this routine is called.

lis instance of the `lis_module`

The calling sequence is:

date2time (2.4.21)
to convert current date to a floating point format

timemgr_print display the contents of the time manager

2.4.4 timemgr_set (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine timemgr_set(lis,yr,mo,da,hr,mn,ss)
```

USES:

```
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

```
type(lisdec) :: lis
integer :: yr,mo,da,hr,mn,ss
```

DESCRIPTION:

sets the time manager clock based on the input time specification. This method is used to initialize the time mangager when the clock is passed down to LIS from a parent component

lis instance of the `lis_module`

yr year

mn month

da day of the month

hr hour of the day

mn minute of the hour

ss second

The calling sequence is:

date2time (2.4.21)
convert date to a flotating point format

2.4.5 timemgr_print (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine timemgr_print(lis)
```

2.4.6 advance_timestep (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine advance_timestep(lis)
```

USES:

```
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

```
type(lisdec) :: lis
```

DESCRIPTION:

Increments time manager clock by the model timestep. In case of LIS running multiple nests, each running at different timesteps, the smallest timestep is chosen to advance the clock.

lis instance of the `lis_module`

The calling sequence is:

date2time (2.4.21)
convert date to a flotating point format

2.4.7 get_step_size (Source File: listime_mgr.F90)

INTERFACE:

```
function get_step_size(lis)

implicit none
```

ARGUMENTS:

```
type(lisdec) :: lis
integer :: get_step_size
```

DESCRIPTION:

Return the timestep used by the clock in the time manager.

lis instance of the `lis_module`

get_step_size timestep value

2.4.8 get_nstep (Source File: listime_mgr.F90)

INTERFACE:

```
function get_nstep(lis,n)

  implicit none
```

ARGUMENTS:

```
  integer, intent(in) :: n
  type(lisdec) :: lis
  integer :: get_nstep
```

DESCRIPTION:

Return the timestep number for each nest.

lis instance of the `lis_module`

n index of the nest

get_nstep timestep number

2.4.9 get_curr_day (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine get_curr_date(lis, yr, mon, day, tod, offset)

  implicit none
```

ARGUMENTS:

```
  type(lisdec) :: lis
  integer, intent(out) ::yr,mon,day,tod
  integer, optional, intent(in) :: offset
```

DESCRIPTION:

Return date components valid at end of current timestep with an optional offset (positive or negative) in seconds.
The arguments are:

lis instance of the `lis_module`

yr year

mon month

day day of the month

tod time of day (seconds past 0Z)

offset offset from current time in seconds positive for future times, negative for previous times.

2.4.10 get_julhr (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine get_julhr(yr,mo,da,hr,mn,ss,julhr)

  implicit none
```

ARGUMENTS:

```
  integer, intent(in)     :: yr
  integer, intent(in)     :: mo
  integer, intent(in)     :: da
  integer, intent(in)     :: hr
  integer, intent(in)     :: mn
  integer, intent(in)     :: ss
  integer                 :: julhr
```

DESCRIPTION:

Returns the julian hour. In this convention, julian day began at midnight at the beginning of May 24, 1968. Interestingly, this convention was introduced by NASA for the space program.
The arguments are:

yr year

mo month

da day of the month

hr hour of day

mn minute

ss second

julhr julian hour

2.4.11 get_curr_calday (Source File: listime_mgr.F90)

INTERFACE:

```
function get_curr_calday(lis,offset)

 implicit none
```

ARGUMENTS:

```
 type(lisdec) :: lis
 integer, optional, intent(in) :: offset
 real(r8) :: get_curr_calday
```

DESCRIPTION:

Return calendar day at end of current timestep with optional offset. Calendar day 1.0 = 0Z on Jan 1.
The arguments are:

lis instance of the `lis_module`

offset offset from current time in seconds, positive for future times, negative for previous times.

get_curr_calday calendar day value

2.4.12 is_last_step (Source File: listime_mgr.F90)

INTERFACE:

```
function is_last_step(lis)
   implicit none
```

ARGUMENTS:

```
   type(lisdec) :: lis
   logical :: is_last_step
```

DESCRIPTION:

Function returns true on last timestep.

lis instance of the `lis_module`

is_last_step result of the function

2.4.13 setMonthlyAlarm (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine setMonthlyAlarm(alarmTime)

  implicit none
```

ARGUMENTS:

```
  real*8 :: alarmTime
```

DESCRIPTION:

Initializes the monthly alarm
The arguments are:

alarmTime time of the monthly alarm

2.4.14 isMonthlyAlarmRinging1 (Source File: listime_mgr.F90)

INTERFACE:

```
!Private name: call using isMonthlyAlarmRinging
subroutine isMonthlyAlarmRinging1(lis, alarmTime, interval, &
     midmonth, ringFlag)

  implicit none
```

ARGUMENTS:

```
  type(lisdec), intent(in) :: lis
  real*8, intent(inout) :: alarmTime
  logical, intent(inout):: ringFlag
  integer, intent(in) :: interval
  logical, intent(in) :: midmonth
```

DESCRIPTION:

checks if the monthly alarm is ringing. The function returns true when the elapsed alarm time is greater than the number of months specified in the alarm's interval. If the midmonth flag is enabled, the elapsed time is counted from the middle of the month to middle of another month, rather than from the beginning of the month to the end of another month.
The arguments are:

lis instance of the `lis_module`

alarmTime the elapsed alarm time

interval the alarm's frequency

midmonth flag to indicate if the elapsed time is to be counted from the beginning or the middle of the month

ringflag flag indicating the status of the call

The calling sequence is:

date2time (2.4.21)
convert date to a flotating point format

2.4.15 isMonthlyAlarmRinging2 (Source File: listime_mgr.F90)

INTERFACE:

```
!Private name: call using isMonthlyAlarmRinging
subroutine isMonthlyAlarmRinging2(lis, alarmTime, interval, ringFlag)
```

2.4.16 setHourlyAlarm (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine setHourlyAlarm(alarmTime)

  implicit none
```

ARGUMENTS:

```
  real*8 :: alarmTime
```

DESCRIPTION:

Initializes the hourly alarm
The arguments are:

alarmTime time of the monthly alarm

2.4.17 isHourlyAlarmRinging (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine isHourlyAlarmRinging(lis, alarmTime, interval, ringFlag)

  implicit none
```

ARGUMENTS:

```
  type(lisdec), intent(in) :: lis
  real*8, intent(inout) :: alarmTime
  integer, intent(in)   :: interval
  logical, intent(inout) :: ringFlag
```

DESCRIPTION:

checks if the hourly alarm is ringing. The function returns true when the elapsed alarm time is greater than the number of hours specified in the alarm's interval.
The arguments are:

lis instance of the `lis_module`

alarmTime the elapsed alarm time

interval the alarm's frequency in hours

ringflag flag indicating the status of the call

The calling sequence is:

tick (2.4.23)
advance the time by the specified increment

2.4.18 computeTimeBookEnds (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine computeTimeBookEnds(lis,interval,time1,time2)
  implicit none
```

ARGUMENTS:

```
  type(lisdec), intent(in) :: lis
  integer, intent(in) :: interval
  real*8, intent(inout) :: time1,time2
```

DESCRIPTION:

compute the time end points based on the interval. If the interval is 3 hours, and the current time is 1:30Z, time1 will be set to 1Z and time2 will be set to 4Z.
The arguments are:

lis instance of the `lis_module`

interval time interval in hours

time1 previous time

time2 next time

The calling sequence is:

tick (2.4.23)
advance the time by the specified increment

2.4.19 computeMonthlyWeights (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine computeMonthlyWeights(lis,interval,mo1,mo2,wt1,wt2)

  implicit none
```

ARGUMENTS:

```
  type(lisdec) :: lis
  integer :: interval
  integer :: mo1,mo2
  real    :: wt1,wt2
```

DESCRIPTION:

This routine computes the time interpolation weights based on the climatology interval, current time, and the previous and next month. For example, a monthly climatology data to the current date can be interpolated as value = value1 *wt1+valu2 *wt2 where value1 is the value from the previous month's climatology, and value2 is the value from the next month's climatology and value is the interpolated value.
The arguments are:

lis instance of the `lis_module`

interval interval of the climatology

mo1 previous month

mo2 next month

wt1, wt2 interpolation weights to be used on the climatology data to interpolate to the current day.

The calling sequence is:

date2time (2.4.21)
converts date to a floating point format

2.4.20 computeMonthlyWeights (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine computeMonthlyWeights(lis,interval,mo1,mo2,wt1,wt2)

  implicit none
```

ARGUMENTS:

```
  type(lisdec) :: lis
  integer :: interval
  integer :: mo1,mo2
  real    :: wt1,wt2
```

DESCRIPTION:

This routine computes the time interpolation weights based on the climatology interval, current time, and the previous and next month. For example, a monthly climatology data to the current date can be interpolated as value = value1 *wt1+valu2 *wt2 where value1 is the value from the previous month's climatology, and value2 is the value from the next month's climatology and value is the interpolated value.
The arguments are:

lis instance of the `lis_module`

interval interval of the climatology

mo1 previous month

mo2 next month

wt1, wt2 interpolation weights to be used on the climatology data to interpolate to the current day.

The calling sequence is:

date2time (2.4.21)
converts date to a floating point format

2.4.21 date2time (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine date2time(time,doy,gmt,yr,mo,da,hr,mn,ss)

  implicit none
```

ARGUMENTS:

```
  integer :: yr,mo,da,hr,mn,ss, doy
  real*8  :: time
  real    :: gmt
```

DESCRIPTION:

determines the time, time in GMT, and the day of the year based on the value of year, month, day of month, hour of the day, minute and second. This method is the inverse of time2date
The arguments are:

yr year

mo month

da day of the month

hr hour of day

mn minute

ss second

time lis time

gmt time in GMT

doy day of the year

2.4.22 time2date (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine time2date(time,doy,gmt,yr,mo,da,hr,mn)

  implicit none
```

ARGUMENTS:

```
  integer :: yr,mo,da,hr,mn,ss,doy
  real*8  :: time
  real    :: gmt
```

DESCRIPTION:

determines the value of the year, month, day of month, hour of the day, minute and second based on the specified time. This method is the inverse of date2time
The arguments are:

yr year

mo month

da day of the month

hr hour of day

mn minute

ss second

time lis time

gmt time in GMT

doy day of the year

2.4.23 tick (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine tick(time,doy,gmt,yr,mo,da,hr,mn,ss,ts)
  implicit none
```

ARGUMENTS:

```
  real*8  :: time
  integer :: yr,mo,da,hr,mn,ss,ts,doy
  real    :: gmt
```

DESCRIPTION:

Method to advance or retract the time by the specified time increment
The arguments are:

yr year

mo month

da day of the month

hr hour of day

mn minute

ss second

ts time increment in seconds

time lis time

gmt time in GMT

doy day of the year

2.4.24 julhr_date (Source File: listime_mgr.F90)

REVISION HISTORY:

```
15 oct 1998  initial version.........................mr moore/dnxm
10 aug 1999  ported to ibm sp2.  added intent attributes to
             arguments...............................mr gayno/dnxm
29 oct 2005  Sujay Kumar, Adopted in LIS
```

INTERFACE:

```
subroutine julhr_date( julhr, yyyy,mm,dd,hh)
```

USES:

```
use lis_logmod, only : lis_abort

implicit none
```

ARGUMENTS:

```
integer,          intent(in)   :: julhr
integer                        :: yyyy
integer                        :: mm
integer                        :: dd
integer                        :: hh
```

DESCRIPTION:

to convert from a julian hour to a 10 digit date/time group (yyyymmddhh)

2.4.25 Method

: call utility routine tmjul4 to convert from julian hours to year, month, day, hour.
perform several checks to ensure date information passed back from tmjul4 is valid.
if date is good, convert from integer data to a 10 digit date/time group and pass back to calling routine.

The arguments are:

julhr input julian hour

yyyy four digit year

mm month of the year

dd day of the month

hh time of the day in hours

The calling sequence is:

tmjul4 (2.4.26)
convert julian hour to hour, day, month, and year

lis_abort (2.27.3)
abort the code in case of error

2.4.26 tmjul4 (Source File: listime_mgr.F90)

REVISION HISTORY:

```
18 mar 98 initial version.........................sra milburn/dnxm
08 jul 99 fixed error which initialzed done flag in a data
          data statement.  variables must be initialized
          using an assignment statement.  ported to ibm sp2.......
          ..............................................mr gayno/dnxm
29 oct 2005  Sujay Kumar, Adopted in LIS
```

INTERFACE:

```
subroutine tmjul4( hour, day, month, year, julhr )

  implicit none
```

ARGUMENTS:

```
  integer,  intent(out)       :: day
  integer,  intent(out)       :: hour
  integer,  intent(in)        :: julhr
  integer,  intent(out)       :: month
  integer,  intent(out)       :: year
```

DESCRIPTION:

uses the julian hour to determine the 2-digit zulu time, day, month, and 4-digit year.

2.4.27 Method

- determine the current zulu hour
- determine the total number of elapsed days
- count forward to the current day/month/year

The arguments are:

hour the zulu time of the julian hour

day day of the month (1..31)

month month of the year (1..12)

year four digit year

julhr the julian hour being processed

2.5 Fortran: Module Interface spmdMod (Source File: spmdMod.F90)

The primary mode of parallel processing in LIS uses the SPMD style, which refers to "single program, multiple data". In this mode, all processors use the same program, though each has its own data. The SPMD mode is enabled through the use of Message Passing Interface (MPI) in LIS.
This module provides variables and routines to be used in a parallel processing environment. This includes variables to keep track of processor resources and domain decomposition information.

USES:

```
  use ESMF_Mod
#if (defined SPMD)
  use mpishorthand
#endif
 ---------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
 ---------------------------------------------------------------------------
  public :: spmd_init      ! intialize variables
  public :: spmd_setup     ! allocate memory
  public :: spmd_finalize  ! cleanup allocated structures
 ---------------------------------------------------------------------------
```

PUBLIC TYPES:

```
 ---------------------------------------------------------------------------
  public :: masterproc ! =0 if the current processor is the master
  public :: iam        ! current processor's id
  public :: npes       ! total number of processors
  public :: c_str      ! starting E-W index of each nest
  public :: c_end      ! ending E-W index of each nest
  public :: r_str      ! starting N-S index of each nest
  public :: r_end      ! ending N-S index of each nest
  public :: deltas     ! size of unmasked grid space buffers
  public :: offsets    ! offsets unmasked grid space buffers
  public :: tdeltas    ! size of tile space buffers
  public :: toffsets   ! offsets of tile space buffers
  public :: gdeltas    ! size of grid space buffers
  public :: goffsets   ! offsets of grid space buffers
  public :: nchs       ! size of tile spaces
  public :: ngrids     ! size of grid spaces
```

2.5.1 spmd_init (Source File: spmdMod.F90)

INTERFACE:

```
  interface spmd_init
```

PRIVATE MEMBER FUNCTIONS:

```
     module procedure spmd_init_offline
     module procedure spmd_init_coupled
```

DESCRIPTION:

This interface intializes the MPI variables, routines for SPMD style of parallel processing. The private routines handle the differences in initializations in an offline and a coupled mode.

2.5.2 spmd_init_offline (Source File: spmdMod.F90)

INTERFACE:

```
subroutine spmd_init_offline(lisvm)
```

DESCRIPTION:

Initializes MPI and retrieves the number of CPUs and the processors IDs. The MPI initialization is done in an offline mode. In a coupled mode, the initialized environment is passed to LIS from a parent component.

2.5.3 spmd_init_coupled (Source File: spmdMod.F90)

INTERFACE:

```
subroutine spmd_init_coupled()
```

DESCRIPTION:

Obtains the number of processors and the processors IDs from an already initialized MPI state. The initialization is assumed to be performed in a parent component.

2.5.4 spmd_setup (Source File: spmdMod.F90)

INTERFACE:

```
subroutine spmd_setup(nnest)

  implicit none
```

ARGUMENTS:

```
  integer, intent(in):: nnest
```

DESCRIPTION:

Allocates memory for the variables describing domain decomposition.
The arguments are:

nnest Number of nests or domains

2.5.5 spmd_finalize (Source File: spmdMod.F90)

INTERFACE:

```
subroutine spmd_finalize
```

DESCRIPTION:

This routine issues the invocation to deallocate and cleanup any allocated data structures.

2.6 Fortran: Module Interface output_metaMod (Source File: output_metaMod.F90)

This module is used by the user to define the metadata associated with model output. The module lists a superset of the land surface model variables. The user can choose a subset from this list through the lis configuration file for model output. Currently this list includes the variable definitions from ALMA specification.
http://www.lmd.jussieu.fr/ALMA/

REVISION HISTORY:

```
21 Oct 2005    Sujay Kumar  Initial Specification
19 Jan 2007    Chuck Alonge Added Snow Depth Option (Future use in WPS)

 real*8  :: time        ! time to begin writing output
 integer :: year        ! year to begin writing output
 integer :: month       ! month to begin writing output
 integer :: day         ! day to begin writing output
 integer :: hour        ! hour to begin writing output
 integer :: min         ! minutes to begin writing output
 integer :: sec         ! seconds to begin writing output

 integer :: swnet        ! Net shortwave radiation (surface) (W/m2)
 integer :: lwnet        ! Net longwave radiation (surface) (W/m2)
 integer :: qle          ! Latent Heat Flux (W/m2)
 integer :: qh           ! Sensible Heat Flux (W/m2)
 integer :: qg           ! Ground Heat Flux (W/m2)
 integer :: qf           ! Energy of fusion (W/m2)
 integer :: qv           ! Energy of sublimation (W/m2)
 integer :: qtau         ! Momentum flux (N/m2)
 integer :: qa           ! Advective flux (W/m2)
 integer :: delsurfheat  ! Change in surface heat storage (J/m2)
 integer :: delcoldcont  ! Change in snow water content (J/m2)

 integer :: snowf        ! Snowfall rate (kg/m2s)
 integer :: rainf        ! Rainfall rate (kg/m2s)
 integer :: evap         ! Evapotranspiration (kg/m2s)
 integer :: qs           ! Surface Runoff(kg/m2s)
 integer :: qrec         ! Recharge from river to the floodplain (kg/m2s)
 integer :: qsb          ! Subsurface Runoff (kg/m2s)
 integer :: qsm          ! Snowmelt (kg/m2s)
 integer :: qfz          ! Refreezing of water in the snowpack (kg/m2s)
 integer :: qst          ! Snow throughfall (kg/m2s)
 integer :: delsoilmoist ! DelSoilMoist
 integer :: delswe       ! DelSWE
 integer :: delsurfstor  ! Change in surface water storage (kg/m2)
 integer :: delintercept ! Change in interception storage (kg/m2)

 integer :: snowt        ! Snow surface temperature (K)
 integer :: vegt         ! Vegetation canopy temperature (K)
 integer :: baresoilt    ! Temperature of bare soil (K)
 integer :: avgsurft     ! Average Surface Temperature (K)
 integer :: radt         ! Surface Radiative Tempearture (K)
 integer :: albedo       ! Surface Albedo (-)
 integer :: swe          ! Snow water equivalent (kg/m2)
 integer :: sweveg       ! SWE intercepted by vegetation (kg/m2)
 integer :: snowfrac     ! Grid cell snow covered fraction
 integer :: snowh        ! Snow Depth(m)
```

```
integer :: surfstor     ! Surface water storage (kg/m2)

integer :: soilmoist
integer :: soiltemp
integer :: sliqfrac   ! fraction of SWE which is in the liquid phase
integer :: smliqfrac  ! Average layer fraction of liquid
                                     ! moisture
integer :: smfrozfrac ! Average layer fraction of liquid
                                     ! moisture

integer :: soilwet      ! Total Soil Wetness (-)

integer :: potevap      ! Potential Evapotranspiration (kg/m2s)
integer :: ecanop       ! Interception evaporation (kg/m2s)
integer :: tveg         ! Vegetation transpiration (kg/m2s)
integer :: esoil        ! Bare soil evaporation (kg/m2s)
integer :: ewater       ! Open water evaporation (kg/m2s)
integer :: rootmoist    ! Root zone soil moisture (kg/m2)
integer :: canopint     ! Total canopy water storage (kg/m2s)
integer :: evapsnow     ! Snow evaporation (kg/m2s)
integer :: subsnow      ! Snow sublimation (kg/m2s)
integer :: subsurf      ! Sublimation of the snow free area (kg/m2s)
integer :: acond        ! Aerodynamic conductance (m/s)

integer :: etpndx       ! Ponded water evaporation (kg/m2s)
integer :: infxsrt      ! Infiltration excess (kg/m2)
integer :: sfcheadrt    ! Surface overland flow head (kg/m2)
```

2.7 Fortran: Module Interface domain_module.F90 (Source File: domain_module.F90)

The code in this file provides interfaces to manages different running domain implementations

2.7.1 Overview

The definition of a domain in LIS is includes two dimensions. A LIS domain is defined by the projection of the running grid and the input data ordering rules. For example, a domain using latlon projection with a SW to NE data ordering is considered different from a domain using latlon projection with NW to SE data ordering. Each input data used with a domain definition is expected to be transformed to the specified projection and data ordering rules through the interfaces defined in the `param_module`.

REVISION HISTORY:

```
 17 Feb 2004    Sujay Kumar  Initial Specification

implicit none
-----------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
-----------------------------------------------------------------------------
 public :: LIS_domain_init          !initialize specified domains
 public :: LIS_domain_finalize      !cleanup allocated structures
```

2.7.2 LIS_domain_init (Source File: domain_module.F90)

INTERFACE:

```
subroutine LIS_domain_init
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use domain_pluginMod, only : domain_plugin
use grib_module,      only : grib_init
```

DESCRIPTION:

This routine invokes the registry that defines the domain implementations, first. This is followed by calling the routines to read the runtime specific parameters for each domain instance. A call to create the domain is issued next, which is expected to generate the grid, tile, and ensemble spaces for each domain. This routine is also expected to perform any domain decomposition needed for parallel processing. Finally, the domain decomposition information is disseminated to individual processors.
The calling sequence is:

domain_plugin (3.2.1)
sets up function table registries for implemented domains

readinput (2.34.4)
invokes the generic method in the registry to read domain specific input options from the configuration file

makedomain (2.34.2)
invokes the generic method in the registry to create the grid and tile spaces.

2.7.3 LIS_domain_finalize (Source File: domain_module.F90)

INTERFACE:

```
subroutine LIS_domain_finalize
```

DESCRIPTION:

This routine issues the invocation to deallocate and cleanup any allocated data structures in the specific instance of the domain implementation.

2.8 Fortran: Module Interface lsm_module (Source File: lsm_module.F90)

The code in this file provides interfaces to manage the operation of different land surface models

2.8.1 Overview

This module defines the interface plugins for the incorporation of different land surface models. These interfaces provide entry points for introducing a new land surface scheme in LIS. The following implementations for each LSM are expected to specify methods to initialize and set the LSM specific variables and parameters, provide methods for model simulation, model output, and restart operations. A number of other optional interfaces need to specified depending on the mode of operation of the LSM. For example, if the LSM is

used for a coupled simulation with an atmospheric component, the `LIS_lsm_setexport` interface need to be implemented. Similar implementations need to be specified for the use of the LSM in data assimilation applications.

REVISION HISTORY:

```
 14 Nov 2002    Sujay Kumar  Initial Specification

use ESMF_Mod

implicit none
-------------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
-------------------------------------------------------------------------------
public :: LIS_lsm_init       ! initialize lsm variables, memory
public :: LIS_setuplsm       ! set land surface parameters
public :: LIS_readrestart    ! read the restart file
public :: LIS_force2tile     ! transfer forcing to model tiles
public :: LIS_lsm_main       ! execute the land model
public :: LIS_lsm_output     ! write model output
public :: setLSMDynParams    ! set the time dependent parameters
public :: LIS_writerestart   ! write the restart file
public :: LIS_lsm_setexport  ! set the variables to be exported to the
                             ! atmospheric component
public :: LIS_perturb_states ! perturbs the prognostic variables
public :: LIS_lsm_finalize   ! cleanup allocated structures
```

2.8.2 LIS_lsm_init (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_lsm_init()
```

USES:

```
use lisdrv_module,    only : lis
use lsm_pluginMod,    only : lsm_plugin
use lsmcpl_pluginMod, only : lsmcpl_plugin
use lsmda_pluginMod,  only : lsmda_plugin
use lis_logmod,       only : logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber
```

DESCRIPTION:

Interface for initializing the land model. The intialization includes the allocation of memory for LSM specific variables and datastructures and specification of any runtime specific options.
The calling sequence is:

lsm_plugin (3.5.1)
sets up function table registries for implemented land surface models

lsmcpl_plugin (3.11.1)
sets up function table registries for implemented land surface models used in coupled modes

lsmda_plugin (3.10.1)
sets up function table registries for land surface models used in data assimilation

lsmini (2.39.2)
invokes the generic method in the registry to initialize the land surface model

perturbinit (2.40.2)
invokes the generic method in the registry to perturb the prognostic variables using the specified algorithm.

2.8.3 LIS_setuplsm (Source File: lsm_module.F90)

INTERFACE:

```
subroutine  LIS_setuplsm
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

The setup interfaces are used for the LSM specification of LSM bparameters. If a parameter falls outside the LIS-specified generic parameter list, the LSM is expected to provide routines for the handling of any input data, specific to those parameters.
The calling sequence is:

lsmsetup (2.39.8)
invokes the generic method in the registry to set up the land surface model

---

2.8.4 LIS_lsm_main (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_lsm_main(n)
```

USES:

```
use lisdrv_module, only: lis
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This interaface provides the entry point to the LSM routines that invokes the land surface model physics. The arguments are:

n index of the nest or domain

The calling sequence is:

lsmrun (2.39.4)
invokes the generic method in the registry to run the land surface model

---

2.8.5 LIS_perturb_states (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_perturb_states(n)
```

USES:

```
use lisdrv_module, only : lis
```

ARGUMENTS:

```
integer,    intent(IN)  :: n
```

DESCRIPTION:

This interaface provides the entry point to the LSM routines that computes the perturbations on the LSM prognostic state variables
The arguments are:

n index of the nest or domain

The calling sequence is:

perturbmethod (2.40.4)
invokes the abstract method to invoke the perturbation algorithm to perturb LSM prognostic variables

getLSMstateVar (2.39.18)
obtains the list of prognostic variables

applyLSMPert (2.8.13)
applies the specified perturbations to the LSM state

qcLSMState (2.39.32)
performs the QC of the perturbed LSM state

setLSMstateVar (2.39.20)
assigns the prognostic variables back to the model states

2.8.6 LIS_readrestart (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_readrestart
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

This interface provides the entry point to read LSM specific model restart files.
The calling sequence is:

lsmrestart (2.39.10)
invokes the generic method in the registry to read restart files for the land surface model

---

2.8.7 LIS_lsm_output (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_lsm_output(n)
```

USES:

```
use lisdrv_module, only: lis
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This interface provides the entry point to write LSM specific model output files.
The arguments are:

n index of the nest or domain

The calling sequence is:

lsmoutput (2.39.12)
invokes the generic method in the registry to write the land surface model output

---

2.8.8 setLSMDynparams (Source File: lsm_module.F90)

INTERFACE:

```
subroutine setLSMDynparams(n)
```

USES:

```
use lisdrv_module, only : lis
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This interface provides an entry point to update any time dependent land surface model parameters in an LSM.
The arguments are:

n index of the nest or domain

The calling sequence is:

lsmdynsetup (2.39.30)
invokes the generic method in the registry to set up time dependent parameters for the land surface model

---

2.8.9 LIS_force2tile (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_force2tile(n)
```

USES:

```
use lisdrv_module, only: lis
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This interface is used to transfer the forcing variables to the actual model tile space. Any forcing perturbations that need to be applied to the input forcing is applied at this stage as well.
The arguments are:

n index of the nest or domain

The calling sequence is:

lsmf2t (2.39.14)
invokes the generic method in the registry to transfer the forcing to the land surface model tiles

---

2.8.10 LIS_writerestart (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_writerestart(n)
```

USES:

```
use lisdrv_module, only : lis
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This interface provides the entry point to read the LSM specific model restart files.
The arguments are:

n index of the nest or domain

The calling sequence is:

lsmwrst (2.39.16)
invokes the generic method in the registry to write restart files for the land surface model

---

2.8.11 LIS_lsm_setexport (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_lsm_setexport(n)
```

USES:

```
use lisdrv_module, only : lis
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This interface provides the entry point for specifying an export state (a list of model specific variables) from a land surface model. The routine is used in a coupled simulation to provide feedback to a different model component such as an atmospheric model.
The arguments are:

n index of the nest or domain

The calling sequence is:

lsmsetexport (2.39.28)
invokes the generic method in the registry to set the export state from the land surface model

---

2.8.12 LIS_lsm_finalize (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_lsm_finalize()
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

This routine issues the invocation to deallocate and cleanup any allocated data structures in the specific instance of a land surface model
The calling sequence is:

lsmfinalize (2.39.6)
invokes the generic method in the registry to cleanup the LSM related datastructures

---

2.8.13 applyLSMPert (Source File: lsm_module.F90)

INTERFACE:

```
subroutine applyLSMPert(n, LSM_State, LSM_Pert_State)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod
```

ARGUMENTS:

```
integer, intent(IN) :: n
type(ESMF_State)    :: LSM_State
type(ESMF_State)    :: LSM_Pert_State
```

DESCRIPTION:

This routine applies the specified perturbations to the LSM prognostic state variables.
The arguments are:

n index of the nest

LSM_State ESMF State with prognostic variables

LSM_Pert_State ESMF State with prognostic variable perturbations

---

2.9 Fortran: Module Interface baseforcing_module (Source File: baseforcing_module.F90)

The code in this file provides interfaces to manage different meteorological forcing analyses

2.9.1 Overview

This module defines interface plugins for the incorporation of various meteorological forcing analyses. The analyses implemented by extending these interfaces should contain all (and possibly more) of the basic meteorological forcing variables. Analyses containing one or more of the basic variables, but not the entire set, should be implmented as a supplemental forcing. The following is a list of the basic meterological forcing variables.

- `T 2m: Temperature interpolated to 2m` (K)
- `Q 2m: Instantaneous specific humidity interpolated to 2m` (kg/kg)
- `SWdown: Downward shortwave flux at the ground` (W/m^2)
- `LWdown: Downward shortwave flux at the ground` (W/m^2)
- `U 10m: Instantaneous zonal wind interpolated to 10m` (m/s)
- `V 10m: Instantaneous meridional wind interpolated to 10m` (m/s)
- `Psurf: Instantaneous surface pressure` (Pa)
- `rainf: Total precipitation` (mm/s)
- `rainf_c: Convective precipitation` (mm/s)
- `snowf: Total snowfall` (mm/s)

REVISION HISTORY:

```
 14 Nov 2002    Sujay Kumar  Initial Specification
 10 Oct 2006:   Sujay Kumar: Switched to using ESMF_State for storing
                             forcing data.

use ESMF_Mod

implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: LIS_baseforcing_init      !initialize base forcing setup
public :: LIS_get_base_forcing      !retrieve data and interpolate
                                    !spatially and temporally
public :: LIS_baseforcing_finalize  !cleanup allocated structures
```

PUBLIC TYPES:

```
public :: lisforc
public :: FORC_State
public :: FORC_Pert_State
```

2.9.2 LIS_baseforcing_init (Source File: baseforcing_module.F90)

INTERFACE:

```
subroutine LIS_baseforcing_init_offline
```

USES:

```
use lisdrv_module, only: lis
use baseforcing_pluginMod
```

DESCRIPTION:

This routine sets up the structures to include meteorological forcing analyses for an offline run. The registry that defines the implemented forcing schemes are invoked followed by the routines to initialize the specific instance of the forcing scheme.
The methods invoked are:

baseforcing_plugin (3.3.1)
sets up function table registries for implemented base forcing analyses

allocate_forcing_mem (2.9.8)
allocate memory for required structures

defineNative (2.35.6)
invokes the generic method in the registry to define the native domain of the base forcing scheme

perturbSetup (2.9.4)
setup and allocate structures required for forcing perturbations.

2.9.3 LIS_baseforcing_init_coupled (Source File: baseforcing_module.F90)

INTERFACE:

```
subroutine LIS_baseforcing_init_coupled(flag)
```

USES:

```
use lisdrv_module, only: lis

integer, intent(in) :: flag
```

DESCRIPTION:

This routine sets up the structures to include meteorological forcing analyses in a coupled runmode. The routine simply allocates memory for the forcing variables.
The methods invoked are:

allocate_forcing_mem (2.9.8)
allocate memory for required structures

perturbSetup (2.9.4)
setup and allocate structures required for forcing perturbations.

2.9.4 perturbSetup (Source File: baseforcing_module.F90)

INTERFACE:

```
subroutine perturbSetup
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
use lis_logmod,    only : logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber
```

DESCRIPTION:

This routine sets up and allocates structures required for forcing pertubations. The ESMF state objects for perturbed forcing fields and the perturbations themselves are created. The perturbation related attributed are also assigned in this routine.

2.9.5 LIS_perturb_forcing (Source File: baseforcing_module.F90)

INTERFACE:

```
subroutine LIS_perturb_forcing(n)
```

USES:

```
use lisdrv_module, only : lis
```

ARGUMENTS:

```
integer,  intent(IN)   :: n
```

DESCRIPTION:

This routine computes the forcing perturbations and then applies them.

2.9.6 LIS_get_base_forcing (Source File: baseforcing_module.F90)

INTERFACE:

```
subroutine LIS_get_base_forcing(n)
```

USES:

```
use lisdrv_module, only: lis, lisdom
use spmdMod, only : iam
```

ARGUMENTS:

```
integer,intent(in) :: n
```

DESCRIPTION:

This routine issues the calls to retrieve the forcing variables from the specific forcing scheme. The retrieval call is expected to read the data, perform any spatial interpolation and other transformations needed to grid the data to the running domain and resolution. This invocation is followed by the call to temporally interpolate the data to the current model timestep. The temporal interpolation is performed based on the two consecutive forcing analyses. A zenith angle based interpolation is performed for the temporal disaggregation of downward shortwave radiation. Finally any topographic corrections and downscaling to the variables are applied.
The arguments are:

n index of the domain or nest.

The methods invoked are:

retrieveforcing (2.35.2)
invokes the generic method in the registry to retrieve the base forcing data

timeinterp (2.35.4)
invokes the generic method in the registry to perform temporal interpolation

lapseRateCorrection (2.24.7)
method to apply topographical corrections

microMetCorrection (2.27.7)
method to apply topographical corrections

2.9.7 LIS_baseforcing_finalize (Source File: baseforcing_module.F90)

INTERFACE:

```
subroutine LIS_baseforcing_finalize
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

This routine issues the invocation to deallocate and cleanup any allocated data structures in the specific instance of the forcing scheme.
The methods invoked are:

forcingfinalize (2.35.8)
invokes the generic method in the registry to cleanup the allocated structures for the base forcing scheme.

2.9.8 allocate_forcing_mem (Source File: baseforcing_module.F90)

INTERFACE:

```
subroutine allocate_forcing_mem()
```

USES:

```
use lisdrv_module, only: lis
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber
use lis_logmod, only       : logunit
```

DESCRIPTION:

This subroutine allocates the memory for the forcing variables for an offline run. The ESMF object for forcing state is created with placeholders for all meteorological forcing variables

---

2.9.9 allocate_forcing_mem_coupled (Source File: baseforcing_module.F90)

INTERFACE:

```
subroutine allocate_forcing_mem_coupled()
```

USES:

```
use lisdrv_module, only: lis
```

DESCRIPTION:

This subroutine allocates the memory for the forcing variables for a coupled run. The ESMF object for forcing state is created with placeholders for all meteorological forcing variables

---

2.10 Fortran: Module Interface suppforcing_module (Source File: suppforcing_module.F90)

The code in this file provides interfaces to manage supplemental forcing analyses

2.10.1 Overview

The module provides interfaces for incorporating supplemental forcing analyses into LIS. Supplemental forcing is defined as any analysis that includes a subset of the meteorological forcing variables defined in `baseforcing_module`. Examples include the incorporating precipitation analyses from various sources such as CMAP, CMORPH, etc. If a supplemental forcing is provided, the correspoinding variables set by the baseforcing is overwritten.

REVISION HISTORY:

```
14 Nov 2002    Sujay Kumar  Initial Specification
```

USES:

```
use spmdMod
implicit none
-----------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
-----------------------------------------------------------------------------
public :: LIS_suppforcing_init ! initializes supplemental forcing
public :: LIS_get_supp_forcing ! retrieve supplemental forcing data
                               ! and interpolate spatially and temporally
public :: LIS_suppforcing_finalize !cleanup allocated structures

-----------------------------------------------------------------------------
```

PUBLIC TYPES:

```
-----------------------------------------------------------------------------
public :: suppforc
```

2.10.2 LIS_suppforcing_init (Source File: suppforcing_module.F90)

INTERFACE:

```
subroutine LIS_suppforcing_init
```

USES:

```
use lisdrv_module, only : lis
use suppforcing_pluginMod, only : suppforcing_plugin
use lis_logmod,            only : logunit
```

DESCRIPTION:

This routine sets up the structures to include supplemental forcing analyses. The registry that defines the implemented schemes are invoked, followed by the routines to initialize the specific instance of the forcing scheme.
The methods invoked are:

suppforcing_plugin (3.4.1)
sets up function table registries for implemented supplemental forcing analyses

defineNativeSupp (2.43.2)
invokes the generic method in the registry to define the native domain of the supplemental forcing scheme

---

2.10.3 LIS_get_supp_forcing (Source File: suppforcing_module.F90)

INTERFACE:

```
subroutine LIS_get_supp_forcing(n)
```

USES:

```
use lisdrv_module, only: lis, lisdom

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine issues the call to retrieve data from the specified supplemental forcing analysis. The retrieval call is expected to read the data, perform any spatial interpolation and other transformations needed to grid the data to the running domain and resolution. This invocation is followed by the call to temporally interpolate the data to the current model timestep. The temporal interpolation details are similar to the description in `baseforcing_module`.
The arguments are:

n index of the domain or nest.

The methods invoked are:

getsuppforc (2.43.4)
invokes the generic method in the registry to retrieve the supplemental forcing data

timeinterpsupp (2.43.6)
invokes the generic method in the registry to perform temporal interpolation

---

2.10.4 LIS_suppforcing_finalize (Source File: suppforcing_module.F90)

INTERFACE:

```
subroutine LIS_suppforcing_finalize
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

This routine issues the invocation to deallocate and cleanup any allocated data structures in the specific instance of the supplemental forcing scheme.
The methods invoked are:

suppforcingfinalize (2.43.8)
invokes the generic method in the registry to cleanup the allocated structures for the supplemental forcing scheme.

2.11 Fortran: Module Interface dataassim_module (Source File: dataassim_module.F90)

The code in this file provides abstractions to manage different data assimilation implementations.

2.11.1 Overview

This module defines some of the interface plugins for the incorporation of different data assimilation implementations. There are three main levels of abstractions identified in a data assimilation implementation:

- Data Assimilation Algorithm: various methods such as direct insertion, extended kalman filter, ensemble kalman filter, etc.
- Observations/Variable being assimilated: this abstraction identifies the source of variables, and the translation of observations to model state variables.
- Land surface model: the model used in the assimilation implementation.
- Pertubation techniques: methods to perturb the forcing and model prognostic variables

A complete implementation would require the interaction between the specific instances of the components listed above. This module includes methods to invoke some of the above.

REVISION HISTORY:

```
 27 Feb 2005    Sujay Kumar; Initial Specification

implicit none
-----------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
-----------------------------------------------------------------------------
 public :: LIS_dataassim_init      !initialize data assimilation routines
 public :: LIS_dataassim_run       !invoke data assimilation algorithms
 public :: LIS_dataassim_finalize  !cleanup allocated structures
```

2.11.2 LIS_dataassim_init (Source File: dataassim_module.F90)

INTERFACE:

```
subroutine LIS_dataassim_init
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis, config_lis
use dataassim_pluginMod
use biasEstimation_pluginMod
```

DESCRIPTION:

The routine performs the initializations required for data assimilation. The registries that define the instances of data assimilation algorithms, observations, and perturbation methods are invoked, followed by the creation of required data structures.
The calling sequence is:

dataassim_plugin (3.7.1)
sets up function table registries for implemented data assimilation algorithms

dataassiminit (2.33.8)
invokes the generic method in the registry to set up the data assimilation algorithm- structures

---

2.11.3 LIS_dataassim_run (Source File: dataassim_module.F90)

INTERFACE:

```
subroutine LIS_dataassim_run(n)
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

The routine initiates the call to the specific assimilation algorithm.
The calling sequence is:

computeIncrements (2.33.12)
computes the increments from the analysis

applyIncrements (2.33.10)
applies the increments from the analysis

2.11.4 LIS_dataassim_output (Source File: dataassim_module.F90)

INTERFACE:

```
subroutine LIS_dataassim_output(n)
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

This routine issues the invocation to write data assimilation specific output
The calling sequence is:

daoutput (2.33.16)
invokes the generic method in the registry to write data assimilation specific output

---

2.11.5 LIS_dataassim_finalize (Source File: dataassim_module.F90)

INTERFACE:

```
subroutine LIS_dataassim_finalize
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

This routine issues the invocation to deallocate and cleanup any allocated data structures in the specific instance of the data assimilation implementations.
The calling sequence is:

dafinalize (2.33.14)
invokes the generic method in the registry to cleanup the allocated structures for the data assimilation algorithm

---

2.12 Fortran: Module Interface observations_module (Source File: observations_module.F90)

The code in this file implements routines to read various sources of observations to be used for data assimilation.

REVISION HISTORY:

```
 21 Jun 2006: Sujay Kumar; Initial implementation
use ESMF_Mod

implicit none

-----------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
-----------------------------------------------------------------------------
 public::  LIS_readobservations
-----------------------------------------------------------------------------
```

PUBLIC TYPES:

```
-----------------------------------------------------------------------------
 type(ESMF_State), pointer :: OBS_State(:)
 type(ESMF_State), pointer :: OBS_Pert_State(:)
```

2.12.1 LIS_initObservations (Source File: observations_module.F90)

INTERFACE:

```
subroutine LIS_initObservations
```

DESCRIPTION:

This routine allocates the structures required for managing observational data used for data assimilation. The routine creates an empty ESMF State that will be used to incorporte observational data.
The methods invoked are:

dataobs_plugin (3.8.1)
sets up function table registries for implemented observation sources to be assimilated

readObsDataConfig (2.33.2)
invokes the generic method in the registry to set up the structures to read the observation data, for assimilation

perturbinit (2.40.2)
invokes the generic method in the registry to perturb the prognostic variables using the specified algorithm.

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only    : logunit
use dataobs_pluginMod, only : dataobs_plugin
```

2.12.2 LIS_readobservations (Source File: observations_module.F90)

INTERFACE:

```
subroutine LIS_readobservations(n)
```

USES:

```
use lisdrv_module, only : lis
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routines calls the method to read the specific observation source.
The methods invoked are:

readDAObs (2.33.4)
method to read the observations and incorporate into the observation state

---

2.12.3 LIS_perturb_observations (Source File: observations_module.F90)

INTERFACE:

```
subroutine LIS_perturb_observations(n)
```

USES:

```
use lisdrv_module, only : lis
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routines calls the method to perturb the observations based on the specified choice.
The methods invoked are:

perturbmethod (2.40.4)
invokes the specified algorithm to perturb the observations.

2.13 Fortran: Module Interface perturb_module (Source File: perturb_module.F90)

The code in this file provides abstractions to manage different perturbation implementations. This includes methods to perturb forcing, observations, and land surface states.

REVISION HISTORY:

```
25 Jun 2006: Sujay Kumar; Initial implementation
```

2.13.1 LIS_perturb_init (Source File: perturb_module.F90)

INTERFACE:

```
subroutine LIS_perturb_init
```

DESCRIPTION:

The routines invoked are:

perturb_plugin (3.9.1)
sets up the function table registries for implemented perturbation algorithms

USES:

```
use perturb_pluginMod
```

2.14 Fortran: Module Interface param_module (Source File: param_module.F90)

The code in this file provides interfaces to manage different sources of parameter datasets.

2.14.1 Overview

This module contains interface plugins for the incorporating I/O associated with various sources of land surface parameter maps. These interfaces act as the entry points for handing issues such as format, ordering,and projection associated with a dataset. The parameter reading routines are expected to perform all these transformations and convert the data to same grid, domain, and data ordering as the ones used in the running domain of LIS. The `param_module` provides interfaces to incorporate the following sources of parameter maps.

soils sand, silt, clay fractions, soil color, soil texture

albedo climatology and max albedo over deep snow

greenness climatology

Leaf Area Index climatology

Stem Area Index climatology

Topography static elevation, slope, aspect, curvature data

Bottom Temperature static

REVISION HISTORY:

```
    26 Oct 2005    Sujay Kumar  Initial Specification
```

```
   implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
  ------------------------------------------------------------------------------
   public :: LIS_param_init  !initializes structures, read static data
   public :: LIS_setDynParams !read time dependent data
   public :: LIS_param_finalize !cleanup allocated structures
  ------------------------------------------------------------------------------
```

PUBLIC TYPES:

```
  ------------------------------------------------------------------------------
   public :: paramAlarmTime

   real*8 :: paramAlarmTime
```

2.14.2 LIS_param_init (Source File: param_module.F90)

INTERFACE:

```
   subroutine LIS_param_init()
```

USES:

```
use lisdrv_module, only: lis
use gfrac_module, only : greenness_setup
use albedo_module, only : albedo_setup
use soils_module, only : soils_setup,read_soils
use lai_module, only : lai_setup,read_lai
use sai_module, only : sai_setup,read_sai
use snow_module, only : snow_setup
use prism_module, only : LIS_prism_init
use listime_mgr, only : setMonthlyAlarm
```

DESCRIPTION:

This interface provides the entry point for the initialization of data structures required for reading in parameter datasets. It also invokes routines to read static datasets such as soils.
The calling sequence is:

soils_setup (2.20.2)
call to set up the soil parameters options

read_soils (2.20.3)
call to read the chosen soil parameters

albedo_setup (2.15.2)
call to setup albedo parameter options

greenness_setup (2.16.2)
call to setup greenness parameter options

lai_setup (2.17.2)
call to setup LAI parameter options

sai_setup (2.18.2)
call to setup SAI parameter options

snow_setup (2.19.1)
call to setup snow parameter options

2.14.3 LIS_setDynparams (Source File: param_module.F90)

INTERFACE:

```
subroutine LIS_setDynparams(n)
```

USES:

```
use lisdrv_module, only : lis
use albedo_module, only : read_albedo
use gfrac_module, only : read_greenness
use lai_module, only : read_lai
use sai_module, only : read_sai
use snow_module, only : read_snow
use lsm_module, only : setLSMDynparams
use prism_module, only : prism_dataops
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This interface provides the entry point for routines to update any time dependent land surface parameters. Current implementation includes datasets such as albedo, greenness, LAI, and SAI.
The arguments are:

n index of the domain or nest.

The calling sequence is:

read_greenness (2.16.3)
call to read the greenness data source

read_albedo (2.15.4)
call to read the albedo data source

read_lai (2.17.3)
call to read the LAI data source

read_sai (2.18.3)
call to read the SAI data source

read_snow (2.19.2)
call to read the snow data source

2.14.4 LIS_output_params (Source File: param_module.F90)

INTERFACE:

```
subroutine LIS_output_params(n)
```

USES:

```
use lisdrv_module,    only : lis
use listime_mgr,      only : isMonthlyAlarmRinging
use lis_fileIOMod,    only : create_output_directory, &
     create_output_filename
use lis_fileUnitsMod, only : getNextUnitNumber, &
     releaseUnitNumber
use spmdMod,          only : masterproc
use soils_module,     only : writesoils
use gfrac_module,     only : writegfrac
use LAI_module,       only : writeLAI
use SAI_module,       only : writeSAI
use albedo_module,    only : writealbedo
```

ARGUMENTS:

```
implicit none
integer, intent(in) :: n
```

DESCRIPTION:

This routine is used to enable the output of land surface parameters to a file, once a month.
The arguments are:

n index of the nest

The routines called are:

isMonthlyAlarmRinging (2.4.1)
check to see if the alarm set for parameter output is ringing

create_output_directory (2.25.2)
creates a time stamped output directory

create_output_filename (2.25.3)
creates a time stamped output filename

2.14.5 LIS_param_finalize (Source File: param_module.F90)

INTERFACE:

```
subroutine LIS_param_finalize()
```

USES:

```
use lisdrv_module, only : lis
use albedo_module, only : albedo_finalize
use gfrac_module, only : greenness_finalize
use lai_module, only : lai_finalize
use sai_module, only : sai_finalize
use snow_module, only : snow_finalize
```

DESCRIPTION:

This routine issues the invocation to deallocate and cleanup any allocated data structures used in the parameter dataset implementations.
The calling sequence is:

albedo_finalize (2.15.5)
call to cleanup albedo related structures

greenness_finalize (2.16.4)
call to cleanup greenness related structures

lai_finalize (2.17.4)
call to LAI related structures

sai_finalize (2.18.4)
call to SAI related structures

2.14.6 writelandmask (Source File: param_module.F90)

INTERFACE:

```
subroutine writelandmask(n,ftn)
```

USES:

```
use lisdrv_module,  only : lis, lisdom
use drv_output_mod, only : drv_writevar_bin
```

ARGUMENTS:

```
implicit none
integer, intent(in)   :: n
integer, intent(in)   :: ftn
```

DESCRIPTION:

This routine writes the LIS landmask to the parameter output file.
The arguments are:

n index of the nest

ftn file unit number to be used

The routines called are:

drv_writevar_bin (2.22.2)
generic routine to write a single variable to a flat binary output file.

2.14.7 writelandcover (Source File: param_module.F90)

INTERFACE:

```
subroutine writelandcover(n,ftn)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use drv_output_mod, only : drv_writevar_bin
```

ARGUMENTS:

```
implicit none
integer, intent(in)   :: n
integer, intent(in)   :: ftn
```

DESCRIPTION:

This routine writes the LIS landcover to the parameter output file.
The arguments are:

n index of the nest

ftn file unit number to be used

The routines called are:

drv_writevar_bin (2.22.2)
generic routine to write a single variable to a flat binary output file.

2.14.8 writetopography (Source File: param_module.F90)

INTERFACE:

```
subroutine writetopography(n,ftn)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use drv_output_mod, only : drv_writevar_bin
```

ARGUMENTS:

```
implicit none
integer, intent(in)   :: n
integer, intent(in)   :: ftn
```

DESCRIPTION:

This routine writes the LIS topography to the parameter output file.
The arguments are:

n index of the nest

ftn file unit number to be used

The routines called are:

drv_writevar_bin (2.22.2)
generic routine to write a single variable to a flat binary output file.

2.15 Fortran: Module Interface albedo_module (Source File: albedo_module.F90)

The code in this file implements routines to read various sources of albedo.

2.15.1 Overview

This routines in this module reads two sources of albedo:

- Albedo climatology
- Static, maximum albedo expected over deep snow

This module provides routines to read the albedo climatology data and allows the users to specify the frequency of climatology (in months). The climatological data is temporally interpolated between months to the current simulation date.

REVISION HISTORY:

```
 8 Aug 2005: Sujay Kumar; Initial implementation
implicit none

-----------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
-----------------------------------------------------------------------------
 public :: albedo_setup       !allocates memory for required variables
 public :: read_albedo        !reads climatological albedo
 public :: albedo_finalize    !cleanup allocated structures

-----------------------------------------------------------------------------
```

PUBLIC TYPES:

```
-----------------------------------------------------------------------------
 public :: lis_alb      !data structure containing albedo data
```

2.15.2 albedo_setup (Source File: albedo_module.F90)

INTERFACE:

```
subroutine albedo_setup
```

DESCRIPTION:

Allocates memory for data structures for reading in albedo datasets
The routines invoked are:

read_mxsnalb (2.15.3)
method to read the max snow albedo

setMonthlyAlarm (2.4.13)
sets the monthly alarm to read the albedo climatology

USES:

```
use lisdrv_module, only : lis
use listime_mgr, only : setMonthlyAlarm
```

2.15.3 read_mxsnalb (Source File: albedo_module.F90)

INTERFACE:

```
subroutine read_mxsnalb(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Reads the static albedo upper bound over deep snow for each domain.
The arguments are:

n index of the domain or nest.

The routines invoked are:

readmxsnoalb (2.32.2)
Calls the generic method in the registry to read the max snow albedo

---

2.15.4 read_albedo (Source File: albedo_module.F90)

INTERFACE:

```
subroutine read_albedo(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use listime_mgr, only : isMonthlyAlarmRinging, computeMonthlyWeights
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Reads the snow free albedo climatology data and temporally interpolates it to the current day
The arguments are:

n index of the domain or nest.

The routines invoked are:

isMonthlyAlarmRinging (2.4.1)
checks to see if it is time to read the climatology data

computeMonthlyWeights (2.4.19)
computes the interpolation weights

readalbedo (2.32.4)
invokes the generic method in the registry to read the albedo climatology data

---

2.15.5 albedo_finalize (Source File: albedo_module.F90)

INTERFACE:

```
subroutine albedo_finalize
```

USES:

```
use lisdrv_module, only : lis
implicit none
```

DESCRIPTION:

Deallocates objects created in this module

2.15.6 writealbedo (Source File: albedo_module.F90)

INTERFACE:

```
subroutine writealbedo(n,ftn)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use drv_output_mod, only : drv_writevar_bin
```

ARGUMENTS:

```
implicit none
integer, intent(in)   :: n
integer, intent(in)   :: ftn
```

DESCRIPTION:

This routine writes the LIS albedo to the parameter output file.
The arguments are:

n index of the nest

ftn file unit number to be used

The routines called are:

drv_writevar_bin (2.22.2)
generic routine to write a single variable to a flat binary output file.

2.16 Fortran: Module Interface gfrac_module (Source File: gfrac_module.F90)

The code in this file implements routines to read greenness fraction data.

2.16.1 Overview

This routines in this module provides routines to read the greenness fraction climatology data and allows the users to specify the frequency of climatology (in months). The climatological data is temporally interpolated between months to the current simulation date.

REVISION HISTORY:

```
 08 Aug 2005: Sujay Kumar; Initial implementation
implicit none

------------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
------------------------------------------------------------------------------
 public :: greenness_setup    !allocates memory for required structures
 public :: read_greenness     !reads the greenness data
 public :: greenness_finalize !cleanup allocated structures

------------------------------------------------------------------------------
```

PUBLIC TYPES:

```
------------------------------------------------------------------------------
 public :: lis_gfrac !data structure containing greenness fraction data.
```

2.16.2 greenness_setup (Source File: gfrac_module.F90)

INTERFACE:

```
subroutine greenness_setup
```

USES:

```
  use lisdrv_module, only : lis
  use listime_mgr, only : setMonthlyAlarm
```

DESCRIPTION:

Allocates memory for data structures for reading the greenness fraction datasets
The routines invoked are:

setMonthlyAlarm (2.4.13)
sets the monthly alarm to read the greenness climatology

2.16.3 read_greenness (Source File: gfrac_module.F90)

INTERFACE:

```
subroutine read_greenness(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use listime_mgr, only : isMonthlyAlarmRinging, computeMonthlyWeights
use spmdMod, only : masterproc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Reads the greenness fraction climalotogy and temporally interpolates it to the current day.
The arguments are:

n index of the domain or nest.

The routines invoked are:

isMonthlyAlarmRinging (2.4.1)
checks to see if it is time to read the climatology data

computeMonthlyWeights (2.4.19)
computes the interpolation weights

readgfrac (2.36.2)
invokes the generic method in the registry to read the greenness climatology data

---

2.16.4 greenness_finalize (Source File: gfrac_module.F90)

INTERFACE:

```
subroutine greenness_finalize
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

Deallocates objects created in this module

2.16.5 writegfrac (Source File: gfrac_module.F90)

INTERFACE:

```
subroutine writegfrac(n,ftn)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use drv_output_mod, only : drv_writevar_bin
```

ARGUMENTS:

```
implicit none
integer, intent(in)   :: n
integer, intent(in)   :: ftn
```

DESCRIPTION:

This routine writes the LIS gfrac to the parameter output file.
The arguments are:

n index of the nest

ftn file unit number to be used

The routines called are:

drv_writevar_bin (2.22.2)
generic routine to write a single variable to a flat binary output file.

2.17 Fortran: Module Interface lai_module (Source File: lai_module.F90)

The code in this file provides interfaces to manage different sources of Leaf Area Index (LAI) data.

2.17.1 Overview

This routines in this module provides routines to read the LAI climatology data. The climatological data is temporally interpolated between months to the current simulation date.
Note that currently the module assumes the frequency of LAI climatology to be monthly.

REVISION HISTORY:

```
08 Aug 2005: Sujay Kumar; Initial implementation
```

USES:

```
use precision
implicit none
```

```
-----------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
-----------------------------------------------------------------------------
public :: lai_setup ! allocates memory for required structures
public :: read_lai  ! reads the LAI data
public :: lai_finalize ! cleanup allocated structures
-----------------------------------------------------------------------------
```

PUBLIC TYPES:

```
-----------------------------------------------------------------------------
public :: lis_lai ! data structure containing LAI data.
```

2.17.2 lai_setup (Source File: lai_module.F90)

INTERFACE:

```
subroutine lai_setup
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr, only : setMonthlyAlarm
```

DESCRIPTION:

Allocates memory and other structures for reading LAI datasets
The routines invoked are:

setMonthlyAlarm (2.4.13)
sets the monthly alarm to read the LAI climatology

2.17.3 read_lai (Source File: lai_module.F90)

INTERFACE:

```
subroutine read_lai(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use listime_mgr, only : isMonthlyAlarmRinging, computeMonthlyWeights

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Reads the LAI climatology and temporally interpolates it to the current day.
The arguments are:

n index of the domain or nest.

The routines invoked are:

readlai (2.37.2)
invokes the generic method in the registry to read the LAI climatology data

---

2.17.4 lai_finalize (Source File: lai_module.F90)

INTERFACE:

```
subroutine lai_finalize
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

Deallocates objects created in this module

2.17.5 writeLAI (Source File: lai_module.F90)

INTERFACE:

```
subroutine writeLAI(n,ftn)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use drv_output_mod, only : drv_writevar_bin
```

ARGUMENTS:

```
implicit none
integer, intent(in)   :: n
integer, intent(in)   :: ftn
```

DESCRIPTION:

This routine writes the LIS LAI to the parameter output file.
The arguments are:

n index of the nest

ftn file unit number to be used

The routines called are:

drv_writevar_bin (2.22.2)
generic routine to write a single variable to a flat binary output file.

2.18 Fortran: Module Interface sai_module (Source File: sai_module.F90)

The code in this file provides interfaces to manage different sources of Leaf Area Index (SAI) data.

2.18.1 Overview

This routines in this module provides routines to read the SAI climatology data. The climatological data is temporally interpolated between months to the current simulation date.
Note that currently the module assumes the frequency of SAI climatology to be monthly.

REVISION HISTORY:

```
08 Aug 2005: Sujay Kumar; Initial implementation
```

USES:

```
use precision
implicit none
```

```
-----------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
-----------------------------------------------------------------------------
 public :: sai_setup ! allocates memory for required structures
 public :: read_sai  ! reads the SAI data
 public :: sai_finalize ! cleanup allocated structures
-----------------------------------------------------------------------------
```

PUBLIC TYPES:

```
-----------------------------------------------------------------------------
 public :: lis_sai ! data structure containing SAI data.
```

2.18.2 sai_setup (Source File: sai_module.F90)

INTERFACE:

```
subroutine sai_setup
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr, only : setMonthlyAlarm
```

DESCRIPTION:

Allocates memory and other structures for reading SAI datasets
The routines invoked are:

setMonthlyAlarm (2.4.13)
sets the monthly alarm to read the SAI climatology

2.18.3 read_sai (Source File: sai_module.F90)

INTERFACE:

```
subroutine read_sai(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use listime_mgr, only : isMonthlyAlarmRinging, computeMonthlyWeights

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Reads the SAI climatology and temporally interpolates it to the current day.
The arguments are:

n index of the domain or nest.

The routines invoked are:

readsai (2.37.4)
invokes the generic method in the registry to read the SAI climatology data

---

2.18.4 sai_finalize (Source File: sai_module.F90)

INTERFACE:

```
subroutine sai_finalize
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

Deallocates objects created in this module

2.18.5 writeSAI (Source File: sai_module.F90)

INTERFACE:

```
subroutine writeSAI(n,ftn)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use drv_output_mod, only : drv_writevar_bin
```

ARGUMENTS:

```
implicit none
integer, intent(in)   :: n
integer, intent(in)   :: ftn
```

DESCRIPTION:

This routine writes the LIS SAI to the parameter output file.
The arguments are:

n index of the nest

ftn file unit number to be used

The routines called are:

drv_writevar_bin (2.22.2)
generic routine to write a single variable to a flat binary output file.

2.19 Fortran: Module Interface snow_module (Source File: snow_module.F90)

The code in this file implementts routines to read various sources of snow depth data. Currently this module is used to ingest the SNODEP products from AFWA, which is obtained at 12z daily.

REVISION HISTORY:

```
08 Aug 2005: Sujay Kumar; Initial implementation
```

USES:

```
implicit none

-----------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
-----------------------------------------------------------------------------
public :: snow_setup       !allocates memory for required variables
public :: read_snow        !reads snow data
public :: snow_finalize    !cleanup allocated structures

-----------------------------------------------------------------------------
```

PUBLIC TYPES:

```
-----------------------------------------------------------------------------
public :: snow_struc      !data structure containing snow variables
```

2.19.1 snow_setup (Source File: snow_module.F90)

Allocates memory for data structures used for reading snow datasets. The snowdepth field is updated by the external files. The snow water equivalent fields are expected to be set by the model.

INTERFACE:

```
subroutine snow_setup
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr, only   : setHourlyAlarm
```

2.19.2 read_snow (Source File: snow_module.F90)

This routine reads the snowdepth data at 12Z, every day.
The arguments are:

n index of the nest

INTERFACE:

```
subroutine read_snow(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use listime_mgr, only   : isHourlyAlarmRinging
```

ARGUMENTS:

```
implicit none

integer, intent(in) :: n
```

2.19.3 snow_finalize (Source File: snow_module.F90)

This routine cleans up snow-related structures

INTERFACE:

```
subroutine snow_finalize
```

USES:

```
use lisdrv_module, only : lis
```

2.20 Fortran: Module Interface soils_module (Source File: soils_module.F90)

The code in this file implements routines to read various sources of soil parameter data.

2.20.1 Overview

This module provides routines for reading and manipulating various parameters related to soil properties. The following list of soil parameters are currently supported.

sand, silt, clay fractions

soil color data

soil texture data

soil porosity data

hydraulic conductivity data

thermal conductivity data

b parameter data

quartz data

REVISION HISTORY:

```
 21 Oct 2005: Sujay Kumar; Initial implementation
implicit none
-----------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
-----------------------------------------------------------------------------
public :: soils_setup  ! initializes data structures and memory
public :: read_soils   ! read soil parameters
public :: soils_finalize !cleanup allocated structures
-----------------------------------------------------------------------------
```

PUBLIC TYPES:

```
-----------------------------------------------------------------------------
public :: lis_soils      !data structure containing soils data
```

2.20.2 soils_setup (Source File: soils_module.F90)

INTERFACE:

```
subroutine soils_setup()
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

Allocates memory for data structures for reading soils datasets

---

2.20.3 read_soils (Source File: soils_module.F90)

INTERFACE:

```
subroutine read_soils()
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

Reads the soils data based on the choice of options specified in the lis configuration.
The routines invoked are:

readsoiltexture (2.42.8)
invokes the generic method in the registry to read the soil texture data

readsand (2.42.2)
invokes the generic method in the registry to read the sand fraction data

readclay (2.42.4)
invokes the generic method in the registry to read the clay fraction data

readsilt (2.42.6)
invokes the generic method in the registry to read the silt fraction data

readcolor (2.42.20)
invokes the generic method in the registry to read the soil color data

readporosity (2.42.10)
invokes the generic method in the registry to read the porosity data

readpsisat (2.42.12)
invokes the generic method in the registry to read the saturated matric potential data

readksat (2.42.14)
invokes the generic method in the registry to read the saturated hydraulic conductivity data

readbexp (2.42.16)
invokes the generic method in the registry to read the b parameter data

readquartz (2.42.18)
invokes the generic method in the registry to read the quartz data

---

2.20.4 soils_finalize (Source File: soils_module.F90)

INTERFACE:

```
subroutine soils_finalize()
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

deallocates memory for all datastructures used for reading soils datasets. This method is typically called after the information is translated to the LSM model tiles.

2.20.5 writesoils (Source File: soils_module.F90)

INTERFACE:

```
subroutine writesoils(n,ftn)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use drv_output_mod, only : drv_writevar_bin
```

ARGUMENTS:

```
implicit none
integer, intent(in)   :: n
integer, intent(in)   :: ftn
```

DESCRIPTION:

This routine writes the LIS soils to the parameter output file.
The arguments are:

n index of the nest

ftn file unit number to be used

The routines called are:

drv_writevar_bin (2.22.2)
generic routine to write a single variable to a flat binary output file.

2.20.6 check_error (Source File: check_error.F90)

INTERFACE:

```
subroutine check_error(ierr,msg,iam)
```

ARGUMENTS:

```
implicit none
integer, intent(in)          :: ierr, iam
character(len=*), intent(in) :: msg
```

DESCRIPTION:

This is an error check routine. Program exits in case of error with an associated error message written to the 'abort message' file.

2.21 Fortran: Module Interface constantsmod.F90 (Source File: constantsMod.F90)

The code in this file provides values of physical constants for consistent use across different components.

REVISION HISTORY:

```
    17 Feb 2004    Sujay Kumar  Initial Specification

!-----------------------------------------------------------------------------
! physical constants (all data public)
!-----------------------------------------------------------------------------
   public
!   real*8,parameter :: CONST_PI     = 3.14159265358979323846  ! pi
   real*8,parameter :: CONST_PI     = 3.14159265   ! pi
   real*8,parameter :: CONST_CDAY   = 86400.0      ! sec in calendar day ~ sec
   real*8,parameter :: CONST_SDAY   = 86164.0      ! sec in siderial day ~ sec
   real*8,parameter :: CONST_OMEGA  = 2.0*CONST_PI/CONST_SDAY ! earth rot ~ rad/sec
   real*8,parameter :: CONST_REARTH = 6.37122e6    ! radius of earth ~ m
   real*8,parameter :: CONST_G      = 9.80616      ! acceleration of gravity ~ m/s^2
   real*8,parameter :: CONST_PSTD   = 101325.0     ! standard pressure ~ pascals

   real*8,parameter :: CONST_STEBOL = 5.67e-8      ! Stefan-Boltzmann constant ~ W/m^2/K^4
   real*8,parameter :: CONST_BOLTZ  = 1.38065e-23  ! Boltzmann's constant ~ J/K/molecule
   real*8,parameter :: CONST_AVOGAD = 6.02214e26   ! Avogadro's number ~ molecules/kmole
   real*8,parameter :: CONST_RGAS   = CONST_AVOGAD*CONST_BOLTZ ! Universal gas constant ~ J/K/kmole
   real*8,parameter :: CONST_MWDAIR = 28.966       ! molecular weight dry air ~ kg/kmole
   real*8,parameter :: CONST_MWWV   = 18.016       ! molecular weight water vapor
   real*8,parameter :: CONST_RDAIR  = CONST_RGAS/CONST_MWDAIR  ! Dry air gas constant ~ J/K/kg
   real*8,parameter :: CONST_RWV    = CONST_RGAS/CONST_MWWV    ! Water vapor gas constant ~ J/K/kg
   real*8,parameter :: CONST_ZVIR   = (CONST_RWV/CONST_RDAIR)-1.0   ! RWV/RDAIR - 1.0
   real*8,parameter :: CONST_KARMAN = 0.4          ! Von Karman constant

   real*8,parameter :: CONST_TKFRZ  = 273.16       ! freezing T of fresh water ~ K (intentionally made =
   real*8,parameter :: CONST_TKTRIP = 273.16       ! triple point of fresh water ~ K

   real*8,parameter :: CONST_RHODAIR=CONST_PSTD/ &
     (CONST_RDAIR*CONST_TKFRZ)         ! density of dry air at STP   ~ kg/m^3
   real*8,parameter :: CONST_RHOFW  = 1.000e3      ! density of fresh water ~ kg/m^3
   real*8,parameter :: CONST_RHOSW  = 1.026e3      ! density of sea water ~ kg/m^3
   real*8,parameter :: CONST_RHOICE = 0.917e3      ! density of ice   ~ kg/m^3
   real*8,parameter :: CONST_CPDAIR = 1.00464e3    ! specific heat of dry air ~ J/kg/K
   real*8,parameter :: CONST_CPFW   = 4.188e3      ! specific heat of fresh h2o ~ J/kg/K
   real*8,parameter :: CONST_CPSW   = 3.996e3      ! specific heat of sea h2o ~ J/kg/K
   real*8,parameter :: CONST_CPWV   = 1.810e3      ! specific heat of water vap ~ J/kg/K
```

```
real*8,parameter :: CONST_CPICE  = 2.11727e3    ! specific heat of fresh ice ~ J/kg/K
real*8,parameter :: CONST_LATICE = 3.337e5      ! latent heat of fusion ~ J/kg
real*8,parameter :: CONST_LATVAP = 2.501e6      ! latent heat of evaporation ~ J/kg
real*8,parameter :: CONST_LATSUB = CONST_LATICE + CONST_LATVAP ! latent heat of sublimation ~ J/kg

real*8,parameter :: CONST_OCN_REF_SAL = 34.7    ! ocn ref salinity (psu)
real*8,parameter :: CONST_ICE_REF_SAL =  4.0    ! ice ref salinity (psu)
real*8,parameter :: CONST_SOLAR  =  1369.2    !Solar constant (W/m2)
```

2.22 Fortran: Module Interface drv_output_mod (Source File: drv_output_mod.F90)

The code in this file provides interfaces to manage LIS model output in different file formats.

2.22.1 Overview

The module provides generic interfaces to manage output from different land surface models. Currently LIS supports three data formats.

- binary: in binary, big-endian.
- grib: WMO grib1 format
- netcdf: NCAR unidata netcdf format

This module also provides generic interfaces for writing and reading model restart files. LIS uses the following convention on the filename extensions:

- .1gd4r: binary output, 1 correspondes to one variable, g represents gridded data, d represnts direct access, 4r represents 4 byte real
- .grb: files in grib1 format
- .nc: files in netcdf format
- *rst: restart files. Currently all restart files are written as binary format

This module also manages any data aggregatiion from computing nodes when parallel processing is employed.

REVISION HISTORY:

```
10 Feb 2004; Sujay Kumar; Initial Specification
```

USES:

```
#if ( defined USE_NETCDF )
  use netcdf
#endif
  use lis_logmod, only :lis_log_msg

#if (defined SPMD)
#if (defined USE_INCLUDE_MPI)
  implicit none
  include 'mpif.h'
#else
  use mpi
```

```
  implicit none
#endif
#endif
```

PUBLIC MEMBER FUNCTIONS:

```
  public :: drv_writevar_bin    ! write a variable into a binary file
  public :: drv_writevar_grib   ! write a variable into a grib file
  public :: drv_writevar_netcdf ! write a variable into a netcdf file
  public :: drv_writevar_restart ! writes a variable into a restart file
  public :: drv_readvar_restart ! reads a variable from a restart file
  public :: drv_readvar_gridded ! read a variable from a gridded file
  public :: drv_tile2grid       ! convert a tilespace variable to gridspace
  public :: drv_write_innov     ! writes innovations to a gridded file
  public :: drv_write_dastats   ! writes DA-related diagnostics to an ASCII file
```

2.22.2 drv_writevar_bin (Source File: drv_output_mod.F90)

INTERFACE:

```
interface drv_writevar_bin
```

PRIVATE MEMBER FUNCTIONS:

```
    module procedure writevar_bin_real
    module procedure writevar_bin_withstats_real
```

DESCRIPTION:

This interface provides routines for writing variables (integer or real) in a binary file. The aggregation from decomposed tile spaces in each processor to their respective grid spaces and the aggregations of these grid spaces from each processor are also performed by this routine. The interface also provides options to write some diagnostic statistics about the variable being written.

2.22.3 drv_writevar_restart (Source File: drv_output_mod.F90)

INTERFACE:

```
interface drv_writevar_restart
```

PRIVATE MEMBER FUNCTIONS:

```
    module procedure writevar_restart_int
    module procedure writevar_restart_real
    module procedure writevar_restart_char
```

DESCRIPTION:

This interface provides routines for writing variables (integer or real) in a restart file. By definition, the restart files are written in the model tile space. The aggregations from tile spaces of each processors are also performed by this routine.

2.22.4 drv_readvar_restart (Source File: drv_output_mod.F90)

INTERFACE:

```
interface drv_readvar_restart
```

PRIVATE MEMBER FUNCTIONS:

```
module procedure readvar_restart_int
module procedure readvar_restart_real
module procedure readvar_restart_char
```

DESCRIPTION:

This interface provides routines for reading variables (integer or real) from a restart file. The restart files are read by the master processor. The domain decomposition of the "global" tile space on the master processor to individual processors are also performed by this routine.

2.22.5 drv_readvar_gridded (Source File: drv_output_mod.F90)

INTERFACE:

```
interface drv_readvar_gridded
```

PRIVATE MEMBER FUNCTIONS:

```
module procedure readvar_gridded_real
```

DESCRIPTION:

This interface provides routines for reading variables (real) from a gridded (output) file into a tile space. The gridded files are read by the master processor. The domain decomposition of the "global" grid space on the master processor to individual processors are also performed by this routine.

2.22.6 drv_writevar_grib (Source File: drv_output_mod.F90)

INTERFACE:

```
interface drv_writevar_grib
```

PRIVATE MEMBER FUNCTIONS:

```
module procedure writevar_grib_real
module procedure writevar_grib_withstats_real
module procedure writevar_grib_withstats_real3d
```

DESCRIPTION:

This interface provides routines for writing variables (integer or real) in a grib file. The aggregation from decomposed tile spaces in each processor to their respective grid spaces and the aggregations of these grid spaces from each processor are also performed by this routine. The interface also provides options to write some diagnostic statistics about the variable being written.

2.22.7 drv_writevar_netcdf (Source File: drv_output_mod.F90)

INTERFACE:

```
interface drv_writevar_netcdf
```

PRIVATE MEMBER FUNCTIONS:

```
module procedure writevar_netcdf_real
module procedure writevar_netcdf_real3d
module procedure writevar_netcdf_withstats_real
```

DESCRIPTION:

This interface provides routines for writing variables (2d or 3d) in a netcdf file. The aggregation from decomposed tile spaces in each processor to their respective grid spaces and the aggregations of these grid spaces from each processor are also performed by this routine.

2.22.8 writevar_bin_withstats_real (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call using drv_writevar_bin
subroutine writevar_bin_withstats_real(ftn,ftn_stats,n,flag,var,mvar,form)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod,       only : iam, masterproc, gdeltas, goffsets,&
     r_str,c_str,r_end,c_end, npes, tdeltas, toffsets
use lis_logmod, only : lis_flush

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn
integer, intent(in) :: ftn_stats
real, intent(in)    :: var(lis%nch(n))
character (len=*)   :: mvar
integer, intent(in) :: form
integer, intent(in) :: flag
```

DESCRIPTION:

Write a real variable to a binary output file with some diagnostic statistics written to a text file.
The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

ftn_stats unit number of the ASCII text statistics file

var variables being written, dimensioned in the tile space

mvar name of the variable being written (will be used in the stats file)

form format to be used in the stats file (1-decimal format, 2-scientific format)

flag option to determine if the variable needs to be written (1-write, 0-do not write)

The routines invoked are:

stats (2.29.4)
call to compute the diagnostic statistics

2.22.9 writevar_bin_real (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call using drv_writevar_bin
subroutine writevar_bin_real(ftn, n, var)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod,       only : iam, masterproc,gdeltas,goffsets,&
     r_str,c_str,r_end,c_end,npes, tdeltas, toffsets

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn
real, intent(in)    :: var(lis%nch(n))
```

DESCRIPTION:

Write a real variable to a binary output file.
The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

var variables being written, dimensioned in the tile space

2.22.10 writevar_restart_int (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call using drv_writevar_restart
subroutine writevar_restart_int(ftn, n, var)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod,       only : masterproc, iam, tdeltas,toffsets,&
     r_str,r_end,c_str,c_end,npes

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: ftn
integer, intent(in) :: n
integer, intent(in) :: var(lis%nch(n))
```

DESCRIPTION:

Writes an integer variable to a binary restart file.
The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

var variables being written, dimensioned in the tile space

2.22.11 writevar_restart_real (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call using drv_writevar_restart
subroutine writevar_restart_real(ftn, n, var)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod,       only : masterproc,iam, toffsets,tdeltas,&
     r_str,r_end,c_str,c_end,npes

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: ftn
integer, intent(in) :: n
real, intent(in)    :: var(lis%nch(n))
```

DESCRIPTION:

Writes a real variable to a binary restart file.
The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

var variables being written, dimensioned in the tile space

2.22.12 writevar_restart_char (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call using drv_writevar_restart
subroutine writevar_restart_char(ftn, n, var)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod,       only : masterproc,iam, toffsets,tdeltas,&
     r_str,r_end,c_str,c_end,npes

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: ftn
integer, intent(in) :: n
character*1, intent(in)    :: var(lis%nch(n))
```

DESCRIPTION:

Writes a character variable to a binary restart file.
The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

var variables being written, dimensioned in the tile space

2.22.13 readvar_restart_int (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call using drv_readvar_restart
subroutine readvar_restart_int(ftn, n, var)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod,       only : r_str,r_end,c_str,c_end,iam

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: ftn
integer, intent(in)    :: n
integer, intent(inout) :: var(lis%nch(n))
```

DESCRIPTION:

Reads an integer variable from a binary restart file.
The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

var variables being written, dimensioned in the tile space

2.22.14 readvar_restart_real (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call drv_readvar_restart
subroutine readvar_restart_real(ftn, n, var)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod,       only : r_str,r_end,c_str,c_end,iam

implicit none
```

ARGUMENTS:

```
integer, intent(in)   :: ftn
integer, intent(in)   :: n
real, intent(inout)   :: var(lis%nch(n))
```

DESCRIPTION:

Reads a real variable from a binary restart file.
The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

var variables being written, dimensioned in the tile space

2.22.15 readvar_restart_char (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call using drv_readvar_restart
subroutine readvar_restart_char(ftn, n, var)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod,       only : r_str,r_end,c_str,c_end,iam

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: ftn
integer, intent(in)    :: n
character*1, intent(inout) :: var(lis%nch(n))
```

DESCRIPTION:

Reads an character variable from a binary restart file.
The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

var variables being written, dimensioned in the tile space

2.22.16 readvar_gridded_real (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call drv_readvar_gridded
subroutine readvar_gridded_real(ftn, n, var)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod,       only : iam, r_str,r_end,c_str,c_end

implicit none
```

ARGUMENTS:

```
integer, intent(in)   :: ftn
integer, intent(in)   :: n
real, intent(inout)   :: var(lis%ngrid(n))
```

DESCRIPTION:

Reads a real variable from a binary sequential access, gridded file. After reading the global data, the routine subroutine subsets the data for each processor's domain.
The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

var variables being written, dimensioned in the tile space

2.22.17 writevar_netcdf_real (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call using drv_writevar_netcdf
subroutine writevar_netcdf_real(ftn,n, var,dim1,varid)
```

USES:

```
use lisdrv_module, only : lis

implicit none
```

ARGUMENTS:

```
integer,intent(in) :: ftn
integer,intent(in) :: n
integer,intent(in) :: dim1
integer            :: varid
real, intent(in)   :: var(lis%glbnch(n))
```

DESCRIPTION:

Writes a 2-D variable to the netcdf file
The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

dim1 index of the data write in the netcdf file (typically the time index)

varid netcdf id for the variable being written.

var variables being written, dimensioned in the tile space

2.22.18 writevar_netcdf_real3d (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call using drv_writevar_netcdf
subroutine writevar_netcdf_real3d(ftn,n, var,dim1,dim2,varid)
```

USES:

```
    use lisdrv_module, only : lis
#if ( defined USE_NETCDF )
    use netcdf
#endif
    implicit none
```

ARGUMENTS:

```
    integer, intent(in) :: ftn
    integer, intent(in) :: dim1
    integer, intent(in) :: n
    integer, intent(in) :: dim2
    integer, intent(in) :: varid
    real, intent(in)   :: var(lis%glbnch(n))
```

DESCRIPTION:

Writes a 3-D variable to the netcdf file
The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

dim1 first index of the data write in the netcdf file (typically the time index)

dim2 second index of the data write in the netcdf file (typically the time index)

varid netcdf id for the variable being written.

var variables being written, dimensioned in the tile space

2.22.19 writevar_netcdf_withstats_real (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call using drv_writevar_netcdf
subroutine writevar_netcdf_withstats_real(ftn, ftn_stats, n, flag, &
                                          var, dim1, varid, form)
```

USES:

```
    use lisdrv_module, only : lis,lisdom
    use spmdMod,       only : iam, masterproc, gdeltas, goffsets,&
         c_str,c_end,r_str,r_end,npes
#if ( defined USE_NETCDF )
    use netcdf
#endif

    implicit none
```

ARGUMENTS:

```
    integer, intent(in) :: n
    integer, intent(in) :: ftn,ftn_stats
    real, intent(in)    :: var(lis%nch(n))
    integer, intent(in) :: dim1
    integer, intent(in) :: varid
    integer, intent(in) :: form
    integer, intent(in) :: flag
```

DESCRIPTION:

Write a real variable to a netcdf output file with some diagnostic statistics written to a text file. The arguments are:

ftn unit number of the binary output file

ftn_stats unit number of the ASCII text statistics file

n index of the domain or nest.

flag option to determine if the variable needs to be written (1-write, 0-do not write)

var variables being written, dimensioned in the tile space

dim1 first index of the data write in the netcdf file (typically the time index)

varid netcdf id for the variable being written.

form format to be used in the stats file (1-decimal format, 2-scientific format)

2.22.20 writevar_grib_real (Source File: drv_output_mod.F90)

INTERFACE:

```
subroutine writevar_grib_real(ftn, n,var, kpds, writeint,&
     today, yesterday)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod,       only : iam, masterproc,gdeltas,goffsets,&
     c_str,c_end,r_str,r_end,npes
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: ftn
integer, intent(in)      :: n
real, intent(in)         :: var(lis%nch(n))
integer, intent(in)      :: kpds(25)
real, intent(in)         :: writeint
character*8,intent(in)   :: today
character*8, intent(in)  :: yesterday
```

DESCRIPTION:

Writes a real variable to a grib file
The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

var variables being written, dimensioned in the tile space

The routines invoked are:

makepdsn (2.27.6)
computes the PDS array for the variable being written

2.22.21 writevar_grib_withstats_real (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call using drv_writevar_grib
subroutine writevar_grib_withstats_real(ftn, ftn_stats, n,                &
                                        flag, var, mvar, form,            &
                                        toplev, botlev, kdim)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod,       only : npes, iam, masterproc, goffsets, gdeltas,&
     c_str,c_end,r_str,r_end
use lis_logmod, only : logunit
use grib_module
implicit none
```

ARGUMENTS:

```
integer, intent(in)         :: ftn
integer, intent(in)         :: ftn_stats
integer, intent(in)         :: n
integer, intent(in)         :: flag
real, intent(in)            :: var(lis%nch(n))
character(len=*),intent(in) :: mvar
integer, intent(in)         :: form

integer                     :: kdim
real,    intent(in)         :: toplev(kdim), botlev(kdim)
```

DESCRIPTION:

Write a real variable to a grib output file with some diagnostic statistics written to a text file. The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

ftn_stats unit number of the ASCII text statistics file

var variables being written, dimensioned in the tile space

mvar name of the variable being written (will be used in the stats file)

flag option to determine if the variable needs to be written (1-write, 0-do not write)

form format to be used in the stats file (1-decimal format, 2-scientific format)

info Array containing the sizes of the various grib sections

sect1 Array for storing the contents of Section 1.

s1opt Array for storing optional Section 1 information (not used)

sect2 Array for storing the contents of Section 2.

sect3 Array for storing the contents of Section 3.

toplev Array of the heights of the top of each layer

botlev Array of the heights of the bottom of each layer

kdim Grid dimension k-direction (number of levels/layers)

size_3 Size of the sect3 array

The routines invoked are:

gribdat (2.24.6)
calls the low-level grib routines contained in the grib library

stats (2.29.4)
call to compute diagnostic statistics of a variable

2.22.22 writevar_grib_withstats_real3d (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call using drv_writevar_grib
subroutine writevar_grib_withstats_real3d(ftn, ftn_stats, n,                &
                                          flag, var, mvar, form,            &
                                          toplev, botlev, kdim)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod,       only : iam, masterproc, gdeltas, goffsets,&
     c_str,c_end,r_str,r_end,npes
use lis_logmod, only : logunit
use grib_module
implicit none
```

ARGUMENTS:

```
integer                     :: kdim

integer, intent(in)         :: ftn
integer, intent(in)         :: ftn_stats
integer, intent(in)         :: n
integer, intent(in)         :: flag
real, intent(in)            :: var(lis%nch(n),kdim)
character(len=*),intent(in) :: mvar
integer, intent(in)         :: form
real,    intent(in)         :: toplev(kdim), botlev(kdim)
```

DESCRIPTION:

Write a real variable to a grib output file with some diagnostic statistics written to a text file. The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

ftn_stats unit number of the ASCII text statistics file

var variables being written, dimensioned in the tile space

mvar name of the variable being written (will be used in the stats file)

flag option to determine if the variable needs to be written (1-write, 0-do not write)

form format to be used in the stats file (1-decimal format, 2-scientific format)

info Array containing the sizes of the various grib sections

sect1 Array for storing the contents of Section 1.

s1opt Array for storing optional Section 1 information (not used)

sect2 Array for storing the contents of Section 2.

sect3 Array for storing the contents of Section 3.

toplev Array of the heights of the top of each layer

botlev Array of the heights of the bottom of each layer

kdim Grid dimension k-direction (number of levels/layers)

size_3 Size of the sect3 array

The routines invoked are:

gribdat (2.24.6)
calls the low-level grib routines contained in the grib library

stats (2.29.4)
call to compute diagnostic statistics of a variable

2.22.23 drv_tile2grid (Source File: drv_output_mod.F90)

INTERFACE:

```
subroutine drv_tile2grid(n,gvar,tvar)
```

USES:

```
use lisdrv_module,      only : lis,lisdom

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real              :: gvar(lis%lnc(n),lis%lnr(n))
real, intent(in)  :: tvar(lis%nch(n))
```

DESCRIPTION:

This routine converts a tile space variable to the corresponding grid space. The aggregation involves weighted average of each tile in a grid cell based on the vegetation distribution.
The arguments are:

n index of the domain or nest.

tvar variable dimensioned in the tile space.

gvar variable after converstion to the grid space

2.22.24 drv_write_innov (Source File: drv_output_mod.F90)

INTERFACE:

```
subroutine drv_write_innov(ftn, n, var)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod,       only : iam, masterproc, odeltas, ooffsets, &
     c_str,c_end,r_str,r_end,npes

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn
real, intent(in)    :: var(lis%ngrid(n)*lis%nobtypes(n))
```

DESCRIPTION:

Writes the innovations to a binary file in a gridded format.
The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

var variables being written, dimensioned in the observation space

---

2.22.25 drv_write_dastats (Source File: drv_output_mod.F90)

INTERFACE:

```
subroutine drv_write_dastats(ftn,n,var)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod,       only : iam, masterproc, tdeltas, toffsets,&
     c_str,c_end,r_str,r_end,npes

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn
real, intent(in)    :: var(lis%nch(n))
```

DESCRIPTION:

Write a real variable to a ASCII text output file with DA-related diagnostic statistics written to a text file.
The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

ftn_stats unit number of the ASCII text statistics file

var variables being written, dimensioned in the tile space

mvar name of the variable being written (will be used in the stats file)

form format to be used in the stats file (1-decimal format, 2-scientific format)

range range of variables to be written

The routines invoked are:

stats_da (2.29.5)
call to compute the diagnostic statistics

2.22.26 endrun (Source File: endrun.F90)

REVISION HISTORY:

```
    14 Nov 2002    Sujay Kumar  Initial Specification
```

INTERFACE:

```
 subroutine endrun
```

USES:

```
   use lis_logmod, only : lis_flush, logunit
 #if (defined SPMD)
   use mpishorthand, only: MPI_COMM_WORLD
 #endif
   implicit none
```

DESCRIPTION:

Routine to be called to terminate the program. This routines flushes the output streams and aborts the mpi processes.
The routines invoked are:

lis_flush (2.27.5)
flushes the output streams

2.23 Fortran: Module Interface gswp_module (Source File: gswp_module.F90)

This module contains useful routines that generates indices for reading the GSWP netcdf data.

REVISION HISTORY:

```
    24 Feb 2004    Sujay Kumar  Initial Specification

  ------------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
  ------------------------------------------------------------------------------
   public :: getgswp_monindex
   public :: getgswp_timeindex
```

2.23.1 getgswp_monindex (Source File: gswp_module.F90)

INTERFACE:

```
subroutine getgswp_monindex(yr,mo,index)
```

ARGUMENTS:

```
integer, intent(out) :: index
integer, intent(in) :: yr, mo
```

DESCRIPTION:

This routine computes the GSWP monthly index (base time being 1982), to be used to read monthly data.

2.23.2 getgswp_timeindex (Source File: gswp_module.F90)

INTERFACE:

```
subroutine getgswp_timeindex(yr,mo,da,hr,mn,ss,index)

  implicit none
```

ARGUMENTS:

```
integer, intent(in)  :: yr, mo, da, hr, mn, ss
integer, intent(out) :: index
```

DESCRIPTION:

This routine computes the GSWP timestep index (base time being 1982), to be used to read the forcing files. Note: GSWP forcing data are written into monthly files with a frequency of 3 hours. The first entry corresponds to 03:00:00 of the first day of the given month. The last entry corresponds to 00:00:00 of the first day of the next month.
E.g.; for Tair_cru198207.nc, the data run from 1982-07-01T03:00:00 through 1982-08-01T00:00:00, inclusive. So, when you are at hour 0 on the first day of the month, reset the day to the last day of the previous month, and reset the hour from 0 to 24. This will compute the correct index for the last entry in the forcing file.
E.g.; 1982-08-01T00:00:00 should be re-written as 1982-07-31T24:00:00.

2.24 Fortran: Module Interface grib_module (Source File: grib_module.F90)

The code in this file sets up various grib related variables used for writing grib output.

REVISION HISTORY:

```
15 May 2006: James Geiger; Initial code
15 Dec 2006: Charles Alonge; Updated parameter ids, added
             support for multiple grid output, and code cleanup
 4 Apr 2007: Sujay Kumar; Added a derived data type and support
             for parallel runs
```

2.24.1 grib_init (Source File: grib_module.F90)

INTERFACE:

```
subroutine grib_init()
```

USES:

```
use mpishorthand
use lisdrv_module, only : lis, lisdom
use spmdMod,       only : masterproc, iam, gdeltas, goffsets,&
     c_str,r_str,c_end,r_end,npes
```

DESCRIPTION:

Initializes the grib_pds_vals array. This array contains the variable-dependent parameters for Section 1 (PDS) of the GRIB record. This routine also initializes the landmask. The landmask is then used create a logical bitmap for use in the GRIB records.

2.24.2 grib1_finalize (Source File: grib_module.F90)

INTERFACE:

```
subroutine grib1_finalize(index, sect1, time_unit, time_P1, time_P2, &
                          time_range)
```

DESCRIPTION:

This routine completes the definition of the sect1 array. It sets the variable-dependent parameters for Section 1 (PDS) of the GRIB record.
The arguments are:

index Index into the grib_pds_vals array corresponding to the variable being processed

sect1 Parameters for Section 1 (PDS) of the GRIB record

time_unit Units in time for output contained in GRIB record (See GRIB Table 4)

time_P1 Time1 for GRIB record descriptor (See GRIB Table 5)

time_P2 Time2 for GRIB record descriptor (See GRIB Table 5)

time_range Time range descriptor for P1 and P2 (See GRIB Table 5)

2.24.3 grib1_setup (Source File: grib_module.F90)

INTERFACE:

```
subroutine grib1_setup (sect1, info, bitmap, date10_12)
```

DESCRIPTION:

This routine initializes the sect1 array. It sets the variable-independent parameters for Section 1 (PDS) of the GRIB record.
The arguments are:

sect1 Parameters for Section 1 (PDS) of the GRIB record

info Sizes of grib sections

bitmap Logical varible indicating if a bitmap is used in grib

date10_12 10 digit date/time group of daily cycle – yyyymmddhh

---

2.24.4 grib2_setup (Source File: grib_module.F90)

INTERFACE:

```
subroutine grib2_setup (sect2, info, imax, jmax, n)
```

DESCRIPTION:

This routine initializes the sect2 array. It sets the parameters for Section 2 (GDS) of the GRIB record. The arguments are:

sect2 Parameters for Section 2 (GDS) of the GRIB record

info Sizes of grib sections

imax Number of points in the i-direction

jmax Number of points in the j-direction

n Number of LIS nest

2.24.5 grib3_setup (Source File: grib_module.F90)

INTERFACE:

```
subroutine grib3_setup (sect3, imax, jmax, numpts, &
                        info, bitmap, size_3, numoct_3, &
                        land )
```

DESCRIPTION:

This routine initializes the sect3 array. It sets the parameters for Section 3 (BMS) of the GRIB record. The arguments are:

sect3 Parameters for Section 3 (BMS) of the GRIB record

imax Number of points in the i-direction

jmax Number of points in the j-direction

numpts Number of land points

info Sizes of grib sections

bitmap Logical varible indicating if a bitmap is used in grib

size_3 Number of elements in the sect3 array.

numoct_3 Number of octets in the sect3 array.

land Input land mask (1-Land, 0-Water) to create bitmap.

2.24.6 gribdat (Source File: grib_module.F90)

INTERFACE:

```
SUBROUTINE GRIBDAT (INFO, SECT1, S1OPT, SECT2, SECT3, DUM3D, &
                    TOPLEV, BOTLEV, IDIM, JDIM, KDIM, SIZE_3)
```

DESCRIPTION:

This routine calls the low-level grib routines, contained in the grib library, to pack the grib record corresponding to the variable contained in dum3d. Then it writes the grib record.
The arguments are:

info Sizes of grib sections

sect1 Parameters for Section 1 (PDS) of the GRIB record

s1opt Optional parameters for Section 1 (PDS) of the GRIB record

sect2 Parameters for Section 2 (GDS) of the GRIB record

sect3 Parameters for Section 3 (BMS) of the GRIB record

dum3d Array of data to be processed

toplev Array of depths for the top of each layer

botlev Array of depths for the bottom of each layer

idim Number of points in the i-direction

jdim Number of points in the j-direction

kdim Number of layers

size_3 Number of elements in the sect3 array.

2.24.7 lapseRateCorrection (Source File: lapseRateCorrection.F90)

REVISION HISTORY:

```
 11 Apr 2000: Brian Cosgrove; Initial Code
 12 May 2000: Brian Cosgrove; Corrected for zero humidities
 25 Jan 2001: Matt Rodell; Compute number of input and output
grid points, use to allocate local arrays
 15 Mar 2001: Jon Gottschalck; if-then to handle negative vapor
pressures in long wave correction
 14 Nov 2003: Sujay Kumar; Adopted in LIS
```

INTERFACE:

```
subroutine lapseRateCorrection(nest)
```

USES:

```
use ESMF_Mod
use constantsMod,  only: CONST_G,CONST_TKFRZ
use lisdrv_module, only: lis,lisdom
use baseforcing_module, only : lisforc,FORC_State
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: nest
```

DESCRIPTION:

Corrects Temperature, Pressure, Humidity and Longwave Radiation forcing values for differences in elevation between the LIS running grid and the native forcing grid. The corrections are based on the lapse-rate and hypsometric adjustments to these variables described in Cosgrove et. al (2003).
Cosgrove, B.A. et.al, Real-time and retrospective forcing in the North American Land Data Assimilation (NLDAS) project, Journal of Geophysical Research, 108(D22), 8842, DOI: 10.1029/2002JD003118, 2003.
The arguments are:

nest index of the domain or nest.

2.25 Fortran: Module Interface lis_fileIOMod (Source File: lis_fileIOMod.F90)

This module contains a number of routines useful for various file I/O operations in LIS. The module provides routines to create output directories, filenames, that can be used in the model output routines.

REVISION HISTORY:

```
08 Apr 2004    James Geiger Initial Specification
```

USES:

```
implicit none
-------------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
-------------------------------------------------------------------------------
public :: create_output_filename   ! create an output filename
public :: create_output_directory  ! create the output directory
public :: create_restart_filename  ! create a restart filename
public :: create_stats_filename    ! create a stats filename
public :: create_innov_filename    ! create an innovations filename
public :: create_daspread_filename ! create an innovations filename
public :: putget                   !a generic read/write method.
```

2.25.1 putget (Source File: lis_fileIOMod.F90)

INTERFACE:

```
interface putget
```

PRIVATE MEMBER FUNCTIONS:

```
   module procedure putget_int
   module procedure putget_real
```

DESCRIPTION:

This is a generic method to read from or write to direct access files.
method:
- open file
- if iofunc = r, read buffer from file, abort on error
- if iofunc = w, write buffer to file, abort on error
- if iofunc = anything else, abort

REVISION HISTORY:

```
04 aug 1999 initial version ..........................mr gayno/dnxm
02 nov 2005 incorporated into LIS                      sujay kumar
```

2.25.2 create_output_directory (Source File: lis_fileIOMod.F90)

INTERFACE:

```
subroutine create_output_directory(mname,dir_name,style)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only : lis_log_msg
implicit none
```

ARGUMENTS:

```
character(len=*)  :: mname
character(len=40), optional   :: dir_name
integer, intent(IN), optional :: style
```

DESCRIPTION:

Create the output directory for the output data files. The call creates a hierarchy of directories in the following format, if the directory name is not specified.
style option 1:

```
<output directory>/EXP<expno>/<model name>/<yr>/<yrmoda>
```

style option 2:

```
<output directory>/EXP<expno>/<model name>
```

Once the directory name is created, the subroutine issues a system call to create the structure.
The arguments are:

mname a string describing the name of the model

dir_name name of the directory to override the above format

2.25.3 create_output_filename (Source File: lis_fileIOMod.F90)

INTERFACE:

```
subroutine create_output_filename(n, fname, model_name, writeint, wout, style)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only : lis_log_msg

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
character(len=*), intent(out)          :: fname
character(len=*), intent(in), optional :: model_name ! needed for gswp run
integer, intent(in), optional          :: writeint ! output writing interval
integer, intent(in), optional          :: wout ! output format
integer, intent(in), optional          :: style ! output directory style
```

DESCRIPTION:

Create the file name for the output data files. It creates both the GSWP style of output filenames and the standard LIS style. The convention used in LIS creates a filename in the following default format (style==1) ¡output directory¿/EXP¡expno¿/¡model name¿/¡yr¿/¡yrmoda¿/¡yrmodahrmn¿.¡extension¿ Style option ==2 corresponds to the following style:
¡output directory¿/EXP¡expno¿/¡model name¿.¡yrmodahrmn¿.¡extension¿ Style option ==3 : ¡output directory¿/EXP¡expno¿/¡model name¿.¡extension¿
The arguments are:

n index of the domain or nest

fname the created file name.

model_name string describing the name of the model

writeint output writing interval of the model

wout output writing format

style style option as described above

2.25.4 create_restart_filename (Source File: lis_fileIOMod.F90)

INTERFACE:

```
subroutine create_restart_filename(n, fname,model_name,extn)
```

USES:

```
use lisdrv_module, only : lis

implicit none
```

ARGUMENTS:

```
integer, intent(in)           :: n
character(len=*), intent(in)  :: model_name ! needed for gswp run
character(len=*), intent(out) :: fname
character(len=*), intent(in)  :: extn
```

DESCRIPTION:

Create the file name for the restart data files. The convention used in LIS creates a restart filename in the following format.
¡output directory¿/EXP¡expno¿/¡model name¿/¡yr¿/¡yrmoda¿/¡yrmodahrmn¿.¡extn¿
The arguments are:

n index of the domain or nest

fname the created file name.

model_name string describing the name of the model

extn string extension to be used in the filename

2.25.5 create_stats_filename (Source File: lis_fileIOMod.F90)

INTERFACE:

```
subroutine create_stats_filename(n, fname, mname)
```

USES:

```
use lisdrv_module, only : lis

implicit none
```

ARGUMENTS:

```
integer, intent(in)           :: n
character(len=*), intent(out) :: fname
character(len=*), intent(in)  :: mname
```

DESCRIPTION:

Create the file name for the stats files. The convention used in LIS creates a restart filename in the following format.
¡output directory¿/EXP¡expno¿/¡model name¿stats.dat
The arguments are:

n index of the domain or nest

fname the created file name.

mname string describing the name of the model

2.25.6 create_innov_filename (Source File: lis_fileIOMod.F90)

INTERFACE:

```
subroutine create_innov_filename(n, fname, mname)
```

USES:

```
use lisdrv_module, only : lis

implicit none
```

ARGUMENTS:

```
integer, intent(in)           :: n
character(len=*), intent(out) :: fname
character(len=*), intent(in)  :: mname
```

DESCRIPTION:

Create the file name for the innovations files. The convention used in LIS creates a restart filename in the following format.
¡output directory¿/EXP¡expno¿/¡model name¿¡timestamp¿.innov
The arguments are:

n index of the domain or nest

fname the created file name.

mname string describing the name of the model

2.25.7 create_daspread_filename (Source File: lis_fileIOMod.F90)

INTERFACE:

```
subroutine create_daspread_filename(n, fname, vname)
```

USES:

```
use lisdrv_module, only : lis

implicit none
```

ARGUMENTS:

```
integer, intent(in)           :: n
character(len=*), intent(out) :: fname
character(len=*), intent(in)  :: vname
```

DESCRIPTION:

Create the file name for the data assimilation diagnostic - the ensemble spread. The convention used in LIS creates a restart filename in the following format.
¡output directory¿/EXP¡expno¿/¡model name¿¡timestamp¿.spread
The arguments are:

n index of the domain or nest

fname the created file name.

mname string describing the name of the model

2.25.8 putget_int (Source File: lis_fileIOMod.F90)

INTERFACE:

```
!Private name: call using putget()
  subroutine putget_int ( buffer, iofunc, file_name, &
       routine_name, imax, jmax )
```

USES:

```
use lis_logmod, only : lis_abort
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer,       intent(in)     :: imax
integer,       intent(in)     :: jmax
integer,       intent(inout)  :: buffer (imax , jmax)
character*1,   intent(in)     :: iofunc
character*100, intent(in)     :: file_name
character*10,  intent(in)     :: routine_name
```

DESCRIPTION:

put or get any size integer array routine
The arguments are:

buffer integer buffer array

iofunc I/O function (r=read, w=write)

file_name input file name

routine_name name of the calling routine

imax grid dimension of the buffer array - East/West direction

jmax grid dimension of the buffer array - North/South direction

The methods invoked are:

lis_abort (2.27.3)
abort the program in case of a fatal error

2.25.9 putget_real (Source File: lis_fileIOMod.F90)

INTERFACE:

```
!Private name: call using putget()
  subroutine putget_real ( buffer, iofunc, file_name, &
       routine_name, imax, jmax )
```

USES:

```
use lis_logmod, only : lis_abort
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber
implicit none

integer,       intent(in)     :: imax
integer,       intent(in)     :: jmax
real,         intent(inout)   :: buffer      (imax , jmax)
character*100, intent(in)     :: file_name
character*1,   intent(in)     :: iofunc
character*10,  intent(in)     :: routine_name
```

DESCRIPTION:

put or get any size real array routine
The arguments are:

buffer integer buffer array

iofunc I/O function (r=read, w=write)

file_name input file name

routine_name name of the calling routine

imax grid dimension of the buffer array - East/West direction

jmax grid dimension of the buffer array - North/South direction

The methods invoked are:

lis_abort (2.27.3)
abort the program in case of a fatal error

2.25.10 create_gain_filename (Source File: lis_fileIOMod.F90)

INTERFACE:

```
subroutine create_gain_filename(n, fname, mname)
```

USES:

```
use lisdrv_module, only : lis

implicit none
```

ARGUMENTS:

```
integer, intent(in)           :: n
character(len=*), intent(out) :: fname
character(len=*), intent(in)  :: mname
```

DESCRIPTION:

Create the file name for the gainations files. The convention used in LIS creates a restart filename in the following format.
¡output directory¿/EXP¡expno¿/¡model name¿¡timestamp¿.gain
The arguments are:

n index of the domain or nest

fname the created file name.

mname string describing the name of the model

2.26 Fortran: Module Interface lis_fileUnitsMod (Source File: lis_fileUnitsMod.F90)

This module contains some utilties to handle the use of unit numbers used in the routines in LIS

REVISION HISTORY:

```
27 Sep 2007    Sujay Kumar; Initial Specification
```

USES:

```
 implicit none
-------------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
-------------------------------------------------------------------------------
 public :: getNextUnitNumber   !get the next available unit number
 public :: releaseUnitNumber   !release the unit number!
```

2.26.1 getNextUnitNumber (Source File: lis_fileUnitsMod.F90)

INTERFACE:

```
integer function getNextUnitNumber()
```

DESCRIPTION:

get next available Fortran unit number
Method: Get next available Fortran unit number itst. This routine is modified from the CCSM codes

---

2.26.2 releaseUnitNumber (Source File: lis_fileUnitsMod.F90)

INTERFACE:

```
subroutine releaseUnitNumber(iunit)
```

DESCRIPTION:

close and release Fortran unit no longer in use

2.27 Fortran: Module Interface lis_logmod (Source File: lis_logmod.F90)

The code in this file provides routines for diagnostic and error logging in LIS. A file unit number is reserved for the diagnostic log file. When a multiprocessor run is conducted, a diagnostic file from each processor is written to a separate file. An entry to the diagnostic log can be made by using:

```
write(logunit,*) msg
```

REVISION HISTORY:

```
 12 Mar 2004: James Geiger; Initial version
-------------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
-------------------------------------------------------------------------------
 public :: lis_log_init ! initializes LIS logging.
 public :: lis_log_msg ! writes a time-processor stamped message
                       ! to the diagnostic log
 public :: lis_abort  ! generates a standard abort message
 public :: lis_alert  ! generates a standard alert message
 public :: lis_flush ! flushes any unwrittend writes to the log
-------------------------------------------------------------------------------
```

PUBLIC TYPES:

```
-------------------------------------------------------------------------------
 public :: logunit ! file unit number used for diagnostic logging
```

2.27.1 lis_log_init (Source File: lis_logmod.F90)

INTERFACE:

```
subroutine lis_log_init(iunit)
  integer, intent(IN) :: iunit
```

DESCRIPTION:

Initializes the lis log management. Reserves the assigned file unit number for lis diagnostic logging.

2.27.2 lis_log_msg (Source File: lis_logmod.F90)

INTERFACE:

```
subroutine lis_log_msg(msg)
```

USES:

```
use spmdMod, only : iam,masterproc

implicit none
```

ARGUMENTS:

```
character(len=*), intent(in) :: msg
```

DESCRIPTION:

This routine formats a given message by prepending a time stamp and appending the process id number. This newly formatted message is then written to standard out.
The arguments are:

msg message to be written to the logfile

2.27.3 lis_abort (Source File: lis_logmod.F90)

REVISION HISTORY:

```
03 mar 1999 initial version ........................mr moore/dnxm
09 aug 1999 ported to IBM SP-2. increase size of abort message
            to 100 characters. removed stop 255 with a call
            to IBM utility "abort"..................mr gayno/dnxm
29 oct 2005  Sujay Kumar, Adopted in LIS
```

INTERFACE:

```
subroutine lis_abort( abort_message )

  use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

  implicit none

  character*100              :: abort_message(20)
```

DESCRIPTION:

to generate standard abort message and abort calling program.

- open abort file, write abort message and then call the system abort routine. this will cause program termination and cause script calling this routine to abort.
- if there is an error opening or writing abort file, send message to unit 6 and call the system abort routine

The arguments are:

abort_message abort message

2.27.4 lis_alert (Source File: lis_logmod.F90)

REVISION HISTORY:

```
03 mar 1999 initial version ........................mr moore/dnxm
09 aug 1999 ported to IBM SP-2.  increase size of message to
            100 characters.  added intent attributes to
            arguments..............................mr gayno/dnxm
29 oct 2005  Sujay Kumar, Adopted in LIS
```

INTERFACE:

```
subroutine lis_alert( program_name, alert_number, message )
```

USES:

```
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber
```

2.27.5 lis_flush (Source File: lis_logmod.F90)

REVISION HISTORY:

```
16 Nov 2004    James Geiger Initial Specification
```

```
subroutine lis_flush(unit)

  implicit none

  integer :: unit
```

DESCRIPTION:

This routine is a generic interface to the system flush routine.
The arguments are:

unit unit to be flushed out

2.27.6 makepdsn (Source File: makepdsn.F90)

REVISION HISTORY:

```
19 Mar 2002  Cosgrove; Initial specification
```

INTERFACE:

```
subroutine makepdsn(yesterday, beforeyester, kpds, hour, writeint)

  implicit none
```

ARGUMENTS:

```
character*8 :: yesterday, beforeyester
integer     :: kpds(25), hour
real        :: writeint
```

DESCRIPTION:

This routine computes the coefficients and terms required for grib output
The arguments are:

yesterday character representation of previous time

beforeyester character representation of uber previous time

kpds grib description array

hour current hour

writeint output writing interval in seconds

2.27.7 microMetCorrection (Source File: microMetCorrection.F90)

REVISION HISTORY:

```
21 Dec 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine microMetCorrection(nest)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis,lisdom
use baseforcing_module, only : lisforc, FORC_State
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: nest
```

DESCRIPTION:

Corrects Temperature, Pressure, Humidity and Longwave Radiation values for topography (based on Liston et al; 2005) WARNING: This routine is in the priliminary testing phase.
The arguments are:

nest index of the nest

2.28 Fortran: Module Interface mpishorthand.F90 (Source File: mpishorthand.F90)

REVISION HISTORY:

```
02 Jan 2002 : Sujay Kumar: Initial Version
```

USES:

```
#if (defined SPMD)
#if (defined USE_INCLUDE_MPI)
  include 'mpif.h'
#else
  use mpi
#endif
#endif
```

DESCRIPTION:

Data and parameters used for MPI. Some shorthand variables with shorter names than the standard MPI parameters. Also some variables used for heap management. Adopted from CLM

2.29 Fortran: Module Interface precision (Source File: precision.F90)

REVISION HISTORY:

```
14 Nov 2002; Sujay Kumar  Initial Specification
```

ARGUMENTS:

```
integer, parameter :: r4 = selected_real_kind(5)
integer, parameter :: r8 = selected_real_kind(6)
integer, parameter :: i8 = selected_int_kind(13)
```

DESCRIPTION:

Define the precision to use for floating point and integer operations throughout the model.

2.29.1 readcard (Source File: readcard.F90)

REVISION HISTORY:

```
 23 Feb 2001: Urszula Jambor; Added GEOS or GDAS forcing option
 27 Mar 2001: Jon Gottschalck; Revision of subroutine by implementing namelists
 15 Apr 2002: Urszula Jambor; Added ECMWF forcing options, also
              adding 1 & 1/2 degree GLDAS domain options.
 28 Apr 2002: Kristi Arsenault; Added NOAH LSM code
 14 Nov 2003: Sujay Kumar; Modified card file that includes regional
              modeling options
 02 Feb 2006: Sujay Kumar; Switched to the Inpak format
 19 Jan 2007: Chuck Alonge; Added read for Snow Depth (snowh)
              and added parameter output option (wparm)
```

INTERFACE:

```
subroutine readcard()
```

USES:

```
use lisdrv_module, only : lis, metadata_output, config_lis
use listime_mgr, only : date2time
use param_pluginMod, only : soils_plugin, laisai_plugin,tbot_plugin,&
     alb_plugin, gfrac_plugin, landcover_plugin,topo_plugin, &
```

```
    snow_plugin
use runmode_pluginMod, only : runmode_plugin
use spmdMod, only :iam, masterproc, npes
use lis_logmod, only :logunit
use ESMF_Mod
```

DESCRIPTION:

The code in this file initializes the LIS configuration management utility. The generic, model-independent runtime specifications of a LIS simulation are read by this routine. The routine also initializes the LIS log buffers, and calls some of the registries that set up the component plugin definitions.
The routines invoked are:

date2time (2.4.21)
convert the date format to a floating point time format

runmode_plugin (3.1.1)
sets up function table registries for implemented runmodes

soils_plugin (3.6.2)
sets up function table registries for implemented soil data sources

laisai_plugin (3.6.3)
sets up function table registries for implemented LAI/SAI data sources

tbot_plugin (3.6.5)
sets up function table registries for implemented bottom temperature data sources

alb_plugin (3.6.7)
sets up function table registries for implemented albedo data sources

snow_plugin (3.6.8)
sets up function table registries for implemented snow depth data sources

gfrac_plugin (3.6.6)
sets up function table registries for implemented greenness fraction data sources

landcover_plugin (3.6.4)
sets up function table registries for implemented landcover data sources

topo_plugin (3.6.1)
sets up function table registries for implemented topography data sources

---

2.29.2 readkpds (Source File: readkpds.F90)

REVISION HISTORY:

```
23 Oct 2003; Sujay Kumar; Initial Version
```

INTERFACE:

```
subroutine readkpds(ftn, kpds, writeint)

  implicit none
```

ARGUMENTS:

```
integer     :: kpds(25),ftn,writeint
```

DESCRIPTION:

Reads the kpds array from the grib table
The arguments are:

ftn the file unit number to be read from

kpds grid description array

writeint model output interval

2.29.3 soiltype (Source File: soiltype.F90)

3 Jul 2001: Matt Rodell Initial specifiation 13 Dec 2004: Sujay Kumar Updated with support for STATSGO classification

INTERFACE:

```
subroutine soiltype(class,nc,nr,sand,clay,silt,soiltyp)
```

USES:

```
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: class
integer, intent(in)    :: nc,nr
real, intent(in)       :: sand(nc,nr),clay(nc,nr),silt(nc,nr)
integer, intent(inout) :: soiltyp(nc,nr)
```

DESCRIPTION:

This subroutine uses the percentages of sand, silt, and clay to convert to soil texture data. The transformation is based on the type of classification used. This routine supports the transformation to a Zobler (9 classes) or the STATSGO (19 classes) scheme.
The arguments are:

class soil classification scheme (1-zobler, 2-statsgo)

nc dimension of the grid along the east-west direction

nr dimension of the grid along the north-south direction

sand array containing the sand fraction data

clay array containing the clay fraction data

silt array containing the silt fraction data

soiltyp array containing the derived soil texture

2.29.4 stats (Source File: stats.F90)

Calculates some diagnostic statistics for a given variable for model output. The routine provides statistics such as mean, standard deviation, minimum and maximum values of a given variable.

REVISION HISTORY:

```
Nov 11 1999:  Jon Radakovich; Initial code
```

INTERFACE:

```
subroutine stats(var,udef,nch,mean,stdev,min,max)
```

ARGUMENTS:

```
integer, intent(in) :: nch
real, intent(in)    :: var(nch), udef
real, intent(out)   :: mean,stdev,min,max
```

2.29.5 stats_da (Source File: stats_da.F90)

Calculates some diagnostic statistics for a given variable for data assimilation output. The routine provides statistics such as mean, standard deviation, minimum and maximum values and spread of a given variable.

REVISION HISTORY:

```
aug 22 2006:  Sujay Kumar; Initial code
```

INTERFACE:

```
subroutine stats_da(var,udef,nch,mean,spread, stdev,min,max)
```

ARGUMENTS:

```
integer, intent(in) :: nch
real, intent(in)    :: var(nch), udef
real, intent(out)   :: mean,stdev,min,max,spread
```

2.29.6 zterp (Source File: zterp.F90)

REVISION HISTORY:

```
10/2/98 Brian Cosgrove
6/28/00 Brian Cosgrove; changed code so that it uses  LDAS%UDEF and
        not a hard-wired undefined value of -999.999.  Also changed
        call to ZTERP subroutine so that LDAS and GRID mod files
        are brought in and can be accessed in this subroutine
2/27/01 Brian Cosgrove; Added czmodel into call for ZTERP subroutine
7/24/07 Brian Cosgrove; Fixed bug in the loop which calculates the
        average zenith angle...had been one time interval
        too far into the future, gmt now is not incremented by
        interval until second time through loop
```

INTERFACE:

```
subroutine zterp (iflag,lat,lon,btime,etime, &
     mbtime,julianb,weight1,weight2,czbegdata, &
     czenddata,czmodel,lis)
```

USES:

```
use lis_module
use lis_logmod, only :lis_log_msg
implicit none
```

ARGUMENTS:

```
type (lisdec)  :: lis
integer        :: iflag
real           :: lat,lon
real           :: mbtime
integer        :: julianb
real           :: interval
real           :: btime
real           :: etime
real           :: weight1
real           :: weight2
```

DESCRIPTION:

This subroutine is based, in part, on modified subroutines from Jean C. Morrill of the GSWP project. The program temporally interpolates time averge or instantaneous data to that needed by the model at the current timestep. It does this by combining a linear interpolation approach with a solar zenith angle approach in a fashion suitable for use with data such as short wave radiation values. It cannot be used with input data points which are more than 24 hours apart. The program outputs two weights which can then be applied to the original data to arrive at the interpolated data. If IFLAG=0, then WEIGHT1 is the weight which should be applied to the time averaged data (from the time period which the model is currently in) to arrive at the interpolated value and weight 2 is not used at all. If IFLAG=1, then WEIGHT1 should be applied to the original instantaneous data located just prior to the model time step, and WEIGHT2 should be applied to the original instantaneous data located just after the model time step.
i.e. (IF IFLAG=0) interpolated data = (WEIGHT1 * time averaged data from time period that model is currently in)
i.e. (IF IFLAG=1) interp. data = (WEIGHT1*past data)+(WEIGHT2*future data)
The arguments are:

iflag Flag specifying if input data is time averaged or not.

lis instance of the `lis_module`

lat latitude of the current point

lon longitude of the current point

btime beginning time of the original averaged data or time of original instantaneous data point which is located at or just prior to the current model time step (IFLAG=1). Expects GMT time in hour fractions. i.e., 6:30 Z would be 6.5.

etime Ending time of orig. avg. data (IFLAG=0) or time of original instantaneous data point which is located at or just after the current model time step (IFLAG=1).

mbtime Time of current time step.

julianb Julian day upon which BTIME falls

weight1 weight applied to original time averaged data (IFLAG=0) or weight applied to orig instantaneous data point located just prior to the current model time step

weight2 weight applied to orig instantaneous data point located just after the current model time step (IFLAG=1) If IFLAG=0, then this weight is meaningless and should not be used

czbegdata cosine of zenith angle at the beginning timestep

czenddata cosine of zenith angle at the ending timestep

czmodel cosine of the zenith angle at the current timestep

The routines invoked are:

LIS_localtime (2.29.8)
compute the LIS_localtime based on the GMT

coszenith (2.29.7)
compute the cosine of zenith angle

2.29.7 coszenith (Source File: zterp.F90)

Returns the cosine of the zenith angle from the following parameters
1) Day angle (GAMMA)
2) Solar DEClination
3) Equation of time
4) Local apparent time
5) Hour angle
6) Cosine of zenith angle
All equations come from "An Introduction to Solar Radition" By Muhammad Iqbal, 1983.

INTERFACE:

```
subroutine coszenith (lon,latd,lhour,zone,julian,czenith,dec,omega)

  implicit none
```

ARGUMENTS:

```
  integer :: zone                ! time zone (1-24) gmt=12
  integer :: julian              ! julian day
  real ::  czenith             ! cosine of zenith angle (radians)
  real ::  dec                 ! solar declination (radians)
  real ::  et                  ! equation of time (minutes)
  real ::  gamma               ! day angle (radians)
  real ::  latime              ! local apparent time
  real ::  lcorr               ! longitudical correction
  real ::  lhour               ! local standard time
  real ::  lon                 ! local longitude (deg)
  real ::  llat                ! local latitude in radians
  real ::  latd                ! local latitude in degrees
  real ::  ls                  ! standard longitude (deg)
  real ::  omegad              ! omega in degrees
  real ::  omega               ! omega in radians
  real ::  pi                  ! universal constant pi [-]
  real ::  zenith              ! zenith angle(radians)
```

2.29.8 LIS_localtime (Source File: zterp.F90)

Calculates the local time based on GMT

INTERFACE:

```
subroutine LIS_localtime (gmt,lon,lhour,zone)
```

ARGUMENTS:

```
real:: gmt                 ! GMT time (0-23)
real:: lon                 ! longitude in degrees
real:: change              ! the change in number of hours between
real:: lhour               ! local hour (0-23) 0= midnight, 23= 11:00 p.m.

integer::  i              ! working integer
integer:: zone            ! time zone (1-24)
```

2.29.9 gasdev (Source File: gasdev.F90)

Returns a normally distributed deviate with zero mean and unit variance, using ran2(idum) as the source of uniform deviates. This routine is adapted from the Numerical Recipies for Fortran

REVISION HISTORY:

```
27 Feb 2005: Sujay Kumar : Specification in LIS.
```

INTERFACE:

```
FUNCTION gasdev(idum)
```

2.29.10 ran2 (Source File: gasdev.F90)

Long period (¿2!1e18) random number generator of L'Ecuyer with Bays-Durham shuffle and added safeguards. Returns a uniform random deviate between 0.0 and 1.0 (exclusive of the endpoint values). Call with "idum" a negative integer to initialize; thereafter, do not alter "idum" between successive deviates in a sequence. RNMX should approximate the largest floating value that is less than 1. This function is adapted from Numerical Recipies for Fortran.

REVISION HISTORY:

```
27 Feb 2005: Sujay Kumar : Specification in LIS.
```

INTERFACE:

```
FUNCTION ran2(idum)
```

2.30 Fortran: Module Interface infnan (Source File: infnan.F90)

Set parameters for the floating point flags "inf" Infinity and "nan" not-a-number. As well as "bigint" the point at which integers start to overflow. These values are used to initialize arrays with as a way to detect if arrays are being used before being set.

REVISION HISTORY:

```
14 Nov 2005    Sujay Kumar  Initial Specification
```

2.31 Fortran: Module Interface optimization_module (Source File: optimization_module.F90)

REVISION HISTORY:

```
15 May 2007: Sujay Kumar; Initial implementation
```

2.31.1 LIS_optimization_init (Source File: optimization_module.F90)

INTERFACE:

```
subroutine LIS_optimization_init
```

DESCRIPTION:

This routine is under development.

USES:

2.31.2 readObsAttributes (Source File: readObsAttributes.F90)

REVISION HISTORY:

```
14 Oct 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine readObsAttributes(n, vname,&
     ssdev,varmin,varmax)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod,    only : logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer            :: n
character*40       :: vname(lis%nobtypes(n))
integer            :: perttype(lis%nobtypes(n))
real               :: ssdev(lis%nobtypes(n))
real               :: stdmax(lis%nobtypes(n))
integer            :: zeromean(lis%nobtypes(n))
real               :: tcorr(lis%nobtypes(n))
real               :: xcorr(lis%nobtypes(n))
real               :: ycorr(lis%nobtypes(n))
real               :: varmin(lis%nobtypes(n))
real               :: varmax(lis%nobtypes(n))
integer            :: ftn
```

DESCRIPTION:

This routine reads the observation attributes for each observation type. The format of the attributes file is:
Variable name standard deviation Max Value Min Value
The arguments are:

n index of nest

vname name of the observation variables

ssdev error rate in observation

varmin minimum value of the variable

varmax maximum value of the variable

2.31.3 readObsPertAttributes (Source File: readObsPertAttributes.F90)

REVISION HISTORY:

```
14 Oct 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine readObsPertAttributes(n, vname,&
     perttype,ssdev,stdmax,zeromean,tcorr,&
     xcorr, ycorr, ccorr)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod,    only : logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer            :: n
character*40       :: vname(lis%nobtypes(n))
integer            :: perttype(lis%nobtypes(n))
real               :: ssdev(lis%nobtypes(n))
real               :: stdmax(lis%nobtypes(n))
integer            :: zeromean(lis%nobtypes(n))
real               :: tcorr(lis%nobtypes(n))
real               :: xcorr(lis%nobtypes(n))
real               :: ycorr(lis%nobtypes(n))
real               :: ccorr(lis%nobtypes(n),lis%nobtypes(n))
```

DESCRIPTION:

This routine reads the observation perturbation attributes for each observation type. The format of the perturbation attributes file is:
¡Variable name¿ ¡perturbation type¿ ¡standard deviation¿ ¡standard normal max¿ (contd.) ¡ensure zeromean?¿ ¡temporal correlation scale¿ (contd.) ¡x correlation scale¿ ¡y correlation scale¿ ¡cross correlations¿
The arguments are:

n index of nest

vname name of the observation variables

perttype type of perturbation

ssdev standard deviation of perturbation

stdmax standard normal max for the perturbation distribution

zeromean flag to specify if zero mean is to be ensured

tcorr temporal correlation scale

xcorr horizontal x-correlation scale

ycorr horizontal y-correlation scale

2.31.4 tile_module (Source File: tile_module.F90)

The code in this file provides a description of the tile data structure in LIS

2.31.5 Overview

This module contains the tile data structure used in LIS. The tile space in LIS is used to simulate subgrid scale variability, where each tile represents a vegetation class within a grid cell. The tile space is created from the vegetation distribution within a grid cell.
The tile data structure contains the following variables:

col column index of the corresponding grid cell

row row index of the corresponding grid cell

index index of the corresponding grid cell

vegt vegetation type of the tile

ensem ensemble index of the tile

fgrd fraction of grid covered by the tile

REVISION HISTORY:

```
14 Nov 2002: Sujay Kumar; Optimized version of tile_module
```

2.31.6 verify (Source File: verify.F90)

INTERFACE:

```
subroutine verify(ierr)
```

ARGUMENTS:

```
use ESMF_Mod

implicit none
integer, intent(in)          :: ierr
```

DESCRIPTION:

This is an error check routine. Program exits in case of error with an associated error message written to the 'abort message' file.

2.31.7 ft_check_index (Source File: FTable.c)

INTERFACE:

```
void ft_check_index(int index, int max, char * regrtn)
```

DESCRIPTION:

When registering a plugin into a function pointer table, one must specify where (the index) into the table to place this plugin. This routine checks the index into the function pointer table against the table's maximum value.
The purpose of this check is to avoid overwriting a function pointer table and causing a memory error.
This routines exits with a -1 if the index fails the test.
The arguments are:

index Position into the function pointer table

max Maximum value of index

regrtn String containing the name of the register routine. Used for printing an error message.

2.32 Fortran: Module Interface LIS_alb_FTable (Source File: LIS_alb_FTable.c)

Function table registries for storing the interface implementations for managing different sources of albedo parameter data

---

2.32.1 registerreadmxsnoalb (Source File: LIS_alb_FTable.c)

INTERFACE:

```
void FTN(registerreadmxsnoalb)(int *i, int *j,void (*func)(int*, float*))
```

DESCRIPTION:

Creates a entry in the registry for the routine to open and read albedo upper bound over deep snow
The arguments are:

i index of the domain

j index of the albedo source

2.32.2 readmxsnoalb (Source File: LIS_alb_FTable.c)

INTERFACE:

```
void FTN(readmxsnoalb)(int *n, int *i, int *j,float *array)
```

DESCRIPTION:

Invokes the routines from the registry for reading albedo upper bound over deep snow data
The arguments are:

n index of the nest

i index of the domain

j index of the albedo source

array pointer to the array containing the albedo data.

2.32.3 registerreadalbedo (Source File: LIS_alb_FTable.c)

INTERFACE:

```
void FTN(registerreadalbedo)(int *i, int *j,void (*func)(int*, int*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read albedo climatology data
The arguments are:

i index of the domain

j index of the albedo source

2.32.4 readalbedo (Source File: LIS_alb_FTable.c)

INTERFACE:

```
void FTN(readalbedo)(int *n, int *i, int *j, int *k, float *array)
```

DESCRIPTION:

Invokes the routines from the registry for reading climatology albedo files
The arguments are:

n index of the nest

i index of the domain

j index of the albedo source

k time index value

array pointer to the array containing the albedo data.

2.33 Fortran: Module Interface LIS_dataassim_FTable (Source File: LIS_dataassim_FTable.c

Function table registries for storing the interface implementations for the operation of data assimilation algorithms and observations.

2.33.1 registerdaobssetup (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(registerdaobssetup)(int *i, void (*func)(void*, void*))
```

DESCRIPTION:

Creates an entry in the registry for the routine that sets up structures for handling observation data for data assimilation
The arguments are:

i index of the variable being assimilated

2.33.2 readobsdataconfig (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(readobsdataconfig)(int *i, void *obs, void *pert)
```

DESCRIPTION:

Invokes the routine from the registry to set up structures for handling observation data for data assimilation
The arguments are:

i index of the variable being assimilated

obs ESMF state that contain the observations

pert ESMF state that contain the observation perturbations

2.33.3 registerdaobs (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(registerdaobs)(int *i, void (*func)(int*, void*))
```

DESCRIPTION:

Creates an entry in the registry for the routine that reads observations for data assimilation.
The arguments are:

i index of the observations variable

2.33.4 readdaobs (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(readdaobs)(int *i,int *n, void *obsstate)
```

DESCRIPTION:

Invokes the routine from the registry to read observations for data assimilation. This routine also packages the observations into an ESMF state
The arguments are:

i index of the observations variable

n index of the nest

obsstate pointer to the ESMF observation state

2.33.5 registergetnso (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(registergetnso)(int *i, void (*func)(int*, int*, int*, int*))
```

DESCRIPTION:

Creates an entry in the registry for the routine that computes the number of selected observations for the specified grid point.
The arguments are:

i index of the observations variable

2.33.6 getselectedobsnumber (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(getselectedobsnumber)(int *i, int *n, int *gid, int *sid, int *eid)
```

DESCRIPTION:

Invokes the routine from the registry that computes the number of selected observations for a particular modeling point
The arguments are:

i index of the observations variable

n index of the nest

gid model grid point index

sid starting index of the selected observations

eid ending index of the selected observations

2.33.7 registerdainit (Source File: LIS_dataassim_FTable.c)

Creates an entry in the registry for the routine to set up algorithm specific structures for the data assimilation method used.
The arguments are:

i index of the assimilation algorithm

INTERFACE:

```
void FTN(registerdainit)(int *i, void (*func)(int*))
```

2.33.8 dataassiminit (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(dataassiminit)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to initialize the specific structures for the data assimilation algorithm used.
The arguments are:

i index of the assimilation algorithm

2.33.9 registerapplyincrements (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(registerapplyincrements)(int *i, void (*func)(int*))
```

DESCRIPTION:

Creates an entry in the registry for the routine that applies the analysis increments
The arguments are:

i index of the assimilation algorithm

2.33.10 applyincrements (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(applyincrements)(int *i, int *n)
```

DESCRIPTION:

Invoke the routine from the registry that applies analysis increments
The arguments are:

i index of the assimilation algorithm

n index of the nest

2.33.11 registercomputeincrements (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(registercomputeincrements)(int *i, void (*func)(int*))
```

DESCRIPTION:

Creates an entry in the registry for the routine that computes the analysis increments
The arguments are:

i index of the assimilation algorithm

2.33.12 computeincrements (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(computeincrements)(int *i, int *n)
```

DESCRIPTION:

Invoke the routine from the registry that computes analysis increments
The arguments are:

i index of the assimilation algorithm

n index of the nest

2.33.13 registerdafinalize (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(registerdafinalize)(int *i, void (*func)(int*))
```

DESCRIPTION:

Cretes an entry in the registry for the routine to cleanup allocated structures
The arguments are:

i index of the assimilation algorithm

2.33.14 dafinalize (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(dafinalize)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry that cleans up the data assimilation specific structures
The arguments are:

i index of the assimilation algorithm

2.33.15 registerdaoutput (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(registerdaoutput)(int *i, void (*func)(int*))
```

DESCRIPTION:

Cretes an entry in the registry for the writing diagnostic output from the data assimilation routines
The arguments are:

i index of the assimilation algorithm

2.33.16 daoutput (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(daoutput)(int *i,int *n)
```

DESCRIPTION:

Invokes the routine from the registry to write data assimilation specific output
The arguments are:

i index of the assimilation algorithm

n index of the nest

2.34 Fortran: Module Interface LIS_domain_FTable (Source File: LIS_domain_FTable.c)

Function table registries for storing the interface implementations for managing the operation of different domain specifications

---

2.34.1 registerdomain (Source File: LIS_domain_FTable.c)

INTERFACE:

```
void FTN(registerdomain)(int *i,void (*func)())
```

DESCRIPTION:

Creates an entry in the registry for the routine to create grid and tile spaces for a particular domain definition
The arguments are:

i index of the domain

2.34.2 makedomain (Source File: LIS_domain_FTable.c)

INTERFACE:

```
void FTN(makedomain)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to create grid and tile spaces for a particular domain definition
The arguments are:

i index of the domain

2.34.3 registerinput (Source File: LIS_domain_FTable.c)

INTERFACE:

```
void FTN(registerinput)(int *i,void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to read the runtime domain specifics
The arguments are:

i index of the domain

2.34.4 readinput (Source File: LIS_domain_FTable.c)

INTERFACE:

```
void FTN(readinput)(int *i)
```

DESCRIPTION:

Calls the routine from the registry to read the the runtime domain specifics
The arguments are:

i index of the domain

2.35 Fortran: Module Interface LIS_forcing_FTable (Source File: LIS_forcing_FTable.c)

Function table registries for storing the interface implementations for managing different meteorological analyses implmented as "baseforcings"

2.35.1 registerget (Source File: LIS_forcing_FTable.c)

INTERFACE:

```
void FTN(registerget)(int *i,void (*func)(int*))
```

DESCRIPTION:

Creates an entry in the registry for the routines to open and read base forcing
The arguments are:

i index of the forcing type

2.35.2 retrieveforcing (Source File: LIS_forcing_FTable.c)

INTERFACE:

```
void FTN(retrieveforcing)(int *i, int *n)
```

DESCRIPTION:

Invokes the routine from the registry to open and read the base forcing
The arguments are:

i index of the forcing type

n index of the nest

2.35.3 registertimeinterp (Source File: LIS_forcing_FTable.c)

INTERFACE:

```
void FTN(registertimeinterp)(int *i,void (*func)(int*))
```

DESCRIPTION:

Creats an entry in the registry for the routine to perform temporal interpolation
The arguments are:

i index of the forcing type

2.35.4 timeinterp (Source File: LIS_forcing_FTable.c)

INTERFACE:

```
void FTN(timeinterp)(int *i, int *n)
```

DESCRIPTION:

Invokes the routine from the registry to perform temporal interpolation
The arguments are:

i index of the forcing type

n index of the nest

2.35.5 registerdefinenative (Source File: LIS_forcing_FTable.c)

INTERFACE:

```
void FTN(registerdefinenative)(int *i, void (*func)())
```

DESCRIPTION:

Creates an entry in the registry for the routine that define the native domain of the base forcing scheme.
The arguments are:

i index of the forcing type

2.35.6 definenative (Source File: LIS_forcing_FTable.c)

INTERFACE:

```
void FTN(definenative)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry that defines the native domain of the base forcine scheme.
The arguments are:

i index of the forcing type

2.35.7 registerforcingfinal (Source File: LIS_forcing_FTable.c)

INTERFACE:

```
void FTN(registerforcingfinal)(int *i,void (*func)())
```

DESCRIPTION:

Creates an entry in the registry for the routine to cleanup allocated structures
The arguments are:

i index of the forcing type

2.35.8 forcingfinalize (Source File: LIS_forcing_FTable.c)

INTERFACE:

```
void FTN(forcingfinalize)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to cleanup the allocated structures associated with the base forcing
The arguments are:

i index of the forcing type

2.35.9 registerreadforcingelev (Source File: LIS_forcing_FTable.c)

INTERFACE:

```
void FTN(registerreadforcingelev)(int *i,int *j, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to read forcing elevation data
The arguments are:

i index of the forcing type

j index of the domain

2.35.10 readforcingelev (Source File: LIS_forcing_FTable.c)

INTERFACE:

```
void FTN(readforcingelev)(int *n,int *i, int *j, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the forcing elevation data
The arguments are:

n index of the nest

i index of the forcing type

j index of the domain

array pointer to the elevation array

2.36 Fortran: Module Interface LIS_gfrac_FTable (Source File: LIS_gfrac_FTable.c)

Function table registries for storing the interface implementations for managing different sources of greenness fraction data

---

2.36.1 registerreadgfrac (Source File: LIS_gfrac_FTable.c)

INTERFACE:

```
void FTN(registerreadgfrac)(int *i,int *j,void (*func)(int*, int*, float*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to read gfrac data
The arguments are:

i index of the domain

j index of the greenness data source

2.36.2 readgfrac (Source File: LIS_gfrac_FTable.c)

INTERFACE:

```
void FTN(readgfrac)(int *n, int *i,int *j,int *month, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to reading gfrac data
The arguments are:

n index of the nest

i index of the domain

j index of the greenness data source

month month

array pointer to the greenness data

2.37 Fortran: Module Interface LIS_laisai_FTable (Source File: LIS_laisai_FTable.c)

Function table registries for storing the interface implementations for managing different sources of LAI and SAI data

---

2.37.1 registerreadlai (Source File: LIS_laisai_FTable.c)

INTERFACE:

```
void FTN(registerreadlai)(int *i, int *j, void (*func)(int*, int*,float*))
```

DESCRIPTION:

Creates an entry in the registry for the routines to read LAI data
The arguments are:

i index of domain

j index of the LAI source

2.37.2 readlai (Source File: LIS_laisai_FTable.c)

INTERFACE:

```
void FTN(readlai)(int *n, int *i, int *j, int *month, float* array)
```

DESCRIPTION:

Invokes the routine from the registry to read LAI data.
The arguments are:

n index of the nest

i index of domain

j index of the LAI source

month month

array pointer to the LAI data

2.37.3 registerreadsai (Source File: LIS_laisai_FTable.c)

INTERFACE:

```
void FTN(registerreadsai)(int *i, int *j, void (*func)(int*, int*,float*))
```

DESCRIPTION:

Creates an entry in the registry for the routines to read SAI data
The arguments are:

i index of domain

j index of the SAI source

2.37.4 readsai (Source File: LIS_laisai_FTable.c)

INTERFACE:

```
void FTN(readsai)(int *n, int *i, int *j, int *month, float* array)
```

DESCRIPTION:

Invokes the routine from the registry to read SAI data.
The arguments are:

n index of nest

i index of domain

j index of the SAI source

month month

array pointer to the SAI data

2.38 Fortran: Module Interface LIS_landcover_FTable (Source File: LIS_landcover_FTable.c)

Function table registries for storing the interface implementations for managing different sources of landcover data

---

2.38.1 registerreadlc (Source File: LIS_landcover_FTable.c)

INTERFACE:

```
void FTN(registerreadlc)(int *i,int *j, void (*func)())
```

DESCRIPTION:

Creates an entry in the registry for the routine to read landcover data
The arguments are:

i index of the domain

j index of the landcover source

2.38.2 readlandcover (Source File: LIS_landcover_FTable.c)

Invokes the routine from the registry for reading landcover data.
The arguments are:

n index of the nest

i index of the domain

j index of the landcover source

array pointer to the landcover data

INTERFACE:

```
void FTN(readlandcover)(int *n, int *i, int *j,float *array)
```

2.38.3 registerreadmask (Source File: LIS_landcover_FTable.c)

INTERFACE:

```
void FTN(registerreadmask)(int *i,int *j, void (*func)())
```

DESCRIPTION:

Creates an entry in the registry for the routine to read landmask data
The arguments are:

i index of the domain

j index of the landcover source

2.38.4 readlandmask (Source File: LIS_landcover_FTable.c)

INTERFACE:

```
void FTN(readlandmask)(int *n,int *i, int *j,float *array)
```

DESCRIPTION:

Invokes the routine from the registry for reading landmask data.
The arguments are:

n index of the nest

i index of the domain

j index of the landcover source

array pointer to the landmask data

2.39 Fortran: Module Interface LIS_lsm_FTable (Source File: LIS_lsm_FTable.c)

Function table registries for storing the interface implementations for managing the operations of different land surface models

---

2.39.1 registerlsmini (Source File: LIS_lsm_FTable.c)

Creates an entry in the registry for the routine to perform land surface model initialization

INTERFACE:

```
void FTN(registerlsmini)(int *i, void (*func)())
```

2.39.2 lsmini (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmini)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to perform land surface model initialization

i index of the LSM

2.39.3 registerlsmrun (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmrun)(int *i, void (*func)(int*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to run the land surface model

i index of the LSM

2.39.4 lsmrun (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmrun)(int *i,int *n)
```

DESCRIPTION:

Invokes the routine from the registry to run the land surface model

i index of the LSM

n index of the nest

2.39.5 registerlsmfinalize (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmfinalize)(int *i, void (*func)())
```

DESCRIPTION:

Creates an entry in the registry for the routine to cleanup allocated structures specific to the land surface model

i index of the LSM

2.39.6 lsmfinalize (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmfinalize)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry for cleaning up allocated structures specific to the land surface model

i index of the LSM

2.39.7 registerlsmsetup (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmsetup)(int *i, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to set up land surface model parameters

i index of the LSM

2.39.8 lsmsetup (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmsetup)(int *i)
```

DESCRIPTION:

Invokes the routine in the registry to set up land surface model parameters

i index of the LSM

2.39.9 registerlsmrestart (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmrestart)(int *i, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to restart the land surface model from a previously saved state

i index of the LSM

2.39.10 lsmrestart (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmrestart)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to restart the land surface model from a previously saved state

i index of the LSM

2.39.11 registerlsmoutput (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmoutput)(int *i, void (*func)(int*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to perform land surface model output

i index of the LSM

2.39.12 lsmoutput (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmoutput)(int *i, int *n)
```

DESCRIPTION:

Invokes the routine from the registry to perform land surface model output

i index of the LSM

n index of the nest

2.39.13 registerlsmf2t (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmf2t)(int *i, int *j, void (*func)(int*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to transfer forcing to model tiles

i index of the LSM

j index of the runmode

2.39.14 lsmf2t (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmf2t)(int *i, int *j, int*n)
```

DESCRIPTION:

Invokes the routine from the registry to transfer forcing to model tiles

i index of the LSM

j index of the runmode

n index of the nest

2.39.15 registerlsmwrst (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmwrst)(int *i, void (*func)(int*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to write restart files for a land surface model

i index of the LSM

2.39.16 lsmwrst (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmwrst)(int *i, int *n)
```

DESCRIPTION:

Invokes the routine from the registry to write restart files for a land surface model

i index of the LSM

n index of the nest

2.39.17 registergetstatevar (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registergetstatevar)(int *i, int *j, void (*func)(int*, void*))
```

DESCRIPTION:

Makes an entry in the registry for the routine for obtaining the specified prognostic variables from the land surface model

i index of the LSM

j index of the variable

2.39.18 getlsmstatevar (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(getlsmstatevar)(int *i, int *j, int *n, void *state)
```

DESCRIPTION:

Invokes the routine from the registry for obtaining the specified prognostic variables from the land surface model

i index of the LSM

j index of the variable

n index of the nest

state pointer to the prognostic variable state

2.39.19 registersetstatevar (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registersetstatevar)(int *i, int *j, void (*func)(int*, void*))
```

DESCRIPTION:

Makes an entry in the registry for updating the specified state variable in a land surface model

i index of the LSM

j index of the variable

2.39.20 setlsmstatevar (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(setlsmstatevar)(int *i, int *j, int *n, void *statevar)
```

DESCRIPTION:

Invokes the routine from the registry for updating the specified state variable in a land surface model

i index of the LSM

j index of the variable

n index of the nest

statevars pointer to the prognostic variable state

2.39.21 registerobstransform (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerobstransform)(int *i, int *j, void (*func)(int*, void*))
```

DESCRIPTION:

Makes an entry in the registry to perform the translation of observations to state variables

i index of the LSM

j index of the variable being assimilated

k index of the observation data

2.39.22 obstransform (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(obstransform)(int *i, int *j, int *n, void *obs)
```

DESCRIPTION:

Invokes the routine from the registry to translate the observations to the prognostic variable space

i index of the LSM

j index of the variable being assimilated

n index of the nest

obs transformed variable

2.39.23 registergetobspred (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registergetobspred)(int *i, int *j, void (*func)(int*,float*))
```

DESCRIPTION:

Makes an entry in the registry for the routine that provides an LSM's estimate of the observations.

i index of the LSM

j index of the observation data

2.39.24 getobspred (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(getobspred)(int *i, int *j, int *n, float *pred)
```

DESCRIPTION:

Invokes the routine from the registry to translate the observations to state variables

i index of the LSM

j index of the observation data

n index of the nest

pred model's estimated observation prediction

2.39.25 registermapobstolsm (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registermapobstolsm)(int *i, int *j, void (*func)(int*, void*, void*))
```

DESCRIPTION:

Makes an entry in the registry to perform the translation of observations to state variables

i index of the LSM

j index of the observation data

2.39.26 mapobstolsm (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(mapobstolsm)(int *i, int *j, int *n, void *obs, void *lsm)
```

DESCRIPTION:

Invokes the routine from the registry to translate the observations to state variables

i index of the LSM

j index of the observation data

n index of the nest

obs observations to be mapped

lsm updated lsm states

2.39.27 registerlsmexport (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmexport)(int *i, int *j, void (*func)(int*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to set export states for a LSM

i index of the LSM

2.39.28 lsmsetexport (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmsetexport)(int *i, int *j, int *n)
```

DESCRIPTION:

Invokes the routine in the registry to set the export states for a LSM

i index of the LSM

n index of the nest

2.39.29 registerlsmdynsetup (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmdynsetup)(int *i, void (*func)(int*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to set the time dependent land surface parameters

i index of the LSM

2.39.30 lsmdynsetup (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmdynsetup)(int *i, int *n)
```

DESCRIPTION:

Invokes the routine from the registry to set the time dependent land surface parameters

i index of the LSM

n index of the nest

2.39.31 registerqclsmstate (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerqclsmstate)(int *i, int *j, void (*func)(int*, void*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to QC the updated LSM state

i index of the LSM

j index of the assimilated variable

2.39.32 qclsmstate (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(qclsmstate)(int *i, int *j, int *n, void *LSM_State)
```

DESCRIPTION:

Invokes the routine from the registry to set the QC the updated LSM variables

i index of the LSM

j index of the assimilated variable

n index of the nest

LSM_State The LSM state being qc'd

2.39.33 registerscalelsmstatevar (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerscalelsmstatevar)(int *i, int *j, void (*func)(int*, void*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to scale the LSM state variables

i index of the LSM

j index of the assimilated variable

2.39.34 scalelsmstatevar (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(scalelsmstatevar)(int *i, int *j, int *n, void *LSM_State)
```

DESCRIPTION:

Invokes the routine from the registry to scale the LSM state variables

i index of the LSM

j index of the assimilated variable

n index of the nest

LSM_State The LSM state being scaled

2.39.35 registerdescalelsmstatevar (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerdescalelsmstatevar)(int *i, int *j, void (*func)(int*, void*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to descale the LSM state variables

i index of the LSM

j index of the assimilated variable

2.39.36 descalelsmstatevar (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(descalelsmstatevar)(int *i, int *j, int *n, void *LSM_State)
```

DESCRIPTION:

Invokes the routine from the registry descale the LSM state variables

i index of the LSM

j index of the assimilated variable

n index of the nest

LSM_State The LSM state being descaled

2.39.37 registerupdatelsmstate (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerupdatelsmstate)(int *i, int *j, void (*func)(int*, void*, void*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to apply the LSM state increments to LSM state

i index of the LSM

j index of the assimilation set

2.39.38 updatelsmstate (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(updatelsmstate)(int *i, int *j, int *n, void *LSM_State, void *LSM_Incr_State)
```

DESCRIPTION:

Invokes the routine from the registry to apply the LSM state increments to LSM State

i index of the LSM

j index of the assimilated variable

n index of the nest

LSM_State The LSM state being updated

LSM_Incr_State The LSM incr state

2.40 Fortran: Module Interface LIS_perturb_FTable (Source File: LIS_perturb_FTable.c)

Function table registries for storing the interface implementations for managing the operations of different perturbation algorithms

---

2.40.1 registerperturbinit (Source File: LIS_perturb_FTable.c)

INTERFACE:

```
void FTN(registerperturbinit)(int *i, void (*func)(int*, void*, void*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to initialize the pertubation scheme.
The arguments are:

i index of the perturbation algorithm .

2.40.2 perturbinit (Source File: LIS_perturb_FTable.c)

INTERFACE:

```
void FTN(perturbinit)(int *i, int *n, void *base, void *pert)
```

DESCRIPTION:

Invokes the routine from the registry to initialize the pertubation scheme
The arguments are:

i index of the perturbation algorithm

n index of the nest

base base state for perturbation

pert output perturbation state

2.40.3 registerperturbmethod (Source File: LIS_perturb_FTable.c)

INTERFACE:

```
void FTN(registerperturbmethod)(int *i, void (*func)(int*, void*, void*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to invoke the pertubation scheme.
The arguments are:

i index of the perturbation algorithm .

2.40.4 perturbmethod (Source File: LIS_perturb_FTable.c)

INTERFACE:

```
void FTN(perturbmethod)(int *i, int *n, void *base, void *pert)
```

DESCRIPTION:

Invokes the routine from the registry to invoke the pertubation scheme
The arguments are:

i index of the perturbation algorithm

n index of the nest

base base state for perturbation

pert output perturbation state

2.41 Fortran: Module Interface LIS_runmode_FTable (Source File: LIS_runmode_FTable.c)

Function table registries for storing the interface implementations of different running modes in LIS

2.41.1 registerlisinit (Source File: LIS_runmode_FTable.c)

INTERFACE:

```
void FTN(registerlisinit)(int *i, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the LIS initialization method for a certain running mode
The arguments are:

i index of the running mode

2.41.2 lisinit (Source File: LIS_runmode_FTable.c)

INTERFACE:

```
void FTN(lisinit)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to perform the LIS initialization for the specified running mode
The arguments are:

i index of the running mode

2.41.3 registerlisrun (Source File: LIS_runmode_FTable.c)

INTERFACE:

```
void FTN(registerlisrun)(int *i, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the LIS run method for a certain running mode.
The arguments are:

i index of the running mode

2.41.4 lisrun (Source File: LIS_runmode_FTable.c)

INTERFACE:

```
void FTN(lisrun)(int *i)
```

DESCRIPTION:

Invokes the LIS run method from the registry for the specified running mode
The arguments are:

i index of the running mode

2.41.5 registerlisfinalize (Source File: LIS_runmode_FTable.c)

INTERFACE:

```
void FTN(registerlisfinalize)(int *i, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to perform LIS finalization for the specified running mode
The arguments are:

i index of the running mode

2.41.6 lisfinalize (Source File: LIS_runmode_FTable.c)

INTERFACE:

```
void FTN(lisfinalize)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to perform LIS finalization call for the specified running mode
The arguments are:

i index of the running mode

2.42 Fortran: Module Interface LIS_soils_FTable (Source File: LIS_soils_FTable.c)

Function table registries for storing the interface implementations for managing different sources of soil parameters data

---

2.42.1 registerreadsand (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(registerreadsand)(int *i,int *j, void (*func)(int *, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read the sand fraction data
The arguments are:

i index of the domain

j index of the soils source

2.42.2 readsand (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readsand)(int *n, int *i, int *j,float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the sand fraction data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the sand array

2.42.3 registerreadclay (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(registerreadclay)(int *i, int *j,void (*func)(int*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read the clay fraction data
The arguments are:

i index of the domain

j index of the soils source

2.42.4 readclay (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readclay)(int *n, int *i, int *j,float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the clay fraction data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the clay data

2.42.5 registerreadsilt (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(registerreadsilt)(int *i, int *j,void (*func)(int*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read the silt fraction data
The arguments are:

i index of the domain

j index of the soils source

2.42.6 readsilt (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readsilt)(int *n, int *i, int *j,float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the silt fraction data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the silt data

2.42.7 registerreadtexture (Source File: LIS_soils_FTable.c)

Creates an entry in the registry for the routine to read the soil texture data
The arguments are:

i index of the domain

j index of the soils source

INTERFACE:

```
void FTN(registerreadsoiltexture)(int *i, int *j,void (*func)(int*, int*))
```

2.42.8 readsoiltexture (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readsoiltexture)(int *n, int *i, int *j,int *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the soil texture data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the texture array

2.42.9 registerreadporosity (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(registerreadporosity)(int *i, int *j,void (*func)(int*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read the porosity data
The arguments are:

i index of the domain

j index of the soils source

2.42.10 readporosity (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readporosity)(int *n, int *i, int *j,float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the porosity data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the porosity array

2.42.11 registerreadpsisat (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(registerreadpsisat)(int *i, int *j,void (*func)(int*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read the saturated matric potential data
The arguments are:

i index of the domain

j index of the soils source

2.42.12 readpsisat (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readpsisat)(int *n, int *i, int *j,float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the saturated matric potential data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the saturated matric potential data

2.42.13 registerreadksat (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(registerreadksat)(int *i, int *j,void (*func)(int*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read the saturated hydraulic conductivity data
The arguments are:

i index of the domain

j index of the soils source

2.42.14 readksat (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readksat)(int *n, int *i, int *j,float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the saturated hydraulic conductivity data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the ksat data

2.42.15 registerreadbexp (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(registerreadbexp)(int *i, int *j,void (*func)(int*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read the b parameter data
The arguments are:

i index of the domain

j index of the soils source

2.42.16 readbexp (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readbexp)(int *n, int *i, int *j,float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the b parameter data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the b parameter data

2.42.17 registerreadquartz (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(registerreadquartz)(int *i, int *j,void (*func)(int*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read the quartz fraction data
The arguments are:

i index of the domain

j index of the soils source

2.42.18 readquartz (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readquartz)(int *n, int *i, int *j,float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the quartz fraction data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the quartz data

2.42.19 registerreadcolor (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(registerreadcolor)(int *i,int *j, void (*func)(int*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read the soil color data
The arguments are:

i index of the domain

j index of the soils source

2.42.20 readcolor (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readcolor)(int *n, int *i, int *j,float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the soil color data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the color array

2.43 Fortran: Module Interface LIS_suppforcing_FTable (Source File: LIS_suppforcing_FTab

Function table registries for storing the interface implementations for managing different supplemental forcing analyses

---

2.43.1 registerdefinenativesupp (Source File: LIS_suppforcing_FTable.c)

INTERFACE:

```
void FTN(registerdefinenativesupp)(int *i, void (*func)())
```

DESCRIPTION:

Creates an entry in the registry for the routine that define the native domain of the supplemental forcing scheme.
The arguments are:

i index of the forcing type

2.43.2 definenativesupp (Source File: LIS_suppforcing_FTable.c)

INTERFACE:

```
void FTN(definenativesupp)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry that defines the native domain of the supplemental forcine scheme.
The arguments are:

i index of the forcing type

2.43.3 registerreadsupp (Source File: LIS_suppforcing_FTable.c)

INTERFACE:

```
void FTN(registerreadsupp)(int *i,void (*func)(int*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to open and read the supplemental forcing
The arguments are:

i index of the forcing type

2.43.4 getsuppforc (Source File: LIS_suppforcing_FTable.c)

INTERFACE:

```
void FTN(getsuppforc)(int *n, int *i)
```

DESCRIPTION:

Invokes the routine from the registry to open and read the supplemental forcing
The arguments are:

n index of the nest

i index of the forcing type

2.43.5 registersuppti (Source File: LIS_suppforcing_FTable.c)

INTERFACE:

```
void FTN(registersuppti)(int *i,void (*func)(int*))
```

DESCRIPTION:

Creats an entry in the registry for the routine to perform temporal interpolation of the supplemental forcing
The arguments are:

i index of the forcing type

2.43.6 timeinterpsupp (Source File: LIS_suppforcing_FTable.c)

INTERFACE:

```
void FTN(timeinterpsupp)(int *n, int *i)
```

DESCRIPTION:

Invokes the routine from the registry to perform temporal interpolation of the supplemental forcing
The arguments are:

n index of the nest

i index of the forcing type

2.43.7 registersuppforcingfinal (Source File: LIS_suppforcing_FTable.c)

INTERFACE:

```
void FTN(registersuppforcingfinal)(int *i,void (*func)())
```

DESCRIPTION:

Creates an entry in the registry for the routine to cleanup allocated structures
The arguments are:

i index of the forcing type

2.43.8 suppforcingfinalize (Source File: LIS_suppforcing_FTable.c)

INTERFACE:

```
void FTN(suppforcingfinalize)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to cleanup the allocated structures associated with the supplemental forcing
The arguments are:

i index of the forcing type

2.44 Fortran: Module Interface LIS_tbot_FTable (Source File: LIS_tbot_FTable.c)

Function table registries for storing the interface implementations for managing different sources of bottom temperature data

---

2.44.1 registerreadtbot (Source File: LIS_tbot_FTable.c)

INTERFACE:

```
void FTN(registerreadtbot)(int *i,void (*func)(int*, float*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to read the bottom temperature data
The arguments are:

i index of the domain

2.44.2 readtbot (Source File: LIS_tbot_FTable.c)

INTERFACE:

```
void FTN(readtbot)(int *n, int *i, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the bottom temperature data
The arguments are:

n index of the nest

i index of the domain

array array containing bottom temperature data.

2.45 Fortran: Module Interface LIS_topo_FTable (Source File: LIS_topo_FTable.c)

Function table registries for storing the interface implementations for managing different sources of topography data

---

2.45.1 registerreadelev (Source File: LIS_topo_FTable.c)

INTERFACE:

```
void FTN(registerreadelev)(int *i,int *j, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to read elevation data
The arguments are:

i index of the domain

j index of the topography source

2.45.2 readelev (Source File: LIS_topo_FTable.c)

INTERFACE:

```
void FTN(readelev)(int *n,int *i, int *j, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the elevation data
The arguments are:

n index of the nest

i index of the domain

j index of the topography source

array pointer to the elevation array

2.45.3 registerreadslope (Source File: LIS_topo_FTable.c)

INTERFACE:

```
void FTN(registerreadslope)(int *i,int *j, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to read slope data
The arguments are:

i index of the domain

j index of the topography source

2.45.4 readslope (Source File: LIS_topo_FTable.c)

INTERFACE:

```
void FTN(readslope)(int *n,int *i, int *j, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the slope data
The arguments are:

n index of the nest

i index of the domain

j index of the topography source

array pointer to the slope array

2.45.5 registerreadaspect (Source File: LIS_topo_FTable.c)

INTERFACE:

```
void FTN(registerreadaspect)(int *i,int*j, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to read aspect data
The arguments are:

i index of the domain

j index of the topography source

2.45.6 readaspect (Source File: LIS_topo_FTable.c)

INTERFACE:

```
void FTN(readaspect)(int *n,int *i, int *j, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the aspect data
The arguments are:

n index of the nest

i index of the domain

j index of the topography source

array pointer to the aspect array

2.45.7 registerreadcurv (Source File: LIS_topo_FTable.c)

INTERFACE:

```
void FTN(registerreadcurv)(int *i,int *j, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to read curvature data
The arguments are:

i index of the domain

j index of the topography source

2.45.8 readcurv (Source File: LIS_topo_FTable.c)

INTERFACE:

```
void FTN(readcurv)(int *n,int *i, int *j, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the curvature data
The arguments are:

n index of the nest

i index of the domain

j index of the topography source

array pointer to the curvature array

2.46 Fortran: Module Interface LIS_snowd_FTable (Source File: LIS_snowd_FTable.c)

Function table registries for storing the interface implementations for managing different sources of snow depth data.

2.46.1 registerreadsnowdepth (Source File: LIS_snowd_FTable.c)

INTERFACE:

```
void FTN(registerreadsnowdepth)(int *i,int *j,void (*func)(int*, float*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to read snow depth data.
The arguments are:

i index of the domain

j index of the snow depth data source

2.46.2 readsnowdepth (Source File: LIS_snowd_FTable.c)

INTERFACE:

```
void FTN(readsnowdepth)(int *n, int *i,int *j,float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read snow depth data
The arguments are:

n index of the nest

i index of the domain

j index of the snow depth data source

array pointer to the snow depth data

2.46.3 lis_log_msgC.c (Source File: lis_log_msgC.c)

This file provides methods for diagnostic and error logging in LIS for codes written in the C language. The diagnostic messages are appended with the appropriate time-date and processor stamps.

2.46.4 lis_log_msgC (Source File: lis_log_msgC.c)

This routine is a C wrapper to the lis_log_msg routine.

REVISION HISTORY:

```
12 Mar 2004: James Geiger; Initial version
```

INTERFACE:

```
void lis_log_msgC(char * string)
```

2.46.5 lis_memory_managementC (Source File: lis_memory_managementC.c)

This file provides methods for dynamic memory management in LIS for codes written in the C language.

2.46.6 lis_calloc (Source File: lis_memory_managementC.c)

Generic call to the C calloc function used in LIS.

REVISION HISTORY:

```
Mar 2004, James Geiger; Initial Specification
```

INTERFACE:

```
void * lis_calloc(size_t n, size_t size, char * caller)
```

2.46.7 lis_malloc (Source File: lis_memory_managementC.c)

Generic call to the C malloc function used in LIS.

REVISION HISTORY:

```
Mar 2004, James Geiger; Initial Specification
```

INTERFACE:

```
void * lis_malloc(size_t size, char * caller)
```

2.46.8 listask_for_point (Source File: listask_for_point.c)

This file provides an algorithm for the decomposition of a given domain. The method is adopted from the Weather Research and Forecasting (WRF) source code.

REVISION HISTORY:

```
Feb 2006  Sujay Kumar Adopted in LIS
```

Part IV
Plugin Interfaces in LIS

3 Plugin interfaces in LIS

The plugin interfaces in LIS are the functional abstractions to represent various user-defined components in LIS. The plugins contain the hot spots or extensible interfaces for incorporating new domains, running modes, LSMS, meteorological analyses, data assimilation tools, and parameters. These adaptable interfaces in LIS enable the reuse of the broad set of data, high performance computing, data management, visualization tools, and the land modeling infrastructure in LIS.

3.1 Fortran: Module Interface runmode_pluginMod (Source File: runmode_pluginMod.F90)

This module contains the definition of the functions used for LIS initialization, execution, and finalization for different running modes in LIS

REVISION HISTORY:

```
21 Oct 05    Sujay Kumar  Initial Specification
```

3.1.1 runmode_plugin (Source File: runmode_pluginMod.F90)

This is a custom-defined plugin point for introducing a new running mode. The interface mandates that the following routines be implemented and registered for each of the running modes in LIS

Initialization Defining the init routines needed for each running mode in LIS. (to be registered using `registerlisinit` and later called using `lisinit`)

Run Define the execution routines needed for each running mode in LIS (to be registered using `registerlisrun` and later called using `lisrun`)

Finalize Define the finalize routines needed for each LIS running mode. (to be registered using `registerlisfinalize` and later called using `lisfinalize`)

The user-defined functions are included in the registry using a single index. For example, consider the definition of a retrospective running mode in LIS. The methods should be defined in the registry as follows, if the index of the mode is defined to be 1.

```
call registerlisinit(1,lisinit_retrospective)
call registerlisrun(1,lisrun_retrospective)
call registerlisfinalize(1,lisfinal_retrospective)
```

The functions registered above are invoked using generic calls as follows:

```
call lisinit(1)  - calls lisinit_retrospective
call lisrun(1)   - calls lisrun_retrospective
call lisfinal(1)   - calls lisfinal_retrospective
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call lisinit(lis%runmode)
call lisrun(lis%runmode)
call lisfinal(lis%runmode)
```

where *lis%runmode* is set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine runmode_plugin
```

3.2 Fortran: Module Interface domain_pluginMod (Source File: domain_pluginMod.F90)

This module contains the definition of the functions used for defining routines that initialize various LIS-domains. The user defined functions are incorporated into the appropriate registry to be later invoked through generic calls.

REVISION HISTORY:

```
17 Feb 2004;   Sujay Kumar  Initial Specification
```

3.2.1 domain_plugin (Source File: domain_pluginMod.F90)

This is a plugin point for introducing a new LIS-domain. The interface mandates that the following interfaces be implemented and registered for each LIS-domain.

read input Routines to read domain specific options and accordingly set up domain decomposition. (to be registered using `registerinput` and later invoked through `readinput` method)

create grid and tile spaces Routines to create the grid and tile spaces (to be registered using `registerdomain` and later invoked through `makedomain` method)

The user-defined functions are included in the registry using a single index. For example, consider a domain definition using lat/lon projection with a SW to NE data ordering. The methods should be defined in the registry as follows, if the index of the domain is defined to be 1.

```
call registerinput(1,readinput_latlon)
call registerdomain(1,createtiles_latlon)
```

The functions registered above are invoked using generic calls as follows:

```
call readinput(1)   - calls readinput_latlon
call makedomain(1)  - calls createtiles_latlon
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call readinput(lis%domain)
call makedomain(lis%domain)
```

where *lis%domain* is set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine domain_plugin
  use pluginIndices
```

3.3 Fortran: Module Interface baseforcing_pluginMod (Source File: baseforcing_pluginMod.F90)

This module contains the definition of the functions used for incorporating a new base forcing scheme. The user defined functions are incorporated into the appropriate registry to be later invoked through generic calls.

REVISION HISTORY:

```
11 Dec 03    Sujay Kumar  Initial Specification
```

3.3.1 baseforcing_plugin (Source File: baseforcing_pluginMod.F90)

INTERFACE:

```
subroutine baseforcing_plugin
```

DESCRIPTION:

This is a plugin point for introducing a new base forcing scheme. The interface mandates that the following interfaces be implemented and registered for each base forcing scheme.

retrieval of forcing data Routines to retrieve forcing data and to interpolate them. (to be registered using `registerget` and later invoked through `retrieveforcing` method)

definition of native domain Routines to define the native domain (to be registered using registerdefinenative and later invoked through `defineNative` method)

temporal interpolation Interpolate forcing data temporally. (to be registered using `registertimeinterp` and later invoked through `timeinterp` method)

The index used in the register calls should be used to select the appropriate base forcing scheme. For example, assume that the GDAS forcing scheme is incorporated in the registry by the following calls.

```
call registerget(1,getgdas)
call registerdefinenative(1,defineNativeGDAS)
call registertimeinterp(1,time_interp_gdas)
call registerforcingfinal(1,gdasforcing_finalize)
```

The index used here to register these functions is 1. To invoke these methods, the corresponding calls will be:

```
call retrieveforcing(1)  - calls getgdas
call defineNative(1)  - calls defineNativeGDAS
call timeinterp(1)  - calls time_interp_gdas
call forcingfinalize(1) - calls gdasforcing_finalize
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call retrieveforcing(lis%force,n)
call defineNative(lis%force)
call timeinterp(lis%force,n)
call forcingfinalize(lis%force)
```

where $lis\%force$ is set through the congiguration utility, enabling the user to select any of the base forcing schemes, at runtime.

3.4 Fortran: Module Interface suppforcing_pluginMod (Source File: suppforcing_pluginMod.F90)

This module contains the definition of the functions used for incorporating a new supplemental forcing scheme. The user defined functions are incorporated into the appropriate registry to be later invoked through generic calls.

REVISION HISTORY:

```
12 Dec 03    Sujay Kumar  Initial Specification
```

3.4.1 suppforcing_plugin (Source File: suppforcing_pluginMod.F90)

This is a plugin point for introducing a new supplemental forcing scheme. The interface mandates that the following interfaces be implemented and registered for each base forcing scheme.

retrieval of forcing data Routines to retrieve forcing data and to interpolate them. (to be registered using `registerreadsupp` and later invoked through `getsuppforc` method)

definition of native domain Routines to define the native domain (to be registered using `registerdefinenativesupp` and later invoked through `defineNativesupp` method)

temporal interpolation Interpolate forcing data temporally. (to be registered using `registersuppti` and later invoked through `timeinterpsupp` method)

Finalize Cleanup allocated structures (to be registered using `registersuppforcingfinal` and later invoked through `suppforcingfinalize` method)

The index used in the register calls should be used to select the appropriate base forcing scheme. For example, assume that the CMAP precipitation forcing scheme is incorporated in the registry by the following calls.

```
call registerreadsupp(1,getcmap)
call registerdefinenativesupp(1,defineNativeCMAP)
call registersuppti(1,time_interp_cmap)
call registersuppforcingfinal(1,cmapforcing_finalize)
```

The index used here to register these functions is 1. To invoke these methods, the corresponding calls will be:

```
call getsuppforc(1)  - calls getcmap
call defineNativesupp(1)  - calls defineNativeCMAP
call timeinterpsupp(1)  - calls time_interp_cmap
call suppforcingfinalize(1) - calls cmapforcing_finalize
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call getsuppforc(lis%suppforce)
call defineNativeSupp(lis%suppforce)
call timeinterpsupp(lis%suppforce)
call suppforcingfinalize(lis%suppforce)
```

where *lis%suppforce* is set through the congiguration utility, enabling the user to select any of the base forcing schemes, at runtime. INTERFACE:

```
subroutine suppforcing_plugin
```

3.5 Fortran: Module Interface lsm_pluginMod (Source File: lsm_pluginMod.F90)

This module contains the definition of the functions used for land surface model initialization, execution, reading and writing of restart files and other relevant land surface model computations, corresponding to each of the LSMs used in LIS.

REVISION HISTORY:

```
09 Oct 03    Sujay Kumar  Initial Specification
```

3.5.1 lsm_plugin (Source File: lsm_pluginMod.F90)

This is a custom-defined plugin point for introducing a new LSM. The interface mandates that the following routines be implemented and registered for each of the LSM that is included in LIS.

Initialization Definition of LSM variables (to be registered using `registerlsmini` and later called using `lsmini`)

Setup Initialization of parameters (to be registered using `registerlsmsetup` and later called using `lsmsetup`)

Run Routines to execute LSM on a single gridcell for single timestep (to be registered using `registerlsmrun` and later called using `lsmrun`)

Read restart Routines to read a restart file for an LSM run (to be registered using `registerlsmrestart` and later called using `lsmrestart`)

Output Routines to write output (to be registered using `registerlsmoutput` and later called using `lsmoutput`)

Forcing transfer to model tiles Routines to transfer an array of given forcing to model tiles (to be registered using `registerlsmf2t` and later called using `lsmf2t`)

Write restart Routines to write a restart file (to be registered using `registerlsmwrst` and later called using `lsmwrst`)

Finalize Routines to cleanup LSM data structures (to be registered using `registerlsmfinalize` and later called using `lsmfinalize`)

The user-defined functions are included in the registry using a single index. For example, consider the Noah LSM is incorporated in the registry with an index of 1 and is invoked later by the following calls

```
call registerlsmini(1,noah_varder_ini)
call registerlsmsetup(1,noah_setup)
call registerlsmf2t(1,noah_f2t)
call registerlsmrun(1,noah_main)
call registerlsmrestart(1,noahrst)
call registerlsmdynsetup(1,noah_dynsetup)
call registerlsmoutput(1,noah_output)
call registerlsmwrst(1,noah_writerst)
call registerlsmfinalize(1,noah_finalize)
```

The functions registered above are invoked using generic calls as follows:

```
call lsmini(1)      - calls noah_varder_ini
call lsmsetup(1)    - calls noah_setup
call lsmf2t(1)      - calls noah_f2t
call lsmrun(1)      - calls noah_main
```

```
call lsmdynsetup(1) - calls noah_dynsetup
call lsmoutput(1)   - calls noah_output
call lsmwrst(1)     - calls noah_writerst
call lsmfinalize(1) - calls noah_finalize
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call lsmini(lis%lsm)
call lsmsetup(lis%lsm)
call lsmf2t(lis%lsm)
call lsmrun(lis%lsm)
call lsmdynsetup(lis%lsm)
call lsmoutput(lis%lsm)
call lsmwrst(lis%lsm)
call lsmfinalize(lis%lsm)
```

where *lis%lsm* is set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine lsm_plugin
```

3.6 Fortran: Module Interface param_pluginMod (Source File: param_pluginMod.F90)

This module contains the definition of the functions used for defining routines to read various sources of parameters maps. The user defined functions are incorporated into the appropriate registry to be later invoked through generic calls.

REVISION HISTORY:

```
11 Dec 03    Sujay Kumar  Initial Specification
```

3.6.1 topo_plugin (Source File: param_pluginMod.F90)

This is a plugin point for introducing new topography datasets. The interface mandates that the following routines be implemented and registered for each parameter data source.

read the elevation data Routines to retrieve the elevation data (to be registered using `registerreadelev` and later called using the generic `readelev` method)

read the slope data Routines to retrieve the slope data (to be registered using `registerreadslope` and later called using the generic `readslope` method)

read the aspect data Routines to retrieve the aspect data (to be registered using `registerreadaspect` and later called using the generic `readaspect` method)

read the curvature data Routines to retrieve the curvature data (to be registered using `registerreadcurv` and later called using the generic `readcurv` method)

The user-defined functions are included in the registry using two indices. For example, consider the incorporation of the topography datasets from the GTOPO30 source with LIS using a latlon projection. The methods should be defined in the registry as follows, if the index of the source is defined to 1 and latlon projection in LIS uses and index of 1

```
call registerreadelev(1,1,read_elev_gtopo30)
call registerreadslope(1,1,read_slope_gtopo30)
call registerreadaspect(1,1,read_aspect_gtopo30)
call registerreadcurv(1,1,read_curv_gtopo30)
```

The functions registered above are invoked using generic calls as follows:

```
call readelev(1,1)   - calls read_elev_gtopo30
call readslope(1,1)  - calls read_slope_gtopo30
call readaspect(1,1)  - calls read_aspect_gtopo30
call readcurv(1,1)  - calls read_curv_gtopo30
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call readelev(lis%domain, lis%toposrc)
call readslope(lis%domain, lis%toposrc)
call readaspect(lis%domain, lis%toposrc)
call readcurv(lis%domain, lis%toposrc)
```

where *lis%domain* and *lis%toposrc* are set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine topo_plugin
```

3.6.2 soils_plugin (Source File: param_pluginMod.F90)

This is a plugin point for introducing new soil parameter datasets. The interface mandates that the following routines be implemented and registered for each parameter data source.

read the texture data Routines to retrieve the texture data (to be registered using `registerreadsoiltexture` and later called using the generic `readtexture` method)

read the sand fraction data Routines to retrieve the sand fraction data (to be registered using `registerreadsand` and later called using the generic `readsand` method)

read the clay fraction data Routines to retrieve the clay fraction data (to be registered using `registerreadclay` and later called using the generic `readclay` method)

read the silt fraction data Routines to retrieve the silt fraction data (to be registered using `registerreadsilt` and later called using the generic `readsilt` method)

read the soil color data Routines to retrieve the soil color data (to be registered using `registerreadcolor` and later called using the generic `readcolor` method)

read the soil porosity data Routines to retrieve the soil porosity data (to be registered using `registerreadporosity` and later called using the generic `readporosity` method)

read the saturated matric potential data Routines to retrieve the saturated matric potential data (to be registered using `registerreadpsisat` and later called using the generic `readpsisat` method)

read the hydraulic conductivity data Routines to retrieve the hydraulic conductivity data (to be registered using `registerreadksat` and later called using the generic `readksat` method)

read the b parameter data Routines to retrieve the b parameter data (to be registered using `registerreadbexp` and later called using the generic `readbexp` method)

read the quartz data Routines to retrieve the quartz data (to be registered using `registerreadquartz` and later called using the generic `readquartz` method)

The user-defined functions are included in the registry using two indices. For example, consider the incorporation of the soil datasets from the FAO source with LIS using a latlon projection. The methods should be defined in the registry as follows (only a subset of the parameters is shown below, others can be defined in the same mannter), if the index of the source is defined to 1 and latlon projection in LIS uses and index of 1

```
call registerreadsand(1,1,read_faosand)
call registerreadclay(1,1,read_faoclay)
call registerreadsilt(1,1,read_faosilt)
```

The functions registered above are invoked using generic calls as follows:

```
call readsand(1,1)  - calls read_faosand
call readclay(1,1)  - calls read_faoclay
call readsilt(1,1)  - calls read_faosilt
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call readsand(lis%domain, lis%soilsrc)
call readclay(lis%domain, lis%soilsrc)
call readsilt(lis%domain, lis%soilsrc)
```

where *lis%domain* and *lis%soilsrc* are set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine soils_plugin
```

3.6.3 laisai_plugin (Source File: param_pluginMod.F90)

This is a plugin point for introducing new LAI/SAI datasets. The interface mandates that the following routines be implemented and registered for each parameter data source.

read the LAI data Routines to retrieve the LAI data (to be registered using `registerreadlai` and later called using the generic `readlai` method)

read the SAI data Routines to retrieve the SAI data (to be registered using `registerreadsai` and later called using the generic `readsai` method)

The user-defined functions are included in the registry using two indices. For example, consider the incorporation of the LAI/SAI datasets from the AVHRR source with LIS using a latlon projection. The methods should be defined in the registry as follows if the index of the source is defined to 1 and latlon projection in LIS uses and index of 1

```
call registerreadlai(1,1,read_ll_avhrrlai)
call registerreadsai(1,1,read_ll_avhrrsai)
```

The functions registered above are invoked using generic calls as follows:

```
call readlai(1,1)  - calls read_ll_avhrrlai
call readsai(1,1)  - calls read_ll_avhrrsai
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call readlai(lis%domain,lislaisrc)
call readsai(lis%domain,lislaisrc)
```

where *lis%domain* and *lis%laisrc* are set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine laisai_plugin
```

3.6.4 landcover_plugin (Source File: param_pluginMod.F90)

This is a plugin point for introducing new landcover datasets. The interface mandates that the following routines be implemented and registered for each parameter data source.

read the landmask data Routines to retrieve the landmask data (to be registered using `registerreadmask` and later called using the generic `readmask` method)

read the landcover data Routines to retrieve the landcover data (to be registered using `registerreadlc` and later called using the generic `readlandcover` method)

The user-defined functions are included in the registry using two indices. For example, consider the incorporation of the landcover datasets from the UMD source with LIS using a latlon projection. The methods should be defined in the registry as follows if the index of the source is defined to 1 and latlon projection in LIS uses and index of 1

```
call registerreadmask(1,1,read_latlon_umdmask)
call registerreadlc(1,1,read_latlon_umdlc)
```

The functions registered above are invoked using generic calls as follows:

```
call readmask(1,1)  - calls read_latlon_umdmask
call readlc(1,1)  - calls read_latlon_umdlc
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call readmask(lis%domain,lis%vegsrc)
call readlandcover(lis%domain,lis%vegsrc)
```

where *lis%domain* and *lis%vegsrc* are set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine landcover_plugin
```

3.6.5 tbot_plugin (Source File: param_pluginMod.F90)

This is a plugin point for introducing new bottom temperature datasets. The interface mandates that the following routines be implemented and registered for each parameter data source.

read the bottome temperature data Routines to retrieve the bottom temperature data (to be registered using `registerreadtbot` and later called using the generic `readtbot` method)

The user-defined functions are included in the registry using a single index. For example, consider the incorporation of the tbot datasets to be used in LIS running lat/lon projection. The methods should be defined in the registry as follows, if the index of the domain is defined to 1.

```
call registerreadtbot(1,read_statictbot)
```

The functions registered above are invoked using generic calls as follows:

```
call readtbot(1)  - calls read_statictbot
```

In the LIS code, the above calls are typically invoked in the following manner.

```
 call readtbot(lis%domain)
```

where *lis%domain* is set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine tbot_plugin
```

3.6.6 gfrac_plugin (Source File: param_pluginMod.F90)

This is a plugin point for introducing new greenness datasets. The interface mandates that the following routines be implemented and registered for each parameter data source.

read the greenness data Routines to retrieve the greenness data (to be registered using `registerreadgfrac` and later called using the generic `readgfrac` method)

The user-defined functions are included in the registry using two indices. For example, consider the incorporation of the greenness datasets from the NCEP source with LIS using a latlon projection. The methods should be defined in the registry as follows if the index of the source is defined to 1 and latlon projection in LIS uses and index of 1

```
call registerreadgfrac(1,1,read_llgfrac)
```

The functions registered above are invoked using generic calls as follows:

```
call readgfrac(1,1)  - calls read_llgfrac
```

In the LIS code, the above calls are typically invoked in the following manner.

```
 call readgfrac(lis%domain,lis%gfracsrc)
```

where *lis%domain* and *lis%gfracsrc* are set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine gfrac_plugin
```

3.6.7 alb_plugin (Source File: param_pluginMod.F90)

This is a plugin point for introducing new albedo datasets. The interface mandates that the following routines be implemented and registered for each parameter data source.

read the max snow albedo data Routines to retrieve the max snow albedo data (to be registered using `registerreadmxsnoalb` and later called using the generic `readmxsnoalb` method)

read the albedo climatology data Routines to retrieve the albedo climatology data (to be registered using `registerreadalbedo` and later called using the generic `readalbedo` method)

The user-defined functions are included in the registry using two indices. For example, consider the incorporation of the albedo datasets from the NCEP source with LIS using a latlon projection. The methods should be defined in the registry as follows if the index of the source is defined to 1 and latlon projection in LIS uses and index of 1

```
call registerreadmxsnoalb(1,1,read_llmxsnoalb)
call registerreadalbedo(1,1,read_llalbedo)
```

The functions registered above are invoked using generic calls as follows:

```
call readmxsnoalb(1,1)  - calls read_llmxsnoalb
call readalbedo(1,1)    - calls read_llalbedo
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call readmxsnoalb(lis%domain,lis%albedosrc)
call readalbedo(lis%domain,lis%albedoarc)
```

where *lis%domain* and *lis%albedosrc* are set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine alb_plugin
```

3.6.8 snow_plugin (Source File: param_pluginMod.F90)

This is a plugin point for introducing new snow datasets. The interface mandates that the following routines be implemented and registered for each parameter data source.

read the snowdepth data Routines to retrieve the snow depth data (to be registered using `registerreadsnowdepth` and later called using the generic `readsnowdepth` method)

The user-defined functions are included in the registry using two indices. For example, consider the incorporation of the snowdepth datasets from AFWA with LIS using a latlon projection. The methods should be defined in the registry as follows if the index of the source is defined to 1 and latlon projection in LIS uses and index of 1

```
call registerreadsnowdepth(1,1,read_llsnowdepth)
```

The functions registered above are invoked using generic calls as follows:

```
call readsnowdepth(1,1)  - calls read_llsnowdepth
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call readsnowdepth(lis%domain,lis%snowsrc)
```

where *lis%domain* and *lis%snowsrc* are set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine snow_plugin
```

3.7 Fortran: Module Interface dataassim_pluginMod.F90 (Source File: dataassim_pluginMod.F90)

This module contains the definition of the functions used for defining routines that performs data assimilation. The user defined functions are incorporated into the appropriate registry to be later invoked through generic calls.

REVISION HISTORY:

```
27 Feb 2005;   Sujay Kumar  Initial Specification
```

3.7.1 dataassim_plugin (Source File: dataassim_pluginMod.F90)

This is a plugin point for introducing a new data assimilation scheme. As explaind in the `dataassim_module`, there are three different abstractions associated with the data assimilation implementation. The implementations defined in this subroutine complete the "wirings" required to complete the data assimilation algorithm-related abstractions. Other required interfaces defined in `lsmda_pluginMod` and `dataobs_pluginMod` should be completed to complete a successful implementation.
The following interfaces should be implemented in this routine.

init Initialization of data and memory structures (to be registered using `registerdainit` and later called through `dataassiminit`)

Assimilate Routines to perform data assimilation using observations. (to be registered using `registerassim` and later called through `assimilate`)

Output Routines to perform data assimilation related output (to be registered using `registerdaoutput` and later called through `daoutput`)

Finalize Routines to perform data assimilation related cleanups (to be registered using `registerdafinalize` and later called throuh `dafinalize`)

A single index is used to register a routine, corresponding to the type of data assimilation algorithm used. For example, consider an instance where Ensemble Kalman Filter (EnKF) algorithm is employed, and is denoted by an index 3. The methods should be defined in the registry as follows.

```
call registerdainit(3,enkf_init)
call registerassim(3,enkf_assim)
call registerdaoutput(3,enkf_output)
call registerdafinalize(3,enkf_final)
```

The functions registered above are invoked using generic calls as follows:

```
call dainit(3) -  calls enkf_init
call assimilate(3) - calls enkf_assim
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call dainit(lis%daalg)
call assimilate(lis%daalg)
```

where *lis%daalg* is set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine dataassim_plugin
```

3.8 Fortran: Module Interface dataobs_pluginMod (Source File: dataobs_pluginMod.F90)

This module contains the definition of the functions that are used to read observation data for data assimiliation. The user defined functions are incorporated into the appropriate registry to be later invoked through generic calls.

REVISION HISTORY:

```
27 Feb 2005;   Sujay Kumar  Initial Specification
```

3.8.1 dataobs_plugin (Source File: dataobs_pluginMod.F90)

This is a plugin point for introducing a routines to handle the observation data for assimilation.As explaind in the `dataassim_module`, there are three different abstractions associated with the data assimilation implementation. The implementations defined in this subroutine complete the "wirings" required to complete the observation data-related abstractions. Other required interfaces defined in `lsmda_pluginMod` and `dataassim_pluginMod` should be completed for a successful implementation.
The following interfaces should be implemented in this routine.

Setup Initialization of data and memory structures (to be registered using `registerdaobssetup` and later called through `daobssetup`)

read observations Routines to read the observation data and perform any spatial transformation. (to be registered using `registerdaobs` and later called through `readobservations`)

get number of selected observations routines to retrieve the number of selected observations for the selected modeling point. (to be registered using `registergetnso` and later called through `getselctedobsnumber`

The user-defined functions are included in the registry using a single index. For example, consider an instance where soil moisture assimilation using TMI soil moisture data is conducted. The methods should be defined in the registry as follows, if the index of the 'assimilation set' (assimilating TMI to update soil moisture variables) is defined to be 1

```
call registerdaobssetup(1,TMIsmobs_setup)
call registerdaobs(1,read_TMIsm)
call registergetnso(1,getNSO_TMIsm)
```

The functions registered above are invoked using generic calls as follows:

```
call daobssetup(1)       -  calls TMIsmobs_setup
call readobservations(1) -  calls read_TMIsm
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call daobssetup(lis%daset, )
call readobservations(lis%daset)
```

where *lis%daset* is set through the configuration utility, enabling the user to make selections at runtime.

INTERFACE:

```
subroutine dataobs_plugin
```

3.9 Fortran: Module Interface perturb_pluginMod (Source File: perturb_pluginMod.F90)

This module contains the definition of the functions used for defining routines that perform pertubations of model states, forcing, or parameters. The user defined functions are incorporated into the appropriate registry to be later invoked through generic calls.

REVISION HISTORY:

```
08Jul2005;   Sujay Kumar  Initial Specification
```

INTERFACE:

3.9.1 perturb_plugin (Source File: perturb_pluginMod.F90)

This is a plugin point for introducing a new forcing perturbation scheme. The interface mandates that the following interfaces be implemented for each scheme.

Setup Initialization of data and memory structures (to be registered using `registerperturbsetup` and later called using the generic call `perturbinit`)

Forecast Routines to compute perturbations (to be registered using `registerperturb` and later called using the generic call `perturb`)

The user-defined functions are included in the registry using a single index. For example, consider a new scheme called 'FOO' The methods should be defined in the registry as follows, if the index of the scheme is defined to be 1.

```
call registerperturbsetup(1,foo_setup)
call registerperturb(1,foo_method)
```

The functions registered above are invoked using generic calls as follows:

```
call perturbinit(1)  - calls foo_setup
call perturb(1)      - calls foo_method
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call perturbinit(lis%perturb)
call perturb(lis%perturb)
```

where *lis%perturb* is set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine perturb_plugin
```

3.10 Fortran: Module Interface lsmda_pluginMod (Source File: lsmda_pluginMod.F90)

This module contains the definition of the functions that need to be defined to enable the use of a land surface model in a data assimilation setup.

REVISION HISTORY:

```
29 Mar 07    Sujay Kumar  Separated from the lsm module implementation.
```

3.10.1 lsmda_plugin (Source File: lsmda_pluginMod.F90)

This is a custom-defined plugin point for introducing a new LSM in a data assimilation mode. The interface mandates that a number of routines be implemented and registered for each of the LSM that is used in a data assimilation setup. Currently two algorithms are supported, Direct Insertion (DI), and Ensemble Kalman Filter (EnKF).
For use with EnKF, the following routines need to be implemented.

Return the state prognostic variables Routine that retrieves the specified state prognostic variable(s) (to be registered using `registergetstatevar` and later called through `getstatevar` method)

Set the state prognostic variables Routine that sets the specified state prognostic variable(s) (to be registered using `registersetstatevar` and later called through `setstatevar` method)

Retrieve "Obspred" Routine that retrieves the model's estimate of observations. (to be registered using `registergetobspred` and later called through `getobspred` method)

QC the LSM state Routine that QCs the given LSM state for physical consistency. (to be registered using `registerqclsmstate` and later called through `qclsmstate` method)

Scale LSM state Scale the LSM state variables in order to change the variables to similar scales so that the matrices are well-conditioned (to be registered using `registerscalelsmstate` and later called through `scalelsmstate` method)

Descale LSM state Descale the LSM state variables from the scaled state. (to be registered using `registerdescalelsmstat` and later called through `descalelsmstate` method)

For use with DI, the following routines are required.

Return the state prognostic variables Routine that retrieves the specified state prognostic variable(s) (to be registered using `registergetstatevar` and later called through `getstatevar` method)

Set the state prognostic variables Routine that sets the specified state prognostic variable(s) (to be registered using `registersetstatevar` and later called through `setstatevar` method)

Transform Observations Routine that transforms observations to the LSM state space (could be as simple as a unit conversion). (to be registered using `registerobstransform` and later called through `obstransform` method)

Map observations to LSM space Routine that maps observations to the LSM state variables. (to be registered using `registermapobstolsm` and later called through `mapobstolsm` method)

The user-defined functions are included in the registry using a user-selected indices. For example, consider the Noah LSM is incorporated in the registry by the following calls. In case of registry functions defined with two indices, the first index refers to Noah LSM and the second index refers to the "assimilation set" (in the following example, soil moisture assimilation using TMI)

```
call registergetstatevar(1,1,noah_getsoilm)
call registersetstatevar(1,1,noah_setsoilm)
```

The functions registered above are invoked using generic calls as follows:

```
call getstatevar(1,1) - calls noah_getsoilm
call setstatevar(1,1) - calls noah_setsoilm
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call getstatevar(lis%lsm, lis%daset)
call setstatevar(lis%lsm, lis%daset)
call obstransform(lis%lsm, lis%daset)
```

where *lis%lsm* and *lis%daset* are set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine lsmda_plugin
```

3.11 Fortran: Module Interface lsmcpl_pluginMod (Source File: lsmcpl_pluginMod.F90)

This module contains the definition of the functions that need to be defined to enable the use of a land surface model in coupled land-atmosphere setups.

REVISION HISTORY:

```
09 Oct 07    Sujay Kumar  Initial Specification
```

3.11.1 lsmcpl_plugin (Source File: lsmcpl_pluginMod.F90)

This is a custom-defined plugin point for introducing a LSM in a coupled land-atmosphere setup. The interface mandates that the following routines be implemented and registered for each of the LSM that is used.
If used for a coupled run with an atmospheric component:

Specify an export state Routines to specify the variables to be sent to the atmos. component (to be registered using `registerlsmsetexport` and later called using `lsmsetexport`)

The user-defined functions are included in the registry using two indices. For example, consider the Noah LSM used in a mode coupled to WRF, the setexport method is incorporated in the registry with an index of 1 corresponding to Noah and 3 corresponding to the runmode coupled to WRF as follows:

```
call registerlsmsetexport(1,3,noah_setwrfexport)
```

The function registered above is invoked using a generic call as follows:

```
call lsmsetexport(1,3)      - calls noah_setwrfexport
```

In the LIS code, the above call is typically invoked in the following manner.

```
call lsmsetexport(lis%lsm, lis%runmode)
```

where *lis%lsm* and utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine lsmcpl_plugin
```

3.12 Fortran: Module Interface pluginIndices (Source File: pluginIndices.F90)

The code in this file provides values of indices used to to register functions in the plugin modules
The index definitions are simply a convention The user may change these options, and the lis.config should be changed appropriately to ensure that the correct function is called at run time

REVISION HISTORY:

```
 23 Oct 2006    Sujay Kumar  Initial Specification

!-------------------------------------------------------------------------
! Run modes
!-------------------------------------------------------------------------
   integer, parameter :: retroId     = 1
   integer, parameter :: agrmetrunId = 2
   integer, parameter :: wrfcplId    = 3
   integer, parameter :: gcecplId    = 4
   integer, parameter :: gfscplId    = 5
!-------------------------------------------------------------------------
! Domains
!-------------------------------------------------------------------------
   integer, parameter :: latlonId    = 1
   integer, parameter :: gswpdomId   = 2
   integer, parameter :: polarId     = 3
   integer, parameter :: lambertId   = 4
   integer, parameter :: mercId      = 5
   integer, parameter :: afwaId      = 6
   integer, parameter :: psglobalId  = 7
   integer, parameter :: catdomainId = 8
   integer, parameter :: gaussId     = 9
   integer, parameter :: hrapId      = 10
!-------------------------------------------------------------------------
! LSMS
```

```
!-------------------------------------------------------------------------
   integer, parameter :: templateLSMId = 0
   integer, parameter :: noahId     = 1
   integer, parameter :: clmId      = 2
   integer, parameter :: vicId      = 3
   integer, parameter :: mosaicId   = 4
   integer, parameter :: hyssibId   = 5
   integer, parameter :: sib2Id     = 6
   integer, parameter :: catchId    = 7
   integer, parameter :: sacId      = 8
   integer, parameter :: snow17Id   = 9
   integer, parameter :: sacsnow17Id= 10
   integer, parameter :: sib3Id     = 11
   integer, parameter :: mlbcId     = 12
   integer, parameter :: csuId      = 13
!-------------------------------------------------------------------------
! base forcings
!-------------------------------------------------------------------------
   integer, parameter :: templateBaseForcId = 0
   integer, parameter :: gdasId             = 1
   integer, parameter :: geosId             = 2
   integer, parameter :: ecmwfId            = 3
   integer, parameter :: nldasId            = 4
   integer, parameter :: gswpId             = 5
   integer, parameter :: ecmwfreanalId      = 6
   integer, parameter :: agrmetId           = 7
   integer, parameter :: nldas2Id           = 8
   integer, parameter :: princetonId        = 9
   integer, parameter :: saldasId           = 10
   integer, parameter :: rhoneId            = 11
   integer, parameter :: gswp1Id            = 12
!-------------------------------------------------------------------------
! supplemental forcings
!-------------------------------------------------------------------------
   integer, parameter :: agrradId      = 1
   integer, parameter :: cmapId        = 2
   integer, parameter :: ceopId        = 4
   integer, parameter :: scanId        = 5
   integer, parameter :: TRMM3B42RTId  = 11
   integer, parameter :: TRMM3B42V6Id  = 12
   integer, parameter :: cmorphId      = 14
   integer, parameter :: stg2Id        = 15
   integer, parameter :: stg4Id        = 16
   integer, parameter :: d2pcpcarId    = 17
   integer, parameter :: d2pcpoklId    = 18
!-------------------------------------------------------------------------
! land surface parameters
!-------------------------------------------------------------------------
!-------------------------------------------------------------------------
! Topography sources
!-------------------------------------------------------------------------
   integer, parameter :: gtopoId = 1
   integer, parameter :: gfsTopoId = 3
!-------------------------------------------------------------------------
! Soils sources
```

```
!-------------------------------------------------------------------------
   integer, parameter :: faoSoilId     = 1
   integer, parameter :: statsgoSoilId = 2
   integer, parameter :: gfssoilId = 3
!-------------------------------------------------------------------------
! LAI/SAI sources
!-------------------------------------------------------------------------
   integer, parameter :: avhrrlaiId  = 1
   integer, parameter :: modislaiId  = 2
   integer, parameter :: catlaiId    = 4
!-------------------------------------------------------------------------
! Landcover sources
!-------------------------------------------------------------------------
   integer, parameter :: umdlcId   = 1
   integer, parameter :: usgslcId  = 2
   integer, parameter :: gfslcId   = 3
   integer, parameter :: igbplcId  = 4
!-------------------------------------------------------------------------
! greenness data sources
!-------------------------------------------------------------------------
   integer, parameter :: gfracClimoId = 1
   integer, parameter :: gfraccatId   = 2
   integer, parameter :: gfsgfracId = 3
!-------------------------------------------------------------------------
! albedo data sources
!-------------------------------------------------------------------------
   integer, parameter :: albedoClimoId = 1
   integer, parameter :: gfsalbId = 3
!-------------------------------------------------------------------------
! snow data sources
!-------------------------------------------------------------------------
   integer, parameter :: snodepId = 1

!-------------------------------------------------------------------------
! data assimilation algorithms
!-------------------------------------------------------------------------
   integer, parameter :: diId         = 1
   integer, parameter :: gmaoenkfId   = 2
!-------------------------------------------------------------------------
! perturbation algorithms
!-------------------------------------------------------------------------
   integer, parameter :: staticId   = 1
   integer, parameter :: gmaopertId = 2

!-------------------------------------------------------------------------
! Assimilation set
! DA variable being updated with observations
!-------------------------------------------------------------------------
   integer, parameter :: synsm1Id     = 1
   integer, parameter :: synsm2Id     = 2
   integer, parameter :: synswe1Id    = 3
   integer, parameter :: synsca1Id    = 4

   integer, parameter :: isscpTskinId = 8
   integer, parameter :: amsresmId    = 9
```

```
   integer, parameter :: synsm3Id  = 5
   integer, parameter :: synsm4Id  = 6
   integer, parameter :: synlst1Id = 7
!-----------------------------------------------------------------------
! Bias Estimation Algorithms
!-----------------------------------------------------------------------
   integer, parameter :: gmaobiasId = 1
```

Part V
Interpolation Tools in LIS

4 Interpolation tools in LIS

The spatial interpolation tools in LIS are used for the geospatial transformation of an input data to the LIS grid. The tools in LIS are based on the ipolates package developed at NCEP. Currently LIS supports the transformation and interpolation of polar stereographic, lambert conformal, and mercator grid projection data on to an equidistant cylindrical (lat/lon) grid projection. Bilinear, conservative and neighbor search interpolations are supported.

4.0.1 bilinear_interp_input (Source File: bilinear_interp_input.F90)

INTERFACE:

```
subroutine bilinear_interp_input (gridDesci,gridDesco,npts,&
     rlat,rlon,n11,n12,n21,n22,w11,w12,w21,w22)

  implicit none
```

ARGUMENTS:

```
  real, intent(in)    :: gridDesci(50)
  real                :: gridDesco(50)
  integer             :: npts
  real                :: rlat(npts)
  real                :: rlon(npts)
  integer             :: n11(npts),n12(npts),n21(npts),n22(npts)
  real                :: w11(npts),w12(npts),w21(npts),w22(npts)
```

DESCRIPTION:

This subprogram performs issues calls to compute the interpolation weights and neighbor information for bilinear interpolation,from any grid to any grid for scalar fields. The grids are defined by their grid description arrays.
The grid description arrays are based on the decoding schemes used by NCEP. However, in order to remove the integer arithmetic employed in the original ipolates, the routines are rewritten using real number manipulations. The general structure remains the same.
The current code recognizes the following projections:
(gridDesc(1)=0) equidistant cylindrical
(gridDesc(1)=1) mercator cylindrical
(gridDesc(1)=3) lambert conformal conical
(gridDesc(1)=4) gaussian cylindrical (spectral native)
(gridDesc(1)=5) polar stereographic azimuthal
where gridDesc could be defined for either the input grid or the output grid.
The arguments are:

gridDesci input grid description parameters

gridDesco output grid description parameters

npts number of points to in the output field

rlat output latitudes in degrees

rlon output longitudes in degrees

w11,w12,w21,w22 weights to be used for interpolation

n11,n12,n21,n22 index of neighbor points

The routines invoked are:

compute_earth_coord (4.0.7)
Computes the earth coordinates for the output grid

compute_grid_coord (4.0.14)
Computes the grid coordinates of the input grid, based on the earth coordinates of the output grid.

get_field_pos (4.0.21)
computes the field position for a given point

4.0.2 bilinear_interp (Source File: bilinear_interp.F90)

REVISION HISTORY:

```
04-10-96  Mark Iredell; Initial Specification
05-27-04  Sujay Kumar : Modified verision with floating point arithmetic
```

INTERFACE:

```
subroutine bilinear_interp(gridDesco,ibi,li,gi,ibo,lo,go,mi,mo, &
     rlat,rlon,w11,w12,w21,w22,n11,n12,n21,n22,udef,iret)
```

USES:

```
implicit none
```

ARGUMENTS:

```
real      :: gridDesco(50)
integer   :: ibi
integer   :: ibo
integer   :: mi
integer   :: mo
logical*1 :: li(mi)
logical*1 :: lo(mo)
real      :: gi(mi)
real      :: go(mo)
real      :: rlat(mo)
real      :: rlon(mo)
real      :: w11(mo),w12(mo), w21(mo),w22(mo)
integer   :: n11(mo),n12(mo),n21(mo),n22(mo)
real      :: udef
integer   :: iret
```

DESCRIPTION:

This subprogram performs bilinear interpolation from any grid to any grid for scalar fields. The routine is based on the spatial interpolation package ipolates from NCEP.
The algorithm simply computes (weighted) averages of bilinearly interpolated points arranged in a square box centered around each output grid point and stretching nearly halfway to each of the neighboring grid points. the grids are defined by their grid description arrays.
The grid description arrays are based on the decoding schemes used by NCEP. However, in order to remove the integer arithmetic employed in the original ipolates, the routines are rewritten using real number manipulations. The general structure remains the same.
The current code recognizes the following projections:
(gridDesc(1)=0) equidistant cylindrical
(gridDesc(1)=1) mercator cylindrical

(gridDesc(1)=3) lambert conformal conical
(gridDesc(1)=4) gaussian cylindrical (spectral native)
(gridDesc(1)=5) polar stereographic azimuthal
where gridDesc could be defined for either the input grid or the output grid. The routine also returns the the number of output grid points and their latitudes and longitudes are also returned. The input bitmaps will be interpolated to output bitmaps. output bitmaps will also be created when the output grid extends outside of the domain of the input grid. the output field is set to 0 where the output bitmap is off.
The arguments are:

gridDesco output grid description parameters

ibi integer input bitmap flags

li logical input bitmaps

gi real input fields to interpolate

ibo integer output bitmap flags

lo logical output bitmaps

go real output fields interpolated

mi integer dimension of input grid fields

mo integer dimension of output grid fields

rlat output latitudes in degrees

rlon output longitudes in degrees

w11,w12,w21,w22 weights to be used for interpolation

n11,n12,n21,n22 index of neighbor points

udef undefined value to be used

iret return code (0-success)

The routines invoked are:

polfixs (4.0.22)
Apply corrections for poles

4.0.3 conserv_interp_input (Source File: conserv_interp_input.F90)

INTERFACE:

```
subroutine conserv_interp_input(gridDesci,gridDesco,npts,&
     rlat2,rlon2,n112,n122,n212,n222,w112,w122,w212,w222)

  implicit none
```

ARGUMENTS:

```
real, intent(in)    :: gridDesci(50)
real                :: gridDesco(50)
integer             :: npts
real                :: rlat2(npts)
real                :: rlon2(npts)
integer             :: n112(npts,25),n122(npts,25),&
     n212(npts,25),n222(npts,25)
real                :: w112(npts,25),w122(npts,25),&
     w212(npts,25),w222(npts,25)
```

DESCRIPTION:

This subprogram performs issues calls to compute the interpolation weights and neighbor information for budget bilinear interpolation,from any grid to any grid for scalar fields. The grids are defined by their grid description arrays.
The grid description arrays are based on the decoding schemes used by NCEP. However, in order to remove the integer arithmetic employed in the original ipolates, the routines are rewritten using real number manipulations. The general structure remains the same.
The current code recognizes the following projections:
(gridDesc(1)=0) equidistant cylindrical
(gridDesc(1)=1) mercator cylindrical
(gridDesc(1)=3) lambert conformal conical
(gridDesc(1)=4) gaussian cylindrical (spectral native)
(gridDesc(1)=5) polar stereographic azimuthal
where gridDesc could be defined for either the input grid or the output grid.
The arguments are:

gridDesci input grid description parameters

gridDesco output grid description parameters

npts number of points to in the output field

rlat2 output latitudes in degrees

rlon2 output longitudes in degrees

w112,w122,w212,w222 weights to be used for interpolation

n112,n122,n212,n222 index of neighbor points

The routines invoked are:

compute_earth_coord (4.0.7
Computes the earth coordinates for the output grid

compute_grid_coord (4.0.14
Computes the grid coordinates of the input grid, based on the earth coordinates of the output grid.

get_field_pos (4.0.21)
computes the field position for a given point

4.0.4 conserv_interp (Source File: conserv_interp.F90)

REVISION HISTORY:

```
04-10-96  Mark Iredell; Initial Specification
05-27-04  Sujay Kumar : Modified verision with floating point arithmetic
```

INTERFACE:

```
subroutine conserv_interp(gridDesco,ibi,li,gi,ibo,lo,go,mi,mo,&
     rlat,rlon,w11,w12,w21,w22,n11,n12,n21,n22,udef,iret)
  implicit none
```

ARGUMENTS:

```
  integer, parameter :: nb3 = 25, nb4 = 25
  real      :: gridDesco(50)
  integer   :: ibi
  integer   :: ibo
  integer   :: mi
  integer   :: mo
  logical*1 :: li(mi)
  logical*1 :: lo(mo)
  real      :: gi(mi)
  real      :: go(mo)
  real      :: rlat(mo)
  real      :: rlon(mo)
  integer   :: n11(mo,nb4),n21(mo,nb4),n12(mo,nb4),n22(mo,nb4)
  real      :: w11(mo,nb4),w21(mo,nb4),w12(mo,nb4),w22(mo,nb4)
  real      :: udef
  integer   :: iret
```

DESCRIPTION:

This subprogram performs budget interpolation from any grid to any grid for scalar fields. The routine is based on the spatial interpolation package ipolates from NCEP.
The algorithm simply computes (weighted) averages of bilinearly interpolated points arranged in a square box centered around each output grid point and stretching nearly halfway to each of the neighboring grid points, using 25 points for computing the average.
The grid description arrays are based on the decoding schemes used by NCEP. However, in order to remove the integer arithmetic employed in the original ipolates, the routines are rewritten using real number manipulations. The general structure remains the same.
The current code recognizes the following projections:
(gridDesc(1)=0) equidistant cylindrical
(gridDesc(1)=1) mercator cylindrical
(gridDesc(1)=3) lambert conformal conical
(gridDesc(1)=4) gaussian cylindrical (spectral native)
(gridDesc(1)=5) polar stereographic azimuthal
where gridDesc could be defined for either the input grid or the output grid. The routine also returns the the number of output grid points and their latitudes and longitudes are also returned. The input bitmaps will be interpolated to output bitmaps. output bitmaps will also be created when the output grid extends outside of the domain of the input grid. the output field is set to 0 where the output bitmap is off.
The arguments are:

gridDesco output grid description parameters

ibi integer input bitmap flags

li logical input bitmaps

gi real input fields to interpolate

ibo integer output bitmap flags

lo logical output bitmaps

go real output fields interpolated

mi integer dimension of input grid fields

mo integer dimension of output grid fields

rlat output latitudes in degrees

rlon output longitudes in degrees

w11,w12,w21,w22 weights to be used for interpolation

n11,n12,n21,n22 index of neighbor points

udef undefined value to be used

iret return code (0-success)

The routines invoked are:

polfixs (4.0.22)
Apply corrections for poles

4.0.5 neighbor_interp_input (Source File: neighbor_interp_input.F90)

INTERFACE:

```
subroutine neighbor_interp_input (gridDesci,gridDesco,npts, rlat2,rlon2,n112)
  implicit none
```

ARGUMENTS:

```
  real, intent(in) :: gridDesci(50)
  real             :: gridDesco(50)
  integer          :: npts
  real             :: rlat2(npts)
  real             :: rlon2(npts)
  integer          :: n112(npts)
```

DESCRIPTION:

This subprogram performs issues calls to compute the neighbor information for neighbor search interpolation,from any grid to any grid for scalar fields. The grids are defined by their grid description arrays.
The grid description arrays are based on the decoding schemes used by NCEP. However, in order to remove the integer arithmetic employed in the original ipolates, the routines are rewritten using real number manipulations. The general structure remains the same.
The current code recognizes the following projections:
(gridDesc(1)=0) equidistant cylindrical
(gridDesc(1)=1) mercator cylindrical
(gridDesc(1)=3) lambert conformal conical
(gridDesc(1)=4) gaussian cylindrical (spectral native)
(gridDesc(1)=5) polar stereographic azimuthal
where gridDesc could be defined for either the input grid or the output grid.
The arguments are:

gridDesci input grid description parameters

gridDesco output grid description parameters

npts number of points to in the output field

rlat2 output latitudes in degrees

rlon2 output longitudes in degrees

n112 index of neighbor points

The routines invoked are:

compute_earth_coord (4.0.7)
Computes the earth coordinates for the output grid

compute_grid_coord (4.0.14)
Computes the grid coordinates of the input grid, based on the earth coordinates of the output grid.

get_field_pos (4.0.21)
computes the field position for a given point

4.0.6 neighbor_interp (Source File: neighbor_interp.F90)

REVISION HISTORY:

```
04-10-96  Mark Iredell; Initial Specification
05-27-04  Sujay Kumar : Modified verision with floating point arithmetic,
```

INTERFACE:

```
subroutine neighbor_interp(gridDesco,ibi,li,gi,ibo,lo,go,mi,mo, &
     rlat,rlon,n11,udef, iret)
  implicit none
```

ARGUMENTS:

```
real      :: gridDesco(50)
integer   :: ibi
integer   :: ibo
integer   :: mi
integer   :: mo
logical*1 :: li(mi)
logical*1 :: lo(mo)
real      :: gi(mi)
real      :: go(mo)
real      :: rlat(mo)
real      :: rlon(mo)
integer   :: n11(mo)
real      :: udef
integer   :: iret
```

DESCRIPTION:

This subprogram performs neighbor search interpolation from any grid to any grid for scalar fields. The routine is based on the spatial interpolation package ipolates from NCEP.
The algorithm simply performs a nearest neighbor search around each output grid point for interpolation. The grids are defined by their grid description arrays.

The grid description arrays are based on the decoding schemes used by NCEP. However, in order to remove the integer arithmetic employed in the original ipolates, the routines are rewritten using real number manipulations. The general structure remains the same.
The current code recognizes the following projections:
(gridDesc(1)=0) equidistant cylindrical
(gridDesc(1)=1) mercator cylindrical
(gridDesc(1)=3) lambert conformal conical
(gridDesc(1)=4) gaussian cylindrical (spectral native)
(gridDesc(1)=5) polar stereographic azimuthal
where gridDesc could be defined for either the input grid or the output grid. The routine also returns the the number of output grid points and their latitudes and longitudes are also returned. The input bitmaps will be interpolated to output bitmaps. output bitmaps will also be created when the output grid extends outside of the domain of the input grid. the output field is set to 0 where the output bitmap is off.
The arguments are:

gridDesco output grid description parameters

ibi integer input bitmap flags

li logical input bitmaps

gi real input fields to interpolate

ibo integer output bitmap flags

lo logical output bitmaps

go real output fields interpolated

mi integer dimension of input grid fields

mo integer dimension of output grid fields

rlat output latitudes in degrees

rlon output longitudes in degrees

n11 index of neighbor points

udef undefined value to be used

iret return code (0-success)

The routines invoked are:

polfixs (4.0.22)
Apply corrections for poles

4.0.7 compute_earth_coord (Source File: compute_earth_coord.F90)

REVISION HISTORY:

```
04-10-96 Mark Iredell;  Initial Specification
05-27-04 Sujay Kumar; Modified verision with floating point arithmetic.
```

INTERFACE:

```
subroutine compute_earth_coord(gridDesc,npts,fill,xpts,ypts,rlon,rlat,nret)

  implicit none
```

ARGUMENTS:

```
real        :: gridDesc(50)
integer     :: npts
real        :: fill
real        :: xpts(npts),ypts(npts)
real        :: rlat(npts)
real        :: rlon(npts)
integer     :: nret
```

DESCRIPTION:

This subroutine computes the earth coordinates (lat/lon values) of the specified domain. This routine is based on the grid decoding routines in the ipolates interoplation package.
The input options include : The current code recognizes the following projections:
(gridDesc(1)=000) equidistant cylindrical
(gridDesc(1)=001) mercator cylindrical
(gridDesc(1)=003) lambert conformal conical
(gridDesc(1)=004) gaussian cylindrical
(gridDesc(1)=005) polar stereographic azimuthal

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts input grid x point coordinates

ypts input grid y point coordinates

rlat output latitudes in degrees

rlon output longitudes in degrees

nret return code (0-success)

The routines invoked are:

compute_earth_coord_latlon (4.0.8)
computes the earth coordinates of a latlon grid

compute_earth_coord_merc (4.0.9)
computes the earth coordinates of a mercator grid

compute_earth_coord_lambert (4.0.10)
computes the earth coordinates of a lambert conformal grid

compute_earth_coord_gauss (4.0.11)
computes the earth coordinates of a gaussian cylindrical grid

compute_earth_coord_polar (4.0.13)
computes the earth coordinates of a polar stereographic grid

4.0.8 compute_earth_coord_latlon (Source File: compute_earth_coord_latlon.F90)

REVISION HISTORY:

```
04-10-96 Mark Iredell;  Initial Specification
05-27-04 Sujay Kumar; Modified verision with floating point arithmetic.
```

INTERFACE:

```
subroutine compute_earth_coord_latlon(gridDesc,npts,fill,xpts,ypts,&
     rlon,rlat,nret)

  implicit none
```

ARGUMENTS:

```
real            :: gridDesc(50)
integer         :: npts
real            :: fill
real            :: xpts(npts),ypts(npts)
real            :: rlat(npts)
real            :: rlon(npts)
```

DESCRIPTION:

This subroutine computes the earth coordinates of the specified domain for an equidistant cylindrical projection. This routine is based on the grid decoding routines in the NCEP interoplation package.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts grid x point coordinates

ypts grid y point coordinates

rlat output latitudes in degrees

rlon output longitudes in degrees

4.0.9 compute_earth_coord_merc (Source File: compute_earth_coord_merc.F90)

REVISION HISTORY:

```
04-10-96 Mark Iredell;  Initial Specification
07-15-05 Sujay Kumar; Modified verision with floating point arithmetic.
```

INTERFACE:

```
subroutine compute_earth_coord_merc(gridDesc,npts,fill,xpts,ypts,&
     rlon,rlat,nret)
```

USES:

```
use map_utils
implicit none
```

ARGUMENTS:

```
real            :: gridDesc(50)
integer         :: npts
real            :: fill
real            :: xpts(npts),ypts(npts)
real            :: rlat(npts)
real            :: rlon(npts)
integer         :: nret
```

DESCRIPTION:

This subroutine computes the earth coordinates of the specified domain for a mercator projection This routine is based on the decoding routines in the NCEP interoplation package and has been modified the adopted module from the Weather Research and Forecasting (WRF) model.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts grid x point coordinates

ypts grid y point coordinates

rlat output latitudes in degrees

rlon output longitudes in degrees

nret return code (0-success)

The routines invoked are:

map_set (4.1.7)
Sets the projection to mercator

ij_to_latlon (4.1.9)
Computes the lat lon values for each i,j

4.0.10 compute_earth_coord_lambert (Source File: compute_earth_coord_lambert.F90)

INTERFACE:

```
subroutine compute_earth_coord_lambert(gridDesc,npts,fill,xpts,ypts,&
     rlon,rlat,nret)
```

USES:

```
use map_utils

implicit none
```

ARGUMENTS:

```
real            :: gridDesc(50)
integer         :: npts
real            :: fill
real            :: xpts(npts),ypts(npts)
real            :: rlat(npts)
real            :: rlon(npts)
integer         :: nret
```

DESCRIPTION:

This subroutine computes the earth coordinates of the specified domain for a lambert conformal projection This routine is based on the decoding routines in the NCEP interoplation package and has been modified the adopted module from the Weather Research and Forecasting (WRF) model.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts grid x point coordinates

ypts grid y point coordinates

rlat output latitudes in degrees

rlon output longitudes in degrees

nret return code (0-success)

The routines invoked are:

map_set (4.1.7)
Sets the projection to lambert conformal

ij_to_latlon (4.1.9)
Computes the lat lon values for each i,j

4.0.11 compute_earth_coord_gauss (Source File: compute_earth_coord_gauss.F90)

REVISION HISTORY:

```
04-10-96 Mark Iredell;  Initial Specification
05-27-04 Sujay Kumar; Modified verision with floating point arithmetic.
```

INTERFACE:

```
subroutine compute_earth_coord_gauss(gridDesc,npts,fill,xpts,ypts,&
     rlon,rlat,nret)

  implicit none
```

ARGUMENTS:

```
real            :: gridDesc(50)
integer         :: npts
real            :: fill
real            :: xpts(npts),ypts(npts)
real            :: rlat(npts)
real            :: rlon(npts)
```

DESCRIPTION:

This subroutine computes the earth coordinates of the specified domain for a gaussian cylindrical projection. This routine is based on the grid decoding routines in the NCEP interoplation package.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts grid x point coordinates

ypts grid y point coordinates

rlat output latitudes in degrees

rlon output longitudes in degrees

The routines invoked are:

gausslat (4.0.20)
Computes latitude values in gaussian

4.0.12 compute_earth_coord_ease (Source File: compute_earth_coord_ease.F90)

REVISION HISTORY:

```
04-10-96 Mark Iredell;  Initial Specification
05-27-04 Sujay Kumar; Modified verision with floating point arithmetic.
06-11-07 Bailing Li; adapted from compute_earth_coord_latlon
```

INTERFACE:

```
subroutine compute_earth_coord_ease(gridDesc,npts,fill,xpts,ypts,&
     rlon,rlat,nret)

  implicit none
```

ARGUMENTS:

```
real            :: gridDesc(50)
integer         :: npts
real            :: fill
real            :: xpts(npts),ypts(npts)
real            :: rlat(npts)
real            :: rlon(npts)
```

DESCRIPTION:

This subroutine computes the earth coordinates of the specified domain for an EASE cylindrical projection.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts grid x point coordinates

ypts grid y point coordinates

rlat output latitudes in degrees

rlon output longitudes in degrees

4.0.13 compute_earth_coord_polar (Source File: compute_earth_coord_polar.F90)

REVISION HISTORY:

```
04-10-96 Mark Iredell;  Initial Specification
07-15-05 Sujay Kumar; Modified verision with floating point arithmetic.
```

INTERFACE:

```
subroutine compute_earth_coord_polar(gridDesc,npts,fill,xpts,ypts,&
     rlon,rlat,nret)
```

USES:

```
use map_utils
implicit none
```

ARGUMENTS:

```
real            :: gridDesc(50)
integer         :: npts
real            :: fill
real            :: xpts(npts),ypts(npts)
real            :: rlat(npts)
real            :: rlon(npts)
integer         :: nret
```

DESCRIPTION:

This subroutine computes the earth coordinates of the specified domain for a polar stereographic projection This routine is based on the decoding routines in the NCEP interoplation package and has been modified the adopted module from the Weather Research and Forecasting (WRF) model.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts grid x point coordinates

ypts grid y point coordinates

rlat output latitudes in degrees

rlon output longitudes in degrees

nret return code (0-success)

The routines invoked are:

map_set (4.1.7)
Sets the projection to polar stereographic

ij_to_latlon (4.1.9)
Computes the lat lon values for each i,j

4.0.14 compute_grid_coord (Source File: compute_grid_coord.F90)

REVISION HISTORY:

```
04-10-96 Mark Iredell;  Initial Specification
05-27-04 Sujay Kumar; Modified verision with floating point arithmetic.
```

INTERFACE:

```
subroutine compute_grid_coord(gridDesc,npts,fill,xpts,ypts,rlon,rlat,nret)
  implicit none
```

ARGUMENTS:

```
  real        :: gridDesc(50)
  integer     :: npts
  real        :: fill
  real        :: xpts(npts),ypts(npts)
  real        :: rlat(npts)
  real        :: rlon(npts)
  integer     :: nret
```

DESCRIPTION:

This subroutine computes the grid coordinates (cartesian) of the specified domain. This routine is based on the grid decoding routines in the ipolates interoplation package.
The input options include : The current code recognizes the following projections: (gridDesc(1)=000) equidistant cylindrical (gridDesc(1)=001) mercator cylindrical (gridDesc(1)=003) lambert conformal conical (gridDesc(1)=004) gaussian cylindrical (gridDesc(1)=005) polar stereographic azimuthal

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts output grid x point coordinates

ypts output grid y point coordinates

rlat input latitudes in degrees

rlon input longitudes in degrees

nret return code (0-success)

The routines invoked are:

compute_grid_coord_latlon (4.0.15
computes the grid coordinates of a latlon grid

compute_grid_coord_merc (4.0.16
computes the grid coordinates of a mercator grid

compute_grid_coord_lambert (4.0.17
computes the grid coordinates of a lambert conformal grid

compute_grid_coord_gauss (4.0.18
computes the grid coordinates of a gaussian cylindrical grid

compute_grid_coord_polar (4.0.20
computes the grid coordinates of a polar stereographic grid

4.0.15 compute_grid_coord_latlon (Source File: compute_grid_coord_latlon.F90)

REVISION HISTORY:

```
04-10-96 Mark Iredell;  Initial Specification
05-27-04 Sujay Kumar; Modified verision with floating point arithmetic.
```

INTERFACE:

```
subroutine compute_grid_coord_latlon(gridDesc,npts,fill,xpts,ypts,&
     rlon,rlat,nret)

  implicit none
```

ARGUMENTS:

```
  real            :: gridDesc(50)
  integer         :: npts
  real            :: fill
  real            :: xpts(npts),ypts(npts)
  real            :: rlat(npts)
  real            :: rlon(npts)
  integer         :: nret
```

DESCRIPTION:

This subroutine computes the grid coordinates of the specified domain for an equidistant cylindrical rojection. This routine is based on the grid decoding routines in the NCEP interoplation package.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts output grid x point coordinates

ypts output grid y point coordinates

rlat input latitudes in degrees

rlon input longitudes in degrees

4.0.16 compute_grid_coord_merc (Source File: compute_grid_coord_merc.F90)

REVISION HISTORY:

```
04-10-96 Mark Iredell;  Initial Specification
07-15-05 Sujay Kumar; Modified verision with floating point arithmetic.
```

INTERFACE:

```
subroutine compute_grid_coord_merc(gridDesc,npts,fill,xpts,ypts,&
     rlon,rlat,nret)
```

USES:

```
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
real            :: gridDesc(50)
integer         :: npts
real            :: fill
real            :: xpts(npts),ypts(npts)
real            :: rlat(npts)
real            :: rlon(npts)
integer         :: nret
```

DESCRIPTION:

This subroutine computes the grid coordinates of the specified domain for a mercator projection. This routine is based on the grid decoding routines in the NCEP interoplation package.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts output grid x point coordinates

ypts output grid y point coordinates

rlat input latitudes in degrees

rlon input longitudes in degrees

!NOTE: This routine is currently unsupported.

4.0.17 compute_grid_coord_lambert (Source File: compute_grid_coord_lambert.F90)

REVISION HISTORY:

```
04-10-96 Mark Iredell;  Initial Specification
07-15-05 Sujay Kumar; Modified verision with floating point arithmetic.
```

INTERFACE:

```
subroutine compute_grid_coord_lambert(gridDesc,npts,fill,xpts,ypts,&
     rlon,rlat,nret)
```

USES:

```
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

```
real            :: gridDesc(50)
integer         :: npts
real            :: fill
real            :: xpts(npts),ypts(npts)
real            :: rlat(npts)
real            :: rlon(npts)
integer         :: nret
```

DESCRIPTION:

This subroutine computes the grid coordinates of the specified domain for a lambert conformal projection. This routine is based on the grid decoding routines in the NCEP interoplation package.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts output grid x point coordinates

ypts output grid y point coordinates

rlat input latitudes in degrees

rlon input longitudes in degrees

!NOTE: This routine is currently unsupported.

4.0.18 compute_grid_coord_gauss (Source File: compute_grid_coord_gauss.F90)

REVISION HISTORY:

```
04-10-96 Mark Iredell;  Initial Specification
05-27-04 Sujay Kumar; Modified verision with floating point arithmetic.
```

INTERFACE:

```
subroutine compute_grid_coord_gauss(gridDesc,npts,fill,xpts,ypts,&
     rlon,rlat,nret)

implicit none
```

ARGUMENTS:

```
real            :: gridDesc(50)
integer         :: npts
real            :: fill
real            :: xpts(npts),ypts(npts)
real            :: rlat(npts)
real            :: rlon(npts)
integer         :: nret
```

DESCRIPTION:

This subroutine computes the grid coordinates of the specified domain for a gaussian cylindrical projection. This routine is based on the grid decoding routines in the NCEP interoplation package.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts output grid x point coordinates

ypts output grid y point coordinates

rlat input latitudes in degrees

rlon input longitudes in degrees

The routines invoked are:

gausslat (4.0.20
Computes latitude values in gaussian

4.0.19 compute_grid_coord_ease (Source File: compute_grid_coord_ease.F90)

REVISION HISTORY:

```
04-10-96 Mark Iredell;  Initial Specification
05-27-04 Sujay Kumar; Modified verision with floating point arithmetic.
06-11-07 Bailing Li; adapted from compute_grid_coord_latlon
```

INTERFACE:

```
subroutine compute_grid_coord_ease(gridDesc,npts,fill,xpts,ypts,&
     rlon,rlat,nret)

  implicit none
```

ARGUMENTS:

```
real            :: gridDesc(50)
integer         :: npts
real            :: fill
real            :: xpts(npts),ypts(npts)
real            :: rlat(npts)
real            :: rlon(npts)
integer         :: nret
```

DESCRIPTION:

This subroutine computes the grid coordinates of the specified domain for an cylindrical EASE projection.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts output grid x point coordinates

ypts output grid y point coordinates

rlat input latitudes in degrees

rlon input longitudes in degrees

4.0.20 compute_grid_coord_polar (Source File: compute_grid_coord_polar.F90)

REVISION HISTORY:

```
04-10-96 Mark Iredell;  Initial Specification
07-15-05 Sujay Kumar; Modified verision with floating point arithmetic.
07-25-06 Sujay Kumar; Updated version to include Map Utils routines
```

INTERFACE:

```
subroutine compute_grid_coord_polar(gridDesc,npts,fill,xpts,ypts,&
     rlon,rlat,nret)

  use map_utils

  implicit none
```

ARGUMENTS:

```
real            :: gridDesc(50)
integer         :: npts
real            :: fill
real            :: xpts(npts),ypts(npts)
real            :: rlat(npts)
real            :: rlon(npts)
integer         :: nret
```

DESCRIPTION:

This subroutine computes the grid coordinates of the specified domain for a polar stereographic projection. This routine is based on the grid decoding routines in the NCEP interoplation package.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts output grid x point coordinates

ypts output grid y point coordinates

rlat input latitudes in degrees

rlon input longitudes in degrees

The routines invoked are:

latlonTopolar (4.0.23)
Converts lat, lons to polar coordinates

map_set (4.1.7)
Determines projection type for grid structure fill

latlon_to_ij (4.1.8)
Converts the input lat/lon values to the Cartesian (i,j) value for the projection type.

!ROUTINE : gausslat

REVISION HISTORY:

```
04-16-92 Mark Iredell; Initial Specification
10-20-97 Mark Iredell; Increased precision
05-14-02 Urzula Jambor; Reduced limit of eps from e-12 to e-7
```

INTERFACE:

```
subroutine gausslat(jmax,slat,wlat)
  implicit none
```

ARGUMENTS:

```
  integer       :: jmax
  real          :: slat(jmax)
  real          :: wlat(jmax)
```

DESCRIPTION:

This subroutine computes gaussian latitudes Computes cosines of colatitude and gaussian weights on the gaussian latitudes. the gaussian latitudes are at the zeroes of the legendre polynomial of the given order. The arguments are:

jmax input number of latitudes

slat cosines of colatitude

wlat gaussian weights

4.0.21 get_field_pos (Source File: get_fieldpos.F90)

REVISION HISTORY:

```
04-10-96  Mark Iredell; Initial Specification
03-11-96  Mark Iredell; Allowed hemispheric grids to wrap over one pole
05-27-04  Sujay Kumar; Modified code with floating point arithmetic
```

INTERFACE:

```
function get_fieldpos(i,j,gridDesc) result(field_pos)

  implicit none
```

ARGUMENTS:

```
  integer     :: field_pos
  real        ::  gridDesc(50)
  integer     ::  i,j
```

DESCRIPTION:

This subprogram returns the field position for a given grid point based on the input grid definition. The arguments are:

i integer x grid point

j integer y grid point

gridDesc grid description parameters

field_pos integer position in grid field to locate grid point

---

4.0.22 polfixs (Source File: polfixs.F90)

REVISION HISTORY:

```
    04-10-96  Mark Iredell; Initial Specification
```

INTERFACE:

```
subroutine polfixs(nm,nx,km,rlat,rlon,ib,lo,go)
  implicit none
```

ARGUMENTS:

```
  integer         :: nm
  integer         :: nx
  integer         :: km
  real            :: rlat(nm)
  real            :: rlon(nm)
  integer         :: ib(km)
  logical*1       :: lo(nx,km)
  real            :: go(nx,km)
```

DESCRIPTION:

This subroutine averages multiple pole scalar values on a latitude/longitude grid. bitmaps may be averaged too.
The arguments are:

nm number of grid points

nx leading dimensition of fields

rlat latitudes in degrees

rlon longitudes in degrees

ib integer bitmap flags

lo logical bitmaps

go returned scalar value

4.0.23 latlonTopolar (Source File: latlonTopolar.F90)

REVISION HISTORY:

```
86-07-17  MCDONELL,J.
88-06-07  R.E.JONES   CLEAN UP CODE, TAKE OUT GOTO, USE THEN, ELSE
89-11-02  R.E.JONES   CHANGE TO CRAY CFT77 FORTRAN
05-27-04  Sujay Kumar Incorporated in LIS
```

INTERFACE:

```
subroutine latlontopolar(alat,along,xmeshl,orient,xi,xj)

  implicit none
```

ARGUMENTS:

```
  real     :: alat
  real     :: along
  real     :: xmeshl
  real     :: orient
  real     :: xi
  real     :: xj
```

DESCRIPTION:

Converts the coordinates of a location on earth from the natural coordinate system of latitude/longitude to the grid (i,j) coordinate system overlaid on a polar stereographic map pro- jection true at 60 degrees n or s latitude.
The arguments are:

alat latitude in degrees

along longitude in degrees

xmeshl mesh length of grid in km at 60deg lat (¡0 if sh)

orient orientation west longitude of the grid

xi I of the point relative to North or South pole

xj J of the point relative to North or South pole

4.0.24 polarToLatLon (Source File: polarToLatLon.F90)

!REVISION HISTORY 86-07-17 R.E.JONES 89-11-01 R.E.JONES CHANGE TO CRAY CFT77 FORTRAN 05-27-04 Sujay Kumar Incorporated in LIS

INTERFACE:

```
subroutine polarToLatLon(xi,xj,xmeshl,orient,alat,along)

  implicit none
```

ARGUMENTS:

```
  real    :: xi
  real    :: xj
  real    :: xmeshl
  real    :: orient
  real    :: alat
  real    :: along
```

DESCRIPTION:

Converts the coordinates of a location from the grid(i,j) coordinate system overlaid on the polar stereographic map projec- tion true at 60 degrees N or S latitude to the natural coordinate system of latitude/longitude on the earth.
The arguments are:

xi I of the point relative to North or South pole

xj J of the point relative to North or South pole

xmeshl mesh length of grid in km at 60deg lat (¡0 if sh)

orient orientation west longitude of the grid

alat latitude in degrees

along longitude in degrees

4.0.25 lltops (Source File: lltops.F90)

REVISION HISTORY:

```
      05 aug 1998 initial version.....................ssgt mccormick/dnxm
      10 aug 1999 ported to ibm sp-2.  added intent attributes to
                  arguments...............................mr gayno/dnxm
      29 oct 2005; Sujay Kumar; Incorporated into LIS
```

INTERFACE:

```
subroutine lltops( pose, rlat, rlon, mesh, hemi, ri, rj )

  implicit none
```

ARGUMENTS:

```
integer,      intent(in)     :: pose
real,         intent(in)     :: rlat
real,         intent(in)     :: rlon
integer,      intent(in)     :: mesh
integer,      intent(out)    :: hemi
real,         intent(out)    :: ri
real,         intent(out)    :: rj
```

DESCRIPTION:

converts lat/lon to polar stereographic grid i/j points
method

calculate map related constants/factors.
adjust the longitude according to the user specified sign convention.
convert longitude to radians.
calculate the distance from the pole to the point in radians.
calculate the center point coordinate on the grid.
calculate the grid coordinates for the point.

The arguments are:

pose longitude increment orientation flag (1-positive east, 0-positive west)

rlat input latitude in degrees

rlon input longitude in degrees

mesh mesh factor

hemi index of the hemisphere (1-nh, 2-sh)

ri i-coordinate

rj j-coordinate

4.0.26 pstoll (Source File: pstoll.F90)

REVISION HISTORY:

```
05 aug 1998 initial version....................ssgt mccormick/dnxm
10 aug 1999 ported to ibm sp2.  added intent attributes to
            arguments...............................mr gayno/dnxm
25 jul 2005 Adopted in LIS
```

INTERFACE:

```
subroutine pstoll( hemi, pose, ri, rj, mesh, rlat, rlon )

  implicit none
```

ARGUMENTS:

```
integer,       intent(in)    :: hemi
integer,       intent(in)    :: pose
real,          intent(in)    :: ri
real,          intent(in)    :: rj
integer,       intent(in)    :: mesh
real,          intent(out)   :: rlat
real,          intent(out)   :: rlon
```

DESCRIPTION:

converts polar stereographic grid i/j points to lat/lon
method:

initialize grid specific constants.
calculate distance between input point and center point of grid.
using this distance, calculate latitude and longitude using trigonometry.
adjust sign of longitude according to user preference.

The arguments are:

hemi index of the hemisphere (1-nh, 2-sh)

pose longitude increment orientation flag (1-positive east, 0-positive west)

ri i-coordinate

rj j-coordinate

mesh mesh factor

rlat output latitude in degrees

rlon output longitude in degrees

4.1 Fortran: Module Interface map_utils (Source File: map_utils.F90)

Module that defines constants, data structures, and subroutines used to convert grid indices to lat/lon and vice versa. (This module is adopted from the Weather Research and Forecasting (WRF) model Standard Initialization(SI) program.

4.1.1 Supported Projections

Cylindrical Lat/Lon (code = PROJ_LATLON)
Mercator (code = PROJ_MERC)
Lambert Conformal (code = PROJ_LC)
Polar Stereographic (code = PROJ_PS)

4.1.2 Remarks

The routines contained within were adapted from routines obtained from the NCEP w3 library. The original NCEP routines were less flexible (e.g., polar-stereo routines only supported truelat of 60N/60S) than what we needed, so modifications based on equations in Hoke, Hayes, and Renninger (AFGWC/TN/79-003) were added to improve the flexibility. Additionally, coding was improved to F90 standards and the routines were combined into this module.

4.1.3 Assumptions

Grid Definition: For mercator, lambert conformal, and polar-stereographic projections, the routines within assume the following:

- Grid is dimensioned (i,j) where i is the East-West direction, positive toward the east, and j is the north-south direction, positive toward the north.
- Origin is at (1,1) and is located at the southwest corner, regardless of hemispere.
- Grid spacing (dx) is always positive.
- Values of true latitudes must be positive for NH domains and negative for SH domains.

For the latlon projection, the grid origin may be at any of the corners, and the deltalat and deltalon values can be signed to account for this using the following convention:

```
Origin Location        Deltalat Sign      Deltalon Sign
---------------        -------------      -------------
 SW Corner                  +                  +
 NE Corner                  -                  -
 NW Corner                  -                  +
 SE Corner                  +                  -
```

4.1.4 Data Definitions:

- Any arguments that are a latitude value are expressed in degrees north with a valid range of -90 -¿ 90
- Any arguments that are a longitude value are expressed in degrees east with a valid range of -180 -¿ 180.
- Distances are in meters and are always positive.
- The standard longitude (stdlon) is defined as the longitude line which is parallel to the y-axis (j-direction), along which latitude increases (NOT the absolute value of latitude, but the actual latitude, such that latitude increases continuously from the south pole to the north pole) as j increases.
- One true latitude value is required for polar-stereographic and mercator projections, and defines at which latitude the grid spacing is true. For lambert conformal, two true latitude values must be specified, but may be set equal to each other to specify a tangent projection instead of a secant projection.

4.1.5 Usage

To use the routines in this module, the calling routines must have the following statement at the beginning of its declaration block: `USE map_utils`
The use of the module not only provides access to the necessary routines, but also defines a structure of `TYPE (proj_info)` that can be used to declare a variable of the same type to hold your map projection information. It also defines some integer parameters that contain the projection codes so one only has to use those variable names rather than remembering the acutal code when using them. The basic steps are as follows:

- Ensure the `USE map_utils` is in your declarations.
- Declare the projection information structure as `type(proj_info): TYPE(proj_info) :: proj`
- Populate your structure by calling the map_set routine:
  `CALL map_set(code,lat1,lon1,dx,stdlon,truelat1,truelat2,nx,ny,proj)` where: code (input) = one of PROJ_LATLON, PROJ_MERC, PROJ_LC, or PROJ_PS

lat1 (input) = Latitude of grid origin point (i,j)=(1,1) (see assumptions!)
lon1 (input) = Longitude of grid origin
dx (input) = grid spacing in meters (ignored for LATLON projections)
stdlon (input) = Standard longitude for PROJ_PS and PROJ_LC, deltalon (see assumptions) for PROJ_LATLON, ignored for PROJ_MERC
truelat1 (input) = 1st true latitude for PROJ_PS, PROJ_LC, and PROJ_MERC, deltalat (see assumptions) for PROJ_LATLON
truelat2 (input) = 2nd true latitude for PROJ_LC, ignored for all others.
nx = number of points in east-west direction
ny = number of points in north-south direction
proj (output) = The structure of `type (proj_info)` that will be fully populated after this call

- Now that the proj structure is populated, you may call any of the following routines:

  ```
  latlon_to_ij(proj, lat, lon, i, j)
  ij_to_latlon(proj, i, j, lat, lon)
  truewind_to_gridwind(lon, proj, ugrid, vgrid, utrue, vtrue)
  gridwind_to_truewind(lon, proj, utrue, vtrue, ugrid, vgrid)
  compare_projections(proj1, proj2, same_proj)
  ```

  It is incumbent upon the calling routine to determine whether or not the values returned are within your domain bounds. All values of i, j, lat, and lon are REAL values.

References
Hoke, Hayes, and Renninger, "Map Preojections and Grid Systems for Meteorological Applications." AFGWC/TN-79/003(Rev), Air Weather Service, 1985.

NCAR MM5v3 Modeling System, REGRIDDER program, module_first_guess_map.F NCEP routines w3fb06, w3fb07, w3fb08, w3fb09, w3fb11, w3fb12

REVISION HISTORY:

```
  27 Mar 2001 - Original Version
                Brent L. Shaw, NOAA/FSL (CSU/CIRA)
  02 Apr 2001 - Added routines to rotate winds from true to grid
                and vice versa.
                Brent L. Shaw, NOAA/FSL (CSU/CIRA)
  09 Apr 2001 - Added compare\_projections routine to compare two
                sets of projection parameters.

 implicit none

  ! mean earth radius in m.  the value below is consistent
  ! with nceps routines and grids.
  real, public, parameter     :: earth_radius_m = 6371200.

  ! define public parameters
 mp
  INTEGER, PUBLIC, PARAMETER  :: PROJ_ROTLAT = 203
 mp
 ------------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
 ------------------------------------------------------------------------------
  public :: map_init     !initializes map projection structure
  public :: map_set      !completes the initializations
```

```
public :: ij_to_latlon !converts ijs to lat lon values
public :: latlon_to_ij !converts lat lon values to ijs.
-----------------------------------------------------------------------------
```

PUBLIC TYPES:

```
-----------------------------------------------------------------------------
! projection codes for proj_info structure:
integer, public, parameter  :: proj_latlon = 0
integer, public, parameter  :: proj_merc = 1
integer, public, parameter  :: proj_lc = 3
integer, public, parameter  :: proj_ps = 5
integer, public, parameter  :: proj_hrap = 10
```

4.1.6 map_init (Source File: map_utils.F90)

INTERFACE:

```
subroutine map_init(proj)
 implicit none
```

ARGUMENTS:

```
type(proj_info), intent(inout)  :: proj
```

DESCRIPTION:

Initializes the map projection structure to missing values
The arguments are:

proj data structure containing the map projection information

---

4.1.7 map_set (Source File: map_utils.F90)

INTERFACE:

```
subroutine map_set(proj_code,lat1,lon1,dx,stdlon,truelat1,truelat2, &
                   idim,jdim,proj)

  implicit none
```

ARGUMENTS:

```
  integer, intent(in)              :: proj_code
  real, intent(in)                 :: lat1
  real, intent(in)                 :: lon1
  real, intent(in)                 :: dx
  real, intent(in)                 :: stdlon
  real, intent(in)                 :: truelat1
  real, intent(in)                 :: truelat2
  integer, intent(in)              :: idim
  integer, intent(in)              :: jdim
  type(proj_info), intent(out)     :: proj
```

DESCRIPTION:

Given a partially filled `proj_info` structure, this routine computes `polei, polej, rsw, and cone` (if LC projection) to complete the structure. This allows us to eliminate redundant calculations when calling the coordinate conversion routines multiple times for the same map. This will generally be the first routine called when a user wants to be able to use the coordinate conversion routines, and it will call the appropriate subroutines based on the `proj%code` which indicates which projection type this is.
The arguments are:

proj_code code to indicate the type of projection

lat1 latitude of origin

lon1 longitude of origin

dx grid spacing in meters

stdlon standard longitude

truelat1 1st true latitude

truelat2 2nd true latitude

idim number of points in the East-West direction

jdim number of points in the North-South direction

proj data structure containing the map projection information

---

4.1.8 latlon_to_ij (Source File: map_utils.F90)

INTERFACE:

```
subroutine latlon_to_ij(proj, lat, lon, i, j)
  implicit none
```

ARGUMENTS:

```
type(proj_info), intent(in)          :: proj
real, intent(in)                     :: lat
real, intent(in)                     :: lon
real, intent(out)                    :: i
real, intent(out)                    :: j
```

DESCRIPTION:

Converts input lat/lon values to the cartesian (i,j) value for the given projection.
The arguments are:

proj data structure containing the map projection information

lat input latitude

lon input longitude

i output i value

j output j value

4.1.9 ij_to_latlon (Source File: map_utils.F90)

INTERFACE:

```
subroutine ij_to_latlon(proj, i, j, lat, lon)
  implicit none
```

ARGUMENTS:

```
type(proj_info),intent(in)          :: proj
real, intent(in)                    :: i
real, intent(in)                    :: j
real, intent(out)                   :: lat
real, intent(out)                   :: lon
```

DESCRIPTION:

Computes geographical latitude and longitude for a given (i,j) point in a grid with a projection of proj
The arguments are:

proj data structure containing the map projection information

i input i value

j input j value

lat output latitude

lon output longitude

4.1.10 compute_stnwts (Source File: compute_stnwts.F90)

REVISION HISTORY:

```
07-15-05 Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine compute_stnwts(nstns, gridDesc,stnlat, stnlon,&
     npts, stnwt)

  implicit none
```

ARGUMENTS:

```
integer     :: nstns
real        :: gridDesc(50)
real        :: stnlat(nstns)
real        :: stnlon(nstns)
integer     :: npts
real        :: stnwt(npts,nstns)
```

DESCRIPTION:

This routine compute the interpolation weights to be applied to a network of stations, to generate a gridded field from observations from stations. It simply uses an inverse distance weighting algorithm to compute the relative weights of each station.

nstns number of stations used in interpolation

gridDesc grid description parameters

stnlat station latitudes in degrees

stnlon station longitudes in degrees

npts integer maximum number of coordinates

stnwt interpolation weights of stations with respect to the each point

The routines invoked are:

compute_earth_coord (4.0.7)
computes the earth coordinates

4.1.11 interp_stndata (Source File: interp_stndata.F90)

REVISION HISTORY:

```
08 Dec 04 Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine interp_stndata(wt,undef,vari,varo,npts,nstns)

  use lisdrv_module, only : lis

  implicit none
```

ARGUMENTS:

```
  integer :: nstns,npts
  real    :: undef
  real    :: wt(npts,nstns)
  real    :: vari(nstns)
  real    :: varo(npts)
```

DESCRIPTION:

This subroutine interpolates station data into a gridded set. Currently works only on a lat/lon grid
The arguments are:

nstns number of stations

npts number of points in the output field

vari input variable (array of observations from the stations)

varo output variable (gridded data on the output field)

wt interpolation weights

4.1.12 normalize_stnwts (Source File: normalize_stnwts.F90)

REVISION HISTORY:

```
08Dec04 Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine normalize_stnwts(stndata,nstns,npts,undef,wt)
  implicit none
```

ARGUMENTS:

```
  integer    :: nstns
  integer    :: npts
  real       :: stndata(nstns)
  real       :: wt(npts,nstns)
  real       :: undef
```

DESCRIPTION:

This subroutine normalizes the interpolation weights to convert station data into a gridded set. The arguments are:

nstns number of stations used in interpolation

npts integer maximum number of coordinates

stndata input data (station observations)

undef undefined value used in observations

wt interpolation weights

4.1.13 ezlh_convert (Source File: ezlh_convert.F90)

REVISION HISTORY:

```
  30-Jan.-1992 H.Maybee
  20-Mar-1992 Ken Knowles  303-492-0644  knowles@kryos.colorado.edu
        16-Dec-1993 MJ Brodzik   303-492-8263  brodzik@jokull.colorado.edu
                    Copied from nsmconv.f, changed resolutions from
                    40-20-10 km to 25-12.5 km
        21-Dec-1993 MJ Brodzik   303-492-8263  brodzik@jokull.colorado.edu
                    Fixed sign of Southern latitudes in ease_inverse.
        12-Sep-1994 David Hoogstrate 303-492-4116 hoogstra@jokull.colorado.edu
      Changed grid cell size. Changed "c","f" to "l","h"
  25-Oct-1994 David Hoogstrate 303-492-4116 hoogstra@jokull.colorado.edu
      Changed row size from 587 to 586 for Mercator projection
      Changed function names to "ezlh-.."
        07-11-2007  Bailing Li; downloaded this code from NSIDC
```

INTERFACE:

```
integer function ezlh_convert (grid, lat, lon, r, s)

  implicit none
```

ARGUMENTS:

```
character*(*) grid
real lat, lon, r, s
```

DESCRIPTION:

convert geographic coordinates (spherical earth) to azimuthal equal area or equal area cylindrical grid coordinates

```
status = ezlh_convert (grid, lat, lon, r, s)

input : grid - projection name '[NSM][lh]'
              where l = "low"  = 25km resolution
                    h = "high" = 12.5km resolution
lat, lon - geo. coords. (decimal degrees)

output: r, s - column, row coordinates

result: status = 0 indicates normal successful completion
-1 indicates error status (point not on grid)
```

4.1.14 ezlh_inverse (Source File: ezlh_inverse.F90)

REVISION HISTORY:

```
30-Jan.-1992 H.Maybee
20-Mar-1992 Ken Knowles  303-492-0644  knowles@kryos.colorado.edu
      16-Dec-1993 MJ Brodzik   303-492-8263  brodzik@jokull.colorado.edu
                  Copied from nsmconv.f, changed resolutions from
                  40-20-10 km to 25-12.5 km
      21-Dec-1993 MJ Brodzik   303-492-8263  brodzik@jokull.colorado.edu
                  Fixed sign of Southern latitudes in ease_inverse.
      12-Sep-1994 David Hoogstrate 303-492-4116 hoogstra@jokull.colorado.edu
    Changed grid cell size. Changed "c","f" to "l","h"
25-Oct-1994 David Hoogstrate 303-492-4116 hoogstra@jokull.colorado.edu
    Changed row size from 587 to 586 for Mercator projection
    Changed function names to "ezlh-.."
      07-11-2007  Bailing Li; downloaded this code from NSIDC
```

INTERFACE:

```
integer function ezlh_inverse (grid, r, s, lat, lon)

implicit none
```

ARGUMENTS:

```
character*(*) grid
real r, s, lat, lon
```

DESCRIPTION:

convert azimuthal equal area or equal area cylindrical grid coordinates to geographic coordinates (spherical earth)

```
status = ezlh_inverse (grid, r, s, lat, lon)

input : grid - projection name '[NSM][lh]'
              where l = "low"  = 25km resolution
                    h = "high" = 12.5km resolution
r, s - grid column and row coordinates

output: lat, lon - geo. coords. (decimal degrees)

result: status = 0 indicates normal successful completion
-1 indicates error status (point not on grid)
```

Part VI
User-defined extensible components in LIS

5 User-defined extensible components in LIS

The following sections describe the details of user-defined extensible components in LIS. These include the following:

Running Modes

Domains

Land Surface Parameters

Meteorological forcing analyses

Supplemental forcing products

Land Surface Models

Data Assimilation

Part VII
Running Modes in LIS

6 Running Modes in LIS

This section contains the implementations of various running modes in LIS. The current implementation includes a retrospective mode and the cycling mode used in the AGRMET system. The interfaces are expected to be extended for other running modes such as coupled, parameter estimation and forecast.

7 Retrospective Running Mode

This is a typical running mode used in LIS to perform retrospective simulations. The mode assumes that all the meteorological forcing analyes required for the specified time period is archived for the simulation.

7.1 Fortran: Module Interface retrospective_runMod (Source File: retrospective_runMod.F90)

This module contains the definition of the functions used for LIS initialization, execution, and finalization for a retrospective runmode in LIS.

REVISION HISTORY:

```
 21Oct05    Sujay Kumar  Initial Specification

implicit none

-----------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
-----------------------------------------------------------------------------
 public :: lis_init_retrospective  !init method for retrospective mode
 public :: lis_run_retrospective   !run method for retrospective mode
 public :: lis_final_retrospective !finalize method for retrospective mode
```

7.1.1 lis_init_retrospective (Source File: retrospective_runMod.F90)

INTERFACE:

```
subroutine lis_init_retrospective
```

USES:

```
  use domain_module,       only : LIS_domain_init
  use lsm_module,          only : LIS_lsm_init, LIS_setuplsm, &
       LIS_readrestart
  use baseforcing_module,  only : LIS_baseforcing_init
  use suppforcing_module,  only : LIS_suppforcing_init
  use observations_module, only : LIS_initObservations
  use perturb_module,      only : LIS_perturb_init
  use dataassim_module,    only : LIS_dataassim_init
  use param_module,        only : LIS_param_init
```

DESCRIPTION:

This is the initialize method for LIS in a retrospective running mode. The following calls are invoked from this method.

LIS_domain_init (2.7.2)
initialize the LIS domains

LIS_param_init (2.14.2)
initialize parameters

LIS_lsm_init (2.8.2)
initialize the land surface model.

LIS_baseforcing_init (2.9.2)
initialize the base forcing

LIS_suppforcing_init (2.10.2)
initialize the supplemental forcing

LIS_initObservations (2.12.1)
initialize structures needed to read observations for data assimilation

LIS_setuplsm (2.8.3)
complete the LSM setups

LIS_readrestart (2.8.6)
read the restart files

7.1.2 lis_run_retrospective (Source File: retrospective_runMod.F90)

INTERFACE:

```
subroutine lis_run_retrospective
```

USES:

```
use lisdrv_module,       only : lis, LIS_endofrun, LIS_timeToRunNest, &
     LIS_ticktime
use lsm_module,          only : LIS_force2tile, LIS_perturb_states, &
     LIS_lsm_main,LIS_lsm_output, LIS_writerestart
use param_module,        only : LIS_setDynparams, LIS_output_params
use baseforcing_module,  only : LIS_get_base_forcing, LIS_perturb_forcing
use suppforcing_module,  only : LIS_get_supp_forcing
use observations_module, only : LIS_readobservations, &
     LIS_perturb_observations
use dataassim_module,    only : LIS_dataassim_run, LIS_dataassim_output
use lis_logmod,          only : lis_flush, logunit
```

DESCRIPTION:

This is the run method for LIS in a retrospective running mode. The following calls are invoked from this method.

LIS_endofrun (2.1.7)
check to see if the end of simulation has reached

LIS_ticktime (2.1.6)
advance model clock

LIS_timeToRunNest (2.1.8)
check to see if the current nest needs to be run.

LIS_setDynparams (2.14.3)
set the time dependent parameters

LIS_get_base_forcing (2.9.6)
retrieve the base forcing

LIS_get_supp_forcing (2.10.3)
retrieve the supplemental forcing

LIS_force2tile (2.8.9)
transfer forcing to model tiles

LIS_readobservations (2.12.2)
read observations to be used for data assimilation

LIS_lsm_main (2.8.4)
run the land surface model

LIS_lsm_output (2.8.7)
write model output

LIS_writerestart (2.8.10)
write model restart files

LIS_output_params (2.14.4)
outputs land surface parameters to a file, once a month

7.1.3 lis_final_retrospective (Source File: retrospective_runMod.F90)

INTERFACE:

```
subroutine lis_final_retrospective
```

USES:

```
use lisdrv_module,       only : lis_finalize
use lsm_module,          only : LIS_lsm_finalize
use param_module,        only : LIS_param_finalize
use baseforcing_module,  only : LIS_baseforcing_finalize
use suppforcing_module,  only : LIS_suppforcing_finalize
```

DESCRIPTION:

This is the finalize method for LIS in a retrospective running mode. The following calls are invoked from this method.

LIS_finalize (2.1.10)
cleanup LIS generic structures

LIS_lsm_finalize (2.8.12)
cleanup land surface model specific structures

LIS_param_finalize (2.14.5)
cleanup parameter specific structures

LIS_baseforcing_finalize (2.9.7)
cleanup baseforcing specific structures

LIS_suppforcing_finalize (2.10.4)
cleanup supplemental forcing specific structures

LIS_dataassim_finalize (2.11.5)
cleanup data assimilation specific structures

Part VIII
Domain implementations in LIS

8 Domain implementations in LIS

This section contains the implementations of interfaces to incorporate a new domain in LIS. Currently LIS supports operations on a lat/lon, polar stereographic, lambert conformal, mercator, and GSWP grids.

9 Lat/Lon domain

This section contains the interface implementations for a LIS instance using a lat/lon projection with SW to NE data ordering

9.0.4 createtiles_latlon (Source File: createtiles_latlon.F90)

REVISION HISTORY:

```
04  Feb 2001: Jon Gottschalck; Added option to read and use Koster tile space
17  Oct 2003: Sujay Kumar ; Initial version of subsetting code
18  Feb 2006: Sujay Kumar ; Added nesting options
```

INTERFACE:

```
subroutine createtiles_latlon()
```

USES:

```
use lisdrv_module, only: lis,lisdom
use grid_module
use spmdMod
use lis_logmod, only :logunit
```

DESCRIPTION:

The code in this routine creates the grid and tile spaces in a lat/lon grid projection, with a SW to NE data ordering. The routine first reads the landmask and landcover data. The topography data is also read in depending on the options specified at runtime. The landmask is used to create the grid space, ignoring the water points. The vegetation distribution from the landcover map is used to create subgrid tiling, based on the specified maximum number of tiles and minimum cutoff percentage.
The routines invoked are:

readlandmask (2.38.4)
calls the generic routine in the registry to read the landmask

readlandcover (2.38.2)
calls the generic routine in the registry to read the landcover data

readelev (2.45.2)
calls the generic routine in the registry to read the elevation data

readslope (2.45.4)
calls the generic routine in the registry to read the slope data

readaspect (2.45.6)
calls the generic routine in the registry to read the aspect data

readcurv (2.45.8)
calls the generic routine in the registry to read the curvature data

calculate_domveg (9.0.5)
computes the dominant vegetation distribution for subgrid tiling

create_vegtilespace_latlon (9.0.6)
computes the grid and tile spaces based on the landmask, landcover, and topography datasets in a latlon projection

---

9.0.5 calculate_domveg (Source File: calculate_domveg.F90)

REVISION HISTORY:

```
09 Sept 2004: Sujay Kumar ; Initial version
```

INTERFACE:

```
subroutine calculate_domveg(n, fgrd)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod, only       : iam

implicit none
```

ARGUMENTS:

```
integer, intent(in)  :: n
real :: fgrd(lis%lnc(n), lis%lnr(n), lis%nt)
```

DESCRIPTION:

This primary goal of this routine is to determine the percentages of dominant vegetation for subgrid tiling. The routine uses the distribution of vegetation within a grid cell and imposes the specified cutoff percentage and the maximum number of veg types allowed to create the number of tiles.
The arguments are:

n index of the nest

fgrd fraction of grid covered by each vegetation type

9.0.6 create_vegtilespace_latlon (Source File: create_vegtilespace_latlon.F90)

REVISION HISTORY:

```
09 Sept 2004: Sujay Kumar ; Initial version
```

INTERFACE:

```
subroutine create_vegtilespace_latlon(n, fgrd, localmask)
      slope, aspect, curv)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod
use lis_logmod, only :logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)   :: n
real                  :: fgrd(lis%lnc(n), lis%lnr(n), lis%nt)
real                  :: localmask(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This routine computes the grid and tile spaces based on the input distribution of vegetation and landmask. The routine also computes the following structures necessary for domain decomposition and other parallel routines.
`ntiles_pergrid` : Number of tiles per each grid
`s_tid` : Index of the first tile in each grid cell

The arguments are:

n index of the nest

fgrd fraction of grid covered by each vegetation type

localmask landmask for the region of interest

9.0.7 readinput_latlon (Source File: readinput_latlon.F90)

REVISION HISTORY:

```
11 Apr 2000: Brian Cosgrove; Added Elevation correction and Forcing
             Mask read statements
14 Nov 2003: Sujay Kumar; Modified card file that includes regional
             modeling options
```

INTERFACE:

```
subroutine readinput_latlon
```

USES:

```
use lisdrv_module, only : lis, config_lis, &
     lisLayout, lisdom
use spmdMod
use lis_logmod, only :logunit
use ESMF_Mod
use map_utils
```

DESCRIPTION:

This routine reads the options specified in the LIS configuration file to determine the region of interest in the LIS simulation, in a lat/lon projection. The routine reads the extents of the running domain. The land surface parameters used for the simulation are read from a file that spans the area of interest. The domain specifications of the parameter maps are also read in by this routine. Based on the number of processors and the processor layout specified, this routine also performs the domain decomposition.
The routines invoked are:

listask_for_point (2.46.8)
routine to perform domain decomposition

10 Mercator domain

This section contains the interface implementations for a LIS instance using a mercator projection with SW to NE data ordering

10.0.8 createtiles_merc (Source File: createtiles_merc.F90)

REVISION HISTORY:

```
17 Jul 2005: Sujay Kumar ; Initial specification
```

INTERFACE:

```
subroutine createtiles_merc()
```

USES:

```
use lisdrv_module,only : lis,lisdom
use spmdMod, only : iam

implicit none
```

DESCRIPTION:

The code in this routine creates the grid and tile spaces in a mercator grid projection, with a SW to NE data ordering. The routine first reads the landmask and landcover data. The topography data is also read in depending on the options specified at runtime. The landmask is used to create the grid space, ignoring the water points. The vegetation distribution from the landcover map is used to create subgrid tiling, based on the specified maximum number of tiles and minimum cutoff percentage.
The routines invoked are:

readlandmask (2.38.4)
calls the generic routine in the registry to read the landmask

readlandcover (2.38.2)
calls the generic routine in the registry to read the landcover data

readelev (2.45.2)
calls the generic routine in the registry to read the elevation data

readslope (2.45.4)
calls the generic routine in the registry to read the slope data

readaspect (2.45.6)
calls the generic routine in the registry to read the aspect data

readcurv (2.45.8)
calls the generic routine in the registry to read the curvature data

calculate_domveg (9.0.5)
computes the dominant vegetation distribution for subgrid tiling

create_vegtilespace_merc (10.0.9)
computes the grid and tile spaces based on the landmask, landcover, and topography datasets for the mercator projection

10.0.9 create_vegtilespace_merc (Source File: create_vegtilespace_merc.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar ; Initial version
```

INTERFACE:

```
subroutine create_vegtilespace_merc(n,fgrd, localmask)
      slope, aspect, curv)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: fgrd(lis%lnc(n), lis%lnr(n), lis%nt)
real                :: localmask(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This routine computes the grid and tile spaces based on the input distribution of vegetation and landmask. The routine also computes the following structures necessary for domain decomposition and other parallel routines.
`ntiles_pergrid` : Number of tiles per each grid
`s_tid` : Index of the first tile in each grid cell

The arguments are:

n index of the nest

fgrd fraction of grid covered by each vegetation type

localmask landmask for the region of interest

10.0.10 readinput_merc (Source File: readinput_merc.F90)

REVISION HISTORY:

```
15 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine readinput_merc
```

USES:

```
use lisdrv_module, only : lis,lisdom, config_lis
use spmdMod
use lis_logmod, only : logunit
use map_utils
use ESMF_Mod
```

DESCRIPTION:

This routine reads the options specified in the LIS configuration file to determine the region of interest in the LIS simulation, in a mercator projection. The routine reads the extents of the running domain. The land surface parameters used for the simulation are read from a file that spans the area of interest. The domain specifications of the parameter maps are also read in by this routine. Based on the number of processors and the processor layout specified, this routine also performs the domain decomposition.
The routines invoked are:

listask_for_point (2.46.8)
routine to perform domain decomposition

11 Lambert Conformal domain

This section contains the interface implementations for a LIS instance using a lambert conformal projection with SW to NE data ordering

11.0.11 createtiles_lambert (Source File: createtiles_lambert.F90)

REVISION HISTORY:

```
17 Jul 2005: Sujay Kumar ; Initial specification
```

INTERFACE:

```
subroutine createtiles_lambert()
```

USES:

```
use lisdrv_module,only : lis,lisdom
use spmdMod, only : iam

implicit none
```

DESCRIPTION:

The code in this routine creates the grid and tile spaces in a lambert conformal grid projection, with a SW to NE data ordering. The routine first reads the landmask and landcover data. The topography data is also read in depending on the options specified at runtime. The landmask is used to create the grid space, ignoring the water points. The vegetation distribution from the landcover map is used to create subgrid tiling, based on the specified maximum number of tiles and minimum cutoff percentage.
The routines invoked are:

readlandmask (2.38.4)
calls the generic routine in the registry to read the landmask

readlandcover (2.38.2)
calls the generic routine in the registry to read the landcover data

readelev (2.45.2)
calls the generic routine in the registry to read the elevation data

readslope (2.45.4)
calls the generic routine in the registry to read the slope data

readaspect (2.45.6)
calls the generic routine in the registry to read the aspect data

readcurv (2.45.8)
calls the generic routine in the registry to read the curvature data

calculate_domveg (9.0.5)
computes the dominant vegetation distribution for subgrid tiling

create_vegtilespace_lambert (11.0.12)
computes the grid and tile spaces based on the landmask, landcover, and topography datasets for lambert conformal projection.

11.0.12 create_vegtilespace_lambert (Source File: create_vegtilespace_lambert.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar ; Initial version
```

INTERFACE:

```
subroutine create_vegtilespace_lambert(n,fgrd, localmask)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
real                   :: fgrd(lis%lnc(n), lis%lnr(n), lis%nt)
real                   :: localmask(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This routine computes the grid and tile spaces based on the input distribution of vegetation and landmask. The routine also computes the following structures necessary for domain decomposition and other parallel routines.
`ntiles_pergrid` : Number of tiles per each grid
`s_tid` : Index of the first tile in each grid cell

The arguments are:

n index of the nest

fgrd fraction of grid covered by each vegetation type

localmask landmask for the region of interest

11.0.13 readinput_lambert (Source File: readinput_lambert.F90)

REVISION HISTORY:

```
15Jul2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine readinput_lambert
```

USES:

```
use lisdrv_module, only : lis,config_lis, lisdom
use spmdMod
use lis_logmod, only : logunit
use map_utils
use ESMF_Mod
implicit none
```

DESCRIPTION:

This routine reads the options specified in the LIS configuration file to determine the region of interest in the LIS simulation, in a lambert conformal projection. The routine reads the extents of the running domain. The land surface parameters used for the simulation are read from a file that spans the area of interest. The domain specifications of the parameter maps are also read in by this routine. Based on the number of processors and the processor layout specified, this routine also performs the domain decomposition.
The routines invoked are:

listask_for_point (2.46.8)
routine to perform domain decomposition

12 Polar stereographic domain

This section contains the interface implementations for a LIS instance using a polar stereographic projection with SW to NE data ordering

12.0.14 createtiles_polar (Source File: createtiles_polar.F90)

REVISION HISTORY:

```
17Jul2005: Sujay Kumar ; Initial specification
```

INTERFACE:

```
subroutine createtiles_polar()
```

USES:

```
use lisdrv_module,only : lis, lisdom
use spmdMod, only : iam
```

DESCRIPTION:

The code in this routine creates the grid and tile spaces in a polar stereographic grid projection, with a SW to NE data ordering. The routine first reads the landmask and landcover data. The topography data is also read in depending on the options specified at runtime. The landmask is used to create the grid space, ignoring the water points. The vegetation distribution from the landcover map is used to create subgrid tiling, based on the specified maximum number of tiles and minimum cutoff percentage.
The routines invoked are:

readlandmask (2.38.4)
calls the generic routine in the registry to read the landmask

readlandcover (2.38.2)
calls the generic routine in the registry to read the landcover data

readelev (2.45.2)
calls the generic routine in the registry to read the elevation data

readslope (2.45.4)
calls the generic routine in the registry to read the slope data

readaspect (2.45.6)
calls the generic routine in the registry to read the aspect data

readcurv (2.45.8)
calls the generic routine in the registry to read the curvature data

calculate_domveg (9.0.5)
computes the dominant vegetation distribution for subgrid tiling

create_vegtilespace_polar (12.0.15)
computes the grid and tile spaces based on the landmask, landcover, and topography datasets in a polar stereographic projection

---

12.0.15 create_vegtilespace_polar (Source File: create_vegtilespace_polar.F90)

REVISION HISTORY:

```
09 Sept 2004: Sujay Kumar ; Initial version
```

INTERFACE:

```
subroutine create_vegtilespace_polar(n, fgrd, localmask)
      slope, aspect, curv)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils

implicit none

integer, intent(in) :: n
real                :: fgrd(lis%lnc(n), lis%lnr(n), lis%nt)
real                :: localmask(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This routine computes the grid and tile spaces based on the input distribution of vegetation and landmask. The routine also computes the following structures necessary for domain decomposition and other parallel routines.
`ntiles_pergrid` : Number of tiles per each grid
`s_tid` : Index of the first tile in each grid cell

The arguments are:

n index of the nest

fgrd fraction of grid covered by each vegetation type

localmask landmask for the region of interest

12.0.16 readinput_polar (Source File: readinput_polar.F90)

REVISION HISTORY:

```
15 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine readinput_polar
```

USES:

```
use lisdrv_module, only : lis,lisdom, config_lis
use spmdMod
use lis_logmod, only : logunit
use map_utils
use ESMF_Mod
implicit none
```

DESCRIPTION:

This routine reads the options specified in the LIS configuration file to determine the region of interest in the LIS simulation, in a polar stereographic projection. The routine reads the extents of the running domain. The land surface parameters used for the simulation are read from a file that spans the area of interest. The domain specifications of the parameter maps are also read in by this routine. Based on the number of processors and the processor layout specified, this routine also performs the domain decomposition.
The routines invoked are:

listask_for_point (2.46.8)
routine to perform domain decomposition

13 GSWP domain

This section contains the interface implementations for a LIS instance using the domain used in the Global Soil Wetness Project (GSWP).

13.0.17 createtiles_gswp (Source File: createtiles_gswp.F90)

REVISION HISTORY:

```
23 Feb 04, Sujay Kumar : Intial Specification
```

INTERFACE:

```
subroutine createtiles_gswp()
```

USES:

```
  use lisdrv_module, only: lis, lisdom
  use lis_logmod, only : logunit
  use spmdMod
#if ( defined USE_NETCDF )
  use netcdf
#endif
```

DESCRIPTION:

The code in this file implements the grid and tile spaces for the GSWP domain. The routine first reads the landmask and the landcover data. The size of the grid space is determined by ignoring the water points. The GSWP domain does not support subgrid tiling. As a result, the grid and tile spaces are equivalent.

13.0.18 readinput_gswp (Source File: readinput_gswp.F90)

REVISION HISTORY:

```
14 Nov 2003: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine readinput_gswp
```

USES:

```
  use lisdrv_module, only : lis, config_lis
  use spmdMod
  use lis_logmod, only :logunit
  use ESMF_Mod
```

DESCRIPTION:

This routine reads the domain-related runtime options for GSWP. The GSWP grid is a lat/lon 1deg global grid spanning from 90S to 60N. The extents of the running domain are specified in the LIS configuration file, and read by this routine. The routine also reads the domain extents of the paramter maps. In GSWP, all the parameters are also specified in a lat/lon 1deg global grid.

Part IX
Land Surface Parameters in LIS

14 Land Surface Parameters in LIS

This section contains the implementations of various land surface parameter maps. The interfaces are primarily geared towards including satellite and observed maps of parameters such as albedo, landcover, greenness, LAI, and soil parameters.

15 Landmask/Landcover Data

This section describes the routines to ingest various sources of landmask/landcover data in different LIS domains

15.0.19 read_latlon_umdmask (Source File: read_latlon_umdmask.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_latlon_umdmask(nest, localmask)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use lis_logmod, only :logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: nest
real                :: localmask(lis%lnc(nest),lis%lnr(nest))
```

DESCRIPTION:

This subroutine reads the UMD landmask data and returns the array in a lat/lon projection.
The arguments are:

n index of nest

localmask landmask for the region of interest

15.0.20 read_latlon_umdlc (Source File: read_latlon_umdlc.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_latlon_umdlc(n, fgrd)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use lis_logmod, only :logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real  :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the UMD landcover data and returns the distribution of vegetation in each grid cell, in a lat/lon projection. The data has 13 vegetation types.
The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

15.0.21 read_latlon_usgsmask (Source File: read_latlon_usgsmask.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_latlon_usgsmask(nest, localmask)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use lis_logmod, only :logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: nest
real                :: localmask(lis%lnc(nest),lis%lnr(nest))
```

DESCRIPTION:

This subroutine reads the USGS landmask data and returns the array in a lat/lon projection.
The arguments are:

n index of nest

localmask landmask for the region of interest

15.0.22 read_latlon_usgslc (Source File: read_latlon_usgslc.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_latlon_usgslc(n, fgrd)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use lis_logmod, only :logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real  :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the USGS landcover data and returns the distribution of vegetation in each grid cell, in a lat/lon projection. The data has 24 vegetation types.
The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

15.0.23 read_latlon_igbplc (Source File: read_latlon_igbplc.F90)

REVISION HISTORY:

```
09 May 2007: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_latlon_igbplc(n, fgrd)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use lis_logmod, only :logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real  :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the IGBP landcover data and returns the distribution of vegetation in each grid cell, in a lat/lon projection. The data has 16 vegetation types.
The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

15.0.24 read_lambert_umdmask (Source File: read_lambert_umdmask.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lambert_umdmask(n,localmask)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: localmask(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine reads the landmask data and returns the array in a lambert conformal projection.
The arguments are:

n index of nest

localmask landmask for the region of interest

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in lambert conformal projection

15.0.25 read_lambert_umdlc (Source File: read_lambert_umdlc.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lambert_umdlc(n,fgrd)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the UMD landcover data and returns the distribution of vegetation in each grid cell, in a lambert conformal projection.
The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in lambert conformal projection

15.0.26 read_lambert_usgsmask (Source File: read_lambert_usgsmask.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lambert_usgsmask(n,localmask)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: localmask(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine reads the landmask data and returns the array in a lambert conformal projection.
The arguments are:

n index of nest

localmask landmask for the region of interest

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in lambert conformal projection

15.0.27 read_lambert_usgslc (Source File: read_lambert_usgslc.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lambert_usgslc(n,fgrd)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the USGS landcover data and returns the distribution of vegetation in each grid cell, in a lambert conformal projection.
The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in lambert conformal projection

15.0.28 read_merc_umdmask (Source File: read_merc_umdmask.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_merc_umdmask(n,localmask)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: localmask(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine reads the landmask data and returns the array in a mercator projection.
The arguments are:

n index of nest

localmask landmask for the region of interest

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in mercator projection

15.0.29 read_merc_umdlc (Source File: read_merc_umdlc.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_merc_umdlc(n,fgrd)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the UMD landcover data and returns the distribution of vegetation in each grid cell, in mercator projection.
The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in mercator projection

15.0.30 read_merc_usgsmask (Source File: read_merc_usgsmask.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_merc_usgsmask(n,localmask)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: localmask(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine reads the landmask data and returns the array in a mercator projection.
The arguments are:

n index of nest

localmask landmask for the region of interest

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in mercator projection

15.0.31 read_merc_usgslc (Source File: read_merc_usgslc.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_merc_usgslc(n,fgrd)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the USGS landcover data and returns the distribution of vegetation in each grid cell, in mercator projection.
The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in mercator projection

15.0.32 read_polar_umdmask (Source File: read_polar_umdmask.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_polar_umdmask(n, localmask)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: localmask(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine reads the landmask data and returns the array in polar stereographic projection.
The arguments are:

n index of nest

localmask landmask for the region of interest

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in polar stereographic projection

---

15.0.33 read_polar_umdlc (Source File: read_polar_umdlc.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_polar_umdlc(n,fgrd)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the UMD landcover data and returns the distribution of vegetation in each grid cell, in a polar stereographic projection.
The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in polar stereographic projection

---

15.0.34 read_polar_usgsmask (Source File: read_polar_usgsmask.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_polar_usgsmask(n, localmask)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: localmask(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine reads the landmask data and returns the array in polar stereographic projection.
The arguments are:

n index of nest

localmask landmask for the region of interest

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in polar stereographic projection

---

15.0.35 read_polar_usgslc (Source File: read_polar_usgslc.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_polar_usgslc(n,fgrd)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the USGS landcover data and returns the distribution of vegetation in each grid cell, in a polar stereographic projection.
The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in polar stereographic projection

---

15.0.36 read_gaussian_gfs_lc (Source File: read_gaussian_gfs_lc.F90)

REVISION HISTORY:

```
22 Sept 2006: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_gaussian_gfs_lc(n, fgrd)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use lis_logmod, only :logunit
use gaussian_mod
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real  :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the GFS landcover data and returns the distribution of vegetation in each grid cell, in a quasi-regular Gaussian lat/lon projection. The data has 13 vegetation types.
The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

15.0.37 read_gaussian_gfs_mask (Source File: read_gaussian_gfs_mask.F90)

REVISION HISTORY:

```
22 Sept 2006: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_gaussian_gfs_mask(nest, localmask)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use lis_logmod, only :logunit
use gaussian_mod
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: nest
real                :: localmask(lis%lnc(nest),lis%lnr(nest))
```

DESCRIPTION:

This subroutine reads the GSF landmask data and returns the array in a quasi-regular Gaussian lat/lon projection.
The arguments are:

n index of nest

localmask landmask for the region of interest

16 Albedo Data

This section describes the routines to ingest various sources of albedo data in different LIS domains

16.0.38 read_gaussian_albedo (Source File: read_gaussian_albedo.F90)

REVISION HISTORY:

```
22 Sept 2006: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_gaussian_albedo(nest, q, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis
use lis_logmod, only :logunit
use gaussian_mod
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: nest
real, intent(inout) :: array(lis%lnc(nest),lis%lnr(nest))
```

DESCRIPTION:

This subroutine retrieves the albedo climatology for the specified month and returns the values in a quasi-regular Gaussian latlon projection
The arguments are:

nest index of the nest

q time index (month or quarter)

array output field with the retrieved albedo

16.0.39 read_gaussian_mxsnoalb (Source File: read_gaussian_mxsnoalb.F90)

REVISION HISTORY:

```
22 Sep 2006: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_gaussian_mxsnoalb(nest,array)
```

USES:

```
use lisdrv_module, only : lis
use gaussian_mod
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber
```

16.0.40 read_gswpalbedo (Source File: read_gswpalbedo.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_gswpalbedo(n,mo, array)
```

USES:

```
#if (defined USE_NETCDF)
  use netcdf
  use spmdMod, only : masterproc
  use lisdrv_module, only : lis,lisdom
  use lis_logmod, only :lis_log_msg
  use gswp_module,   only : getgswp_monindex

  implicit none
```

ARGUMENTS:

```
integer, intent(in)              :: n
integer, intent(in)              :: mo
real, intent(inout)              :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the albedo climatology for the specified month and returns the values in the GSWP data order
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved albedo

The routines invoked are:

getgswp_monindex (2.23.1)
computes the GSWP monthly index

16.0.41 read_gswpmxsnoalb (Source File: read_gswpmxsnoalb.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_gswpmxsnoalb(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_logmod, only : lis_log_msg
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the maximum snow expected over deep snow to be used in the GSWP data order
The arguments are:

n index of the nest

array output field with the retrieved albedo

16.0.42 read_lcalbedo (Source File: read_lcalbedo.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lcalbedo(n, q, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the albedo climatology for the specified month and returns the values in lambert conformal projection
The arguments are:

n index of the nest

q time index (month or quarter)

array output field with the retrieved albedo

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in lambert conformal projection

16.0.43 read_lcmxsnoalb (Source File: read_lcmxsnoalb.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lcmxsnoalb(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the maximum snow expected over deep snow and returns the values in lambert conformal projection
The arguments are:

n index of the nest

array output field with the retrieved albedo

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in lambert conformal projection

16.0.44 read_llalbedo (Source File: read_llalbedo.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
27 Nov 2007: Charles Alonge : Added options for monthly files
```

INTERFACE:

```
subroutine read_llalbedo(n, q, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis
use lis_logmod, only :logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the albedo climatology for the specified month and returns the values in a latlon projection
The arguments are:

n index of the nest

q time index (month or quarter)

array output field with the retrieved albedo

16.0.45 read_llmxsnoalb (Source File: read_llmxsnoalb.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_llmxsnoalb(n,array)
```

USES:

```
use lisdrv_module, only : lis
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber
```

16.0.46 read_mercalbedo (Source File: read_mercalbedo.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_mercalbedo(n,q, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the albedo climatology for the specified month and returns the values in the mercator projection
The arguments are:

n index of the nest

q time index (month or quarter)

array output field with the retrieved albedo

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the mercator projection

16.0.47 read_mercmxsnoalb (Source File: read_mercmxsnoalb.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_mercmxsnoalb(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the maximum snow expected over deep snow and returns the values in mercator projection
The arguments are:

n index of the nest

array output field with the retrieved albedo

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in mercator projection

16.0.48 read_psalbedo (Source File: read_psalbedo.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_psalbedo(n, q, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the albedo climatology for the specified month and returns the values in the polar stereographic projection
The arguments are:

n index of the nest

q time index (month or quarter)

array output field with the retrieved albedo

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the polar stereographic projection

---

16.0.49 read_psmxsnoalb (Source File: read_psmxsnoalb.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_psmxsnoalb(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the maximum snow expected over deep snow and returns the values in the polar stereographic projection
The arguments are:

n index of the nest

array output field with the retrieved albedo

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the polar stereographic projection

17 Greenness Fraction Data

This section describes the routines to ingest various sources of greenness fraction data in different LIS domains

17.0.50 read_gswpgfrac (Source File: read_gswpgfrac.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_gswpgfrac(n,mo, gfracarr)
```

USES:

```
#if (defined USE_NETCDF)
  use netcdf
#endif
  use spmdMod, only : masterproc
  use lisdrv_module, only : lis,lisdom
  use gswp_module, only : getgswp_monindex
  use lis_logmod, only :logunit,lis_log_msg

  implicit none
```

ARGUMENTS:

```
  integer, intent(in) :: n
  integer, intent(in) :: mo
  real, intent(inout) :: gfracarr(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the greenness fraction climatology for the specified month in the GSWP data order. The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved greenness fraction

The routines invoked are:

getgswp_monindex (2.23.1)
computes the GSWP monthly index

17.0.51 read_lcgfrac (Source File: read_lcgfrac.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lcgfrac(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the greenness fraction climatology for the specified month and returns the values in the lambert conformal projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved greenness fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the lambert conformal projection

---

17.0.52 read_llgfrac (Source File: read_llgfrac.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
20 Feb 2006: Sujay Kumar; Modified to support nesting
```

INTERFACE:

```
subroutine read_llgfrac(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the greenness fraction climatology for the specified month and returns the values in the latlon projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved greenness fraction

17.0.53 read_mercgfrac (Source File: read_mercgfrac.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_mercgfrac(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the greenness fraction climatology for the specified month and returns the values in the mercator projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved greenness fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in mercator projection

17.0.54 read_psgfrac (Source File: read_psgfrac.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_psgfrac(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the greenness fraction climatology for the specified month and returns the values in the polar stereographic projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved greenness fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the polar stereographic projection

---

17.0.55 read_catchgfrac (Source File: read_catchgfrac.F90)

REVISION HISTORY:

```
29 Jun 2006: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_catchgfrac(n, mo, array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_logmod, only :logunit,lis_log_msg
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the greenness fraction climatology for the specified month.
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved greenness fraction

17.0.56 read_gaussian_frac (Source File: read_gaussian_gfrac.F90)

REVISION HISTORY:

```
22 Sep 2006: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_gaussian_gfrac(nest, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use gaussian_mod
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: nest
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(nest))
```

DESCRIPTION:

This subroutine retrieves the greenness fraction climatology for the specified month and returns the values in the quasi-regular Gaussian latlon projection
The arguments are:

nest index of the nest

mo time index (month or quarter)

array output field with the retrieved greenness fraction

18 Leaf Area Index Data

This section describes the routines to ingest various sources of LAI data in different LIS domains

18.0.57 read_catchlai (Source File: read_catchlai.F90)

REVISION HISTORY:

```
29 Jun 2006: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_catchlai(n, mo, array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_logmod, only :logunit,lis_log_msg
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
!real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the LAI climatology for the specified month.
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved LAI

18.0.58 read_gswplai (Source File: read_gswplai.F90)

REVISION HISTORY:

```
07 Jul 2005: James Geiger, Initial Specification
```

INTERFACE:

```
subroutine read_gswplai(n, mo, array)
```

USES:

```
   use lisdrv_module, only : lis, lisdom
   use listime_mgr
   use gswp_module,   only : getgswp_monindex
   use lis_logmod, only : logunit, lis_log_msg
#if ( defined USE_NETCDF )
   use netcdf
#endif

   implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, dimension(lis%nch(n)), intent(out) :: array
```

DESCRIPTION:

This program reads in GSWP-2 LAI data
*******Need to be modified ************

18.0.59 read_gswpsai (Source File: read_gswpsai.F90)

REVISION HISTORY:

```
07 Jul 2005: James Geiger, Initial Specification
```

INTERFACE:

```
subroutine read_gswpsai(n, mo, array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use listime_mgr

implicit none
```

ARGUMENTS:

```
integer, intent(in)                    :: n
integer, intent(in)                    :: mo
real, dimension(lis%nch(n))            :: array
```

DESCRIPTION:

This program reads in GSWP-2 SAI data
***** Need to be modified ************

18.0.60 read_lambert_avhrrlai (Source File: read_lambert_avhrrlai.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lambert_avhrrlai(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the AVHRR LAI climatology for the specified month and returns the values in the lambert conformal projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved LAI

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the polar stereographic projection

---

18.0.61 read_lambert_avhrrsai (Source File: read_lambert_avhrrsai.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lambert_avhrrsai(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the AVHRR SAI climatology for the specified month and returns the values in the lambert conformal projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved SAI

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the polar stereographic projection

---

18.0.62 read_lambert_modislai (Source File: read_lambert_modislai.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lambert_modislai(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the MODIS LAI climatology for the specified month and returns the values in the polar stereographic projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved LAI

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the polar stereographic projection

---

18.0.63 read_lambert_modissai (Source File: read_lambert_modissai.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lambert_modissai(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the MODIS SAI climatology for the specified month and returns the values in the polar stereographic projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved SAI

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the polar stereographic projection

---

18.0.64 read_ll_avhrrlai (Source File: read_ll_avhrrlai.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
20 Feb 2006: Sujay Kumar; Modified to support nesting
```

INTERFACE:

```
subroutine read_ll_avhrrlai(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the LAI climatology for the specified month and returns the values in the latlon projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved LAI

18.0.65 read_ll_avhrrsai (Source File: read_ll_avhrrsai.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
20 Feb 2006: Sujay Kumar; Modified to support nesting
```

INTERFACE:

```
subroutine read_ll_avhrrsai(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the SAI climatology for the specified month and returns the values in the latlon projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved SAI

18.0.66 read_ll_modislai (Source File: read_ll_modislai.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
20 Feb 2006: Sujay Kumar; Modified to support nesting
```

INTERFACE:

```
subroutine read_ll_modislai(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the LAI climatology for the specified month and returns the values in the latlon projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved LAI

18.0.67 read_ll_modissai (Source File: read_ll_modissai.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
20 Feb 2006: Sujay Kumar; Modified to support nesting
```

INTERFACE:

```
subroutine read_ll_modissai(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the SAI climatology for the specified month and returns the values in the latlon projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved SAI

18.0.68 read_merc_avhrrlai (Source File: read_merc_avhrrlai.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_merc_avhrrlai(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the AVHRR LAI climatology for the specified month and returns the values in the mercator projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved LAI

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the polar stereographic projection

---

18.0.69 read_merc_avhrrsai (Source File: read_merc_avhrrsai.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_merc_avhrrsai(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the AVHRR SAI climatology for the specified month and returns the values in the mercator projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved SAI

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the polar stereographic projection

---

18.0.70 read_merc_modislai (Source File: read_merc_modislai.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_merc_modislai(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the MODIS LAI climatology for the specified month and returns the values in the mercator projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved LAI

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the polar stereographic projection

---

18.0.71 read_merc_modissai (Source File: read_merc_modissai.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_merc_modissai(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the MODIS SAI climatology for the specified month and returns the values in the mercator projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved SAI

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the polar stereographic projection

---

18.0.72 read_ps_avhrrlai (Source File: read_ps_avhrrlai.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_ps_avhrrlai(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the AVHRR LAI climatology for the specified month and returns the values in the polar stereographic projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved LAI

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the polar stereographic projection

---

18.0.73 read_ps_avhrrsai (Source File: read_ps_avhrrsai.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_ps_avhrrsai(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the AVHRR SAI climatology for the specified month and returns the values in the polar stereographic projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved SAI

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the polar stereographic projection

---

18.0.74 read_ps_modislai (Source File: read_ps_modislai.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_ps_modislai(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the MODIS LAI climatology for the specified month and returns the values in the polar stereographic projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved LAI

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the polar stereographic projection

---

18.0.75 read_ps_modissai (Source File: read_ps_modissai.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_ps_modissai(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%nch(n))
```

DESCRIPTION:

This subroutine retrieves the MODIS SAI climatology for the specified month and returns the values in the polar stereographic projection
The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved SAI

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the polar stereographic projection

19 Soils

This section describes the routines to ingest various sources of soil parameter data in different LIS domains

19.0.76 read_gswp_bexp (Source File: read_gswp_bexp.F90)

REVISION HISTORY:

```
12 Sep 2005: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_gswp_bexp(n,array)
```

USES:

```
#if ( defined USE_NETCDF )
  use netcdf
#endif
  use lisdrv_module, only : lis, lisdom
  use lis_logmod, only : lis_log_msg

  implicit none
```

USES:

```
  integer, intent(in)             :: n
  real, intent(inout)             :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the b parameter data in the GSWP data order
The arguments are:

n index of the nest

array output field with the retrieved b parameter data

19.0.77 read_gswp_ksat (Source File: read_gswp_ksat.F90)

REVISION HISTORY:

```
12 Sep 2005: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_gswp_ksat(n, array)
```

USES:

```
#if ( defined USE_NETCDF )
  use netcdf
#endif
  use lisdrv_module, only : lis,lisdom
  use lis_logmod, only : lis_log_msg

  implicit none
```

ARGUMENTS:

```
integer, intent(in)             :: n
real, intent(inout)             :: array(lis%glbnch(n))
```

DESCRIPTION:

This subroutine retrieves the saturated hydraulic conductivity data in the GSWP data order
The arguments are:

n index of the nest

array output field with the retrieved saturated hydraulic conductivity

---

19.0.78 read_gswp_psisat (Source File: read_gswp_psisat.F90)

REVISION HISTORY:

```
12 Sep 2005: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_gswp_psisat(n, array)
```

USES:

```
#if ( defined USE_NETCDF )
  use netcdf
#endif
  use lisdrv_module, only : lis,lisdom
  use lis_logmod, only : lis_log_msg

  implicit none
```

ARGUMENTS:

```
integer, intent(in)             :: n
real, intent(inout)             :: array(lis%glbnch(n))
```

DESCRIPTION:

This subroutine retrieves the saturated matric potential data in the GSWP data order
The arguments are:

n index of the nest

array output field with the retrieved saturated matric potential

---

19.0.79 read_gswp_soilclass (Source File: read_gswp_soilclass.F90)

REVISION HISTORY:

```
12 Sep 2005: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_gswp_soilclass(n, array)
```

USES:

```
#if ( defined USE_NETCDF )
  use netcdf
#endif
  use lisdrv_module, only : lis, lisdom
  use lis_logmod, only : lis_log_msg

  implicit none
```

ARGUMENTS:

```
  integer, intent(in)             :: n
  real, intent(inout)             :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the soil texture data in the GSWP data order
The arguments are:

n index of the nest

array output field with the retrieved soil texture data

19.0.80 read_gswpclay (Source File: read_gswpclay.F90)

REVISION HISTORY:

```
15 Jun 2005: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_gswpclay(n, array)
```

USES:

```
#if ( defined USE_NETCDF )
  use netcdf
#endif
  use lisdrv_module, only : lis, lisdom
  use lis_logmod, only : lis_log_msg

  implicit none
```

ARGUMENTS:

```
  integer, intent(in)             :: n
  real, intent(inout)             :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the clay fraction data in the GSWP data order
The arguments are:

n index of the nest

array output field with the retrieved clay fraction data

19.0.81 read_gswpsand (Source File: read_gswpsand.F90)

REVISION HISTORY:

```
15 Jun 2005: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_gswpsand(n,array)
```

USES:

```
#if ( defined USE_NETCDF )
  use netcdf
#endif
  use lisdrv_module, only : lis, lisdom
  use lis_logmod, only : lis_log_msg

  implicit none
```

ARGUMENTS:

```
  integer, intent(in)             :: n
  real, intent(inout)             :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the sand fraction data in the GSWP data order
The arguments are:

n index of the nest

array output field with the retrieved sand fraction data

19.0.82 read_gswpsilt (Source File: read_gswpsilt.F90)

REVISION HISTORY:

```
15 Jun 2005: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_gswpsilt(n,array)
```

USES:

```
#if ( defined USE_NETCDF )
  use netcdf
#endif
  use lisdrv_module, only : lis, lisdom
  use lis_logmod, only : lis_log_msg

  implicit none
```

ARGUMENTS:

```
integer, intent(in)             :: n
real, intent(inout)             :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the silt fraction data in the GSWP data order
The arguments are:

n index of the nest

array output field with the retrieved silt fraction data

19.0.83 read_lambert_faoclay (Source File: read_lambert_faoclay.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lambert_faoclay(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in)          :: n
real,          intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO clay fraction data and reprojects it to the lambert conformal projection.
The arguments are:

n index of the nest

array output field with the retrieved clay fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in lambert conformal projection

19.0.84 read_lambert_faocolor (Source File: read_lambert_faocolor.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lambert_faocolor( n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in)          :: n
real,          intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO soil color data and reprojects it to the lambert conformal projection. The arguments are:

n index of the nest

array output field with the retrieved soil color

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in lambert conformal projection

19.0.85 read_lambert_faosand (Source File: read_lambert_faosand.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lambert_faosand(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real,intent(inout)  :: array(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO sand fraction data and reprojects it to the lambert conformal projection. The arguments are:

n index of the nest

array output field with the retrieved sand fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in lambert conformal projection

19.0.86 read_lambert_faosilt (Source File: read_lambert_faosilt.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lambert_faosilt(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO silt fraction data and reprojects it to the lambert conformal projection. The arguments are:

n index of the nest

array output field with the retrieved silt fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in lambert conformal projection

19.0.87 read_lambert_statsgoclay (Source File: read_lambert_statsgoclay.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lambert_statsgoclay(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO clay fraction data and reprojects it to the lambert conformal projection. The arguments are:

n index of the nest

array output field with the retrieved clay fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in lambert conformal projection

19.0.88 read_lambert_statsgosand (Source File: read_lambert_statsgosand.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lambert_statsgosand(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in)  :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO sand fraction data and reprojects it to the lambert conformal projection. The arguments are:

n index of the nest

array output field with the retrieved sand fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in lambert conformal projection

19.0.89 read_lambert_statsgosilt (Source File: read_lambert_statsgosilt.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lambert_statsgosilt(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO silt fraction data and reprojects it to the lambert conformal projection. The arguments are:

n index of the nest

array output field with the retrieved silt fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in lambert conformal projection

19.0.90 read_lambert_statsgotexture (Source File: read_lambert_statsgotexture.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lambert_statsgotexture(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in)   :: n
integer,intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO soil texture data and reprojects it to the lambert conformal projection. The arguments are:

n index of the nest

array output field with the retrieved soil texture

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in lambert conformal projection

19.0.91 read_faoclay (Source File: read_faoclay.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_faoclay(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in)          :: n
real,          intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO clay fraction data and reprojects it to the latlon projection.
The arguments are:

n index of the nest

array output field with the retrieved clay fraction

19.0.92 read_faocolor (Source File: read_faocolor.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_faocolor(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer,       intent(in)    :: n
real,          intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO soil color data and reprojects it to the latlon projection.
The arguments are:

n index of the nest

array output field with the retrieved soil color

19.0.93 read_faosand (Source File: read_faosand.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_faosand(n, array)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in)  :: n !nest index
real,          intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO sand fraction data and reprojects it to the latlon projection.
The arguments are:

n index of the nest

array output field with the retrieved sand fraction

19.0.94 read_faosilt (Source File: read_faosilt.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_faosilt(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer,       intent(in)     :: n
real,          intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO silt fraction data and reprojects it to the latlon projection.
The arguments are:

n index of the nest

array output field with the retrieved silt fraction

19.0.95 read_statsgoclay (Source File: read_statsgoclay.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_statsgoclay(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only :logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer,       intent(in)    :: n
real,          intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO clay fraction data and reprojects it to the latlon projection. The arguments are:

n index of the nest

array output field with the retrieved clay fraction

19.0.96 read_statsgocolor (Source File: read_statsgocolor.F90)

REVISION HISTORY:

```
17 Jan 2007: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_statsgocolor(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only :logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer,       intent(in)    :: n
real,          intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO color fraction data and reprojects it to the latlon projection. The arguments are:

n index of the nest

array output field with the retrieved color fraction

19.0.97 read_statsgososand (Source File: read_statsgosand.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_statsgosand(n, array)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer,       intent(in)    :: n
real,          intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO sand fraction data and reprojects it to the latlon projection. The arguments are:

n index of the nest

array output field with the retrieved sand fraction

19.0.98 read_statsgosilt (Source File: read_statsgosilt.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_statsgosilt(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only :logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer,       intent(in)    :: n
real,          intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO silt fraction data and reprojects it to the latlon projection.
The arguments are:

n index of the nest

array output field with the retrieved silt fraction

19.0.99 read_statsgotexture (Source File: read_statsgotexture.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_statsgotexture(n,array)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod, only : masterproc
use lis_logmod, only :logunit

implicit none
```

ARGUMENTS:

```
integer,intent(in)    :: n
integer,intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO soil texture data and reprojects it to the latlon projection.
The arguments are:

n index of the nest

array output field with the retrieved soil texture

19.0.100 read_merc_faoclay (Source File: read_merc_faoclay.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_merc_faoclay(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod, only       : masterproc
use lis_logmod, only    : logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO clay fraction data and reprojects it to the mercator projection.
The arguments are:

n index of the nest

array output field with the retrieved clay fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the mercator projection

19.0.101 read_merc_faocolor (Source File: read_merc_faocolor.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_merc_faocolor(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO soil color data and reprojects it to the mercator projection.
The arguments are:

n index of the nest

array output field with the retrieved soil color

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in mercator projection

19.0.102 read_merc_faosand (Source File: read_merc_faosand.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_merc_faosand(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod, only       : masterproc
use lis_logmod, only    :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO sand fraction data and reprojects it to the mercator projection.
The arguments are:

n index of the nest

array output field with the retrieved sand fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the mercator projection

19.0.103 read_merc_faosilt (Source File: read_merc_faosilt.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_merc_faosilt(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod, only       : masterproc
use lis_logmod, only    : logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO silt fraction data and reprojects it to the mercator projection.
The arguments are:

n index of the nest

array output field with the retrieved silt fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in the mercator projection

19.0.104 read_merc_statsgoclay (Source File: read_merc_statsgoclay.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_merc_statsgoclay(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO clay fraction data and reprojects it to the mercator projection. The arguments are:

n index of the nest

array output field with the retrieved clay fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in mercator projection

19.0.105 read_merc_statsgosand (Source File: read_merc_statsgosand.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_merc_statsgosand(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO sand fraction data and reprojects it to the mercator projection. The arguments are:

n index of the nest

array output field with the retrieved sand fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in mercator projection

19.0.106 read_merc_statsgosilt (Source File: read_merc_statsgosilt.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_merc_statsgosilt(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real,          intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO silt fraction data and reprojects it to the mercator projection. The arguments are:

n index of the nest

array output field with the retrieved silt fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in mercator projection

19.0.107 read_merc_statsgotexture (Source File: read_merc_statsgotexture.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_merc_statsgotexture(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in)   :: n
integer,intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO soil texture data and reprojects it to the mercator projection. The arguments are:

n index of the nest

array output field with the retrieved soil texture

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in mercator projection

19.0.108 read_polar_faoclay (Source File: read_polar_faoclay.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_polar_faoclay(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO clay fraction data and reprojects it to the polar stereographic projection. The arguments are:

n index of the nest

array output field with the retrieved clay fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in polar stereographic projection

---

19.0.109 read_polar_faocolor (Source File: read_polar_faocolor.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_polar_faocolor(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO soil color data and reprojects it to the polar stereographic projection. The arguments are:

n index of the nest

array output field with the retrieved soil color

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in polar stereographic projection

---

19.0.110 read_polar_faosand (Source File: read_polar_faosand.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_polar_faosand(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO sand fraction data and reprojects it to the polar stereographic projection. The arguments are:

n index of the nest

array output field with the retrieved sand fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in polar stereographic projection

---

19.0.111 read_polar_faosilt (Source File: read_polar_faosilt.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_polar_faosilt(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO silt fraction data and reprojects it to the polar stereographic projection. The arguments are:

n index of the nest

array output field with the retrieved silt fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in polar stereographic projection

---

19.0.112 read_polar_statsgoclay (Source File: read_polar_statsgoclay.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_polar_statsgoclay(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO clay fraction data and reprojects it to the polar stereographic projection. The arguments are:

n index of the nest

array output field with the retrieved clay fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in polar stereographic projection

---

19.0.113 read_polar_statsgosand (Source File: read_polar_statsgosand.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_polar_statsgosand(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO sand fraction data and reprojects it to the polar stereographic projection.
The arguments are:

n index of the nest

array output field with the retrieved sand fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in polar stereographic projection

---

19.0.114 read_polar_statsgosilt (Source File: read_polar_statsgosilt.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_polar_statsgosilt(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod, only       : masterproc
use lis_logmod, only    : logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO silt fraction data and reprojects it to the polar stereographic projection.
The arguments are:

n index of the nest

array output field with the retrieved silt fraction

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in polar stereographic projection

---

19.0.115 read_polar_statsgotexture (Source File: read_polar_statsgotexture.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_polar_statsgotexture(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in)   :: n
integer,intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO soil texture data and reprojects it to the polar stereographic projection. The arguments are:

n index of the nest

array output field with the retrieved soil texture

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in polar stereographic projection

---

19.0.116 read_gfs_texture (Source File: read_gfs_texture.F90)

REVISION HISTORY:

```
22 Sept 2006: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_gfs_texture(nest,array)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use gaussian_mod
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer,intent(in)    :: nest
integer,intent(inout) :: array(lis%lnc(nest),lis%lnr(nest))
```

DESCRIPTION:

This subroutine retrieves GFS soil texture data on a quasi-regular Gaussian latlon projection.
The arguments are:

nest index of the nest

array output field with the retrieved soil texture

20 Snow

This section describes the routines to ingest various sources of snow datasets in different LIS domains

20.0.117 read_llsnowdepth (Source File: read_llsnowdepth.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_llsnowdepth(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This subroutine retrieves the AFWA snowdepth data and reprojects it to the equidistant cylindrical projection. The input data is in inches

---

20.0.118 read_lcsnowdepth (Source File: read_lcsnowdepth.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lcsnowdepth(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This subroutine retrieves the AFWA snowdepth data and reprojects it to the lambert conformal projection. The input data is in inches

---

20.0.119 read_mercsnowdepth (Source File: read_mercsnowdepth.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_mercsnowdepth(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This subroutine retrieves the AFWA snowdepth data and reprojects it to the mercator projection. The input data is in inches

---

20.0.120 read_polarsnowdepth (Source File: read_polarsnowdepth.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_polarsnowdepth(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This subroutine retrieves the AFWA snowdepth data and reprojects it to the polar stereographic projection. The input data is in inches

21 Topography

This section describes the routines to ingest various sources of topography data in different LIS domains

21.0.121 read_lambert_elev_gtopo30 (Source File: read_lambert_elev_gtopo30.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lambert_elev_gtopo30(n,elev)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only :lis_log_msg
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: elev(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static elevation data from the GTOPO30 source and reprojects it to the lambert conformal projection.
The arguments are:

n index of the nest

elev output field with the retrieved elevation

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in lambert conformal projection

21.0.122 read_elev_gtopo30 (Source File: read_elev_gtopo30.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_elev_gtopo30(n,elev)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use lis_logmod, only :lis_log_msg, logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: elev(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static elevation data from the GTOPO30 source and reprojects it to the latlon projection.
The arguments are:

n index of the nest

elev output field with the retrieved elevation

21.0.123 read_slope_gtopo30 (Source File: read_slope_gtopo30.F90)

REVISION HISTORY:

```
22 Dec 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_slope_gtopo30(n,slope)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use lis_logmod, only :lis_log_msg, logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: slope(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static slope data from the GTOPO30 source and reprojects it to the latlon projection.
The arguments are:

n index of the nest

slope output field with the retrieved slope

21.0.124 read_aspect_gtopo30 (Source File: read_aspect_gtopo30.F90)

REVISION HISTORY:

```
22 Dec 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_aspect_gtopo30(n,aspect)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use lis_logmod, only :lis_log_msg, logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber
```

21.0.125 read_curv_gtopo30 (Source File: read_curv_gtopo30.F90)

REVISION HISTORY:

```
22 Dec 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_curv_gtopo30(n,curv)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use lis_logmod, only :lis_log_msg, logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: curv(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static curvature data from the GTOPO30 source and reprojects it to the latlon projection.
The arguments are:

n index of the nest

curv output field with the retrieved curvature

21.0.126 read_merc_elev_gtopo30 (Source File: read_merc_elev_gtopo30.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_merc_elev_gtopo30(n,elev)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only :lis_log_msg
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: elev(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static elevation data from the GTOPO30 source and reprojects it to the mercator projection.
The arguments are:

n index of the nest

elev output field with the retrieved elevation

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in mercator projection

21.0.127 read_polar_elev_gtopo30 (Source File: read_polar_elev_gtopo30.F90)

REVISION HISTORY:

```
03 Sept 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_polar_elev_gtopo30(n,elev)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only :lis_log_msg
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: elev(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static elevation data from the GTOPO30 source and reprojects it to the polar stereographic projection.
The arguments are:

n index of the nest

elev output field with the retrieved elevation

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in polar stereographic projection

---

21.0.128 read_elev_gaussian (Source File: read_elev_gaussian.F90)

REVISION HISTORY:

```
22 Sep 2006: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_elev_gaussian(n,elev)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use lis_logmod, only :lis_log_msg

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: elev(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static elevation data from the GFS source and reprojects it to the quasi-regular Gaussian latlon projection.
The arguments are:

n index of the nest

elev output field with the retrieved elevation

22 Bottom Temperature

This section describes the routines to ingest various sources of bottom boundary temperature data in different LIS domains

22.0.129 read_gswptbot (Source File: read_gswptbot.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_gswptbot(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_logmod, only : lis_log_msg

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static, bottom temperature data in the GSWP data order
The arguments are:

n index of the nest

array output field with the retrieved bottom temperature

22.0.130 read_lambert_statictbot (Source File: read_lambert_statictbot.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_lambert_statictbot(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static, bottom temperature data and reprojects it to the lambert conformal projection.
The arguments are:

n index of the nest

array output field with the retrieved bottom temperature

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in lambert conformal projection

22.0.131 read_statictbot (Source File: read_statictbot.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_statictbot(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static, bottom temperature data and reprojects it to the latlon projection.
The arguments are:

n index of the nest

array output field with the retrieved bottom temperature

22.0.132 read_merc_statictbot (Source File: read_merc_statictbot.F90)

REVISION HISTORY:

```
25 Jan 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_merc_statictbot(n, array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static, bottom temperature data and reprojects it to the mercator projection.
The arguments are:

n index of the nest

array output field with the retrieved bottom temperature

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in mercator projection

22.0.133 read_polar_statictbot (Source File: read_polar_statictbot.F90)

REVISION HISTORY:

```
25 Jul 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_polar_statictbot(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use map_utils
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static, bottom temperature data and reprojects it to the polar stereographic projection.
The arguments are:

n index of the nest

array output field with the retrieved bottom temperature

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in polar stereographic projection

---

22.0.134 read_gaussian_tbot (Source File: read_gaussian_tbot.F90)

REVISION HISTORY:

```
22 Sep 2006: James Geiger; Initial Specification
```

INTERFACE:

```
subroutine read_gaussian_tbot(nest, array)
```

USES:

```
use lisdrv_module, only : lis
use gaussian_mod
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: nest
real, intent(inout) :: array(lis%lnc(nest),lis%lnr(nest))
```

DESCRIPTION:

This subroutine retrieves static, bottom temperature data and reprojects it to the quasi-regular Gaussian latlon projection.
The arguments are:

nest index of the nest

array output field with the retrieved bottom temperature

Part X
Base forcing analyses in LIS

23 Overview

This section contains various model-based and satellite-based meteorological forcing analyses implemented in LIS. These include the algorithms from the AGRMET system of the Air Force Weather Agency (AFWA), Global Data Assimilation System (GDAS) from NCEP, Goddard Earth Observing System (GEOS) from NASA GSFC, forcing products from the ECMWF, Global Soil Wetness Project (GSWP) and the North American Land Data Assimilation System (NLDAS). The details of each scheme are described in the following sections.

24 ECMWF Reanalysis

This section describes the implementation of the bias-corrected atmospheric reanalysis data developed by Aaron Berg. The data is global 0.5 degree in latlon projection.

24.1 Fortran: Module Interface ecmwfreanaldomain_module (Source File: ecmwfreanaldomain_module.F90)

This module contains variables and data structures that are used for the implementation of the bias-corrected ECMWF atmospheric reanalysis data (Berg et al. 2003). The data is global 0.5 degree dataset in latlon projection, and at 6 hourly intervals. The derived data type `ecmwfreanal_struc` includes the variables that specify the runtime options, and the weights and neighbor information to be used for spatial interpolation. They are described below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

fmodeltime1 The nearest, previous 6 hour instance of the incoming data (as a real time).

fmodeltime2 The nearest, next 6 hour instance of the incoming data (as a real time).

remask1d The data used to remask the input data to the LIS mask.

ecmwfreanaldir Directory containing the input data

emaskfile File containing the 0.5 deg land-sea mask used in the input data.

elevfile File with the elevation definition for the input data.

mi Number of points in the input grid

rlat1 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

rlon1 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

n11,n121,n211,n221 Arrays containing the neighbor information of the input grid for each grid point in LIS, for bilinear interpolation.

w111,w121,w211,w221 Arrays containing the weights of the input grid for each grid point in LIS, for bilinear interpolation.

rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

n12,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

Berg, A.A., J.S. Famiglietti, J.P. Walker, and P.R. Houser, 2003: Impact of bias correction to reanalysis products on simulations of North American soil moisture and hydrological fluxes. Journal of Geophysical Research, 108, 4490, DOI: 10.1029/2002JD003334.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
---------------------------------------------------------------------------
 public :: defineNativeECMWFREANAL      !defines the native resolution of
                                  !the input data
---------------------------------------------------------------------------
```

PUBLIC TYPES:

```
---------------------------------------------------------------------------
 public :: ecmwfreanal_struc
```

24.1.1 defineNativeECMWFREANAL (Source File: ecmwfreanaldomain_module.F90)

REVISION HISTORY:

```
26Jan2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine defineNativeECMWFREANAL()
```

USES:

```
use lisdrv_module, only : lis

implicit none
```

DESCRIPTION:

Defines the native resolution of the input forcing for ECMWFREANAL data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readecmwfreanalcrd (24.1.2)
reads the runtime options specified for ECMWFREANAL data

readecmwfreanalmask (24.1.4)
reads the 0.5 degree land sea mask

bilinear_interp_input (4.0.1)
computes the neighbor, weights for bilinear interpolation

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

read_ecmwfreanal_elev (24.1.3)
reads the native elevation of the ecmwfreanal data to be used for topographic adjustments to the forcing

---

24.1.2 readecmwfreanalcrd (Source File: readecmwfreanalcrd.F90)

REVISION HISTORY:

```
26Jan2004; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine readecmwfreanalcrd()
```

USES:

```
use lis_logmod, only : logunit
use lisdrv_module, only : lis, config_lis
use ESMF_Mod
use ecmwfreanaldomain_module, only : ecmwfreanal_struc
```

DESCRIPTION:

This routine reads the options specific to ECMWF reanalysis forcing from the LIS configuration file.

24.1.3 read_ecmwfreanal_elev (Source File: read_ecmwfreanal_elev.F90)

REVISION HISTORY:

```
17Dec2004; Sujay Kumar; Initial Specificaton
3  Dec 2007; Sujay Kumar; Added the abstract method to read the
             EDAS data.
```

INTERFACE:

```
subroutine read_ecmwfreanal_elev(n)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use baseforcing_module, only : lisforc
use ecmwfreanaldomain_module, only : ecmwfreanal_struc
use lis_logmod, only : logunit
```

24.1.4 readecmwfreanalmask (Source File: readecmwfreanalmask.F90)

REVISION HISTORY:

```
16 Apr 2002: Urszula Jambor; Initial Code
24 Nov 2003: Sujay Kumar; Included in LIS
```

INTERFACE:

```
subroutine readecmwfreanalmask(n)
```

USES:

```
use lisdrv_module, only :lis    ! LIS non-model-specific 1-D variables
use ecmwfreanaldomain_module, only : ecmwfreanal_struc
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Reads in land-sea mask for 0.5 degree Reanalysis ECMWF data set, changes grid from ECMWF convention to LIS convention by calling `ecmwfreanalgrid_2_lisgrid`, and transforms grid array into 1D vector for later use. . The arguments are:

n index of the nest

The routines invoked are:

ecmwfreanalgrid_2_lisgrid (24.1.7)
transform the ECMWF Reanalysis data to the LIS grid

24.1.5 getecmwfreanal (Source File: getecmwfreanal.F90)

REVISION HISTORY:

```
11 Apr 2002: Urszula Jambor; original code based on getgeos.f
22 Oct 2002: Urszula Jambor; Limited SW forcing processing to
             land-only grid points
24 Nov 2003: Sujay Kumar; Included the scheme in LIS
```

INTERFACE:

```
subroutine getecmwfreanal(n)
```

USES:

```
use lisdrv_module, only : lis
use baseforcing_module, only : lisforc
use ecmwfreanaldomain_module, only : ecmwfreanal_struc
use listime_mgr
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates 6-hrly, 1/2 degree Reanalysis ECMWF forcing. At the beginning of a simulation, the code reads the most recent past data (nearest 6 hour interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. The strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day . The arguments are:

n index of the nest

The routines invoked are:

tick (2.4.23)
call to advance or retract time

retecmwfreanal (24.1.6)
call to read the ECMWF Reanalysis data and perform spatial interpolation

---

24.1.6 retecmwfreanal (Source File: retecmwfreanal.F90)

REVISION HISTORY:

```
09 Apr 2002: Urszula Jambor; Original code, based on readgeos.f
20 Dec 2002: Matt Rodell; Don't ipolate if running 1/2 deg, also fix
             error in setting v-wind to zero
25 Mar 2003: Urszula Jambor; Modified argument list passed to GEOGFILL2.
24 Nov 2003; Sujay Kumar; Included in LIS
15 Mar 2004; Matt Rodell; Fix test for beginning of restart run
```

INTERFACE:

```
subroutine retecmwfreanal(order, n, yr, mon, da, hr, ferror)
  USES:
  use lisdrv_module, only : lis,lisdom
  use baseforcing_module, only: lisforc
  use ecmwfreanaldomain_module, only : ecmwfreanal_struc
  use spmdMod, only : iam,masterproc
  use lis_logmod, only : logunit

  implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: order
integer, intent(in)    :: n
integer, intent(in)    :: yr
integer, intent(in)    :: mon
integer, intent(in)    :: da
integer, intent(in)    :: hr
integer, intent(inout) :: ferror
```

DESCRIPTION:

For the given time, reads 8 parameters from 1/2 degree Reanalysis ECMWF data, transforms into 9 LIS forcing parameters and interpolates to the LIS domain.
Reanal. ECMWF FORCING VARIABLES available every 6 hours:
1. 2T 2 metre air temperature [K], instantaneous
2. 2D 2 metre dew point temperature [K], instantaneous
3. SSRD Downward shortwave flux, surface [W/m2s], accumulation
4. STRD Downward longwave radiation, surface [W/m2s], accumulation
5. WIND Calculated absolute 10 metre wind speed [m/s], instantaneous
6. P Calculate surface pressure [Pa], instantaneous
7. LSP+CP Total precipitation [m], accumulation
8. CP Convective precipatation [m], accumulation

order flag indicating which data to be read (order=1, read the previous 6 hourly instance, order=2, read the next 6 hourly instance)

n index of the nest

yr current year

mon current month

da current day of the year

hr current hour of day

ferror flag to indicate success of the call (=0 indicates success)

The routines invoked are:

ecmwfreanalgrid_2_lisgrid (24.1.7)
transform the ECMWF REANALYSIS data to the LIS grid

bilinear_interp (4.0.2)
spatially interpolate the forcing data using bilinear interpolation

conserv_interp (4.0.4)
spatially interpolate the forcing data using conservative interpolation

geogfill2 (24.1.10)
fill in any grid points due to mismatched grids.

24.1.7 ecmwfreanalgrid_2_lisgrid (Source File: retecmwfreanal.F90)

REVISION HISTORY:

```
10 Apr 2002: Urszula Jambor;  Code adapted from
             ecmwfgrid_2_grid2catgrid, by R. Reichle
```

INTERFACE:

```
subroutine ecmwfreanalgrid_2_lisgrid( nx, ny, grid_data )

  implicit none
```

ARGUMENTS:

```
integer, intent(in)                   :: nx, ny
real, intent(inout), dimension(nx,ny) :: grid_data
```

DESCRIPTION:

Changes grid data from ECMWF data convention to LIS convention
ECMWF: North-to-South around Greenwich Meridian Global grid. Data are written as flat binary from "upper left to lower right" starting at 0.5-degree grid point center coordinates: 0.25E,89.75N and going to 0.25W,89.75S. Here is the write statement:

```
do i = 1,360
  write(14) (val(j,i),j=1,720)
end do
```

LIS: South-to-North around Date Line Full global grid. Starts at the southernmost latitude and date line, going east and then north.

24.1.8 fillgaps (Source File: retecmwfreanal.F90)

Fills in values for NSIPP tilespace land points where no ECMWF reanalysis data is available via GEOGFILL by assigning most appropriate land-point value along the latitudinal circle of original tilespace point. Developed manually with 17 points in mind.

REVISION HISTORY:

```
 23 Jul 2002: Urszula Jambor
 03 Sep 2002: Urszula Jambor, revised points to reflect
              NSIPP land mask correction.
```

INTERFACE:

```
subroutine fillgaps( nc, nr, v, arr )
```

24.1.9 time_interp_ecmwfreanal (Source File: time_interp_ecmwfreanal.F90)

REVISION HISTORY:

```
24 Nov 2003: Sujay Kumar; Initial version of code adapted from the GLDAS
                          version
 10 Oct 2006: Sujay Kumar: Switched to using ESMF_State for storing
              forcing data.
```

INTERFACE:

```
subroutine time_interp_ecmwfreanal(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis,lisdom
use constantsMod, only  : CONST_SOLAR
use baseforcing_module, only : lisforc, FORC_State
use listime_mgr
use ecmwfreanaldomain_module, only : ecmwfreanal_struc
use spmdMod
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep. Downward shortwave radiation is interpolated using a zenith-angled based approach. Precipitation and longwave radiation are not temporally interpolated, and the previous 6 hourly value is used. All other variables are linearly interpolated between the 6 hourly blocks.
The routines invoked are:

time2date (2.4.22)
converts the time to a date format

zterp (2.29.6)
zenith-angle based interpolation

24.1.10 geogfill2 (Source File: geogfill2.F90)

REVISION HISTORY:

```
20 Sep 2002: Urszula Jambor; Modified original geogfill routine
             for use with Reanalysis forcing sets prepared by
             Aaron Berg via NSIPP
29 Jan 2003: Urszula Jambor; Removed LDAS & GRID modules from list of
             arguments, instead pass only needed variables directly
             (nc,nr,fimask).
27 Jan 2004: Matt Rodell; Make sure all points are filled, rename
             data array to fdata.
```

INTERFACE:

```
subroutine geogfill2(n, nc,nr,geogmask,fdata,v,tmask)
```

USES:

```
use lisdrv_module, only : lisdom

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer             :: nc
integer             :: nr
logical*1           :: geogmask(nc,nr)
real                :: fdata(nc,nr)
integer             :: v
logical*1           :: tmask(nc,nr)
```

DESCRIPTION:

Fill in grid points that are invalid in LIS due to differences in geography between forcing and land surface. Based on original geogfill.F90 for use with Reanalysis forcing fdata, to account for differences in land masks. NOTE** for Reanalysis fdata, v=6 is the V-component of wind, which is always set to ZERO since the U-component (v=5) is assigned the absolute value of wind speed.

For v=1:temperature, v=2:specific humidity, v=4:LW radiation, v=5:wind, and v=7:pressure, data values of zero are not allowed to contribute to average fill-value. For v=3:SW radiation, v=8,9:precipitation, use the mask defined by valid temperature data points to establish whether to seek fill-value for given column and row, rather than include vs exclude zeroes.
The arguments are:

n index of the nest

nc number of columns of the grid (along the east-west dimension)

nr number of rows of the grid (along the north-south dimension)

geogmask interpolated forcing bitmap

fdata interpolated forcing

v index of the variable

tmask valid temperature points (used as the land mask)

24.1.11 ecmwfreanalforcing_finalize (Source File: ecmwfreanalforcing_finalize.F90)

REVISION HISTORY:

```
25 Oct 2005; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine ecmwfreanalforcing_finalize
```

USES:

```
use lisdrv_module, only : lis
use ecmwfreanaldomain_module
```

DESCRIPTION:

Routine to cleanup allocated structures for ecmwf reanalysis forcing.

25 ECMWF

The operational, global analysis products from the European Center for Medium-Range Weather Forecasts (ECMWF) is available on a T_L511 triangular truncation, linear reduced gaussian grid for four synoptic hours: 00, 06, 12, and 18 UTC, which are used in LIS. The radiation products from the data assimilation system at the ECMWF is based on the scheme simulated after Morcrette's work [13]. The estimation of shortwave and longwave radiation includes schemes to include effects of adsorption by water vapor and other gases and aerosol scattering based on different parameterizations [18, 12].

25.1 Fortran: Module Interface ecmwfdomain_module (Source File: ecmwfdomain_module.F90)

This module contains variables and data structures that are used for the implementation of the ECMWF forcing data. The data is global 0.25 degree dataset in latlon projection, and at 3 hourly intervals. The derived data type `ecmwf_struc` includes the variables that specify the runtime options, and the weights and neighbor information to be used for spatial interpolation. They are described below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

nmif Number of forcing variables in the ECMWF data

ecmwftime1 The nearest, previous 3 hour instance of the incoming data (as a real time).

ecmwftime2 The nearest, next 3 hour instance of the incoming data (as a real time).

griduptime1 The time to switch Geopotential field

griduptime2 The time to switch Geopotential field

ecmwfdir Directory containing the input data

elevfile File with the elevation definition for the input data

mi Number of points in the input grid

rlat1 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

rlon1 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

n11,n121,n211,n221 Arrays containing the neighbor information of the input grid for each grid point in LIS, for bilinear interpolation.

w111,w121,w211,w221 Arrays containing the weights of the input grid for each grid point in LIS, for bilinear interpolation.

rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

n12,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: defineNativeECMWF      !defines the native resolution of
                                 !the input data
```

PUBLIC TYPES:

```
public :: ecmwf_struc
```

25.1.1 defineNativeECMWF (Source File: ecmwfdomain_module.F90)

REVISION HISTORY:

```
11Dec2003: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine defineNativeECMWF
```

USES:

```
use lisdrv_module, only: lis
use listime_mgr, only : date2time

implicit none
```

DESCRIPTION:

Defines the native resolution of the input forcing for ECMWF data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readecmwfcrd (25.1.2)
reads the runtime options specified for ECMWF data

bilinear_interp_input (4.0.1)
computes the neighbor, weights for bilinear interpolation

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

---

25.1.2 readecmwfcrd (Source File: readecmwfcrd.F90)

REVISION HISTORY:

```
11 Dec 2003; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine readecmwfcrd()
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis, config_lis
use ecmwfdomain_module, only : ecmwf_struc
use lis_logmod, only : logunit
```

DESCRIPTION:

This routine reads the options specific to ECMWF forcing from the LIS configuration file.

25.1.3 read_ecmwf_elev (Source File: read_ecmwf_elev.F90)

REVISION HISTORY:

```
17Dec2004; Sujay Kumar; Initial Specificaton
```

INTERFACE:

```
subroutine read_ecmwf_elev(n, change)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use baseforcing_module, only :lisforc
use ecmwfdomain_module, only : ecmwf_struc
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: change
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates ECMWF model elevation to the LIS grid. The data will be used to perform any topographical adjustments to the forcing.
The arguments are:

n index of the nest

25.1.4 getecmwf (Source File: getecmwf.F90)

REVISION HISTORY:

```
18 Jun 2003: Urszula Jambor; original code based on getreanlecmwf.F90
20 Feb 2006: Sujay Kumar; Modified with nesting options
```

INTERFACE:

```
subroutine getecmwf(n)
```

USES:

```
use lisdrv_module, only : lis
use baseforcing_module, only : lisforc
use listime_mgr
use ecmwfdomain_module, only : ecmwf_struc
use lis_logmod, only : logunit

implicit none

integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates 3-hrly, 1/4 degree ECMWF forcing. At the beginning of a simulation, the code reads the most recent past data (nearest 3 hour interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. The strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day . The arguments are:

n index of the nest

The routines invoked are:

tick (2.4.23)
call to advance or retract time

retecmwf (25.1.5)
call to read the ECMWF data and perform spatial interpolation

read_ecmwf_elev (25.1.3)
reads the native elevation of the ECMWF data to be used for topographic adjustments to the forcing

---

25.1.5 retecmwf (Source File: retecmwf.F90)

REVISION HISTORY:

```
18 Jun 2003: Urszula Jambor; original code
```

INTERFACE:

```
subroutine retecmwf( order, n, yr, mon, da, hr, ferror )
```

USES:

```
use lisdrv_module, only : lis,lisdom
use baseforcing_module, only : lisforc
use listime_mgr
use ecmwfdomain_module, only : ecmwf_struc
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n  ! lower(1) or upper(2) time interval bdry
integer, intent(in)    :: order  ! lower(1) or upper(2) time interval bdry
integer, intent(in)    :: yr,mon,da,hr      ! data and hour (multiple of 3)
integer, intent(inout) :: ferror           ! set to zero if there's an error
```

DESCRIPTION:

For the given time, reads the parameters from 1/4 degree ECMWF data, transforms into 9 LIS forcing parameters and interpolates to the LIS domain.
ECMWF model output variables used to force LIS fall into 2 categories: –¿ inst3, Instantaneous values, available every 3 hours
–¿ accum, Time integrated values (accumulations), updated every 3 hours

NOTE-1: be aware that ECMWF outputs large-scale and convective precipitation separately. For total precipitation, need to sum the two fields, LSP+CP=TP. Reversed traditional order of model-forcing precip

fields for easier processing – interchange at end of routine to preserve traditional order.
NOTE 2: only time2 SW flux accumulations used in interpolation
NOTE-3: read in of Albedo is currently suppressed. This field is instantaneous and available every 6 hours. At this time, all LIS LSMs replace this parameter.

ECMWF FORCING VARIABLES:
1. T inst3, near-surfcae temperature, 10 metres [K]
2. Q inst3, near-surface specific humidity, 10 metres[kg/kg]
3. SSRD accum, surface solar radiation downwards [W m**-2 s]
4. STRD accum, surface thermal radiation downwards [W m**-2 s]
5. U inst3, zonal wind, 10 metres [m/s]
6. V inst3, meridional wind, 10 metres[m/s]
7. SP inst3, surface pressure [Pa]
8. CP accum, convective precipitation [m]
9. LSP accum, large scale precipitation [m]

order flag indicating which data to be read (order=1, read the previous 3 hourly instance, order=2, read the next 3 hourly instance)

n index of the nest

yr current year

mon current month

da current day of the year

hr current hour of day

ferror flag to indicate success of the call (=0 indicates success)

The routines invoked are:

ret_inst3 (25.1.6)
retrieves the instantaneous variable

ret_accum (25.1.7)
retrieves the accumulated variable

interp_iv (25.1.8)
spatial interpolation from the ECMWF grid to the LIS grid

25.1.6 ret_inst3 (Source File: retecmwf.F90)

INTERFACE:

```
subroutine ret_inst3(dir,yr,mo,da,hr,necmwf,f,id,lb,fret)

  use lis_logmod, only : logunit

  implicit none
```

ARGUMENTS:

```
character*200, intent(in) :: dir
integer, intent(in)       :: yr,mo,da,hr,necmwf,id
real, intent(out)         :: f(necmwf)
logical*1, intent(out)    :: lb(necmwf)
integer, intent(inout)    :: fret
```

DESCRIPTION:

This routine opens the corresponding ECMWF data file to extract the specified variable, which represents an instantaneous value. Should be used for near-surface temperature, specific humidity, winds, and surface pressure.
The arguments are:

dir directory path name for ECMWF data

yr year

mo month

da day of month

hr hour of day

necmwf number of elements of input grid

f 1D input forcing field

id index indicating the forcing variable

lb unpacked input bitmap

fret return code (0-indicates success)

25.1.7 ret_accum (Source File: retecmwf.F90)

INTERFACE:

```
subroutine ret_accum(dir,yr,mo,da,hr,necmwf,f,id,lb,fret)
```

USES:

```
use listime_mgr, only : tick
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
character*200    :: dir
integer          :: yr,mo,da,hr
integer          :: necmwf
real             :: f(necmwf)
integer          :: id
logical*1        :: lb(necmwf)
integer          :: fret
```

DESCRIPTION:

This routine opens the corresponding ECMWF data file to extract the specified variable, which represents an accumulated value. Should be used for radiation and precipitation. Returns field valid for 3-hour interval prior to given time, except for downward sw flux, where interval length varies. Returns forcing in units appropriate to LIS:
W/(m2s) to W/m2
m to mm/s

The arguments are:

dir directory path name for ECMWF data

yr year

mo month

da day of month

hr hour of day

necmwf number of elements of input grid

f 1D input forcing field

id index indicating the forcing variable

lb unpacked input bitmap

fret return code (0-indicates success)

25.1.8 interp_iv (Source File: retecmwf.F90)

INTERFACE:

```
subroutine interp_iv (n, iv,lis_gridDesc,necmwf,nlis,lb,f,lis_1d,mo,iret)
```

USES:

```
use lisdrv_module, only : lis
use ecmwfdomain_module, only : ecmwf_struc
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)   :: n
integer               :: iv
real                  :: lis_gridDesc(50)
integer               :: necmwf
integer               :: nlis
real                  :: f(necmwf)
real, dimension(nlis) :: lis_1d
integer               :: mo
integer               :: iret
```

DESCRIPTION:

This routine interpolates the input forcing to the LIS grid
The arguments are:

n index of the nest

iv field identifier

lis_gridDesc array description of the LIS grid

necmwf number of elements in the input grid

nlis number of elements in the LIS grid

f input data array to be interpolated

lis_1d interpolated array in the LIS grid

mo Number of points in the output grid

iret return code (0 indicates success)

The routines invoked are:

bilinear_interp (4.0.2)
spatially interpolate the forcing data using bilinear interpolation

---

25.1.9 time_interp_ecmwf (Source File: time_interp_ecmwf.F90)

REVISION HISTORY:

```
 1  Oct 1999: Jared Entin; Initial code
 25 Oct 1999: Jared Entin; Significant F90 Revision
 11 Apr 2000: Brian Cosgrove; Fixed name construction error
              in Subroutine ETA6HRFILE
 27 Apr 2000: Brian Cosgrove; Added correction for use of old shortwave
              data with opposite sign convention from recent shortwave data.
              Added capability to use time averaged shortwave & longwave data
              Altered times which are passed into ZTERP--used to be GMT1
              and GMT2, now they are LDAS%ETATIME1 and LDAS%ETATIME2
 30 Nov 2000: Jon Radakovich; Initial code based on geteta.f
 17 Apr 2001: Jon Gottschalck; A few changes to allow model init.
 13 Aug 2001: Urszula Jambor; Introduced missing data replacement.
  5 Nov 2001: Urszula Jambor; Reset tiny negative SW values to zero.
 10 Oct 2006: Sujay Kumar: Switched to using ESMF_State for storing
              forcing data.
```

INTERFACE:

```
subroutine time_interp_ecmwf(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis, lisdom
use baseforcing_module, only : lisforc, FORC_State
use constantsMod,       only : CONST_SOLAR
```

```
use listime_mgr
use spmdMod
use ecmwfdomain_module, only : ecmwf_struc
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep. Downward shortwave radiation is interpolated using a zenith-angled based approach. Precipitation and longwave radiation are not temporally interpolated, and the previous 3 hourly value is used. All other variables are linearly interpolated between the 3 hourly blocks.
The routines invoked are:

time2date (2.4.22)
converts the time to a date format

tick (2.4.23)
advances or retracts time by the specified amount

zterp (2.29.6)
zenith-angle based interpolation

25.1.10 ecmwfforcing_finalize (Source File: ecmwfforcing_finalize.F90)

REVISION HISTORY:

```
25Oct2005; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine ecmwfforcing_finalize
```

USES:

```
use lisdrv_module, only : lis
use ecmwfdomain_module
```

DESCRIPTION:

Routine to cleanup allocated structures for ECMWF forcing.

26 GDAS

The Global Data Assimilation System (GDAS) is the global, operational weather forecast model developed at the Environmental Modeling Center (EMC) of NOAA/NCEP [6]. The radiation algorithms in GDAS include parameterizations for SW↓ and LW↓ radiation interactions between clouds and radiation. The shortwave radiation parameterization follows the the scheme developed by [3] and the longwave radiation scheme is based on the work of [19]. GDAS forcing variables are produced on a quadratic T170 gaussian grid. LIS uses the 00, 03, and as needed, the 06 (hours after current) forecasts, which are produced at 6 hour intervals. From October 2002 onwards, the GDAS outputs are generated at a higher resolution T254 gaussian grid.

26.1 Fortran: Module Interface gdasdomain_module (Source File: gdasdomain_module.F90)

This module contains variables and data structures that are used for the implementation of the forcing data from the Global Data Assimilation System (GDAS) developed at the Environmental Modeling Center (EMC) of NOAA/NCEP. GDAS forcing variables are produced on a quadratic gaussian grid. LIS uses the 00, 03, and as needed, the 06 forecasts, which are produced at 6 hour intervals. The resolution of GDAS forcing varies as follows:
upto 2000/1/24 : T126 (384x190) grid 2001/1/24 - 2002/10/29 : T170 (512x256) grid 2002/10/29 - 2005/5/31 : T254 (768x384) grid 2005/5/31 onwards : T382 (1152x576) grid
The implementation in LIS has the derived data type `gdas_struc` that includes the variables that specify the runtime options, and the weights and neighbor information to be used for spatial interpolation. They are described below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

nmif Number of forcing variables in the GDAS data

gdastime1 The nearest, previous 3 hour instance of the incoming data (as a real time).

gdastime2 The nearest, next 3 hour instance of the incoming data (as a real time).

griduptime1 The time to switch GDAS resolution to T170

griduptime2 The time to switch GDAS resolution to T254

gdasdir Directory containing the input data

elevfile File with the elevation definition for the input data.

mi Number of points in the input grid

rlat1 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

rlon1 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

n11,n121,n211,n221 Arrays containing the neighbor information of the input grid for each grid point in LIS, for bilinear interpolation.

w111,w121,w211,w221 Arrays containing the weights of the input grid for each grid point in LIS, for bilinear interpolation.

rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

n112,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
---------------------------------------------------------------------------
public :: defineNativeGDAS      !defines the native resolution of
                                !the input data
---------------------------------------------------------------------------
```

PUBLIC TYPES:

```
---------------------------------------------------------------------------
public :: gdas_struc
```

26.1.1 defineNativeGDAS (Source File: gdasdomain_module.F90)

REVISION HISTORY:

```
11Dec2003: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine defineNativeGDAS()
```

USES:

```
use lisdrv_module, only: lis
use listime_mgr, only : date2time
```

DESCRIPTION:

Defines the native resolution of the input forcing for GDAS data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readgdascrd (26.1.2)
reads the runtime options specified for GDAS data

date2time (2.4.21)
converts date to the real time format

bilinear_interp_input (4.0.1)
computes the neighbor, weights for bilinear interpolation

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

---

26.1.2 readgdascrd (Source File: readgdascrd.F90)

REVISION HISTORY:

```
11 Dec 2003; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine readgdascrd()
```

USES:

```
use lisdrv_module, only : lis, config_lis
use lis_logmod, only : logunit
use ESMF_Mod
use gdasdomain_module, only: gdas_struc
```

DESCRIPTION:

This routine reads the options specific to GDAS forcing from the LIS configuration file.

26.1.3 read_gdas_elev (Source File: read_gdas_elev.F90)

REVISION HISTORY:

```
17Dec2004; Sujay Kumar; Initial Specificaton
```

INTERFACE:

```
subroutine read_gdas_elev(n, change)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use baseforcing_module, only :lisforc
use gdasdomain_module, only : gdas_struc
use lis_logmod, only : logunit
use gaussian_mod
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: change
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates GDAS model elevation to the LIS grid. The data will be used to perform any topographical adjustments to the forcing.
The arguments are:

n index of the nest

The routines invoked are:

readforcingelev (2.35.10)
Abstract method to read the elevation of the forcing data in the same map projection used in LIS.

26.1.4 getgdas (Source File: getgdas.F90)

REVISION HISTORY:

```
08 Dec 2000: Urszula Jambor; Rewrote geteta.f in fortran90 to
             use GDAS in GLDAS
09 Apr 2001: Urszula Jambor; Added capability of using
             DAAC forcing data every
             6 hours, rather than every 3 hours.
30 May 2001: Urszula Jambor; Changed forcing used: T,q,u fields taken
             from F00 & F03 files, radiation and precip. fields taken
             from F06 & F03 (F03 fields are subtracted out from F06)
11 Dec 2003: Sujay Kumar; Specification of elevation correction options
```

INTERFACE:

```
subroutine getgdas(n)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use baseforcing_module, only: lisforc
use listime_mgr
use gdasdomain_module, only : gdas_struc
use spmdMod
use lis_logmod, only :logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates 3-hrly, GDAS forcing. The idea is to open either the 00 or 03 forecast file associated with the most recent GDAS assimilation (available every 6 hours). Precipitation rates and radiation fluxes will be taken from the F03 and F06 files, since averages are provided. At the beginning of a simulation, the code reads the most recent past data (nearest 3 hour interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. The strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day . The arguments are:

n index of the nest

The routines invoked are:

tick (2.4.23)
determines the GDAS data times

gdasfile (26.1.5)
Puts together appropriate file name for 3 hour intervals

gdasfilef06 (26.1.6)
Puts together appropriate file name for 6 hour intervals

retgdas (26.1.7)
Interpolates GDAS data to LIS grid

bilinear_interp_input (4.0.1)
computes the neighbor, weights for bilinear interpolation

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

read_gdas_elev (26.1.3)
reads the native elevation of the GDAS data to be used for topographic adjustments to the forcing

---

26.1.5 gdasfile (Source File: getgdas.F90)

INTERFACE:

```
subroutine gdasfile( name, gdasdir, yr, mo, da, hr )

   implicit none
```

ARGUMENTS:

```
   character(len=80), intent(in)    :: gdasdir
   integer, intent(in)              :: yr, mo, da, hr
   character(len=80), intent (out)  :: name
```

DESCRIPTION:

This subroutine puts together GDAS file name for 3 hour file intervals
The arguments are:

gdasdir Name of the GDAS directory

yr year

mo month

da day of month

hr hour of day

name name of the timestamped GDAS file

26.1.6 gdasfilef06 (Source File: getgdas.F90)

INTERFACE:

```
subroutine gdasfileF06( name, gdasdir, yr, mo, da, hr )
```

USES:

```
   use listime_mgr

   implicit none
```

ARGUMENTS:

```
character(len=80) :: gdasdir
integer           :: yr, mo, da, hr
character(len=80) :: name
```

DESCRIPTION:

This subroutine puts together GDAS file name for 6 hour forecast files.
The arguments are:

gdasdir Name of the GDAS directory

yr year

mo month

da day of month

hr hour of day

name name of the timestamped GDAS file

26.1.7 retgdas (Source File: retgdas.F90)

REVISION HISTORY:

```
 14 Dec 2000: Urszula Jambor; Rewrote geteta.f to use GDAS forcing in GLDAS
 15 Mar 2001: Jon Gottschalck; Added additional parameters and octets in
              which to search in GRIB files
 01 Jun 2001: Urszula Jambor; Added option to get forcing from different
              files (F00 instantaneous and F06 six hour means)
 29 Jan 2003: Urszula Jambor; Rewrote code, uses GETGB call to replace
              ungribgdas.  Interpolation now occurs in interp_gdas.
              Using GETGB avoids problems with the Oct2002 GDAS
              grid update.
 12 Nov 2003: Matt Rodell; Check to make sure input file exists before
opening and thereby creating a new, empty file.
 14 Nov 2003: Matt Rodell; Ensure lugb varies in call to baopen
 05 Feb 2004: James Geiger; Added GrADS-DODS Server functionality
```

INTERFACE:

```
subroutine retgdas( order, n, name, nameF06, F00flag,ferror,try )
```

USES:

```
use lisdrv_module, only : lis,lisdom
use listime_mgr
use baseforcing_module, only: lisforc
use gdasdomain_module, only : gdas_struc
use spmdMod
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)           :: order
integer, intent(in)           :: n
character(len=80), intent(in) :: name, nameF06
integer, intent(in)           :: F00flag
integer, intent(out)          :: ferror
integer, intent(inout)        :: try
```

DESCRIPTION:

For the given time, reads parameters from GDAS data, transforms into 9 LIS forcing parameters and interpolates to the LIS domain.
The arguments are:

order flag indicating which data to be read (order=1, read the previous 3 hourly instance, order=2, read the next 3 hourly instance)

n index of the nest

name name of the 3 hour GDAS forecast file

nameF06 name of the 6 hour GDAS forecast file

ferror flag to indicate success of the call (=0 indicates success)

try index of the tries (in case of missing data)

The routines invoked are:

interp_gdas (26.1.8)
spatially interpolates a GDAS variable

26.1.8 interp_gdas (Source File: retgdas.F90)

INTERFACE:

```
subroutine interp_gdas(n, kpds,ngdas,f,lb,lis_gds,nc,nr, &
     varfield)
```

USES:

```
use lisdrv_module, only : lis
use gdasdomain_module, only : gdas_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: kpds(200)
integer, intent(in) :: ngdas
real, intent(out)   :: f(ngdas)
logical*1           :: lb(ngdas)
real                :: lis_gds(50)
integer, intent(in) :: nc
integer, intent(in) :: nr
real, intent(out)   :: varfield(nc,nr)
```

DESCRIPTION:

This subroutine interpolates a given GDAS field to the LIS grid. The arguments are:

n index of the nest

kpds grib decoding array

ngdas number of elements in the input grid

f input data array to be interpolated

lb input bitmap

lis_gds array description of the LIS grid

nc number of columns (in the east-west dimension) in the LIS grid

nr number of rows (in the north-south dimension) in the LIS grid

varfield output interpolated field

The routines invoked are:

bilinear_interp (4.0.2)
spatially interpolate the forcing data using bilinear interpolation

conserv_interp (4.0.4)
spatially interpolate the forcing data using conservative interpolation

---

26.1.9 time_interp_gdas (Source File: time_interp_gdas.F90)

REVISION HISTORY:

```
30 Nov 2000: Jon Radakovich; Initial code based on geteta.f
17 Apr 2001: Jon Gottschalck; A few changes to allow model init.
13 Aug 2001: Urszula Jambor; Introduced missing data replacement.
 5 Nov 2001: Urszula Jambor; Reset tiny negative SW values to zero.
10 Oct 2006: Sujay Kumar: Switched to using ESMF_State for storing
             forcing data.
```

INTERFACE:

```
subroutine time_interp_gdas(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only :lis, lisdom
use constantsMod, only  : CONST_SOLAR
use baseforcing_module, only: lisforc, FORC_State
use listime_mgr
use spmdMod
use gdasdomain_module, only : gdas_struc
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep. Downward shortwave radiation is interpolated using a zenith-angled based approach. Precipitation and longwave radiation are not temporally interpolated, and the previous 3 hourly value is used. All other variables are linearly interpolated between the 3 hourly blocks.
The arguments are:

n index of the nest

The routines invoked are:

time2date (2.4.22)
converts the time to a date format

tick (2.4.23)
advances or retracts time by the specified amount

zterp (2.29.6)
zenith-angle based interpolation

26.2 Fortran: Module Interface gdasforcing_finalize (Source File: gdasforcing_finalize.F90)

REVISION HISTORY:

```
25Oct2005; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine gdasforcing_finalize
```

USES:

```
use lisdrv_module, only : lis
use gdasdomain_module
```

DESCRIPTION:

Routine to cleanup allocated structures for GDAS forcing.

27 GEOS

This section describes the implementation of the forcing data available fromt the Goddard Earth Observing System (GEOS) data assimilation system produced by the NASA GSFC's Global Modeling and Assimilation office (GMAO).

27.1 Fortran: Module Interface geosdomain_module (Source File: geosdomain_module.F90)

This module contains variables and data structures that are used for the implementation of the GEOS forcing data. The data is global 1 degree dataset in latlon projection, and at 3 hourly intervals. The derived data type `geos_struc` includes the variables that specify the runtime options, and the weights and neighbor information to be used for spatial interpolation. They are described below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

nmif Number of forcing variables in the ECMWF data

geostime1 The nearest, previous 3 hour instance of the incoming data (as a real time).

geostime2 The nearest, next 3 hour instance of the incoming data (as a real time).

griduptime The time to switch GEOS resolution

geosdir Directory containing the input data

mi Number of points in the input grid

rlat1 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

rlon1 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

n111,n121,n211,n221 Arrays containing the neighbor information of the input grid for each grid point in LIS, for bilinear interpolation.

w111,w121,w211,w221 Arrays containing the weights of the input grid for each grid point in LIS, for bilinear interpolation.

rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

n122,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: defineNativeGEOS      !defines the native resolution of
                                !the input data
```

PUBLIC TYPES:

```
public :: geos_struc
```

27.1.1 defineNativeGEOS (Source File: geosdomain_module.F90)

REVISION HISTORY:

```
11Dec2003: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine defineNativeGEOS
```

USES:

```
use lisdrv_module, only: lis
use listime_mgr, only : date2time
implicit none
```

DESCRIPTION:

Defines the native resolution of the input forcing for GEOS data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readgeoscrd (27.1.2)
reads the runtime options specified for GEOS data

bilinear_interp_input (4.0.1)
computes the neighbor, weights for bilinear interpolation

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

---

27.1.2 readgeoscrd (Source File: readgeoscrd.F90)

REVISION HISTORY:

```
11 Dec 2003; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine readgeoscrd()
```

USES:

```
use lisdrv_module, only : lis, config_lis
use ESMF_Mod
use lis_logmod, only : logunit
use geosdomain_module, only : geos_struc
```

DESCRIPTION:

This routine reads the options specific to GEOS forcing from the LIS configuration file.

27.1.3 getgeos (Source File: getgeos.F90)

REVISION HISTORY:

```
27 Apr 2000: Brian Cosgrove; Added correction for use of old shortwave
             data with opposite sign convention from recent shortwave data.
             Added capability to use time averaged shortwave & longwave data
             Altered times which are passed into ZTERP--used to be GMT1
             and GMT2, now they are LDAS%ETATIME1 and LDAS%ETATIME2
 5 Nov 2001: Urszula Jambor; Reset tiny negative SW values to zero.
10 Feb 2003: Sujay Kumar: Initial Specification in LIS.
```

INTERFACE:

```
subroutine getgeos(n)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod,only : logunit
use listime_mgr
use spmdMod
use baseforcing_module, only: lisforc
use geosdomain_module, only : geos_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates 3-hrly, 1 degree GEOS forcing. At the beginning of a simulation, the code reads the most recent past data (nearest 3 hour interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. The strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day . The arguments are:

n index of the nest

The routines invoked are:

tick (2.4.23)
call to advance or retract time

geosfile (27.1.4)
Puts together appropriate file name for 3 hour intervals

readgeos (27.1.5)
call to read the GEOS data and perform spatial interpolation

bilinear_interp_input (4.0.1)
computes the neighbor, weights for bilinear interpolation

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

27.1.4 geosfile (Source File: getgeos.F90)

INTERFACE:

```
subroutine geosfile(name,geosdir,yr,mo,da,hr,ncold)

  implicit none
```

ARGUMENTS:

```
  character*40, intent(in)  :: geosdir
  integer, intent(in)       :: yr,mo,da,hr,ncold
  character*80, intent(out) :: name
```

DESCRIPTION:

This subroutine puts together GEOS file name for 3 hour file intervals
The arguments are:

geosdir Name of the GEOS directory

yr year

mo month

da day of month

hr hour of day

ncold Number of columns (used as to determine the GEOS resolution)

name name of the timestamped GEOS file

27.1.5 readgeos (Source File: readgeos.F90)

REVISION HISTORY:

```
   11 Apr 2000: Brian Cosgrove; Added read statements for forcing interpolation
   17 Apr 2001: Jon Gottschalck; Added code to perform initialization of
                                 Mosaic with GEOS forcing and new intp. scheme
   14 Aug 2001: Urszula Jambor; Added ferror flag as a routine argument
   07 Dec 2001: Urszula Jambor; Began used of LDAS$%$LDAS_GRIDDESC array
   10 Feb 2003: Sujay Kumar: Initial Specification in LIS.
```

INTERFACE:

```
subroutine readgeos(order,n, name,tscount,ferror)
```

USES:

```
  use lisdrv_module, only : lis,lisdom
  use spmdMod
  use lis_logmod,only : logunit
  use baseforcing_module, only: lisforc
  use geosdomain_module, only : geos_struc

  implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: n
character*80, intent(in) :: name
integer, intent(in)      :: order, tscount
integer, intent(out)     :: ferror
```

DESCRIPTION:

For the given time, reads parameters from GEOS data, transforms into 9 LIS forcing parameters and interpolates to the LIS domain.

GEOS FORCING VARIABLES (unless noted, fields are 3-hr upstream averaged):
1. T 2m Temperature interpolated to 2 metres $[K]$
2. q 2m Instantaneous specific humidity interpolated to 2 metres$[kg/kg]$
3. radswg Downward shortwave flux at the ground $[W/m^2]$
4. lwgdwn Downward longwave radiation at the ground $[W/m^2]$
5. u 10m Instantaneous zonal wind interpolated to 10 metres $[m/s]$
6. v 10m Instantaneous meridional wind interpolated to 10 metres$[m/s]$
7. ps Instantaneous Surface Pressure $[Pa]$
8. preacc Total precipitation $[mm/s]$
9. precon Convective precipatation $[mm/s]$
10. albedo Surface albedo (0-1)

The arguments are:

order flag indicating which data to be read (order=1, read the previous 3 hourly instance, order=2, read the next 3 hourly instance)

n index of the nest

name name of the 3 hour GDAS forecast file

tscount time step count

ferror return error code (0 indicates success)

The routines invoked are:

bilinear_interp (4.0.2)
spatially interpolate the forcing data using bilinear interpolation

conserv_interp (4.0.4)
spatially interpolate the forcing data using conservative interpolation

---

27.1.6 time_interp_geos (Source File: time_interp_geos.F90)

REVISION HISTORY:

```
30 Nov 2000: Jon Radakovich; Initial code based on geteta.f
17 Apr 2001: Jon Gottschalck; A few changes to allow model init.
 5 Nov 2001: Urszula Jambor; Reset tiny negative SW values to zero
10 Feb 2003: Sujay Kumar: Initial Specification in LIS.
10 Oct 2006: Sujay Kumar: Switched to using ESMF_State for storing
             forcing data.
```

INTERFACE:

```
subroutine time_interp_geos(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis, lisdom
use baseforcing_module, only : lisforc, FORC_State
use constantsMod,       only : CONST_SOLAR
use listime_mgr
use lis_logmod,only : logunit
use spmdMod
use geosdomain_module, only :geos_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in):: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep. Downward shortwave radiation is interpolated using a zenith-angled based approach. Precipitation and longwave radiation are not temporally interpolated, and the previous 3 hourly value is used. All other variables are linearly interpolated between the 3 hourly blocks.
The routines invoked are:

time2date (2.4.22)
converts the time to a date format

tick (2.4.23)
advances or retracts time by the specified amount

zterp (2.29.6)
zenith-angle based interpolation

27.1.7 geosforcing_finalize (Source File: geosforcing_finalize.F90)

REVISION HISTORY:

```
25Oct2005; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine geosforcing_finalize
```

USES:

```
use lisdrv_module, only : lis
use geosdomain_module
```

DESCRIPTION:

Routine to cleanup GEOS forcing related memory allocations.

28 GSWP

This section describes the implementation of the forcing data used in the Global Soil Wetness Project Phase II.

28.1 Fortran: Module Interface gswpdomain_module (Source File: gswpdomain_module.F90)

This module contains variables and data structures that are used for the implementation of the forcing data from the Global Soil Wetness Project (GSWP). GSWP forcing variables are produced on a latlon 1degree grid at 3 hour intervals.
The implemenatation in LIS has the derived data type `gswp_struc` that includes the variables that specify the runtime options They are described below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

nmif Number of forcing variables in the GSWP data

gswptime1 The nearest, previous 3 hour instance of the incoming data (as a real time).

gswptime2 The nearest, next 3 hour instance of the incoming data (as a real time).

tair Directory containing the 2m air temperature data

qair Directory containing the 2m specific humidity data

psurf Directory containing the surface pressure data

wind Directory containing the wind data

rainf Directory containing the total precipitation data

snowf Directory containing the total snowfall data

swdown Directory containing the downward shortwave radiation data

swdown Directory containing the downward longwave radiation data

mi Number of points in the input grid

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: defineNativeGSWP      !defines the native resolution of
                                !the input data
```

PUBLIC TYPES:

```
public :: gswp_struc
```

28.1.1 defineNativeGSWP (Source File: gswpdomain_module.F90)

REVISION HISTORY:

```
11Dec2003: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine defineNativeGSWP()
```

USES:

```
use lisdrv_module, only: lis
use listime_mgr, only : date2time
use lis_logmod, only : logunit
```

DESCRIPTION:

Defines the native resolution of the input forcing for GSWP data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readgswpcrd (28.1.2)
reads the runtime options specified for GSWP data

bilinear_interp_input (4.0.1)
computes the neighbor, weights for bilinear interpolation

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

gswp_mask (28.1.9)
reads the GSWP mask

28.1.2 readgswpcrd (Source File: readgswpcrd.F90)

REVISION HISTORY:

```
11 Dec 2003; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine readgswpcrd()
```

USES:

```
use lis_logmod, only : logunit
use lisdrv_module, only : lis, config_lis
use gswpdomain_module, only : gswp_struc
use ESMF_Mod
```

DESCRIPTION:

This routine reads the options specific to GSWP forcing from the LIS configuration file.

28.1.3 getgswp (Source File: getgswp.F90)

REVISION HISTORY:

```
20Feb2004; Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine getgswp(n)
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr
use spmdMod
use baseforcing_module, only: lisforc
use gswpdomain_module, only : gswp_struc
use lis_logmod, only : logunit,lis_log_msg

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates 3-hrly, GSWP forcing. At the beginning of a simulation, the code reads the most recent past data (nearest 3 hour interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. The strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day . The arguments are:

n index of the nest

The routines invoked are:

tick (2.4.23)
determines the GSWP data times

readgswp (28.1.4)
Interpolates GSWP data to LIS grid

28.1.4 readgswp (Source File: readgswp.F90)

REVISION HISTORY:

```
20Feb2004; Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine readgswp(order,n, yr,mo,da,hr,mn,ss)
```

USES:

```
#if ( defined USE_NETCDF )
  use netcdf
#endif
  use lisdrv_module, only : lis,lisdom
  use baseforcing_module, only:lisforc
  use gswp_module, only : getgswp_timeindex
  use gswpdomain_module, only : gswp_struc
  use lis_logmod, only : logunit

  implicit none
```

ARGUMENTS:

```
  integer, intent(in) :: n
```

DESCRIPTION:

For the given time, reads parameters from the GSWP files, transforms into 9 LIS forcing parameters and interpolates to the LIS domain.
The arguments are:

order flag indicating which data to be read (order=1, read the previous 3 hourly instance, order=2, read the next 3 hourly instance)

n index of the nest

yr current year

mo current month

da current day of the month

hr current hour of day

mn current minute

ss current second

The routines invoked are:

getgswp_timeindex (2.23.2)
computes the GSWP time index

gswp_process_forcing (28.1.5)
reads the GSWP forcing from the native NETCDF file

interp_gswp (28.1.6)
interpolates GSWP forcing to the LIS grid

gswp_fillgaps (28.1.12)
fills in the mismatches due to GSWP and LIS masks

28.1.5 gswp_process_forcing (Source File: readgswp.F90)

INTERFACE:

```
subroutine gswp_process_forcing(n, gswp1d, gswp_filled)
```

USES:

```
use gswpdomain_module, only : gswp_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in)                                        :: n
real, dimension(gswp_struc(n)%vector_len), intent(in)      :: gswp1d
real, dimension(gswp_struc(n)%ncold*gswp_struc(n)%nrold), intent(out) :: gswp_filled

real, dimension(gswp_struc(n)%ncold, gswp_struc(n)%nrold) :: gswp2d
```

DESCRIPTION:

Converts the native GSWP vector data to a gridded data, reordered to the LIS format. The arguments are:

n index of the nest

gswp1d native GSWP data as a 1-d vector.

gswp_filled Filled, 1-d vector GSWP field

gswp2d reordered gridded GSWP field

The routines invoked are:

gswp_expand_to_grid (28.1.7)
expand the gswp native data, change to the LIS ordering style.

gspw_2d_to_1d (28.1.8)
convert the 2d field to a 1d vector

28.1.6 interp_gswp (Source File: readgswp.F90)

INTERFACE:

```
subroutine interp_gswp(n,land_len,f,tempgswp)
```

USES:

```
use lisdrv_module,     only : lis, lisdom
use gswpdomain_module, only : gswp_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in)                        :: n
integer, intent(in)                        :: land_len
real, dimension(land_len), intent(in)      :: f
real, dimension(lis%ngrid(n)), intent(out) :: tempgswp
```

DESCRIPTION:

Interpolates the GSWP data onto the LIS grid. The arguments are:

n index of the nest

land_len size of the GSWP data

f native GSWP data

tempgswp interpolated GSWP data

The routines invoked are:

bilinear_interp (4.0.2)
spatially interpolate the forcing data using bilinear interpolation

conserv_interp (4.0.4)
spatially interpolate the forcing data using conservative interpolation

---

28.1.7 gswp_expand_to_grid (Source File: readgswp.F90)

INTERFACE:

```
subroutine gswp_expand_to_grid(n, array1, array2)
```

USES:

```
use gswpdomain_module, only : gswp_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in)                                        :: n
real, dimension(gswp_struc(n)%vector_len), intent(in)      :: array1
real, dimension(gswp_struc(n)%ncold, gswp_struc(n)%nrold), &
      intent(out) :: array2
```

DESCRIPTION:

changes the 1d native GSWP data to a 2-d gridded data, reordered to N-S, E-W style. The arguments are:

n index of the nest

array1 native 1-d GSWP data

array2 reordered 2-d GSWP data.

28.1.8 gswp_2d_to_1d (Source File: readgswp.F90)

INTERFACE:

```
subroutine gswp_2d_to_1d(n, array2, array1)
```

USES:

```
use gswpdomain_module, only : gswp_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in)                                        :: n
real, dimension(gswp_struc(n)%ncold, gswp_struc(n)%nrold),          &
      intent(in)                                           :: array2
real, dimension(gswp_struc(n)%ncold*gswp_struc(n)%nrold),           &
      intent(out)                                          :: array1
```

DESCRIPTION:

reshapes the 2d-array to 1d array.
The arguments are:

n index of the nest

array1 input 2-d array

array2 input 1-d array

28.1.9 gswp_mask (Source File: readgswp.F90)

INTERFACE:

```
subroutine gswp_mask()
#if ( defined USE_NETCDF )
```

USES:

```
use lisdrv_module,     only : lis
use gswpdomain_module, only : gswp_struc
use netcdf
```

DESCRIPTION:

Reads the GSWP mask and computes the masking index.

28.1.10 time_interp_gswp (Source File: time_interp_gswp.F90)

REVISION HISTORY:

```
20 Feb 2004; Sujay Kumar : Initial Specification
10 Oct 2006: Sujay Kumar: Switched to using ESMF_State for storing
             forcing data.
```

INTERFACE:

```
subroutine time_interp_gswp(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis,lisdom
use constantsMod,  only : CONST_SOLAR
```

```
use baseforcing_module, only : lisforc, FORC_State
use listime_mgr
use gswpdomain_module, only :gswp_struc
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep. Downward shortwave radiation is interpolated using a zenith-angled based approach. Precipitation and longwave radiation are not temporally interpolated, and the previous 3 hourly value is used. All other variables are linearly interpolated between the 3 hourly blocks.
The routines invoked are:

time2date (2.4.22)
converts the time to a date format

zterp (2.29.6)
zenith-angle based interpolation

finterp (28.1.11)
routine to compute temporal disaggregation of forcing

28.1.11 finterp (Source File: finterp.F90)

REVISION HISTORY:

```
2002/08/16  15:55:18  guo
```

INTERFACE:

```
subroutine finterp(ip,trp_flag,val0,val1,val2,val3,madtt,istep,vout)
```

USES:

```
use lis_logmod, only : logunit

implicit none

integer,intent(in)           :: ip
integer,intent(in)           :: istep
character(len=1),intent(in)  :: trp_flag
real,intent(in)              :: val0
real,intent(in)              :: val1
real,intent(in)              :: val2
real,intent(in)              :: val3
integer,intent(in)           :: madtt
real                         :: vout
```

DESCRIPTION:

Introduced a new interpolation type "P" which specifies a temporal disaggregation of interpolated precipitation to improve partitioning of infiltration/runoff when forcing data interval is large compared to the typical length of a convective (or total) rain event.

** This routine interpolates atmospheric adtt seconds period forcing data to dtt seconds valtrp time-step for one adtt interval. Interpolation is performed based on the value of 'trp_flag':

"L" or "l" = value represents average over interval ending at current time
"N" or "n" = value represents average over interval beginning at current time
"C" or "c" = value represents average over interval centered on current time
"I" or "i" = instantaneous value at current time (linear interpolation)
"P" or "p" = PDF disaggregation applied in time (for precip)
"0" (zero) = no interpolation, centered on current time
Otherwise = no interpolation, applied beginning at current time

* NOTE:
* For "L", "N", and "C" to conserve the mean over the interval after
* interpolation, 'madtt' MUST BE A MULTIPLE OF 2!

The arguments are:

ip grid point in question

trp_flag type of interpolation

val0 last forcing value (for cases N and C)

val1 current forcing value (all cases)

val2 next forcing value (for all cases but default)

val3 ueber-next forcing value (for all cases C and L)

madtt number of valtrp timesteps in a forcing timestep

vout output forcing value

28.1.12 gswp_fillgaps (Source File: gswp_fillgaps.F90)

INTERFACE:

```
subroutine gswp_fillgaps(n,ip,varfield)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use gswpdomain_module, only : gswp_struc
use lis_logmod, only   : logunit

implicit none
```

USES:

```
integer, intent(in)    :: n
integer, intent(in)    :: ip
real, intent(inout)    :: varfield(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine fills in invalid grid points introduced due to mismatches in landmasks in LIS and GSWP. This routine assumes that the undef or invalid value is the LIS undefined value.
The arguments are:

n index of the nest

ip interpolation option

varfield updated output field

28.2 Fortran: Module Interface gswpforcing_finalize (Source File: gswpforcing_finalize.F90)

REVISION HISTORY:

```
25Oct2005; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine gswpforcing_finalize
```

USES:

```
use gswpdomain_module
```

DESCRIPTION:

Routine to cleanup allocated structures for GSWP forcing.

29 GSWP1

This section describes the implementation of the forcing data used in the Global Soil Wetness Project Phase I.

29.1 Fortran: Module Interface gswp1domain_module (Source File: gswp1domain_module.F

This module contains variables and data structures that are used for the implementation of the forcing data from the Global Soil Wetness Project (GSWP). GSWP forcing variables are produced on a latlon 1degree grid at 3 hour intervals.
The implemenatation in LIS has the derived data type `gswp_struc` that includes the variables that specify the runtime options They are described below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

nmif Number of forcing variables in the GSWP data

gswptime1 The nearest, previous 3 hour instance of the incoming data (as a real time).

gswptime2 The nearest, next 3 hour instance of the incoming data (as a real time).

tair Directory containing the 2m air temperature data

qair Directory containing the 2m specific humidity data

psurf Directory containing the surface pressure data

wind Directory containing the wind data

rainf Directory containing the total precipitation data

snowf Directory containing the total snowfall data

swdown Directory containing the downward shortwave radiation data

swdown Directory containing the downward longwave radiation data

mi Number of points in the input grid

29.1.1 defineNativeGSWP1 (Source File: gswp1domain_module.F90)

REVISION HISTORY:

```
11 Dec 2003: Sujay Kumar, Initial Specification
```

INTERFACE:

```
subroutine defineNativeGSWP1()
```

USES:

```
use lisdrv_module, only: lis
use lis_logmod,    only : logunit
```

DESCRIPTION:

Defines the native resolution of the input forcing for GSWP1 data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readgswp1crd (29.1.2)
reads the runtime options specified for GSWP1 data

29.1.2 readgswp1crd (Source File: readgswp1crd.F90)

REVISION HISTORY:

```
11 Dec 2003: Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine readgswp1crd()
```

USES:

```
use ESMF_Mod
use lis_logmod, only : logunit
use lisdrv_module, only : lis, config_lis
use gswp1domain_module, only : gswp1_struc
```

DESCRIPTION:

This routine reads the options specific to GSWP1 forcing from the LIS configuration file.

29.1.3 getgswp1 (Source File: getgswp1.F90)

REVISION HISTORY:

```
11 Dec 2003: Sujay Kumar, Initial Specification
```

INTERFACE:

```
subroutine getgswp1(n)
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr
use spmdMod
use baseforcing_module, only: lisforc
use gswp1domain_module, only : gswp1_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in)  :: n
```

DESCRIPTION:

Opens, reads, and interpolates hourly, GSWP1 forcing. At the beginning of a simulation, the code reads the most recent past data (nearest hourly interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. The strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day . The arguments are:

n index of the nest

The routines invoked are:

tick (2.4.23)
determines the GSWP data times

gswp1file (29.1.4)
determines the GSWP1 data file

readgswp1 (29.1.5)
reads GSWP1 data to LIS grid

29.1.4 gswp1file (Source File: getgswp1.F90)

INTERFACE:

```
subroutine gswp1file(name,gswp1dir,yr,mo,da,hr,ncold)

implicit none
```

INPUT PARAMETERS:

```
character*100, intent(in) :: gswp1dir
integer, intent(in)      :: yr,mo,da,hr,ncold
```

OUTPUT PARAMETERS:

```
character*80, intent(out) :: name
```

DESCRIPTION:

This subroutine puts together GSWP-1 file name

29.1.5 readgswp1 (Source File: readgswp1.F90)

REVISION HISTORY:

```
11 Dec 2003: Sujay Kumar, Initial Specification
```

INTERFACE:

```
subroutine readgswp1(order,n, name,tscount,ferror)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use baseforcing_module, only : lisforc
use gswp1domain_module, only : gswp1_struc
use lis_logmod, only         : logunit
use spmdMod, only            : iam
```

DESCRIPTION:

Reads in GSWP-1 data and performs interpolation to the LIS domain.

29.2 Core Functions of readgswp1

bilinear_interp Interpolates GSWP-1 data to LIS grid using bilinear interpolation

GSWP-1 FORCING VARIABLES (unless noted, fields are 3-hr upstream averaged):
1. T 2m Temperature interpolated to 2 metres $[K]$
2. q 2m Instantaneous specific humidity interpolated to 2 metres$[kg/kg]$
3. radswg Downward shortwave flux at the ground $[W/m^2]$
4. lwgdwn Downward longwave radiation at the ground $[W/m^2]$
5. u 10m Instantaneous zonal wind interpolated to 10 metres $[m/s]$
6. v 10m Instantaneous meridional wind interpolated to 10 metres$[m/s]$
7. ps Instantaneous Surface Pressure $[Pa]$
8. preacc Total precipitation $[mm/s]$
9. precon Convective precipatation $[mm/s]$
10. albedo Surface albedo (0-1)

29.2.1 check_error_gswp1 (Source File: readgswp1.F90)

INTERFACE:

```
subroutine check_error_gswp1(ierr,msg,iam)
```

USES:

```
use lis_logmod, only : logunit

implicit none
```

DESCRIPTION:

Error check; Program exits in case of error

29.2.2 time_interp_gswp1 (Source File: time_interp_gswp1.F90)

REVISION HISTORY:

```
11 Dec 2003: Sujay Kumar, Initial Specification
```

INTERFACE:

```
subroutine time_interp_gswp1(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis, lisdom
use baseforcing_module, only : lisforc, FORC_State
use listime_mgr, only        : get_nstep, time2date
use constantsMod, only       : CONST_SOLAR
use gswp1domain_module, only : gswp1_struc
use lis_logmod, only         : logunit
```

ARGUMENTS:

```
implicit none

integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep. Downward shortwave radiation is interpolated using a zenith-angled based approach. Precipitation and longwave radiation are not temporally interpolated, and the previous hourly value is used. All other variables are linearly interpolated between the hourly blocks.
The routines invoked are:

time2date (2.4.22)
converts the time to a date format

zterp (2.29.6)
zenith-angle based interpolation

29.3 Fortran: Module Interface gswp1forcing_finalize (Source File: gswp1forcing_finalize.F90

REVISION HISTORY:

```
25Oct2005; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine gswp1forcing_finalize
```

USES:

```
use gswp1domain_module
```

DESCRIPTION:

Routine to cleanup allocated structures for GSWP1 forcing.

30 NLDAS

The atmospheric forcing used in the North American Land Data Assimilation System (NLDAS) [5] features products at an hourly 0.125° spatial resolution, over the continental United States. The SW↓ is generated from a 1/2° product derived at the University of Maryland from NOAA's Geostationary Operational Environmental Satellites (GOES). The LW↓ is derived from 3 hourly NCEP Eta Data Assimilation System (EDAS) output fields [17], and from 3 hourly and 6 hourly Eta mesoscale model forecast fields when EDAS data is unavailable.

30.1 Fortran: Module Interface nldasdomain_module (Source File: nldasdomain_module.F90)

This module contains variables and data structures that are used for the implementation of the forcing data used in the North American Land Data Assimilation System (NLDAS; Cosgrove et al.(2003)). The variables are produced at 0.125 degree spatial resolution, and at hourly intervals.
Cosgrove, B. Real-time and retrospective forcing in the North American Land Data Assimilation System (NLDAS) project. Journal of Geophysical Research, 108 (D22), 8842, DOI:10.1029/2002JD003118
The implemenatation in LIS has the derived data type `nldas_struc` that includes the variables that specify the runtime options, and the weights and neighbor information to be used for spatial interpolation. They are described below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

nldastime1 The nearest, previous hourly instance of the incoming data (as a real time).

nldastime2 The nearest, next hourly instance of the incoming data (as a real time).

nldasdir Directory containing the input data

elevfile File with the elevation definition for the input data.

mi Number of points in the input grid

orig_ediff Original NLDAS elevation difference field

rlat1 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

rlon1 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

n111,n121,n211,n221 Arrays containing the neighbor information of the input grid for each grid point in LIS, for bilinear interpolation.

w111,w121,w211,w221 Arrays containing the weights of the input grid for each grid point in LIS, for bilinear interpolation.

rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

n122,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

rlat3 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for nearest neighbor interpolation)

rlon3 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for nearest neighbor interpolation)

n113 Array containing the neighbor information of the input grid for each grid point in LIS, for nearest neighbor interpolation.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
---------------------------------------------------------------------------
 public :: defineNativeNLDAS      !defines the native resolution of
                                 !the input data
---------------------------------------------------------------------------
```

PUBLIC TYPES:

```
---------------------------------------------------------------------------
 public :: nldas_struc
```

30.1.1 defineNativeNLDAS (Source File: nldasdomain_module.F90)

REVISION HISTORY:

```
02Feb2004: Sujay Kumar; Initial Specification
20Oct2007: Kristi Arsenault; Changed to EDAS elev file for new diff
```

INTERFACE:

```
subroutine defineNativeNLDAS()
```

USES:

```
  use lisdrv_module, only: lis
  use listime_mgr, only : date2time
```

DESCRIPTION:

Defines the native resolution of the input forcing for NLDAS data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readnldascrd (30.1.2)
reads the runtime options specified for NLDAS data

date2time (2.4.21)
converts date to the real time format

bilinear_interp_input (4.0.1)
computes the neighbor, weights for bilinear interpolation

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

neighbor_interp_input (4.0.5)
computes the neighbor, weights for nearest neighbor interpolation

read_edas_elev (30.1.3)
reads the native elevation of the EDAS data to be used for topographic adjustments to the baseforcing data

---

30.1.2 readnldascrd (Source File: readnldascrd.F90)

REVISION HISTORY:

```
02Feb2004; Sujay Kumar, Initial Code
04Mar2007; Kristi Arsenault, Implemented EDAS Elevation Correction
```

INTERFACE:

```
subroutine readnldascrd()
```

USES:

```
use lisdrv_module, only : lis
use nldasdomain_module, only : nldas_struc
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use lisdrv_module, only : config_lis
use ESMF_Mod
```

DESCRIPTION:

This routine reads the options specific to NLDAS forcing from the LIS configuration file.

30.1.3 read_edas_elev (Source File: read_edas_elev.F90)

REVISION HISTORY:

```
17 Dec 2004; Sujay Kumar; Initial Specificaton
20 Dec 2006; Kristi Arsenault; Changed to read EDAS elevation only
3  Dec 2007; Sujay Kumar; Added the abstract method to read the
             EDAS data.
```

INTERFACE:

```
subroutine read_edas_elev(n)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use baseforcing_module, only : lisforc
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates NLDAS-EDAS model elevation to the LIS grid. The data will be used to perform any topographical adjustments to the forcing.
The arguments are:

n index of the nest

The routines invoked are:

readforcingelev (2.35.10)
abstract method to read EDAS elevation data in the map projection used in the LIS model grid.

30.1.4 read_edas_elev_latlon (Source File: read_edas_elev_latlon.F90)

REVISION HISTORY:

```
03 Dec 2007: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_edas_elev_latlon(n,elev)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use nldasdomain_module, only : nldas_struc
use lis_logmod, only :lis_log_msg, logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: elev(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the static NLDAS-EDAS data and reprojects it to the latlon projection.
The arguments are:

n index of the nest

elev output field with the retrieved elevation

The routines invoked are:

getNextUnitNumber (2.26.1)
obtain the next available unit number

releaseUnitNumber (2.26.2)
release the unit number after use.

30.1.5 orig_elevdiff (Source File: orig_elevdiff.F90)

REVISION HISTORY:

```
20 Oct 2006: Kristi Arsenault; Adapted elevation correction code
             to remove such a correction from NLDAS fields
```

INTERFACE:

```
subroutine orig_elevdiff(nest)
```

USES:

```
use nldasdomain_module, only : nldas_struc
use lis_logmod, only :logunit
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: nest
```

DESCRIPTION:

Open and read in original NLDAS elevation difference file.
The arguments are:

nest index of the domain or nest.

The routines invoked are:

getNextUnitNumber (2.26.1)
obtain the next available unit number

releaseUnitNumber (2.26.2)
release the unit number after use.

30.1.6 nldas_ec_removal (Source File: nldas_ec_removal.F90)

REVISION HISTORY:

```
20 Oct 2006: Kristi Arsenault; Adapted elevation correction code
             to remove such a correction from NLDAS fields
```

INTERFACE:

```
subroutine nldas_ec_removal( nest, point, force_tmp, force_hum, &
                             force_lwd, force_prs )
```

USES:

```
use lisdrv_module, only: lis, lisdom
use nldasdomain_module, only : nldas_struc
use lis_logmod, only :logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: nest
```

DESCRIPTION:

Removes Temperature, Pressure, Humidity and Longwave Radiation forcing correction.
The corrections are based on the lapse-rate and hypsometric adjustments to these variables described in Cosgrove et. al (2003).
Cosgrove, B.A. et.al, Real-time and retrospective forcing in the North American Land Data Assimilation (NLDAS) project, Journal of Geophysical Research, 108(D22), 8842, DOI: 10.1029/2002JD003118, 2003.
The arguments are:

nest index of the domain or nest.

point index of the grid point

force_tmp temperature value for the grid point

force_hum specific humidity for the grid point

force_lwd downward longwave radiation for the grid point

force_prs surface pressure for the grid point

30.1.7 getnldas (Source File: getnldas.F90)

REVISION HISTORY:

```
27 Apr 2000: Initial Specification
02 Feb 2004 : Sujay Kumar ; Initial Version in LIS
06 Mar 2007 : Kristi Arsenault; Implemented EDAS-Elevation Correction Var
```

INTERFACE:

```
subroutine getnldas(n)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use baseforcing_module, only : lisforc
use nldasdomain_module, only : nldas_struc
use listime_mgr, only : tick
use lis_logmod, only :logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates hourly, NLDAS forcing. At the beginning of a simulation, the code reads the most recent past data (nearest hourly interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. The strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day . The arguments are:

n index of the nest

The routines invoked are:

tick (2.4.23)
determines the NLDAS data times

nldasfile (30.1.8)
Puts together appropriate timestamped filename

retnldas (30.1.9)
Interpolates NLDAS data to LIS grid

bilinear_interp_input (4.0.1)
computes the neighbor, weights for bilinear interpolation

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

---

30.1.8 nldasfile (Source File: getnldas.F90)

REVISION HISTORY:

```
04 Sep 2001: Brian Cosgrove; Use of NASA data enabled, updated
             reading of data directory structure to read new format
```

INTERFACE:

```
subroutine nldasfile(name,nldasdir,yr,mo,da,hr)

implicit none
```

ARGUMENTS:

```
character*80, intent(out)::  name
character*40, intent(in) ::  nldasdir
integer, intent(in)      :: yr,mo,da,hr
```

DESCRIPTION:

This subroutine puts together NLDAS file name for 1 hour file intervals
The arguments are:

nldasdir Name of the NLDAS directory

yr year

mo month

da day of month

hr hour of day

name name of the timestamped NLDAS file

30.1.9 retnldas (Source File: retnldas.F90)

REVISION HISTORY:

```
11 Apr 2000: Brian Cosgrove; changed code to use Forcing Mask (With inland
             water filled in).  Deleteted unused variables.
27 Apr 2000: Brian Cosgrove; changed code to use the original
             mask again since that is the
             mask which  NCEP has already applied to the forcing data
             by the time NASA gets it......not possible to use the
             expanded NASA forcing mask
1  May 2000: Brian Cosgrove; changed code so that if parameter 11 (sw)
             is not found in hourly ncep data, it will just use
             edas-based shortwave from the hourly ncep files
20 Jun 2000: Brian Cosgrove; changed code so that it uses  LDAS%UDEF and
              not a hard-wired undefined value of -999.9 and -999.0
18 Aug 2000: Brian Cosgrove; changed code so that FMASK and not MASK
             is used when ungribbing.  NCEP data already has a mask applied
             to it and so may not be able to supply forcing data to
             all LDAS land forcing points.  In areas where LDAS
             forcing mask states that land exists, but where NCEP forcing
             data is non-existant, assign undefined value to forcing data.
22 Aug 2000: Brian Cosgrove; Altered code for US/Mexico/Canada Mask
05 Sep 2001: Brian Cosgrove; Removed dirnom and infile variables, changed
             call to ungribncep to match removal.  Added code to make use
             of precip weighting mask
02 Feb 2004: Sujay Kumar; Initial Specification in LIS
20 Oct 2007: Kristi Arsenault; Updated NLDAS Code to Read EDAS Height Field
```

INTERFACE:

```
subroutine retnldas(order,n, name,ferror)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use nldasdomain_module, only : nldas_struc
use baseforcing_module, only : lisforc
use lis_logmod, only :logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: order
integer, intent(in)      :: n
character*80, intent(in) :: name
integer, intent(out)     :: ferror
```

DESCRIPTION:

For the given time, reads parameters from NLDAS data, transforms into 9 LIS forcing parameters and interpolates to the LIS domain.
The arguments are:

order flag indicating which data to be read (order=1, read the previous hourly instance, order=2, read the next hourly instance)

n index of the nest

name name of the 3 hour GDAS forecast file

ferror flag to indicate success of the call (=0 indicates success)

The routines invoked are:

interp_nldas (30.1.10)
spatially interpolates a NLDAS variable

30.1.10 interp_nldas (Source File: interp_nldas.F90)

INTERFACE:

```
subroutine interp_nldas(n, kpds, nldas,f,lb,lis_gds,nc,nr, &
     varfield)
```

USES:

```
use lisdrv_module, only : lis
use nldasdomain_module, only :nldas_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in)   :: n
integer, intent(in)   :: kpds(200)
integer, intent(in)   :: nldas
real, intent(in)      :: f(nldas)
logical*1, intent(in) :: lb(nldas)
real, intent(in)      :: lis_gds(50)
integer, intent(in)   :: nc
integer, intent(in)   :: nr
real, intent(inout)   :: varfield(nc,nr)
```

DESCRIPTION:

This subroutine interpolates a given NLDAS field to the LIS grid. The arguments are:

n index of the nest

kpds grib decoding array

ngdas number of elements in the input grid

f input data array to be interpolated

lb input bitmap

lis_gds array description of the LIS grid

nc number of columns (in the east-west dimension) in the LIS grid

nr number of rows (in the north-south dimension) in the LIS grid

varfield output interpolated field

The routines invoked are:

bilinear_interp (4.0.2)
spatially interpolate the forcing data using bilinear interpolation

conserv_interp (4.0.4)
spatially interpolate the forcing data using conservative interpolation

neighbor_interp (4.0.6)
spatially interpolate the forcing data using neighbor interpolation

---

30.1.11 time_interp_nldas (Source File: time_interp_nldas.F90)

REVISION HISTORY:

```
02Feb2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine time_interp_nldas(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis,lisdom
use constantsMod, only  : CONST_SOLAR
use baseforcing_module, only : lisforc, FORC_State
use nldasdomain_module, only : nldas_struc
use listime_mgr, only : tick, time2date
use spmdMod, only : iam
use lis_logmod, only :logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep. Downward shortwave radiation is interpolated using a zenith-angled based approach. Precipitation and longwave radiation are not temporally interpolated, and the previous 3 hourly value is used. All other variables are linearly interpolated between the 3 hourly blocks.
The routines invoked are:

time2date (2.4.22)
converts the time to a date format

tick (2.4.23)
advances or retracts time by the specified amount

zterp (2.29.6)
zenith-angle based interpolation

30.2 Fortran: Module Interface nldasforcing_finalize (Source File: nldasforcing_finalize.F90)

REVISION HISTORY:

```
25Oct2005; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine nldasforcing_finalize
```

USES:

```
use lisdrv_module, only : lis
use nldasdomain_module
```

DESCRIPTION:

Routine to cleanup nldas forcing related memory allocations.

31 NLDAS2

The atmospheric forcing used in the North American Land Data Assimilation System Phase II (NLDAS1) features products at an hourly 0.125° spatial resolution, over the continental United States.

31.1 Fortran: Module Interface nldas2domain_module (Source File: nldas2domain_module.F

This module contains variables and data structures that are used for the implementation of the forcing data used in the North American Land Data Assimilation System Phase II. The variables are produced at 0.125 degree spatial resolution, and at hourly intervals. For more details please view the forcing files manual available at the following URL:
http://ldas.gsfc.nasa.gov/nldas2/nldas2forcing.pdf
The implemenatation in LIS has the derived data type `nldas2_struc` that includes the variables that specify the runtime options, and the weights and neighbor information to be used for spatial interpolation. They are described below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

nldas2time1 The nearest, previous hourly instance of the incoming data (as a real time).

nldas2time2 The nearest, next hourly instance of the incoming data (as a real time).

nldas2dir Directory containing the input data

elevfile File with the elevation definition for the input data.

mi Number of points in the input grid

rlat1 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

rlon1 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

n111,n121,n211,n221 Arrays containing the neighbor information of the input grid for each grid point in LIS, for bilinear interpolation.

w111,w121,w211,w221 Arrays containing the weights of the input grid for each grid point in LIS, for bilinear interpolation.

rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

n122,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

rlat3 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for n. neighbor interpolation)

rlon3 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for n. neighbor interpolation)

n113 Arrays containing the neighbor information of the input grid for each grid point in LIS, for n. neighbor interpolation.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: defineNativeNLDAS2      !defines the native resolution of
                                 !the input data
```

PUBLIC TYPES:

```
public :: nldas2_struc
```

31.1.1 defineNativeNLDAS2 (Source File: nldas2domain_module.F90)

REVISION HISTORY:

```
02Feb2004: Sujay Kumar; Initial Specification
24 Aug 2007: Chuck Alonge; Modified for use with NLDAS2 data
```

INTERFACE:

```
subroutine defineNativeNLDAS2()
```

USES:

```
use lisdrv_module, only: lis
use listime_mgr, only : date2time
```

DESCRIPTION:

Defines the native resolution of the input forcing for NLDAS2 data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readnldas2crd (31.1.2)
reads the runtime options specified for NLDAS2 data

date2time (2.4.21)
converts date to the real time format

bilinear_interp_input (4.0.1)
computes the neighbor, weights for bilinear interpolation

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

read_nldas2_elev (31.1.3)
reads the native elevation of the NLDAS2 data to be used for topographic adjustments to the forcing

---

31.1.2 readnldas2crd (Source File: readnldas2crd.F90)

REVISION HISTORY:

```
02Feb2004; Sujay Kumar, Initial Code
24 Aug 2007: Chuck Alonge; Modified for use with NLDAS2 data
```

INTERFACE:

```
subroutine readnldas2crd()
```

USES:

```
use lisdrv_module, only : lis
use nldas2domain_module, only : nldas2_struc
use spmdMod, only : masterproc
use lis_logmod, only :logunit
use lisdrv_module, only : config_lis
use ESMF_Mod
```

DESCRIPTION:

This routine reads the options specific to NLDAS2 forcing from the LIS configuration file.

31.1.3 read_nldas2_elev (Source File: read_nldas2_elev.F90)

REVISION HISTORY:

```
17Dec2004; Sujay Kumar; Initial Specificaton
24 Aug 2007: Chuck Alonge; Modified for use with NLDAS2 data
```

INTERFACE:

```
subroutine read_nldas2_elev(n)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use baseforcing_module, only : lisforc
use nldas2domain_module, only : nldas2_struc
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates NLDAS2 model elevation to the LIS grid. The data will be used to perform any topographical adjustments to the forcing.
The arguments are:

n index of the nest

The routines invoked are:

ij_to_latlon (4.1.9)
computes the lat lon values in LIS grid projection

31.1.4 getnldas2 (Source File: getnldas2.F90)

REVISION HISTORY:

```
02Feb 2004 : Sujay Kumar ; Initial Version in LIS
22Aug 2007 : Chuck Alonge ; Updated for NLDAS2 Forcing
```

INTERFACE:

```
subroutine getnldas2(n)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use baseforcing_module, only : lisforc
use nldas2domain_module, only : nldas2_struc
use listime_mgr, only : tick
use lis_logmod, only :logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates hourly, NLDAS2 forcing (NARR based). At the beginning of a simulation, the code reads the most recent past data (nearest hourly interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. The strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day . The arguments are:

n index of the nest

The routines invoked are:

tick (2.4.23)
determines the NLDAS2 data times

nldas2filea (31.1.5)
Puts together appropriate timestamped filename

retnldas2 (31.1.7)
Reads and Interpolates NLDAS2 data to LIS grid

31.1.5 nldas2filea (Source File: getnldas2.F90)

REVISION HISTORY:

```
1  Oct 1999: Jared Entin; Initial code
15 Oct 1999: Paul Houser; Significant F90 Revision
04 Sep 2001: Brian Cosgrove; Use of NASA data enabled, updated
             reading of data directory structure to read new format
24 Aug 2007: Chuck Alonge; Modified for use with NLDAS2 data
```

INTERFACE:

```
subroutine nldas2filea(name,nldas2dir,yr,mo,da,hr)

  implicit none
```

ARGUMENTS:

```
  character*80, intent(out)::  name
  character*40, intent(in) ::  nldas2dir
  integer, intent(in)      :: yr,mo,da,hr
```

DESCRIPTION:

This subroutine puts together NLDAS2 file name for 1 hour file intervals
The arguments are:

nldas2dir Name of the NLDAS2 directory

yr year

mo month

da day of month

hr hour of day

name name of the timestamped NLDAS2 file

31.1.6 nldas2fileb (Source File: getnldas2.F90)

REVISION HISTORY:

```
1  Oct 1999: Jared Entin; Initial code
15 Oct 1999: Paul Houser; Significant F90 Revision
04 Sep 2001: Brian Cosgrove; Use of NASA data enabled, updated
             reading of data directory structure to read new format
24 Aug 2007: Chuck Alonge; Modified for use with NLDAS2 data
```

INTERFACE:

```
subroutine nldas2fileb(name,nldas2dir,yr,mo,da,hr)

  implicit none
```

ARGUMENTS:

```
  character*80, intent(out)::  name
  character*40, intent(in) ::  nldas2dir
  integer, intent(in)      :: yr,mo,da,hr
```

DESCRIPTION:

This subroutine puts together NLDAS2 file name for 3 hour file intervals
The arguments are:

nldas2dir Name of the NLDAS2 directory

yr year

mo month

da day of month

hr hour of day

name name of the timestamped NLDAS2 file

31.1.7 retnldas2 (Source File: retnldas2.F90)

REVISION HISTORY:

```
11 Apr 2000: Brian Cosgrove; changed code to use Forcing Mask (With inland
             water filled in).  Deleteted unused variables.
27 Apr 2000: Brian Cosgrove; changed code to use the original
             mask again since that is the
             mask which  NCEP has already applied to the forcing data
             by the time NASA gets it......not possible to use the
             expanded NASA forcing mask
1  May 2000: Brian Cosgrove; changed code so that if parameter 11 (sw)
             is not found in hourly ncep data, it will just use
             edas-based shortwave from the hourly ncep files
20 Jun 2000: Brian Cosgrove; changed code so that it uses  LDAS%UDEF and
              not a hard-wired undefined value of -999.9 and -999.0
18 Aug 2000: Brian Cosgrove; changed code so that FMASK and not MASK
             is used when ungribbing.  NCEP data already has a mask applied
             to it and so may not be able to supply forcing data to
             all LDAS land forcing points.  In areas where LDAS
             forcing mask states that land exists, but where NCEP forcing
             data is non-existant, assign undefined value to forcing data.
22 Aug 2000: Brian Cosgrove; Altered code for US/Mexico/Canada Mask
05 Sep 2001: Brian Cosgrove; Removed dirnom and infile variables, changed
             call to ungribncep to match removal.  Added code to make use
             of precip weighting mask
02 Feb 2004: Sujay Kumar; Initial Specification in LIS
24 Aug 2007: Chuck Alonge; Modified for use with NLDAS2 data
```

INTERFACE:

```
subroutine retnldas2(order,n, name,ferror)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use nldas2domain_module, only : nldas2_struc
use baseforcing_module, only : lisforc
use lis_logmod, only :logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: order
integer, intent(in)      :: n
character*80, intent(in) :: name
integer, intent(out)     :: ferror
```

DESCRIPTION:

For the given time, reads parameters from NLDAS2 data, transforms into 9 LIS forcing parameters and interpolates to the LIS domain.
The arguments are:

order flag indicating which data to be read (order=1, read the previous hourly instance, order=2, read the next hourly instance)

n index of the nest

name name of the hourly NLDAS2 forecast file

ferror flag to indicate success of the call (=0 indicates success)

The routines invoked are:

interp_nldas2 (31.1.8)
spatially interpolates a NLDAS2 variable

31.1.8 interp_nldas2 (Source File: retnldas2.F90)

INTERFACE:

```
subroutine interp_nldas2(n, kpds, nldas2,f,lb,lis_gds,nc,nr, &
     varfield)
```

USES:

```
use lisdrv_module, only : lis
use nldas2domain_module, only :nldas2_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in)   :: n
integer, intent(in)   :: kpds(200)
integer, intent(in)   :: nldas2
real, intent(out)     :: f(nldas2)
logical*1, intent(in) :: lb(nldas2)
real, intent(in)      :: lis_gds(50)
integer, intent(in)   :: nc
integer, intent(in)   :: nr
real, intent(inout)   :: varfield(nc,nr)
```

DESCRIPTION:

This subroutine interpolates a given NLDAS2 field to the LIS grid. The arguments are:

n index of the nest

kpds grib decoding array

ngdas number of elements in the input grid

f input data array to be interpolated

lb input bitmap

lis_gds array description of the LIS grid

nc number of columns (in the east-west dimension) in the LIS grid

nr number of rows (in the north-south dimension) in the LIS grid

varfield output interpolated field

The routines invoked are:

bilinear_interp (4.0.2)
spatially interpolate the forcing data using bilinear interpolation

conserv_interp (4.0.4)
spatially interpolate the forcing data using conservative interpolation

neighbor_interp (4.0.6)
spatially interpolate the forcing data using nearest neighbor interpolation

---

31.1.9 time_interp_nldas2 (Source File: time_interp_nldas2.F90)

REVISION HISTORY:

```
02Feb2004: Sujay Kumar; Initial Specification
24 Aug 2007: Chuck Alonge; Modified for use with NLDAS2 data
```

INTERFACE:

```
subroutine time_interp_nldas2(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis,lisdom
use constantsMod, only  : CONST_SOLAR
use baseforcing_module, only : lisforc, FORC_State
use nldas2domain_module, only : nldas2_struc
use listime_mgr, only : tick, time2date
use spmdMod, only : iam
use lis_logmod, only :logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep. Downward shortwave radiation is interpolated using a zenith-angled based approach. Precipitation and longwave radiation are not temporally interpolated, and the previous 3 hourly value is used. All other variables are linearly interpolated between the 3 hourly blocks.
The routines invoked are:

time2date (2.4.22)
converts the time to a date format

tick (2.4.23)
advances or retracts time by the specified amount

zterp (2.29.6)
zenith-angle based interpolation

31.2 Fortran: Module Interface nldas2forcing_finalize (Source File: nldas2forcing_finalize.F9

REVISION HISTORY:

```
25Oct2005; Sujay Kumar, Initial Code
24 Aug 2007: Chuck Alonge; Modified for use with NLDAS2 data
```

INTERFACE:

```
subroutine nldas2forcing_finalize
```

USES:

```
use lisdrv_module, only : lis
use nldas2domain_module
```

DESCRIPTION:

Routine to cleanup nldas2 forcing related memory allocations.

32 Princeton

This is the 50 year forcing data [**?**] produced at Princeton University. The data is global at 1 degree spatial resolution, at 3 hourly intervals.

32.1 Fortran: Module Interface princetondomain_module (Source File: princetondomain_module.F90)

This module contains variables and data structures that are used for the implementation of the 50-yr Princeton data (Sheffield et al. 2006). The data is global 1 degree dataset in latlon projection, and at 3 hourly intervals. The derived data type `princeton_struc` includes the variables that specify the runtime options, and the weights and neighbor information to be used for spatial interpolation. They are described below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

fmodeltime1 The nearest, previous 6 hour instance of the incoming data (as a real time).

fmodeltime2 The nearest, next 6 hour instance of the incoming data (as a real time).

princetondir Directory containing the input data

elevfile File with the elevation definition for the input data.

mi Number of points in the input grid

rlat1 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

rlon1 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

n11,n121,n211,n221 Arrays containing the neighbor information of the input grid for each grid point in LIS, for bilinear interpolation.

w111,w121,w211,w221 Arrays containing the weights of the input grid for each grid point in LIS, for bilinear interpolation.

rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

n12,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

Sheffield, J., G. Goteti, and E. F. Wood, 2006: Development of a 50-yr high-resolution global dataset of meteorological forcings for land surface modeling, J. Climate, 19 (13), 3088-3111

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
---------------------------------------------------------------------------
public :: defineNativePRINCETON      !defines the native resolution of
                                !the input data
---------------------------------------------------------------------------
```

PUBLIC TYPES:

```
---------------------------------------------------------------------------
public :: princeton_struc
```

32.1.1 defineNativePRINCETON (Source File: princetondomain_module.F90)

REVISION HISTORY:

```
26Jan2007: Hiroko Kato; Initial Specification
```

INTERFACE:

```
subroutine defineNativePRINCETON()
```

USES:

```
use lisdrv_module, only : lis

implicit none
```

DESCRIPTION:

Defines the native resolution of the input forcing for PRINCETON data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readprincetoncrd (32.1.2)
reads the runtime options specified for PRINCETON data

bilinear_interp_input (4.0.1)
computes the neighbor, weights for bilinear interpolation

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

read_princeton_elev (32.1.7)
reads the native elevation of the princeton data to be used for topographic adjustments to the forcing

---

32.1.2 readprincetoncrd (Source File: readprincetoncrd.F90)

REVISION HISTORY:

```
26 Jan 2007; Hiroko Kato, Initial Code adopted from readprincetoncrd.F90
25 Jun 2007; Hiroko Kato, upgraded to LISv5.0
```

INTERFACE:

```
subroutine readprincetoncrd()
```

USES:

```
use lisdrv_module, only : lis, config_lis
use lis_logmod, only : logunit
use ESMF_Mod
use princetondomain_module, only : princeton_struc
```

DESCRIPTION:

This routine reads the options specific to PRINCETON forcing from the LIS configuration file.

32.1.3 getprinceton (Source File: getprinceton.F90)

REVISION HISTORY:

```
26 Jan 2007: Hiroko Kato; Initial Specification adopted from LIS
```

INTERFACE:

```
subroutine getprinceton(n)
```

USES:

```
use lisdrv_module, only : lis
use baseforcing_module, only : lisforc
use listime_mgr
use princetondomain_module, only : princeton_struc
use lis_logmod, only : logunit

implicit none

integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates 3-hrly, 1 degree Princeton forcing. At the beginning of a simulation, the code reads the most recent past data (nearest 3 hour interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. The strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day . The arguments are:

n index of the nest

The routines invoked are:

tick (2.4.23)
call to advance or retract time

retprinceton (32.1.4)
call to read the Princeton data and perform spatial interpolation

read_princeton_elev (32.1.7)
reads the native elevation of the Princeton data to be used for topographic adjustments to the forcing

---

32.1.4 retprinceton (Source File: retprinceton.F90)

REVISION HISTORY:

```
26 Jan 2007: Hiroko Kato; Initial Specification adopted from LIS/retberg.F90
```

INTERFACE:

```
subroutine retprinceton( order, n, yr, mon, da, hr, ferror )
```

USES:

```
  use lisdrv_module, only : lis,lisdom
  use spmdMod, only : iam,masterproc
  use baseforcing_module, only : lisforc
  use listime_mgr
  use princetondomain_module, only : princeton_struc
  use lis_logmod, only : logunit


  implicit none
#if (defined USE_NETCDF)
  include 'netcdf.inc'
#endif
```

ARGUMENTS:

```
  integer, intent(in)    :: n  ! nest
  integer, intent(in)    :: order  ! lower(1) or upper(2) time interval bdry
  integer, intent(in)    :: yr,mon,da,hr      ! data and hour (multiple of 3)
  integer, intent(inout) :: ferror           ! set to zero if there's an error
```

DESCRIPTION:

For the given time, reads the parameters from 1 degree PRINCETON data, transforms into 9 LIS forcing parameters and interpolates to the LIS domain.

PRINCETON variables used to force LIS are: mean values starting at timestep, available every 3 hours

NOTE-1: be aware that PRINCETON has only total precipitation. NOTE 2: only one wind component, it is magnitude

PRINCETON FORCING VARIABLES:
1. TAS Air Temperature [K]
2. SHUM Specific humidity [kg kg-1]
3. DSWRF Downward shortwave radiation [W m-2]
4. DLWRF Downward longwave radiation [W m-2]
5. WIND Wind speed [m s-1]
6. PRES Surface pressure [Pa]
7. PRCP Precipitation [kg m-2 s-1]

order flag indicating which data to be read (order=1, read the previous 3 hourly instance, order=2, read the next 3 hourly instance)

n index of the nest

yr current year

mon current month

da current day of the year

hr current hour of day

ferror flag to indicate success of the call (=0 indicates success)

The routines invoked are:

princetongrid_2_lisgrid (32.1.5)
transform the PRINCETON data to the LIS grid

bilinear_interp (4.0.2)
spatially interpolate the forcing data using bilinear interpolation

conserv_interp (4.0.4)
spatially interpolate the forcing data using conservative interpolation

---

32.1.5 princetongrid_2_lisgrid (Source File: retprinceton.F90)

REVISION HISTORY:

```
10 Apr 2002: Urszula Jambor;  Code adapted from
             ecmwfgrid_2_grid2catgrid, by R. Reichle
29 Jan 2007: Hiroko Kato; modified berggrid_2_lisgrid for Princeton data
```

INTERFACE:

```
subroutine princetongrid_2_lisgrid( nx, ny, grid_data )

  implicit none
```

ARGUMENTS:

```
integer, intent(in)                   :: nx, ny
real, intent(inout), dimension(nx,ny) :: grid_data
```

DESCRIPTION:

Changes grid data from PRINCETON data convention to LIS convention
PRINCETON: Sorth-to-Nouth around Greenwich Meridian Global grid. Data are written in NetCDF from "lower left to upper right" starting at 0.5-degree grid point center coordinates: 0.5E,89.5S and going to 0.5W,89.5N.
LIS: South-to-North around Date Line Full global grid. Starts at the southernmost latitude and date line, going east and then north.

32.1.6 time_interp_princeton (Source File: time_interp_princeton.F90)

REVISION HISTORY:

```
26 Jan 2007: Hiroko Kato; Initial version of code adapted from LIS
25 Jun 2007: Hiroko Kato; upgraded for LISv5.0
```

INTERFACE:

```
subroutine time_interp_princeton(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis,lisdom
use constantsMod, only  : CONST_SOLAR
use baseforcing_module, only : lisforc, FORC_State
use listime_mgr
use princetondomain_module, only : princeton_struc
use spmdMod
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep. Downward shortwave radiation is interpolated using a zenith-angled based approach. Precipitation and longwave radiation are not temporally interpolated, and the previous 3 hourly value is used. All other variables are linearly interpolated between the 3 hourly blocks. Only total precipitation is available (no convective).
The routines invoked are:

time2date (2.4.22)
converts the time to a date format

zterp (2.29.6)
zenith-angle based interpolation

32.1.7 read_princeton_elev (Source File: read_princeton_elev.F90)

REVISION HISTORY:

```
1 Feb 2007; Hiroko Kato; Initial Specificaton adopted from
                         read_ecmwf_elev.F90
```

INTERFACE:

```
subroutine read_princeton_elev(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use baseforcing_module, only :lisforc
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens and reads PRINCETON model elevation to the LIS grid. The data will be used to perform any topographical adjustments to the forcing. The elevation file needs to be preprocessed to fit the running resolution and domain.
The arguments are:

n index of the nest

The routines invoked are:

readforcingelev (2.35.10)
call the abstract method to get the elevation data in the LIS projection.

32.1.8 read_princeton_elev_latlon (Source File: read_princeton_elev_latlon.F90)

REVISION HISTORY:

```
6 Dec 2007: Sujay Kumar; Abstracted out of the original implementation
```

INTERFACE:

```
subroutine read_princeton_elev_latlon(n, elev)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only :  logunit
use princetondomain_module, only : princeton_struc
use lis_fileUnitsMod, only : getNextUnitNumber, releaseUnitNumber

implicit none
```

ARGUMENTS:

```
integer,   intent(in)    :: n
integer,   intent(inout) :: elev(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

Opens and reads PRINCETON model elevation to the latlon projection.
The arguments are:

n index of the nest

elev array of elevation data.

The routines invoked are:

getNextUnitNumber (2.26.1)
obtain the next available unit number

releaseUnitNumber (2.26.2)
release the unit number after use.

32.1.9 princetonforcing_finalize (Source File: princetonforcing_finalize.F90)

REVISION HISTORY:

```
26 Jan 2007; Hiroko Kato, Initial Code
```

INTERFACE:

```
subroutine princetonforcing_finalize
```

USES:

```
use lisdrv_module, only : lis
use princetondomain_module
```

DESCRIPTION:

Routine to cleanup allocated structures for princeton forcing.

33 Rhone-AGG

This is the forcing data used in the PILPS experiments for the Rhone-AGG domain.

33.1 Fortran: Module Interface rhonedomain_module (Source File: rhonedomain_module.F90)

Contains routines and variables that define the native domain for RHONE model forcing.

33.1.1 defineNativeRHONE (Source File: rhonedomain_module.F90)

REVISION HISTORY:

```
11 Dec 2003: Sujay Kumar, Initial Specification
```

INTERFACE:

```
subroutine defineNativeRHONE()
```

USES:

```
use lisdrv_module, only: lis

implicit none
```

DESCRIPTION:

Defines the native resolution of the input forcing for RHONE AGG data.
The routines invoked are:

readrhonecrd (33.1.2)
reads the runtime options specified for RhoneAGG data

33.1.2 readrhonecrd (Source File: readrhonecrd.F90)

REVISION HISTORY:

```
11 Dec 2003: Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine readrhonecrd()
```

USES:

```
use lisdrv_module, only : lis, config_lis
use lis_logmod, only    : logunit
use ESMF_Mod
use rhonedomain_module, only : rhone_struc
```

DESCRIPTION:

Routine to read RHONE specific parameters from the LIS configuration file.

33.1.3 getrhone (Source File: getrhone.F90)

REVISION HISTORY:

```
5 Nov 2003: Dave Mocko, Initial Specification
```

INTERFACE:

```
subroutine getrhone(n)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use baseforcing_module, only: lisforc
use listime_mgr
use rhonedomain_module, only : rhone_struc
use spmdMod
use lis_logmod, only :logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates RHONE forcing.
TIME1 = most recent past data TIME2 = nearest future data
The strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day.
The arguments are:

n index of the nest

The routines invoked are:

tick (2.4.23)
Determines RHONE data times

rhonefile (33.1.4)
Puts together appropriate file name for 3 hour intervals

readrhone (33.1.5)
Interpolates RHONE data to LIS grid

33.1.4 rhonefile (Source File: getrhone.F90)

This subroutine puts together RHONE file name

INTERFACE:

```
subroutine rhonefile(name,rhonedir,yr,mo,da,hr,ncold)

implicit none
```

INPUT PARAMETERS:

```
character*40, intent(in) :: rhonedir
integer, intent(in)      :: yr,mo,da,hr,ncold
```

OUTPUT PARAMETERS:

```
character*80, intent(out) :: name
```

33.1.5 readrhone (Source File: readrhone.F90)

REVISION HISTORY:

```
5 Nov 2003: Dave Mocko, Initial Specification
```

INTERFACE:

```
subroutine readrhone(n, order,name,tscount,ferror)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use listime_mgr
use baseforcing_module, only: lisforc
use rhonedomain_module, only : rhone_struc
use spmdMod
use lis_logmod, only : logunit
```

DESCRIPTION:

Reads in RHONE data and performs interpolation to the LIS domain.

33.2 Core Functions of readrhone

bilinear_interp Interpolates RHONE data to LIS grid using bilinear interpolation

RHONE FORCING VARIABLES (unless noted, fields are 3-hr upstream averaged):
1. T 2m Temperature interpolated to 2 metres $[K]$
2. q 2m Instantaneous specific humidity interpolated to 2 metres$[kg/kg]$
3. radswg Downward shortwave flux at the ground $[W/m^2]$
4. lwgdwn Downward longwave radiation at the ground $[W/m^2]$
5. u 10m Instantaneous zonal wind interpolated to 10 metres $[m/s]$
6. v 10m Instantaneous meridional wind interpolated to 10 metres$[m/s]$
7. ps Instantaneous Surface Pressure $[Pa]$
8. preacc Total precipitation $[mm/s]$
9. precon Convective precipatation $[mm/s]$
10. albedo Surface albedo (0-1)

33.2.1 check_error_rhone (Source File: readrhone.F90)

Error check; Program exits in case of error

INTERFACE:

```
subroutine check_error_rhone(ierr,msg,iam)
```

33.2.2 time_interp_rhone (Source File: time_interp_rhone.F90)

REVISION HISTORY:

```
5 Nov 2003: Dave Mocko, Initial Specification
```

INTERFACE:

```
subroutine time_interp_rhone(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only :lis, lisdom
use constantsMod, only  : CONST_SOLAR
use baseforcing_module, only: lisforc, FORC_State
use listime_mgr
use spmdMod
use rhonedomain_module, only : rhone_struc
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)     :: n
```

DESCRIPTION:

Opens, reads, and interpolates RHONE forcing.
TIME1 = most recent past data TIME2 = nearest future data
The strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day.

33.3 Core Functions of time_interp_rhone

zterp Performs zenith angle-based temporal interpolation

33.3.1 rhoneforcing_finalize (Source File: rhoneforcing_finalize.F90)

REVISION HISTORY:

```
25Oct2005; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine rhoneforcing_finalize
```

USES:

```
use lisdrv_module, only : lis
use rhonedomain_module
```

DESCRIPTION:

Routine to cleanup RHONE forcing related memory allocations.

34 Template

This is a dummy placeholder implementation of a baseforcing, to be used as a template for new implementations.

34.1 Fortran: Module Interface templatedomain_module (Source File: templatedomain_module.F90)

implicit none

PUBLIC MEMBER FUNCTIONS:

```
  ---------------------------------------------------------------------------
   public :: defineNativetemplate      !defines the native resolution of
                                    !the input data
  ---------------------------------------------------------------------------
```

PUBLIC TYPES:

```
  ---------------------------------------------------------------------------
   public :: template_struc
```

34.1.1 defineNativetemplate (Source File: templatedomain_module.F90)

INTERFACE:

```
   subroutine defineNativetemplate()
```

USES:

```
     implicit none

   end subroutine defineNativetemplate
 end module templatedomain_module


\markboth{Left}{Source File: gettemplate.F90,  Date: Sat Jan  5 17:35:26 EST 2008
}


  ----------------------------------------------------------------------
   NASA Goddard Space Flight Center Land Information System (LIS) V5.0
   Released August 2006
   See SOFTWARE DISTRIBUTION POLICY for software distribution policies
   The LIS source code and documentation are in the public domain,
   available without fee for educational, research, non-commercial and
   commercial purposes.  Users may distribute the binary or source
   code to third parties provided this statement appears on all copies and
   that no charge is made for such copies.
   NASA GSFC MAKES NO REPRESENTATIONS ABOUT THE SUITABILITY OF THE
   SOFTWARE FOR ANY PURPOSE.  IT IS PROVIDED AS IS WITHOUT EXPRESS OR
   IMPLIED WARRANTY.  NEITHER NASA GSFC NOR THE US GOVERNMENT SHALL BE
   LIABLE FOR ANY DAMAGES SUFFERED BY THE USER OF THIS SOFTWARE.
   See COPYRIGHT.TXT for copyright details.
  ----------------------------------------------------------------------
%//////////////////////////////////////////////////////////////

\mbox{}\hrulefill\
```

```
  \subsubsection{gettemplate (Source File: gettemplate.F90)}


    \label{gettemplate}

\bigskip{\sf INTERFACE:}
\begin{verbatim} subroutine gettemplate(n)
```

USES:

```
   implicit none
```

ARGUMENTS:

```
   integer, intent(in) :: n
```

DESCRIPTION:

. The arguments are:

n index of the nest

34.1.2 time_interp_template (Source File: time_interp_template.F90)

forcing data. INTERFACE:

```
 subroutine time_interp_template(n)
```

USES:

```
   implicit none
```

ARGUMENTS:

```
   integer, intent(in) :: n
    The arguments are:
    \begin{description}
    \item[n]
     index of the nest
    \end{description}

   return

 end subroutine time_interp_template
\markboth{Left}{Source File: templateforcing\_finalize.F90,  Date: Sat Jan  5 17:35:26 EST 2008
}


  ------------------------------------------------------------------------
   NASA Goddard Space Flight Center Land Information System (LIS) V5.0
   Released August 2006
   See SOFTWARE DISTRIBUTION POLICY for software distribution policies
   The LIS source code and documentation are in the public domain,
   available without fee for educational, research, non-commercial and
   commercial purposes.  Users may distribute the binary or source
```

```
   code to third parties provided this statement appears on all copies and
   that no charge is made for such copies.
   NASA GSFC MAKES NO REPRESENTATIONS ABOUT THE SUITABILITY OF THE
   SOFTWARE FOR ANY PURPOSE.  IT IS PROVIDED AS IS WITHOUT EXPRESS OR
   IMPLIED WARRANTY.  NEITHER NASA GSFC NOR THE US GOVERNMENT SHALL BE
   LIABLE FOR ANY DAMAGES SUFFERED BY THE USER OF THIS SOFTWARE.
   See COPYRIGHT.TXT for copyright details.
  ---------------------------------------------------------------------------
 #include "misc.h"
%//////////////////////////////////////////////////////////////

\mbox{}\hrulefill\

\subsection{Fortran:  Module Interface templateforcing\_finalize (Source File: templateforcing\_finaliz


   \label{templateforcing_finalie}


\bigskip{\sf INTERFACE:}
\begin{verbatim} subroutine templateforcing_finalize
```

USES:

Part XI
Supplemental forcing analyses in LIS

35 Overview

This section contains some examples of the supplemental forcing implementations in LIS. These implementations, by definition overwrite the corresponding base forcing analyses.

36 CMAP

This section describes the implementation of the precipitation forcing from the Climate Prediction Center's (CPC) Merged Analysis of Precipitation (CMAP)

36.1 Fortran: Module Interface cmapdomain_module (Source File: cmapdomain_module.F90)

This module contains variables and data structures that are used for the implementation of the precipitation data from the Climate Prediction Center (CPC)'s Merged Analysis of Precipitation (CMAP). CMAP merges gauge measurements and satellite estimates including GPI, OPI, SSM/I to produce a global 2.5 degree pentad precipitation analysis. The 6-hourly product obtained by disaggregating CMAP using the GDAS forcing is used in this implementation.
The implementation in LIS has the derived data type `cmap_struc` that includes the variables to specify the runtime options, and the weights and neighbor information for spatial interpolation
They are desribed below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

cmapdir Directory containing the input data

cmaptime The nearest, hourly instance of the incoming data (as a real time).

griduptime1 The time to switch the input resolution to T170

mi Number of points in the input grid

rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

n112,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: defineNativeCMAP      !defines the native resolution of
                                !the input data
```

PUBLIC TYPES:

```
public :: cmap_struc
```

36.1.1 defineNativeCMAP (Source File: cmapdomain_module.F90)

REVISION HISTORY:

```
11Dec2003: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine defineNativeCMAP()
```

USES:

```
use lisdrv_module, only: lis
use listime_mgr, only : date2time
use lis_logmod,  only : logunit
```

DESCRIPTION:

Defines the native resolution of the input forcing for CMAP data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readcmapcrd (36.1.2)
reads the runtime options specified for CMAP data

date2time (2.4.21)
converts date to the real time format

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

---

36.1.2 readcmapcrd (Source File: readcmapcrd.F90)

REVISION HISTORY:

```
11 Dec 2003; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine readcmapcrd()
```

USES:

```
use cmapdomain_module, only : cmap_struc
use lisdrv_module, only : config_lis,lis
use ESMF_Mod
use lis_logmod, only : logunit
```

DESCRIPTION:

This routine reads the options specific to CMAP forcing from the LIS configuration file.

36.1.3 getcmap (Source File: getcmap.F90)

REVISION HISTORY:

```
17 Jul 2001: Jon Gottschalck; Initial code
10 Oct 2001: Jon Gottschalck; Modified to adjust convective precip
              using a ratio of the model convective / total ratio
16 Feb 2007: Chuck Alonge; Changed file name creation to use internal
              files instead of external file (avoids failure in parallel runs)
```

INTERFACE:

```
subroutine getcmap(n)
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr
use lis_logmod, only : logunit
use cmapdomain_module, only : cmap_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates 6-hrly, CMAP forcing. At the beginning of a simulation, the code reads the most recent past data (nearest 6 hour interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. . The arguments are:

n index of the nest

The routines invoked are:

tick (2.4.23)
determines the CMAP data times

cmapfile (36.1.4)
Puts together appropriate file name for 6 hour intervals

glbprecip_cmap (36.1.6)
Interpolates CMAP data to LIS grid

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

---

36.1.4 cmapfile (Source File: getcmap.F90)

INTERFACE:

```
subroutine cmapfile( name, cmapdir, yr, mo, da, hr)

implicit none
```

ARGUMENTS:

```
character(len=80) :: name, cmapdir
character(len=100) :: temp
integer           :: yr, mo, da, hr
```

DESCRIPTION:

This subroutine puts together CMAP file name for 6 hour file intervals
The arguments are:

cmapdir Name of the CMAP directory

yr year

mo month

da day of month

hr hour of day

name name of the timestamped CMAP file

36.1.5 cmapfile_old (Source File: getcmap.F90)

This subroutine puts together CMAP file name for 6 hour file intervals

INTERFACE:

```
 subroutine cmapfile_old( name, cmapdir, yr, mo, da, hr)

!-----------------------------------------------------------------------
! Make variables for the time used to create the file
! We don't want these variables being passed out
!-----------------------------------------------------------------------
  uyr = yr
  umo = mo
  uda = da
  uhr = 6*(hr/6)  !hour needs to be a multiple of 6 hours
  umn = 0
  uss = 0
  ts1 = -24*60*60 !one day interval to roll back date.

  name=''

  write(UNIT=temp, fmt='(a40)') cmapdir
  read(UNIT=temp, fmt='(80a1)') (fbase(i), i=1,80)

  write(UNIT=temp, fmt='(a1,i4,a1)') '/', uyr, '/'
  read(UNIT=temp, fmt='(6a1)') fdir
  do i = 1, 6
     if ( fdir(i) == ' ' ) fdir(i) = '0'
  end do

  write(UNIT=temp, fmt='(a11)') '_drean_smth'
  read (UNIT=temp, fmt='(11a1)') (fsubs(i), i=1,11)
```

```
write(UNIT=temp, fmt='(i4,i2,i2,i2)') uyr, umo, uda, uhr
read(UNIT=temp, fmt='(10a1)') ftime
do i = 1, 10
   if ( ftime(i) == ' ' ) ftime(i) = '0'
end do

write(UNIT=temp, fmt='(a4)') '.grb'
read (UNIT=temp, fmt='(4a1)') (fsubs2(i), i=1,4)
c = 0
do i = 1, 80
   if ( (fbase(i) == ' ') .and. (c == 0) ) c = i-1
end do

write(UNIT=temp, fmt='(80a1)') (fbase(i), i=1,c), (fdir(i), i=1,6),  &
                     (ftime(i), i=1,10),(fsubs(i), i=1,11)

read(UNIT=temp, fmt='(a80)') name

return
```

36.1.6 glbprecip_cmap (Source File: glbprecip_cmap.F90)

REVISION HISTORY:

```
17 Jul 2001: Jon Gottschalck; Initial code
04 Feb 2002: Jon Gottschalck; Added necessary code to use global precip
             observations with domain 3 (2x2.5)
16 Feb 2007: Chuck Alonge; Changed baopen to baopenr for grib
```

INTERFACE:

```
subroutine glbprecip_cmap( n, fname, ferror_cmap, filehr)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_logmod, only : logunit
use cmapdomain_module, only : cmap_struc
use suppforcing_module, only : suppforc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
character(len=80)   :: fname
integer             :: ferror_cmap
integer             :: filehr
```

DESCRIPTION:

For the given time, reads parameters from CMAP data and interpolates to the LIS domain. The arguments are:

n index of the nest

fname name of the 6 hour CMAP file

ferror_cmap flag to indicate success of the call (=0 indicates success)

filehr current file hour

The routines invoked are:

interp_cmap (36.1.7)
spatially interpolates the CMAP data

36.1.7 interp_cmap (Source File: interp_cmap.F90)

INTERFACE:

```
subroutine interp_cmap(n, kpds,ncmap,f,lb,lis_gds,nc,nr, &
     varfield)
```

USES:

```
use cmapdomain_module, only : cmap_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
integer                :: nc
integer                :: nr
integer                :: ncmap
integer                :: kpds(200)
real                   :: lis_gds(50)
real                   :: f(ncmap)
logical*1              :: lb(ncmap)
real, dimension(nc,nr) :: varfield
```

DESCRIPTION:

This subroutine interpolates a given CMAP field to the LIS grid. The arguments are:

n index of the nest

kpds grib decoding array

ncmap number of elements in the input grid

f input data array to be interpolated

lb input bitmap

lis_gds array description of the LIS grid

nc number of columns (in the east-west dimension) in the LIS grid

nr number of rows (in the north-south dimension) in the LIS grid

varfield output interpolated field

The routines invoked are:

conserv_interp (4.0.4)
spatially interpolate the forcing data using conservative interpolation

---

36.1.8 time_interp_cmap (Source File: time_interp_cmap.F90)

REVISION HISTORY:

```
20 Jan 2006:   Yudong Tian: Initial Implementation
10 Oct 2006:   Sujay Kumar: Switched to using ESMF_State for storing
                            forcing data.
```

INTERFACE:

```
subroutine time_interp_cmap(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only: lis, lisdom
use cmapdomain_module, only : cmap_struc
use suppforcing_module, only : suppforc
use baseforcing_module, only : FORC_State

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep.
The arguments are:

n index of the nest

37 TRMM 3B42RT

This section describes the implementation of the precipitation product derived from the TRMM 3B42RT satellite precipitation analysis genereated by George Huffman at NASA GSFC.

37.1 Fortran: Module Interface TRMM3B42RTdomain_module (Source File: TRMM3B42RTdomain_module.F90)

This module contains variables and data structures that are used for the implementation of the precipitation product derived from the TRMM real-time multi-satellite precipitation analysis (3B42RT) from NASA GSFC (Huffman et al.2003)
Huffman, G.J.,R.F. Adler, E.F.Stocker, D.T.Bolvin, and E.J.Nelkin (2003): Analysis of TRMM 3-hourly multi-satellite precipitation estimates computed in real and post-real time. Combined preprints CD-ROM, 83rd AMS Annual Meeting, Paper P4.11 in 12th Conference on Sat. Meteor. and Oceanog. 9-13 Feb 2003, Long Beach CA. 6pp.
The implementation in LIS has the derived data type `TRMM3B42RT_struc` that includes the variables to specify the runtime options, and the weights and neighbor information for spatial interpolation
They are desribed below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

TRMM3B42RTdir Directory containing the input data

TRMM3B42RTtime The nearest, hourly instance of the incoming data (as a real time).

griduptime1 The time to switch the input resolution to T170

mi Number of points in the input grid

rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

n112,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
---------------------------------------------------------------------------
 public :: defineNativeTRMM3B42RT      !defines the native resolution of
                                  !the input data
---------------------------------------------------------------------------
```

PUBLIC TYPES:

```
---------------------------------------------------------------------------
 public :: TRMM3B42RT_struc
```

37.1.1 defineNativeTRMM3B42RT (Source File: TRMM3B42RTdomain_module.F90)

REVISION HISTORY:

```
11Dec2003: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine defineNativeTRMM3B42RT()
```

USES:

```
  use lisdrv_module, only: lis
  use listime_mgr, only : date2time
```

DESCRIPTION:

Defines the native resolution of the input forcing for TRMM 3B42RT data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readTRMM3B42RTcrd (37.1.2)
reads the runtime options specified for TRMM 3B42RT data

date2time (2.4.21)
converts date to the real time format

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

---

37.1.2 readTRMM3B42RTcrd (Source File: readTRMM3B42RTcrd.F90)

REVISION HISTORY:

```
11 Dec 2003; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine readTRMM3B42RTcrd()
```

USES:

```
use TRMM3B42RTdomain_module, only : TRMM3B42RT_struc
use lisdrv_module, only : lis,config_lis
use ESMF_Mod
use lis_logmod, only : logunit
```

DESCRIPTION:

This routine reads the options specific to TRMM 3B42RT forcing from the LIS configuration file.

37.1.3 getTRMM3B42RT (Source File: getTRMM3B42RT.F90)

REVISION HISTORY:

```
17 Jul 2001: Jon Gottschalck; Initial code
10 Oct 2001: Jon Gottschalck; Modified to adjust convective precip
              using a ratio of the model convective / total ratio
```

INTERFACE:

```
subroutine getTRMM3B42RT(n)
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr
use lis_logmod, only : logunit
use spmdMod,only : masterproc
use TRMM3B42RTdomain_module, only : TRMM3B42RT_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates 3-hrly, TRMM 3B42RT forcing. At the beginning of a simulation, the code reads the most recent past data (nearest 3 hour interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. . The arguments are:

n index of the nest

The routines invoked are:

tick (2.4.23)
determines the TRMM 3B42RT data times

TRMM3B42RTfile (37.1.4)
Puts together appropriate file name for 6 hour intervals

glbprecip_TRMM3B42RT (37.1.5)
Interpolates TRMM 3B42RT data to LIS grid

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

---

37.1.4 TRMM3B42RTfile (Source File: getTRMM3B42RT.F90)

INTERFACE:

```
subroutine TRMM3B42RTfile( name, TRMM3B42RTdir, yr, mo, da, hr)

  implicit none
```

ARGUMENTS:

```
  character(len=120) :: name, TRMM3B42RTdir
  integer            :: yr, mo, da, hr
```

DESCRIPTION:

This subroutine puts together TRMM 3B42RT file name for 3 hour file intervals
The arguments are:

TRMM3B42RTdir Name of the TRMM 3B42RT data directory

yr year

mo month

da day of month

hr hour of day

name name of the timestamped TRMM 3B42RT file

37.1.5 glbprecip_TRMM3B42RT (Source File: glbprecip_TRMM3B42RT.F90)

REVISION HISTORY:

```
17 Jul 2001: Jon Gottschalck; Initial code
04 Feb 2002: Jon Gottschalck; Added necessary code to use global precip
             observations with domain 3 (2x2.5)
```

INTERFACE:

```
subroutine glbprecip_TRMM3B42RT (n, name_TRMM3B42RT, ferror_TRMM3B42RT )
```

USES:

```
use lisdrv_module, only : lis, lisdom
use TRMM3B42RTdomain_module, only : TRMM3B42RT_struc
use spmdMod,only : masterproc
use lis_logmod, only : logunit
use suppforcing_module, only : suppforc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

For the given time, reads parameters from TRMM 3B42RT data and interpolates to the LIS domain. The arguments are:

n index of the nest

name_TRMM3B42RT name of the 3 hour TRMM 3B42RT forecast file

ferror_TRMM3B42RT flag to indicate success of the call (=0 indicates success)

The routines invoked are:

interp_TRMM3B42RT (37.1.6)
spatially interpolates the TRMM 3B42RT data

37.1.6 interp_TRMM3B42RT (Source File: interp_TRMM3B42RT.F90)

INTERFACE:

```
subroutine interp_TRMM3B42RT(n, nx, ny, finput, lis_gds, nc, nr, varfield)
```

USES:

```
use TRMM3B42RTdomain_module, only : TRMM3B42RT_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer             :: nx
integer             :: ny
integer             :: nc
integer             :: nr
real                :: lis_gds(50)
real, dimension(nx,ny) :: finput
```

DESCRIPTION:

This subroutine interpolates a given TRMM 3B42RT field to the LIS grid. The arguments are:

n index of the nest

nx number of columns (in the east-west dimension) in the TRMM 3B42RT grid

ny number of rows (in the north-south dimension) in the TRMM 3B42RT grid

finput input data array to be interpolated

lis_gds array description of the LIS grid

nc number of columns (in the east-west dimension) in the LIS grid

nr number of rows (in the north-south dimension) in the LIS grid

varfield output interpolated field

The routines invoked are:

conserv_interp (4.0.4)
spatially interpolate the forcing data using conservative interpolation

---

37.1.7 time_interp_TRMM3B42RT (Source File: time_interp_3B42RT.F90)

REVISION HISTORY:

```
20 Jan 2006:   Yudong Tian: Initial Implementation
10 Oct 2006:   Sujay Kumar: Switched to using ESMF_State for storing
                            forcing data.
```

INTERFACE:

```
subroutine time_interp_TRMM3B42RT(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only:lis,lisdom
use TRMM3B42RTdomain_module, only : TRMM3B42RT_struc
use suppforcing_module, only : suppforc
use baseforcing_module, only : FORC_State

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep.
The arguments are:

n index of the nest

38 TRMM 3B42V6

This section describes the implementation of the precipitation product derived from the TRMM 3B42V6 satellite precipitation analysis genereated by George Huffman at NASA GSFC.

38.0.8 defineNativeTRMM3B42V6 (Source File: TRMM3B42V6domain_module.F90)

REVISION HISTORY:

```
11Dec2003: Sujay Kumar; Initial Specification
25Aug2006: Yudong Tian; Implementation for TRMM 3B42 V6
```

INTERFACE:

```
subroutine defineNativeTRMM3B42V6()
```

USES:

```
use lisdrv_module, only: lis
use listime_mgr, only : date2time
```

DESCRIPTION:

Defines the native resolution of the input forcing for TRMM 3B42V6 data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readTRMM3B42V6crd (38.0.9)
reads the runtime options specified for TRMM 3B42V6 data

date2time (2.4.21)
converts date to the real time format

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

---

38.0.9 readTRMM3B42V6crd (Source File: readTRMM3B42V6crd.F90)

REVISION HISTORY:

```
11 Dec 2003; Sujay Kumar, Initial Code
25 Aug 2006; Yudong Tian, Modification for 3B42, LIS 4.2 release
```

INTERFACE:

```
subroutine readTRMM3B42V6crd()
```

USES:

```
use TRMM3B42V6domain_module, only : TRMM3B42V6_struc
use lisdrv_module, only : config_lis,lis
use ESMF_Mod
use lis_logmod, only : logunit
```

DESCRIPTION:

This routine reads the options specific to TRMM 3B42V6 forcing from the LIS configuration file.

38.0.10 getTRMM3B42V6 (Source File: getTRMM3B42V6.F90)

REVISION HISTORY:

```
25Aug2006: Yudong Tian; Modified for LIS 4.2 release
```

INTERFACE:

```
subroutine getTRMM3B42V6(n)
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr
use lis_logmod, only : logunit
use TRMM3B42V6domain_module, only : TRMM3B42V6_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates 3-hrly TRMM 3B42V6 forcing. At the beginning of a simulation, the code reads the most recent past data (nearest 6 hour interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. . The arguments are:

n index of the nest

The routines invoked are:

tick (2.4.23)
determines the CMAP data times

cmapfile (36.1.4)
Puts together appropriate file name for 6 hour intervals

glbprecip_cmap (36.1.6)
Interpolates CMAP data to LIS grid

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

```
!-----------------------------------------------------------------------
! Set parameter to measure 1.5 hour time offset when using 3B42V6
!-----------------------------------------------------------------------
  gap = 0.0001712328767098370
!-----------------------------------------------------------------------
! Determine required observed precip data times
! (current, accumulation end time)
! Model current time
!-----------------------------------------------------------------------
  yr1 = lis%yr  !current time
  mo1 = lis%mo
  da1 = lis%da
  hr1 = lis%hr
  mn1 = lis%mn
  ss1 = 0
  ts1 = 0
  call tick( ctime, doy1, gmt1, yr1, mo1, da1, hr1, mn1, ss1, ts1 )

!-----------------------------------------------------------------------
! 3B42V6 product end time
!-----------------------------------------------------------------------
  yr3 = lis%yr  !end accumulation time data
  mo3 = lis%mo
  da3 = lis%da
  hr3 = 3*(lis%hr/3)
  mn3 = 0
  ss3 = 0
  ts3 = 3*60*60
  call tick( ftime_TRMM3B42V6, doy3, gmt3, yr3, mo3, da3, hr3, mn3, ss3, ts3 )
  breaktime = ftime_TRMM3B42V6 - ctime
  datatime  = ftime_TRMM3B42V6
  fnametime = ftime_TRMM3B42V6
     if (breaktime .ge. gap) then
        call time2date( datatime, kdoy3, kgmt3, kyr3, &
             kmo3, kda3, khr3, kmn3 )
        call time2date( fnametime, mdoy3, mgmt3, myr3, &
             mmo3, mda3, mhr3, mmn3 )
        flag1 = 1
        if (khr3 == 24) khr3 = 0
        if (mhr3 == 24) mhr3 = 0
        if (kgmt3 .eq. 0.0 .and. flag2 .eq. 2) then
           kts3 = -25.5*60*60
   kss3 = 0
           call tick( datatime, kdoy3, kgmt3, kyr3, kmo3, &
                kda3, khr3, kmn3, kss3, kts3 )
           mts3 = -27*60*60
   mss3 = 0
           call tick( fnametime, mdoy3, mgmt3, myr3, mmo3, &
                mda3, mhr3, mmn3, mss3, mts3 )
        else
           kts3 = -1.5*60*60
   kss3 = 0
           call tick( datatime, kdoy3, kgmt3, kyr3, kmo3, &
                kda3, khr3, kmn3, kss3, kts3 )
           mts3 = -3*60*60
```

```
   mss3 = 0
           call tick( fnametime, mdoy3, mgmt3, myr3, mmo3, &
                mda3, mhr3, mmn3, mss3, mts3 )
        endif
        flag2 = 1
      else
        call time2date( datatime, kdoy3, kgmt3, kyr3, kmo3, &
             kda3, khr3, kmn3 )
        call time2date( fnametime, mdoy3, mgmt3, myr3, mmo3, &
             mda3, mhr3, mmn3 )
        if (get_nstep(lis, n).eq. 1) then
           if (kgmt3 .eq. 0) then
              mts3 = -24*60*60
      mss3 = 0
              call tick( fnametime, mdoy3, mgmt3, myr3, mmo3, &
                   mda3, mhr3, mmn3, mss3, mts3 )
              kts3 = -22.5*60*60
      kss3 = 0
              call tick( datatime, kdoy3, kgmt3, kyr3, kmo3, &
                   kda3, khr3, kmn3, kss3, kts3 )
           else
              mts3 = 0
      mss3 = 0
              call tick( fnametime, mdoy3, mgmt3, myr3, mmo3, &
                   mda3, mhr3, mmn3, mss3, mts3 )
              kts3 = 1.5*60*60
      kss3 = 0
              call tick( datatime, kdoy3, kgmt3, kyr3, kmo3, &
                   kda3, khr3, kmn3, kss3, kts3 )
           endif
        else
           flag1 = 2
           if (flag2 .eq. 1) then
              mts3 = 3*60*60
      mss3 = 0
              call tick( fnametime, mdoy3, mgmt3, myr3, mmo3, &
                   mda3, mhr3, mmn3, mss3, mts3 )
              kts3 = 3*60*60
      kss3 = 0
              call tick( datatime, kdoy3, kgmt3, kyr3, kmo3, &
                   kda3, khr3, kmn3, kss3, kts3 )
           endif
           flag2 = 2
        endif
     endif
!-----------------------------------------------------------------------
! Ensure that data is found during first time step
!-----------------------------------------------------------------------
endtime_TRMM3B42V6 = 0
  if ( get_nstep(lis, n).eq. 1 &
  .or.lis%rstflag(n) .eq. 1) then
     endtime_TRMM3B42V6 = 1
     lis%rstflag(n) = 0
  endif
```

```
!-----------------------------------------------------------------------
! Check for and get 3B42V6 observed Precipitation data
!-----------------------------------------------------------------------
      if ( ctime > TRMM3B42V6_struc(n)%TRMM3B42V6time ) endtime_TRMM3B42V6 = 1
      if ( endtime_TRMM3B42V6 == 1 ) then  !get new time2 data
         ferror_TRMM3B42V6 = 0
         write(logunit, *) 'myr3, mmo3, mda3, mhr3:', myr3, mmo3, mda3, mhr3
         call TRMM3B42V6file( name, n, myr3, mmo3, mda3, mhr3 )
         write(logunit, *)'Getting new TRMM 3B42V6 satellite precip data:', name
         call glbprecip_TRMM3B42V6(n, name, ferror_TRMM3B42V6, mhr3)
         TRMM3B42V6_struc(n)%TRMM3B42V6time = datatime
      endif
!endif
return
```

38.0.11 TRMM3B42V6file (Source File: getTRMM3B42V6.F90)

This subroutine puts together 3B42V6 file name for 3 hour file intervals There are two filename formats: 1. original: 3B42.980131.12.6.precipitation, 3B42.980130.3.6.precipitation which was produced after data are converted from hdf to bin format 2. renamed: TRMM3B42V6.2005110809, same data, just different file name The raw hdf data files are named like this: 3B42.060105.15.6.HDF It is possible that we may implement HDF file reading in LIS code to handle raw input. For now, we do the binary format. –Yudong 8/25/06

INTERFACE:

```
 subroutine TRMM3B42V6file( name, n, yr, mo, da, hr)

   use TRMM3B42V6domain_module, only : TRMM3B42V6_struc
```

38.0.12 glbprecip_TRMM3B42V6.F90 (Source File: glbprecip_TRMM3B42V6.F90)

REVISION HISTORY:

```
    17 Jul 2001: Jon Gottschalck; Initial code
    04 Feb 2002: Jon Gottschalck; Added necessary code to use global precip
                 observations with domain 3 (2x2.5)
    30 Jul 2002: Jon Gottschalck; Added code to use 3B42V6 and Persiann precip data
    25 Aug 2006: Yudong Tian; Modified for LIS 4.2 release
```

INTERFACE:

```
 subroutine glbprecip_TRMM3B42V6 (n, fname, ferror_TRMM3B42V6, filehr )
```

USES:

```
  use lisdrv_module, only : lis, lisdom
  use lis_logmod, only : logunit
  use TRMM3B42V6domain_module, only : TRMM3B42V6_struc
  use suppforcing_module,  only : suppforc

   implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
character(len=80)   :: fname
integer             :: ferror_TRMM3B42V6
integer             :: filehr
```

DESCRIPTION:

For the given time, reads parameters from TRMM 3B42V6 data and interpolates to the LIS domain. The arguments are:

n index of the nest

fname name of the 6 hour TRMM 3B42V6 file

ferror_TRMM3B42V6 flag to indicate success of the call (=0 indicates success)

filehr current file hour

The routines invoked are:

interp_TRMM3B42V6 (38.0.13)
spatially interpolates the TRMM 3B42V6 data

```
!-----------------------------------------------------------------------
! Fill necessary arrays to assure not using old 3B42V6 data
!-----------------------------------------------------------------------
 allocate (precip_regrid(lis%lnc(n),lis%lnr(n)))
 precip = -1.0
 suppforc(n)%suppdata2 = -1.0
!-----------------------------------------------------------------------
! Find 3B42V6 precip data, read it in and assign to forcing precip array.
! Must reverse grid in latitude dimension to be consistent with LDAS grid
!-----------------------------------------------------------------------
  open(unit=97,file=fname, status='old', &
   &          access='direct',recl=xd*yd*4, &
   &          form='unformatted',iostat=ios)

  if (ios .eq. 0) then
     read (97,rec=1) precip
     Do j=1, xd
        Do i=1, yd
        if (precip(j, i) .LT. 0.0 ) precip(j, i) = 0.0    ! reset to 0 for weird values
        End Do
      End Do
!-----------------------------------------------------------------------
! Interpolating to desired domain and resolution
! Global precip datasets not used currently to force NLDAS
!-----------------------------------------------------------------------
   !print*, "Writing un-interpolated 3B42V6 precipitation out "
   !open(71, file="TRMM3B42V6-ungrid.1gd4r", access="direct", &
   !    recl=xd*yd*4, form="unformatted")
   ! write(71, rec=1) precip
   !close(71)

   nTRMM3B42V6 = TRMM3B42V6_struc(n)%ncold*TRMM3B42V6_struc(n)%nrold
```

```
   allocate(lb(nTRMM3B42V6))
   lb = .true.
   call interp_TRMM3B42V6(n, kpds, nTRMM3B42V6, precip, lb, lis%gridDesc, &
          lis%lnc(n),lis%lnr(n),precip_regrid)
   deallocate (lb)

   !print*, "Writing interpolated 3B42V6 precipitation out "
   !open(73, file="TRMM3B42V6-regrid.1gd4r", access="direct", &
   !    recl=lis%d%lnr*lis%d%lnc*4, form="unformatted")
   ! write(73, rec=1) precip_regrid
   !close(73)
   !print*, "Writing interpolated 3B42V6 precipitation out finished"

     do j = 1,lis%lnr(n)
        do i = 1,lis%lnc(n)
           if (precip_regrid(i,j) .ne. -1.0) then
              index = lisdom(n)%gindex(i,j)
              if(index .ne. -1) then
                 suppforc(n)%suppdata2(1,index) = precip_regrid(i,j)   !here is mm/h
              endif
           endif
        enddo
     enddo

  ferror_TRMM3B42V6 = 1
  close(97)
     write(logunit,*) "Obtained 3B42 V6 precipitation data ", fname
  else
     write(logunit,*) "Missing 3B42 V6 precipitation data ", fname
     ferror_TRMM3B42V6 = 0
  endif

  deallocate (precip_regrid)

end subroutine glbprecip_TRMM3B42V6

\markboth{Left}{Source File: interp\_TRMM3B42V6.F90,  Date: Sat Jan  5 17:35:28 EST 2008
}

!-----------------------------------------------------------------------------
! NASA Goddard Space Flight Center Land Information System (LIS) V4.3
! Released November 2006
!
! See SOFTWARE DISTRIBUTION POLICY for software distribution policies
!
! The LIS source code and documentation are in the public domain,
! available without fee for educational, research, non-commercial and
! commercial purposes.  Users may distribute the binary or source
! code to third parties provided this statement appears on all copies and
! that no charge is made for such copies.
!
! NASA GSFC MAKES NO REPRESENTATIONS ABOUT THE SUITABILITY OF THE
! SOFTWARE FOR ANY PURPOSE.  IT IS PROVIDED AS IS WITHOUT EXPRESS OR
! IMPLIED WARRANTY.  NEITHER NASA GSFC NOR THE US GOVERNMENT SHALL BE
! LIABLE FOR ANY DAMAGES SUFFERED BY THE USER OF THIS SOFTWARE.
```

```
!
! See COPYRIGHT.TXT for copyright details.
!
!-----------------------------------------------------------------------
```

38.0.13 interp_TRMM3B42V6.F90 (Source File: interp_TRMM3B42V6.F90)

INTERFACE:

```
subroutine interp_TRMM3B42V6(n, kpds, nTRMM3B42V6, f, lb, lis_gds, nc, nr, varfield)
```

USES:

```
use TRMM3B42V6domain_module, only : TRMM3B42V6_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
integer                :: nc
integer                :: nr
integer                :: nTRMM3B42V6
integer                :: kpds(200)
real                   :: lis_gds(50)
real                   :: f(nTRMM3B42V6)
logical*1              :: lb(nTRMM3B42V6)
real, dimension(nc,nr) :: varfield
```

DESCRIPTION:

This subroutine interpolates a given 3B42 V6 field to the LIS grid. The arguments are:

n index of the nest

kpds grib decoding array

nTRMM3B42V6 number of elements in the input grid

f input data array to be interpolated

lb input bitmap

lis_gds array description of the LIS grid

nc number of columns (in the east-west dimension) in the LIS grid

nr number of rows (in the north-south dimension) in the LIS grid

varfield output interpolated field

The routines invoked are:

conserv_interp (4.0.4)
spatially interpolate the forcing data using conservative interpolation

39 CMORPH

This section describes the implementation of the precipitation product from CPC's MORPHing technique, known as CMORPH.

39.1 Fortran: Module Interface cmordomain_module (Source File: cmordomain_module.F90)

REVISION HISTORY:

```
05Jan2006: Yudong Tian; start from Jon G.'s code for older versions of LIS.
```

DESCRIPTION:

This module contains variables and data structures that are used for the implementation of the precipitation data from the Climate Prediction Center (CPC)'s MORPHing technique (CMORPH). CMORPH products are produced as global fields from 60 N-60S at 8km resolution at 30 minute intervals.
The implementation in LIS has the derived data type `cmor_struc` that includes the variables to specify the runtime options, and the weights and neighbor information for spatial interpolation
They are desribed below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

cmordir Directory containing the input data

cmortime The nearest, hourly instance of the incoming data (as a real time).

griduptime1 The time to switch the input resolution to T170

mi Number of points in the input grid

rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

n112,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: defineNativeCMORPH      !defines the native resolution of
                                  !the input data
```

PUBLIC TYPES:

```
public :: cmor_struc
```

39.1.1 defineNativeCMORPH (Source File: cmordomain_module.F90)

REVISION HISTORY:

```
11Dec2003: Sujay Kumar; Initial Specification
06Jan2006: Yudong Tian; modification for LISv4.2
```

INTERFACE:

```
subroutine defineNativeCMORPH
```

USES:

```
use lisdrv_module, only: lis
use listime_mgr, only : date2time

implicit none

real :: gridDesci(50)
integer :: updoy, yr1,mo1,da1,hr1,mn1,ss1
real :: upgmt
integer :: n
```

DESCRIPTION:

Defines the native resolution of the input forcing for CMORPH data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readcmorcrd (39.1.2)
reads the runtime options specified for CMORPH data

date2time (2.4.21)
converts date to the real time format

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

---

39.1.2 readcmorcrd (Source File: readcmorcrd.F90)

REVISION HISTORY:

```
11 Dec 2003; Sujay Kumar, Initial Code
29 Dec 2003; Luis Gustavo, adapted for CMORPH
06 Jan 2005; Yudong Tian, modified for LISv4.2
```

INTERFACE:

```
subroutine readcmorcrd()
```

USES:

```
use cmordomain_module, only : cmor_struc
use lisdrv_module, only : config_lis, lis
use ESMF_Mod
use lis_logmod, only : logunit
```

DESCRIPTION:

This routine reads the options specific to CMORPH forcing from the LIS configuration file.

39.1.3 getcmor (Source File: getcmor.F90)

REVISION HISTORY:

```
17 Jul 2001: Jon Gottschalck; Initial code
10 Oct 2001: Jon Gottschalck; Modified to adjust convective precip
              using a ratio of the model convective / total ratio
29 Dec 2003: Luis Goncalves; Added CMORPH global observed precip data sources
06 Jan 2005: Yudong Tian; Modified for LISv4.2
```

INTERFACE:

```
subroutine getcmor(n)
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr
use spmdMod,only : masterproc
use cmordomain_module, only :cmor_struc
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates 30 minute CMORPH forcing. At the beginning of a simulation, the code reads the most recent past data (nearest 30min interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. . The arguments are:

n index of the nest

The routines invoked are:

tick (2.4.23)
determines the CMORPH data times

cmorfile (39.1.4)
Puts together appropriate file name for 6 hour intervals

glbprecip_cmor (39.1.5)
Interpolates CMORPH data to LIS grid

conserv_interp_input (4.0.3)
computes the neighbor, weights for conservative interpolation

---

39.1.4 cmorfile (Source File: getcmor.F90)

INTERFACE:

```
subroutine cmorfile( name, cmordir, yr, mo, da, hr)

  implicit none
```

ARGUMENTS:

```
  character(len=100) :: name, cmordir
  integer            :: yr, mo, da, hr
```

DESCRIPTION:

This subroutine puts together CMORPH file name for 30min file intervals
The arguments are:

cmordir Name of the CMORPH directory

yr year

mo month

da day of month

hr hour of day

name name of the timestamped CMORPH file

39.1.5 glbprecip_cmor (Source File: glbprecip_cmor.F90)

REVISION HISTORY:

```
17 Jul 2001: Jon Gottschalck; Initial code
04 Feb 2002: Jon Gottschalck; Added necessary code to use global precip
             observations with domain 3 (2x2.5)
29 Dec 2003: Luis Goncalves; Added code to use CMORPH precip data
06 Jan 2006: Yudong Tian; modified for LISv4.2
```

INTERFACE:

```
subroutine glbprecip_cmor (n, name_cmor, ferror_cmor, iflg )
```

USES:

```
  use lisdrv_module, only : lis, lisdom
  use cmordomain_module, only : cmor_struc
  use spmdMod,only : masterproc
  use lis_logmod, only : logunit
  use suppforcing_module, only : suppforc

  implicit none
```

ARGUMENTS:

```
  integer, intent(in) :: n
```

DESCRIPTION:

For the given time, reads parameters from CMORPH data and interpolates to the LIS domain. The arguments are:

n index of the nest

name_cmor name of the CMORPH file

ferror_cmor flag to indicate success of the call (=0 indicates success)

iflg flag indicating which 1/2 hour to read

The routines invoked are:

interp_cmor (39.1.6)
spatially interpolates the CMORPH data

39.1.6 interp_cmor (Source File: interp_cmor.F90)

INTERFACE:

```
subroutine interp_cmor(n, nx, ny, finput, lis_gds, nc, nr, varfield)
```

USES:

```
use cmordomain_module, only : cmor_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer             :: nx
integer             :: ny
integer             :: nc
integer             :: nr
real, dimension(nx,ny) :: finput
real, dimension(nc,nr) :: varfield
```

DESCRIPTION:

This subroutine interpolates a given CMORPH field to the LIS grid. The arguments are:

n index of the nest

nx number of columns (in the east-west dimension) in the CMORPH grid

ny number of rows (in the north-south dimension) in the CMORPH grid

finput input data array to be interpolated

lis_gds array description of the LIS grid

nc number of columns (in the east-west dimension) in the LIS grid

nr number of rows (in the north-south dimension) in the LIS grid

varfield output interpolated field

The routines invoked are:

conserv_interp (4.0.4)
spatially interpolate the forcing data using conservative interpolation

---

39.1.7 time_interp_cmor (Source File: time_interp_cmor.F90)

REVISION HISTORY:

```
20 Jan 2006:   Yudong Tian: Initial Implementation
10 Oct 2006:   Sujay Kumar: Switched to using ESMF_State for storing
                            forcing data.
```

INTERFACE:

```
subroutine time_interp_cmor(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only:lis,lisdom
use cmordomain_module, only : cmor_struc
use suppforcing_module, only : suppforc
use baseforcing_module, only : FORC_State
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep.
The arguments are:

n index of the nest

40 STAGEII

This section describes the implementation of the precipitation product from NCEP, known as STAGEII

40.1 Fortran: Module Interface stg2domain_module (Source File: stg2domain_module.F90)

This module contains variables and data structures that are used for the implementation of the precipitation data from the National Center for Environmental Prediction (NCEP) Stage II (STAGE2) Doppler Radar+gage product. The Stage II is a national level product, on an hourly interval, and supplements mainly the which base forcing precipitation (e.g., NLDAS) is being used.
The implementation in LIS has the derived data type `stg2_struc` that includes the variables to specify the runtime options, and the weights and neighbor information for spatial interpolation
They are desribed below:

ncol Number of columns (along the east west dimension) for the input data

nrow Number of rows (along the north south dimension) for the input data

stg2dir Directory containing the input data

stg2time The nearest, hourly instance of the incoming data (as a real time).

mi Number of points in the input grid

rlat1 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

rlon1 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

n111,n121,n211,n221 Arrays containing the neighbor information of the input grid for each grid point in LIS, for bilinear interpolation.

w111,w121,w211,w221 Arrays containing the weights of the input grid for each grid point in LIS, for bilinear interpolation.

n112,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: defineNativeSTG2      !defines the native resolution of
                                !the input data
```

PUBLIC TYPES:

```
public :: stg2_struc
```

40.1.1 defineNativeSTG2 (Source File: stg2domain_module.F90)

REVISION HISTORY:

```
11Dec2003: Sujay Kumar; Initial Specification
25May2006: Kristi Arsenault; Data and code implementation
```

INTERFACE:

```
subroutine defineNativeSTG2()
```

USES:

```
use lisdrv_module, only: lis
use listime_mgr, only : date2time
```

DESCRIPTION:

Defines the native resolution of the input forcing for STAGE2 data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readstg2crd (40.1.2)
reads the runtime options specified for STAGE2 data

bilinear_interp_input (4.0.1)
computes the neighbor, weights for bilinear interpolation

date2time (2.4.21)
converts date to the real time format - time of grid change

40.1.2 readstg2crd (Source File: readstg2crd.F90)

REVISION HISTORY:

```
11 Dec 2003; Sujay Kumar, Initial Code
05 Jun 2006; Kristi Arsenault, Code and data implementation
```

INTERFACE:

```
subroutine readstg2crd()
```

USES:

```
use ESMF_Mod
use stg2domain_module, only : stg2_struc
use lisdrv_module, only : config_lis, lis
use lis_logmod, only : logunit
```

DESCRIPTION:

This routine reads the options specific to STAGE2 forcing from the LIS configuration file.

40.1.3 getstg2 (Source File: getstg2.F90)

REVISION HISTORY:

```
25 May 2006: Kristi Arsenault;  Data and Code implementation
```

INTERFACE:

```
subroutine getstg2(n)
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr
use lis_logmod, only : logunit
use stg2domain_module, only : stg2_struc
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates hourly STAGE2 forcing. At the beginning of a simulation, the code reads the most recent past data (nearest the hour interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep.
The arguments are:

n index of the nest

lis_time Current LIS Time

stg2_file_time End boundary time of Stage II file

file_name Stage II filename - passed back to getstg

The routines invoked are:

tick (2.4.23)
determines the STAGE2 data times

bilinear_interp_input (4.0.1)
Computes the neighbor, weights for bilinear interpolation

conserv_interp_input (4.0.3)
Computes the neighbor, weights for conservative interpolation

stg2file (40.2.4)
Puts together appropriate file name for 1-hour intervals

precip_stg2 (40.2.3)
Interpolates STAGE2 data to LIS grid

40.2 Fortran: Module Interface stg2_finalize (Source File: stg2_finalize.F90)

REVISION HISTORY:

```
25Oct2005; Sujay Kumar, Initial Code
17Jul2006; K. Arsenault, Added Stage II
```

INTERFACE:

```
subroutine stg2_finalize
```

USES:

```
use lisdrv_module, only : lis
use stg2domain_module
```

DESCRIPTION:

Routine to cleanup Stage II forcing related memory allocations.

40.2.1 interp_stg2 (Source File: interp_stg2.F90)

INTERFACE:

```
subroutine interp_stg2 (n, ksec1, nstg2, ppt_field, lb, lis_gds, &
                        nc, nr, varfield )
```

USES:

```
use stg2domain_module, only : stg2_struc
use lisdrv_module, only: lis

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
integer                :: nc      ! Number of columns (in the E-W dimension) in the LIS grid
integer                :: nr      ! Number of rows (in the N-S dimension) in the LIS grid
integer                :: nstg2   ! Number of points in original STAGE II grid
integer                :: ksec1(100)
real                   :: lis_gds(50)
real                   :: ppt_field(nstg2)
logical*1              :: lb(nstg2)
real, dimension(nc,nr) :: varfield
real                   :: udef
```

DESCRIPTION:

This subroutine interpolates a given STG2 field to the LIS grid. The arguments are:

n index of the nest

ksec1 grib decoding array

nstg2 number of elements in the input grid

ppt_field input data array to be interpolated

lb input bitmap

lis_gds array description of the LIS grid

nc number of columns (in the east-west dimension) in the LIS grid

nr number of rows (in the north-south dimension) in the LIS grid

varfield output interpolated field

The routines invoked are:

bilinear_interp (4.0.2)
Spatially interpolate the forcing data using bilinear interpolation, or

conserv_interp (4.0.4)
spatially interpolate the forcing data using conservative (budget bil.) interpolation

40.2.2 time_interp_stg2 (Source File: time_interp_stg2.F90)

REVISION HISTORY:

```
25May2006: Kristi Arsenault; Initial implementation
10 Oct 2006:   Sujay Kumar: Switched to using ESMF_State for storing
                            forcing data.
```

INTERFACE:

```
subroutine time_interp_stg2(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only: lis, lisdom
use stg2domain_module, only : stg2_struc
use suppforcing_module, only : suppforc
use baseforcing_module, only : FORC_State

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep.
The arguments are:

n index of the nest

40.2.3 precip_stg2 (Source File: precip_stg2.F90)

REVISION HISTORY:

```
25 May 2006: Kristi Arsenault;  Data and code implementation
```

INTERFACE:

```
subroutine precip_stg2( n, fname, ferror_stg2 )
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_logmod, only : logunit
use suppforcing_module, only : suppforc
use stg2domain_module, only : stg2_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
character(len=80)   :: fname
integer             :: ferror_stg2
```

DESCRIPTION:

For the given time, reads parameters from STAGE2 datasets and interpolates to a designated user-domain. NOTE:: These subroutines use the READ_GRIB routines for for opening and reading the STAGE II grib files.
The arguments are:

n index of the nest

fname name of the hourly STAGE2 file

ferror_stg2 flag to indicate success of the call (=0 indicates success)

filehr current file hour

The routines invoked are:

interp_stg2 (40.2.1)
spatially interpolates the STG2 data

40.2.4 stg2file (Source File: stg2file.F90)

INTERFACE:

```
subroutine stg2file( name, stg2dir, yr, mo, da, hr)

  implicit none
```

DESCRIPTION:

This subroutine puts together a STAGE2 filename for one hour file intervals
The arguments are:

stg2dir Name of the STAGE II directory

yr year

mo month

da day of month

hr hour of day

name name of the time-stamped STAGE II file

41 STAGEIV

This section describes the implementation of the precipitation product from NCEP, known as STAGEIV

41.1 Fortran: Module Interface stg4domain_module (Source File: stg4domain_module.F90)

This module contains variables and data structures that are used for the implementation of the precipitation data from the National Center for Environmental Prediction (NCEP) Stage IV (STAGE4) Doppler Radar+gage product. The Stage IV is a national level product, on an hourly interval, and supplements mainly the which base forcing precipitation (e.g., NLDAS) is being used.
The implementation in LIS has the derived data type `stg4_struc` that includes the variables to specify the runtime options, and the weights and neighbor information for spatial interpolation
They are desribed below:

ncol Number of columns (along the east west dimension) for the input data

nrow Number of rows (along the north south dimension) for the input data

stg4dir Directory containing the input data

stg4time The nearest, hourly instance of the incoming data (as a real time).

mi Number of points in the input grid

rlat1 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

rlon1 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

n111,n121,n211,n221 Arrays containing the neighbor information of the input grid for each grid point in LIS, for bilinear interpolation.

w111,w121,w211,w221 Arrays containing the weights of the input grid for each grid point in LIS, for bilinear interpolation.

n112,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: defineNativeSTG4      !defines the native resolution of
                                !the input data
```

PUBLIC TYPES:

```
public :: stg4_struc
```

41.1.1 defineNativeSTG4 (Source File: stg4domain_module.F90)

REVISION HISTORY:

```
11Dec2003: Sujay Kumar; Initial Specification
25May2006: Kristi Arsenault; Data and code implementation
```

INTERFACE:

```
subroutine defineNativeSTG4()
```

USES:

```
use lisdrv_module, only: lis
use listime_mgr, only : date2time
```

DESCRIPTION:

Defines the native resolution of the input forcing for STAGE4 data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readstg4crd (41.1.2)
reads the runtime options specified for STAGE4 data

bilinear_interp_input (4.0.1)
computes the neighbor, weights for bilinear interpolation

date2time (2.4.21)
converts date to the real time format - time of grid change

41.1.2 readstg4crd (Source File: readstg4crd.F90)

REVISION HISTORY:

```
11 Dec 2003; Sujay Kumar, Initial Code
05 Jun 2006; Kristi Arsenault, Code and data implementation
```

INTERFACE:

```
subroutine readstg4crd()
```

USES:

```
use ESMF_Mod
use stg4domain_module, only : stg4_struc
use lisdrv_module, only : config_lis, lis
use lis_logmod, only : logunit
```

DESCRIPTION:

This routine reads the options specific to STAGE4 forcing from the LIS configuration file.

41.1.3 getstg4 (Source File: getstg4.F90)

REVISION HISTORY:

```
25 May 2006: Kristi Arsenault;  Data and Code implementation
```

INTERFACE:

```
subroutine getstg4(n)
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr
use lis_logmod, only : logunit
use stg4domain_module, only : stg4_struc
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates hourly STAGE4 forcing. At the beginning of a simulation, the code reads the most recent past data (nearest the hour interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep.
The arguments are:

n index of the nest

lis_time Current LIS Time

stg4_file_time End boundary time of Stage IV file

file_name Stage IV filename - passed back to getstg

The routines invoked are:

tick (2.4.23)
determines the STAGE4 data times

bilinear_interp_input (4.0.1)
Computes the neighbor, weights for bilinear interpolation

conserv_interp_input (4.0.3)
Computes the neighbor, weights for conservative interpolation

stg4file (41.1.7)
Puts together appropriate file name for 1-hour intervals

precip_stg4 (41.1.4)
Interpolates STAGE4 data to LIS grid

41.1.4 precip_stg4 (Source File: precip_stg4.F90)

REVISION HISTORY:

```
25 May 2006: Kristi Arsenault;  Data and code implementation
```

INTERFACE:

```
subroutine precip_stg4( n, fname, ferror_stg4 )
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_logmod, only : logunit
use stg4domain_module, only : stg4_struc
use suppforcing_module, only : suppforc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
character(len=80)   :: fname
integer             :: ferror_stg4
```

DESCRIPTION:

For the given time, reads parameters from STAGE4 datasets and interpolates to a designated user-domain. NOTE:: These subroutines use the READ_GRIB routines for for opening and reading the STAGE IV grib files.
The arguments are:

n index of the nest

fname name of the hourly STAGE4 file

ferror_stg4 flag to indicate success of the call (=0 indicates success)

filehr current file hour

The routines invoked are:

interp_stg4 (41.1.5)
spatially interpolates the STAGE4 data

41.1.5 interp_stg4 (Source File: interp_stg4.F90)

INTERFACE:

```
subroutine interp_stg4 (n, ksec1, nstg4, ppt_field, lb, lis_gds, &
                        nc, nr, varfield )
```

USES:

```
use stg4domain_module, only : stg4_struc
use lisdrv_module, only: lis

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
integer                :: nc      ! Number of columns (in the E-W dimension) in the LIS grid
integer                :: nr      ! Number of rows (in the N-S dimension) in the LIS grid
integer                :: nstg4   ! Number of points in original STAGE IV grid
integer                :: ksec1(100)
real                   :: lis_gds(50)
real                   :: ppt_field(nstg4)
logical*1              :: lb(nstg4)
real, dimension(nc,nr) :: varfield
real                   :: udef
```

DESCRIPTION:

This subroutine interpolates a given STAGE4 field to the LIS grid. The arguments are:

n index of the nest

ksec1 grib decoding array

nstg4 number of elements in the input grid

ppt_field input data array to be interpolated

lb input bitmap

lis_gds array description of the LIS grid

nc number of columns (in the east-west dimension) in the LIS grid

nr number of rows (in the north-south dimension) in the LIS grid

varfield output interpolated field

The routines invoked are:

bilinear_interp (4.0.2)
Spatially interpolate the forcing data using bilinear interpolation, or

conserv_interp (4.0.4)
spatially interpolate the forcing data using conservative (budget bil.) interpolation

---

41.1.6 time_interp_stg4 (Source File: time_interp_stg4.F90)

REVISION HISTORY:

```
25May2006: Kristi Arsenault; Initial implementation
10 Oct 2006:   Sujay Kumar: Switched to using ESMF_State for storing
                            forcing data.
```

INTERFACE:

```
subroutine time_interp_stg4(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only: lis, lisdom
use stg4domain_module, only : stg4_struc
use suppforcing_module, only : suppforc
use baseforcing_module, only : FORC_State
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep.
The arguments are:

n index of the nest

41.1.7 stg4file (Source File: stg4file.F90)

INTERFACE:

```
subroutine stg4file( name, stg4dir, yr, mo, da, hr)

implicit none
```

DESCRIPTION:

This subroutine puts together a STAGE4 filename for one hour file intervals
The arguments are:

stg4dir Name of the STAGE IV directory

yr year

mo month

da day of month

hr hour of day

name name of the time-stamped STAGE IV file

41.2 Fortran: Module Interface stg4_finalize (Source File: stg4_finalize.F90)

REVISION HISTORY:

```
25Oct2005; Sujay Kumar, Initial Code
17Jul2006; K. Arsenault, Added Stage IV
```

INTERFACE:

```
subroutine stg4_finalize
```

USES:

```
use lisdrv_module, only : lis
use stg4domain_module
```

DESCRIPTION:

Routine to cleanup Stage IV forcing related memory allocations.

42 AGRMET radiation

This section describes the implementation of the AGRMET radiation product.

42.1 Fortran: Module Interface agrrad_module (Source File: agrrad_module.F90)

This module contains variables and data structures that are used for the implementation of the AGRMET radiation data. The AGRMET data is in polar stereographic projection at 48km resolution.
The implementation in LIS has the derived data type `agrrad_struc` that includes the variables to specify the runtime options, and the weights and neighbor information for spatial interpolation
They are desribed below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

cmapdir Directory containing the input data

cmaptime The nearest, hourly instance of the incoming data (as a real time).

griduptime1 The time to switch the input resolution to T170

mi Number of points in the input grid

rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

n112,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
------------------------------------------------------------------------------
 public :: defineNativeAGRRAD      !defines the native resolution of
                                   !the input data
------------------------------------------------------------------------------
```

PUBLIC TYPES:

```
------------------------------------------------------------------------------
 public :: agrrad_struc
```

42.1.1 defineNativeAGRRAD (Source File: agrrad_module.F90)

REVISION HISTORY:

```
11Dec2003: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine defineNativeAGRRAD()
```

USES:

```
use lisdrv_module, only: lis
use listime_mgr, only : date2time
```

DESCRIPTION:

Defines the native resolution of the input forcing for AGRMET radiation data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readagrradcrd (42.1.2)
reads the runtime options specified for AGRMET radiation data

date2time (2.4.21)
converts date to the real time format

bilinear_interp_input (4.0.1)
computes the neighbor, weights for conservative interpolation

---

42.1.2 readagrradcrd (Source File: readagrradcrd.F90)

REVISION HISTORY:

```
29Jul2005; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine readagrradcrd()
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis, config_lis
use agrrad_module, only : agrrad_struc

implicit none
```

DESCRIPTION:

This routine reads the options specific to AGRMET algorithms from the LIS configuration file.

42.1.3 getagrrad (Source File: getagrrad.F90)

REVISION HISTORY:

```
29 Jul 2005; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine getagrrad(n)
```

USES:

```
use lisdrv_module, only       : lis
use suppforcing_module, only  : suppforc
use agrrad_module,      only  : agrrad_struc
use listime_mgr, only         : tick
use lis_logmod, only          : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This is the entry point for calling various AGRMET radiation analyses. At the beginning of a simulation, the code invokes call to read the most recent past data (nearest hourly interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. . The arguments are:

n index of the nest

The routines invoked are:

readagrrad (42.1.4)
routines that reads the AGRMET radiation and interpolates it to the LIS projection.

tick (2.4.23)
computes the AGRMET read/processing times.

42.1.4 readagrrad (Source File: readagrrad.F90)

REVISION HISTORY:

```
29Jul2005; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine readagrrad(n,order,yr,mo,da,hr,error)
```

USES:

```
use lisdrv_module,      only : lis,lisdom
use agrrad_module,      only : agrrad_struc
use suppforcing_module, only : suppforc
use lis_logmod,         only : logunit
use listime_mgr,        only : tick

implicit none
```

ARGUMENTS:

```
integer,   intent(IN)    :: n
integer,   intent(IN)    :: order
integer,   intent(INOUT) :: error
integer,   intent(IN)    :: yr,mo,da,hr
```

DESCRIPTION:

This routine opens the AGRMET files and reads the radiation fields. The fields are then spatially interpolated and gaps in the interpolation dues ot mismatches in landmask (between LIS and AGRMET) are filled. The arguments are:

n index of the nest

order flag indicating which data to be read (order=1, read the previous hourly instance, order=2, read the next hourly instance)

error output error code to indicate if the data was read and interpolated successfully.

yr,mo,da,hr current year, month, day and hour.

The routines invoked are:

tick (2.4.23)
determines the AGRMET data times

agrrad_filename (42.1.5)
generates the name of the AGRMET file to be read

bilinear_interp (4.0.2)
spatially interpolate the forcing data using bilinear interpolation

agrmet_fillgaps (**??**)
fills in the mismatches due to AGRMET and LIS masks

42.1.5 agrrad_filename (Source File: readagrrad.F90)

INTERFACE:

```
subroutine agrrad_filename(name,rootdir,yr,mo,da,hr)

  implicit none
```

ARGUMENTS:

```
  character(len=*)          :: name
  character(len=*)          :: rootdir
  integer, intent(IN)       :: yr
  integer, intent(IN)       :: mo
  integer, intent(IN)       :: da
  integer, intent(IN)       :: hr
```

DESCRIPTION:

This routines generates the name of the surface temperature file by appending the hemisphere and timestamps to the root directory.
The arguments are:

name filename to be generated

rootdir name of the root directory that contains the forcing

yr year of data

mo month of data

da day of data

hr hour of data

42.1.6 agrrad_fillgaps (Source File: readagrrad.F90)

INTERFACE:

```
subroutine agrrad_fillgaps(n,varfield)
```

USES:

```
  use lisdrv_module, only : lis,lisdom
  use agrrad_module, only : agrrad_struc
  use lis_logmod, only   : logunit

  implicit none
```

USES:

```
  integer, intent(in)    :: n
  real, intent(inout)    :: varfield(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine fills in invalid grid points introduced due to reprojection from PS to lat/lon. This routine assumes that the undef or invalid value is the LIS undefined value.
The arguments are:

n index of the nest

ip interpolation option

varfield updated output field

42.1.7 time_interp_agrrad (Source File: time_interp_agrrad.F90)

REVISION HISTORY:

```
08 Dec 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine time_interp_agrrad(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module,      only : lis,lisdom
use baseforcing_module, only : FORC_State
use suppforcing_module, only : suppforc
use agrrad_module,      only : agrrad_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the AGRMET radiation forcing data to the model timestep.

42.1.8 agrrad_finalize (Source File: agrrad_finalize.F90)

REVISION HISTORY:

```
29 Jul 2005; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine agrrad_finalize
```

USES:

```
use agrrad_module,      only  : agrrad_struc

implicit none
```

DESCRIPTION:

This routine deallocates and cleans up allocated memory structures for AGRMET radiation forcing implementation.

43 CEOP forcing

This section describes the implementation of the CEOP station data.

43.1 Fortran: Module Interface ceopdomain_module (Source File: ceopdomain_module.F90)

Contains routines and data structures that are used for the implementation of the station data from various CEOP stations. The stations report estimates of meteorological forcing terms, which is spatially interpolated using the inverse distance weighting scheme (IDW).
The implementation in LIS has the derived data type `ceop_struc` that includes the variables to specify the runtime options, and the calculation of weights for spatial interpolation.

REVISION HISTORY:

```
08Dec2004: Sujay Kumar; Initial Specification

implicit none
-----------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
-----------------------------------------------------------------------------
public :: defineNativeCEOP      !defines the native resolution of
                                !the input data
-----------------------------------------------------------------------------
```

PUBLIC TYPES:

```
-----------------------------------------------------------------------------
public :: ceop_struc
```

43.1.1 defineNativeCEOP (Source File: ceopdomain_module.F90)

REVISION HISTORY:

```
08Dec2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine defineNativeCEOP()
```

USES:

```
use lisdrv_module,only : lis
use listime_mgr, only : date2time
use lis_logmod, only : logunit
```

DESCRIPTION:

This routines reads the runtime configurations for using the CEOP station data. Using the metadata provided for the stations, this routine invokes the call to compute the interpolation weights to be later used.
The routines invoked are:

readceopcrd (43.1.2)
reads the runtime options specified for CEOP station data

date2time (2.4.21)
converts date to the real time format

compute_stnwts (4.1.10)
computes the weights for spatial interpolation

43.1.2 readceopcrd (Source File: readceopcrd.F90)

REVISION HISTORY:

```
08 Dec 2004; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine readceopcrd()
```

USES:

```
use ESMF_Mod
use lis_logmod, only : logunit
use lisdrv_module, only : lis,config_lis
use ceopdomain_module, only : ceop_struc
```

DESCRIPTION:

This routine reads the options specific to CEOP station data forcing from the LIS configuration file.

43.1.3 getceop (Source File: getceop.F90)

REVISION HISTORY:

```
08 Dec 2004: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine getceop(n)
```

USES:

```
use ceopdomain_module, only : ceop_struc
use lisdrv_module, only : lis
use listime_mgr, only : tick
use lis_logmod, only : logunit
use suppforcing_module, only : suppforc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates the CEOP station data. At the beginning of a simulation, the code reads the most recent past data, and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. . The arguments are:

n index of the nest

The routines invoked are:

tick (2.4.23)
determines the CEOP data times

readceop (43.1.4)
Interpolates the appropriate CEOP station data to LIS grid

43.1.4 readceop (Source File: readceop.F90)

REVISION HISTORY:

```
08 Dec 2002: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine readceop(n,order)
```

USES:

```
use lis_logmod, only        : logunit
use lisdrv_module, only     : lis,lisdom
use ceopdomain_module, only : ceop_struc
use suppforcing_module, only : suppforc

implicit none
```

ARGUMENTS:

```
integer, intent(in)           :: order
integer, intent(in)           :: n
```

DESCRIPTION:

For the given time, reads parameters from the correct CEOP station data (ASCII), transforms into LIS forcing parameters and interpolates to the LIS domain.
The arguments are:

order flag indicating which data to be read (order=1, read the previous instance, order=2, read the next instance)

n index of the nest

The routines invoked are:

readbon (43.1.5)
reads the Bondville, IL data

readfpk (43.1.6)
reads the Fort Peck, MT data

readsgp (43.1.7)
reads the SGP data

normalize_stnwts (4.1.12)
renormalizes the station weights accounting for missing data

interp_stndata (4.1.11)
spatially interpolates the station data onto the LIS grid.

43.1.5 readbon (Source File: readbon.F90)

REVISION HISTORY:

```
08 Dec 2002: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine readbon(n,order,ftn,tair,qair,swdown,lwdown,u,v,psurf,pcp)
```

USES:

```
use lisdrv_module, only : lis
use ceopdomain_module, only : ceop_struc
use listime_mgr, only : tick
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer,  intent (IN)     :: n
integer , intent (IN)     :: ftn
integer , intent (IN)     :: order
real    , intent (INOUT)  :: psurf(ceop_struc(n)%nstns)
real    , intent (INOUT)  :: tair(ceop_struc(n)%nstns)
real    , intent (INOUT)  :: qair(ceop_struc(n)%nstns)
real    , intent (INOUT)  :: u(ceop_struc(n)%nstns),v(ceop_struc(n)%nstns)
real    , intent (INOUT)  :: pcp(ceop_struc(n)%nstns)
real    , intent (INOUT)  :: swdown(ceop_struc(n)%nstns)
real    , intent (INOUT)  :: lwdown(ceop_struc(n)%nstns)
```

DESCRIPTION:

Reads the CEOP station data for Bondville, IL. The format of the data is ASCII.
The arguments are:

n index of the nest

order flag indicating which data to be read (order=1, read the previous instance, order=2, read the next instance)

ftn unit number used to open the file

tair array for storing the 2m air temperature for all stations

qair array for storing the 2m relative humidity for all stations

swdown array for storing the downward shortwave radiation for all stations

lwdown array for storing the downward longwave radiation for all stations

u array for storing the u component of wind for all stations

v array for storing the v component of wind for all stations

psurf array for storing the surface pressure for all stations

pcp array for storing the precipitation for all stations

43.1.6 readfpk (Source File: readfpk.F90)

REVISION HISTORY:

```
08 Dec 2002: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine readfpk(n,order,ftn,tair,qair,swdown,lwdown,u,v,psurf,pcp)
```

USES:

```
use lisdrv_module, only : lis
use ceopdomain_module, only :ceop_struc
use listime_mgr, only : tick
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer,  intent (IN)     :: n
integer , intent (IN)     :: ftn
integer , intent (IN)     :: order
real    , intent (INOUT)  :: psurf(ceop_struc(n)%nstns)
real    , intent (INOUT)  :: tair(ceop_struc(n)%nstns)
real    , intent (INOUT)  :: qair(ceop_struc(n)%nstns)
real    , intent (INOUT)  :: u(ceop_struc(n)%nstns),v(ceop_struc(n)%nstns)
real    , intent (INOUT)  :: pcp(ceop_struc(n)%nstns)
real    , intent (INOUT)  :: swdown(ceop_struc(n)%nstns)
real    , intent (INOUT)  :: lwdown(ceop_struc(n)%nstns)
```

DESCRIPTION:

Reads the CEOP station data for Fort Peck, MT. The format of the data is ASCII.
The arguments are:

n index of the nest

order flag indicating which data to be read (order=1, read the previous instance, order=2, read the next instance)

ftn unit number used to open the file

tair array for storing the 2m air temperature for all stations

qair array for storing the 2m relative humidity for all stations

swdown array for storing the downward shortwave radiation for all stations

lwdown array for storing the downward longwave radiation for all stations

u array for storing the u component of wind for all stations

v array for storing the v component of wind for all stations

psurf array for storing the surface pressure for all stations

pcp array for storing the precipitation for all stations

43.1.7 readsgp (Source File: readsgp.F90)

REVISION HISTORY:

```
08 Dec 2002: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine readsgp(n,order,tair,qair,swdown,lwdown,u,v,psurf,pcp, &
     errorcode)
```

USES:

```
use lisdrv_module, only : lis
use ceopdomain_module, only : ceop_struc
use listime_mgr, only : tick, date2time
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer,  intent (IN)     :: n
integer , intent (IN)     :: order
real    , intent (INOUT)  :: psurf(ceop_struc(n)%nstns)
real    , intent (INOUT)  :: tair(ceop_struc(n)%nstns)
real    , intent (INOUT)  :: qair(ceop_struc(n)%nstns)
real    , intent (INOUT)  :: u(ceop_struc(n)%nstns),v(ceop_struc(n)%nstns)
real    , intent (INOUT)  :: pcp(ceop_struc(n)%nstns)
real    , intent (INOUT)  :: swdown(ceop_struc(n)%nstns)
real    , intent (INOUT)  :: lwdown(ceop_struc(n)%nstns)
integer , intent (INOUT)  :: errorcode
```

DESCRIPTION:

Reads the CEOP station data for SGP. The format of the data is ASCII.
The arguments are:

n index of the nest

order flag indicating which data to be read (order=1, read the previous instance, order=2, read the next instance)

tair array for storing the 2m air temperature for all stations

qair array for storing the 2m relative humidity for all stations

swdown array for storing the downward shortwave radiation for all stations

lwdown array for storing the downward longwave radiation for all stations

u array for storing the u component of wind for all stations

v array for storing the v component of wind for all stations

psurf array for storing the surface pressure for all stations

pcp array for storing the precipitation for all stations

43.1.8 time_interp_ceop (Source File: time_interp_ceop.F90)

REVISION HISTORY:

```
08 Dec 2004: Sujay Kumar; Initial Specification
10 Oct 2006:   Sujay Kumar: Switched to using ESMF_State for storing
                            forcing data.
```

INTERFACE:

```
subroutine time_interp_ceop(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis,lisdom
use ceopdomain_module, only : ceop_struc
use baseforcing_module, only : FORC_State
use suppforcing_module, only : suppforc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the CEOP forcing data to the model timestep. All variables except precipitation is linearly interpolated.

44 SCAN forcing

This section describes the implementation of the SCAN station data.

44.1 Fortran: Module Interface scandomain_module (Source File: scandomain_module.F90)

Contains routines and data structures that are used for the implementation of the station data from various SCAN stations. The stations report estimates of meteorological forcing terms, which is spatially interpolated using the inverse distance weighting scheme (IDW).
The implementation in LIS has the derived data type `scan_struc` that includes the variables to specify the runtime options, and the calculation of weights for spatial interpolation.

REVISION HISTORY:

```
13Apr2007: Bailing Li:  Initial Specification

implicit none
-----------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
-----------------------------------------------------------------------------
public :: defineNativeSCAN      !defines the native resolution of
                                !the input data
-----------------------------------------------------------------------------
```

PUBLIC TYPES:

```
-----------------------------------------------------------------------------
public :: scan_struc
```

44.1.1 defineNativeSCAN (Source File: scandomain_module.F90)

INTERFACE:

```
subroutine defineNativeSCAN()
```

USES:

```
use lisdrv_module,only : lis
use listime_mgr, only : date2time
use lis_logmod, only : logunit
```

DESCRIPTION:

This routines reads the runtime configurations for using the SCAN station data. Using the metadata provided for the stations, this routine invokes the call to compute the interpolation weights to be later used.
The routines invoked are:

readscancrd (44.1.2)
reads the runtime options specified for SCAN station data

date2time (2.4.21)
converts date to the real time format

compute_stnwts (4.1.10)
computes the weights for spatial interpolation

44.1.2 readscancrd (Source File: readscancrd.F90)

REVISION HISTORY:

```
13 Apr 2007; Bailing Li, Initial Code
```

INTERFACE:

```
subroutine readscancrd()
```

USES:

```
use ESMF_Mod
use lis_logmod, only : logunit
use lisdrv_module, only : lis,config_lis
use scandomain_module, only : scan_struc
```

DESCRIPTION:

This routine reads the options specific to SCAN station data forcing from the LIS configuration file.

44.1.3 getscan (Source File: getscan.F90)

REVISION HISTORY:

```
13 Apr 2007: Bailing Li; Initial Specification
```

INTERFACE:

```
subroutine getscan(n)
```

USES:

```
use scandomain_module, only : scan_struc
use lisdrv_module, only : lis
use listime_mgr, only : tick
use lis_logmod, only : logunit
use suppforcing_module, only : suppforc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates the SCAN station data. At the beginning of a simulation, the code reads the most recent past data, and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. . The arguments are:

n index of the nest

The routines invoked are:

tick (2.4.23)
determines the SCAN data times

readscan (44.1.4)
Interpolates the appropriate SCAN station data to LIS grid

44.1.4 readscan (Source File: readscan.F90)

REVISION HISTORY:

```
13 Apr 2007: Bailing Li; Initial Specification
```

INTERFACE:

```
subroutine readscan(n,ftn,order)
```

USES:

```
use lis_logmod, only        : logunit
use lisdrv_module, only     : lis,lisdom
use scandomain_module, only : scan_struc
use suppforcing_module, only : suppforc

implicit none
```

ARGUMENTS:

```
integer, intent(in)           :: order
integer, intent(in)           :: n
integer, intent(in)           :: ftn
```

DESCRIPTION:

For the given time, reads parameters from the correct SCAN station data (ASCII), transforms into LIS forcing parameters and interpolates to the LIS domain.
The arguments are:

order flag indicating which data to be read (order=1, read the previous instance, order=2, read the next instance)

n index of the nest

ftn unit number for the SCAN station data

The routines invoked are:

normalize_stnwts (4.1.12)
renormalizes the station weights accounting for missing data

interp_stndata (4.1.11)
spatially interpolates the station data onto the LIS grid.

44.1.5 time_interp_scan (Source File: time_interp_scan.F90)

REVISION HISTORY:

```
13 Apr 2007: Bailing Li; Initial Specification
```

INTERFACE:

```
subroutine time_interp_scan(n)
```

USES:

```
use ESMF_Mod
use lis_logmod, only        : logunit
use lisdrv_module, only : lis,lisdom
use scandomain_module, only : scan_struc
use baseforcing_module, only : FORC_State
use suppforcing_module, only : suppforc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the SCAN forcing data to the model timestep. All variables except precipitation is linearly interpolated.

Part XII
Data Assimilation in LIS

A Data Assimilation Algorithms in LIS

This section contains the description of the implementations of data assimilation algorithms in LIS. These algorithms are used in conjunction with the specific interface implementations of the land surface model and the observations, which are described in sections XIV and XIII, respectively.

B Direct Insertion

B.1 Fortran: Module Interface directInsertion_module (Source File: directInsertion_module.F90)

This module contains interfaces and subroutines that control the incorporation of a data set using direct insertion in a land surface model.

REVISION HISTORY:

```
 27Feb05    Sujay Kumar;   Initial Specification
 21Jun06    Sujay Kumar:   Updated implementation with ESMF
                           structures
```

USES:

```
  use lisdrv_module, only : lis

  implicit none
 ------------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
 ------------------------------------------------------------------------------
  public :: di_init  ! Initialization for Direct Insertion
  public :: di_increments ! Routine to compute DI increments
  public :: di_update     ! Routine to apply DI increments
  public :: di_diagnostics ! write DI related diagnostics
  public :: di_final ! Finalization for Direct Insertion
```

B.1.1 di_init (Source File: directInsertion_module.F90)

INTERFACE:

```
  subroutine di_init
```

DESCRIPTION:

This method performs the required initializations for the direct insertion method.

B.1.2 di_increments (Source File: directInsertion_module.F90)

INTERFACE:

```
subroutine di_increments(n)
```

DESCRIPTION:

This method computes the analysis increments from direct insertion. The direct insertion method simply retrieves the LSM state, prompts the land surface model to transform the OBS state to an LSM state. The tranformed LSM state is then used to overwrite the current LSM state.
The methods invoked are:

getLSMstateVar (**??**)
obtain the specified LSM state prognostic variables

ObsTransform (2.39.22)
transform the observation state into an LSM space

mapObsToLSM (2.39.26)
map the observation state to generate the analysis increments

USES:

```
use ESMF_Mod
use lsm_module, only          : LSM_State, LSM_Incr_State
use observations_module, only : OBS_State
use lis_logmod, only          : logunit
```

B.1.3 di_update (Source File: directInsertion_module.F90)

INTERFACE:

```
subroutine di_update(n)
```

USES:

```
use ESMF_Mod
use lsm_module, only          : LSM_State, LSM_Incr_State
```

ARGUMENTS:

```
integer, intent(in)    :: n
```

DESCRIPTION:

This method applies the analysis increments computed by the direct insertion method and updates the model prognostic variables.
The methods invoked are:

setLSMstateVar (**??**)
set the specified LSM state prognostic variables

B.1.4 di_diagnostics (Source File: directInsertion_module.F90)

INTERFACE:

```
subroutine di_diagnostics(n)
```

DESCRIPTION:

This subroutine generates the EnKF related diagnostics and outputs it to a file.

B.1.5 di_final (Source File: directInsertion_module.F90)

INTERFACE:

```
subroutine di_final
```

DESCRIPTION:

This method performs the finalization for all direct insertion related structures and subroutines.

C GMAO Ensemble Kalman Filter

C.1 Fortran: Module Interface gmaoenkf_module (Source File: gmaoenkf_module.F90)

This module contains interfaces and subroutines that control the incorporation of a data set using the ensemble kalman filter (EnKF) method, into a land surface model.
The EnKF algorithm is based on the work of Rolf Reichle at the NASA Global Modeling and Assimilation Office (GMAO) at the NASA GSFC.

REVISION HISTORY:

```
27 Feb 2005: Sujay Kumar; Initial Specification
```

USES:

```
use ESMF_Mod
use enkf_types
use lisdrv_module, only : lis

implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
---------------------------------------------------------------------------
public :: gmaoenkf_init  ! Initialization for EnKF
public :: gmaoenkf_increments ! compute analysis increments
public :: gmoenkf_update ! apply the analysis increments
public :: gmaoenkf_diagnostics ! write EnKF related diagnostics
public :: gmaoenkf_final ! Finalization for EnKF
---------------------------------------------------------------------------
```

PUBLIC TYPES:

```
-----------------------------------------------------------------------------
 public :: gmaoenkf_struc ! data structure containing EnKF diagnostics
```

C.1.1 gmaoenkf_init (Source File: gmaoenkf_module.F90)

INTERFACE:

```
subroutine gmaoenkf_init
```

USES:

```
    use ESMF_Mod
    use lisdrv_module, only       : lis,config_lis
    use observations_module, only : OBS_State
    use lis_logmod,    only       : logunit
```

DESCRIPTION:

This method performs the required initializations for the GMAO EnKF method. The method reads the runtime settings from the LIS configuration file.

C.1.2 gmaoenkf_increments (Source File: gmaoenkf_module.F90)

INTERFACE:

```
subroutine gmaoenkf_increments(n)
```

USES:

```
    use lisdrv_module,       only : lis, lisdom
    use lsm_module,          only : LSM_State, LSM_Pert_State,&
         LSM_Incr_State
    use observations_module, only : OBS_State, OBS_Pert_State
    use enkf_general,        only : enkf_increments
    use lis_logmod,          only : logunit
    use spmdMod,             only : masterproc
    use my_matrix_functions, only : row_variance
```

ARGUMENTS:

```
    integer, intent(IN)    :: n
```

DESCRIPTION:

This routine computes the analysis increments for EnKF. The state variable increments are computed and stored int the Increment State.
The arguments and variables are:

n index of the nest

data_status flag indicating if the observations are new

N_state Number of state prognostic variables

N_ens Number of ensembles

N_obs_size total number of observations

N_selected_obs number of selected observations

st_id,en_id indices of the selected observations

stvar state prognostic variables

Obs_pert observation perturbations

Obs_cov observations error covariances

Observations observations object used in the GMAO routines

The methods invoked are:

generateObservations C.1.10
obtain the observations

getObsPred (2.39.24)
obtain model's estimate of the observations

getObsPert (C.1.9)
obtain the observation perturbations

generateObsparam (C.1.11)
generate the 'obsparam' (metadata for observations)

getLSMstateVar 2.39.18 obtain the specified LSM prognostic variables.

assemble_obs_cov (C.1.6) assembles the observation error covariance

getSelectedObsNumber(2.33.6)
obtain the number of selected observations for each modeling point.

enkf_increments (C.2.1)
apply the EnKF filter to obtain the prognostic variable increments.

row_variance (C.4.2)
computes the row variance HPH'

setLSMstateVar (2.39.20)
assigns the specified LSM state prognostic variables

scaleLSMstatVar (2.39.34)
scales the state variables for computational stability

descaleLSMstatVar (2.39.36)
descales the state variables to the original state

qcLSMState 2.39.32 QC the updated LSM state

C.1.3 gmaoenkf_update (Source File: gmaoenkf_module.F90)

INTERFACE:

```
subroutine gmaoenkf_update(n)
```

USES:

```
use lsm_module,          only : LSM_State, LSM_Incr_State
use observations_module, only : OBS_State
use lis_logmod,          only : logunit
implicit none

integer,       intent(in)    :: n
```

DESCRIPTION:

This routine updates the model prognostics using the analysis increments computed earlier.

C.1.4 gmaoenkf_diagnostics (Source File: gmaoenkf_module.F90)

INTERFACE:

```
subroutine gmaoenkf_diagnostics(n)
```

USES:

```
use spmdMod,         only : masterproc
use lisdrv_module,   only : lis, lisdom
use lsm_module,      only : LSM_State
use observations_module, only : OBS_State
use lis_fileIOMod, only   : &
     create_output_directory, create_daspread_filename,&
     create_innov_filename
use drv_output_mod,  only : drv_write_dastats, drv_write_innov
use lis_logmod, only          : logunit
```

ARGUMENTS:

```
integer, intent(IN)    :: n
```

DESCRIPTION:

This subroutine generates the EnKF related diagnostics and outputs it to a file. This includes a text output of selected ensemble members, their mean, standard deviation, observations, and normalized innovations. The frequency of diagnostic outputs can be specified in the LIS configuration file.
The routines invoked are:

create_output_directory (2.25.2)
call to create output directory for DA statistics

getLSMvarnames (C.1.8)
retrieve the names of the LSM prognostic variables

pruneVarname (C.1.5)
trims the variable name, eliminating white spaces

create_stats_filename (2.25.5)
creates the filename for statistics

create_innov_filename (2.25.6)
creates the name of the innovations file

getLSMstateVar (2.39.18)
retrieve the lsm state variables

getLSMdata (**??**)
unpack the LSM state and retrive the data

drv_writevar_txt (**??**)
call to write a variable in a text file

writeDAObsStats (**??**)
writes a formatted observation output

C.1.5 pruneVarname (Source File: gmaoenkf_module.F90)

INTERFACE:

```
subroutine pruneVarname(varname)
```

ARGUMENTS:

```
character(len=*), intent(INOUT) :: varname
```

DESCRIPTION:

This routine generates a filename based on the names of the state prognostic variables

varname name of the variable filename

C.1.6 assemble_obs_cov (Source File: gmaoenkf_module.F90)

INTERFACE:

```
subroutine assemble_obs_cov(nob, N_obs, obs_param, &
    Observations, Obs_cov)

implicit none
```

ARGUMENTS:

```
integer, intent(in)                              :: nob, N_obs
type(obs_param_type), dimension(nob), intent(in) :: obs_param
type(obs_type), dimension(N_obs), intent(in)     :: Observations
real, intent(out), dimension(N_obs,N_obs)        :: Obs_cov
```

DESCRIPTION:

assemble measurements error covariance (reichle, 27 Jul 2005)

nob number of observation types

N_obs number of observations

obs_param observation settings

Observations Observations data type

Obs_cov observation error covariance

C.1.7 gmaoenkf_final (Source File: gmaoenkf_module.F90)

INTERFACE:

```
subroutine gmaoenkf_final
```

DESCRIPTION:

This method performs the finalization for all EnKF related structures and subroutines.

C.1.8 getLSMvarnames (Source File: gmaoenkf_module.F90)

INTERFACE:

```
subroutine getLSMvarnames(LSM_State, dim1, varname)
```

ARGUMENTS:

```
type(ESMF_State)      :: LSM_State
integer               :: dim1
character(len=*)      :: varname(dim1)
```

DESCRIPTION:

This routine retrieves the names of the state prognostic variables from the LSM state.

C.1.9 getObsPert (Source File: gmaoenkf_module.F90)

INTERFACE:

```
subroutine getObsPert(OBS_Pert_State, dim1, dim2, pert)
```

ARGUMENTS:

```
type(ESMF_State)      :: OBS_Pert_State
integer               :: dim1, dim2
real                  :: pert(dim1,dim2)
```

DESCRIPTION:

This routine retrieves the perturbations to be applied to the observations from the Observation perturbation state
The arguments are:

OBS_Pert_State Observation perturbation state

dim1,dim2 dimensions of the perturbations array

pert perturbation array

C.1.10 generateObservations (Source File: gmaoenkf_module.F90)

INTERFACE:

```
subroutine generateObservations(n, Nobjs, N_obs_size, OBS_State, Observations )

  use lisdrv_module, only        : lisdom
```

ARGUMENTS:

```
  integer,     intent(IN)        :: n
  integer,     intent(IN)        :: Nobjs
  integer,     intent(IN)        :: N_obs_size
  type(ESMF_State)               :: OBS_State
  type(obs_type)                 :: Observations(N_obs_size)
```

DESCRIPTION:

This subroutine unpacks the Observation state and packages them into the 'obs_type' datastructure.

n index of the nest

Nobjs Number of observation types

OBS_State ESMF State containing observations

Observations Observations datastructure being returned.

C.1.11 generateObsparam (Source File: gmaoenkf_module.F90)

INTERFACE:

```
subroutine generateObsparam(Nobjs, OBS_Pert_State, obs_param)

  implicit none
```

ARGUMENTS:

```
  integer                      :: Nobjs
  type(ESMF_State)             :: OBS_Pert_State
  type(obs_param_type)         :: obs_param(Nobjs)
```

DESCRIPTION:

This routine obtains the metadata information associated with the observations and puts them into the obs_param datastructure
The arguments are:

Nobjs Number of observation types

OBS_Pert_State ESMF state containing observation perturbations

obs_param obs_param datastructure being returned

C.1.12 getLSMData (Source File: gmaoenkf_module.F90)

INTERFACE:

```
subroutine getLSMData(LSM_State, dim1, dim2, lsmdata)
```

ARGUMENTS:

```
type(ESMF_State)      :: LSM_State
integer               :: dim1, dim2
real                  :: lsmdata(dim1, dim2)
```

DESCRIPTION:

This routine retrives the LSM state prognostic variables from the LSM state.
The arguments are:

LSM_State LSM State containing prognostic variables

dim1,dim2 dimensions of the prognostic variable array

lsmdata prognostic variable array

C.2 Fortran: Module Interface enkf_general (Source File: enkf_general.F90)

this file contains a collection of general Ensemble Kalman filter subroutines and compact support subroutines

REVISION HISTORY:

```
reichle, 20 Apr 2001
reichle, 18 Mar 2004 - optional arguments
reichle, 27 Jan 2005 - eliminated use of module select_kinds
reichle, 19 Jul 2005 - merged compact_support.f90 and enkf_general.f90
reichle,  1 Aug 2005 - eliminated tile_coord
reichle, 18 Oct 2005 - return increments instead of updated State
```

C.2.1 enkf_increments (Source File: enkf_general.F90)

INTERFACE:

```
subroutine enkf_increments( gid, &
     N_state, N_obs, N_ens, &
     Observations, Obs_pred, Obs_err, Obs_cov, &
     State_incr, &
     State_lon, State_lat, xcompact, ycompact )
```

USES:

```
use enkf_types
use my_matrix_functions
use lis_logmod

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: gid
integer, intent(in) :: N_state, N_obs, N_ens
type(obs_type), intent(in), dimension(N_obs) :: Observations
real, intent(in), dimension(N_obs,N_ens) :: Obs_pred
real, intent(in), dimension(N_obs,N_ens) :: Obs_err
real, intent(in), dimension(N_obs,N_obs) :: Obs_cov
real, intent(inout), dimension(N_state,N_ens) :: State_incr

! optional inputs
real, dimension(N_state), intent(in), optional :: State_lon, State_lat
real, intent(in), optional :: xcompact       ! [deg] longitude
real, intent(in), optional :: ycompact       ! [deg] latitude
```

DESCRIPTION:

perform EnKF update
IMPORTANT: on input, State_incr must contain State_minus(1:N_state,1:N_ens) on output, State_incr contains the increments
if optional inputs State_lon, State_lat, xcompact, and ycompact are present, Hadamard product is applied to HPHt and PHt

---

C.2.2 hadamard_for_repr_matrix (Source File: enkf_general.F90)

INTERFACE:

```
subroutine hadamard_for_repr_matrix( N_obs, Observations,            &
     xcompact, ycompact,                                             &
     Repr_Matrix )
```

USES:

```
use enkf_types                     ! for obs_type

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: N_obs

type(obs_type), intent(in), dimension(N_obs) :: Observations

real, intent(in) :: xcompact       ! [deg] longitude
real, intent(in) :: ycompact       ! [deg] latitude

real, dimension(N_obs,N_obs), intent(inout) :: Repr_matrix
```

C.2.3 gaspari_cohn (Source File: enkf_general.F90)

INTERFACE:

```
function gaspari_cohn( d )
```

DESCRIPTION:

evaluate 5th-order polynomial from Gaspari and Cohn, 1999, Eq (4.10)
On input, d = separation distance relative to the distance at which all correlations vanish. In the isotropic case, Gaspari and Cohn, 1999, Eq. (4.10)
d = sqrt(dx**2 + dy**2) / (2*c) = —z— / (2*c)
or in the anisotropic case
d = sqrt((dx/xcompact)**2 + (dy/ycompact)**2)
*** Use —z—/c = 2*d. All correlations vanish for d ¿ 1. ***

C.2.4 get_gaspari_cohn (Source File: enkf_general.F90)

INTERFACE:

```
function get_gaspari_cohn( dx, dy, xcompact, ycompact )
```

DESCRIPTION:

for a given lat/lon distance, compute the anisotropic compact support (Gaspari and Cohn) weights
input coordinates must be in degrees latitude/longitude
dx = longitude separation of two points dy = latitude separation of two points
xcompact = longitude scale of compact support ycompact = latitude scale of compact support
for the isotropic Gaspari and Cohn function, the relative distance is in Gaspari and Cohn, 1999, notation (Eq (4.10))
d = sqrt(dx**2 + dy**2) / 2*c
get_gaspari_cohn() uses a generalized anisotropic Gaspari and Cohn approach (essentially coordinate stretching, see handwritten notes for details)
d = sqrt((dx/xcompact)**2 + (dy/ycompact)**2)
All correlations vanish outside of an ellipse with semi-axes xcompact and ycompact, ie Gaspari and Cohn weights vanish for d ¿ 1 (note the factor 2!)
When the anisotropic case is reduced back to the isotropic case, (ie if xcompact==ycompact) then c = xcompact/2 = ycompact/2.

C.3 Fortran: Module Interface enkf_types (Source File: enkf_types.F90)

This module contains the definition of types for the GMAO EnKF

REVISION HISTORY:

```
reichle, 19 Jul 2005
```

C.4 Fortran: Module Interface my_matrix_functions (Source File: my_matrix_functions.F90)

this file contains a collection of matrix operation subroutines

C.4.1 (Source File: my_matrix_functions.F90)

reichle, 1 May 01 reichle, 18 Apr 06 - renamed subroutine sort()

C.4.2 row_variance (Source File: my_matrix_functions.F90)

INTERFACE:

```
subroutine row_variance( M, N, A, var )

  implicit none
```

ARGUMENTS:

```
  integer, intent(in) :: M, N

  real, intent(in), dimension(M,N) :: A

  real, intent(out), dimension(M)  :: var
```

DESCRIPTION:

compute variance of each row of an M-by-N matrix A

C.4.3 row_third_moment (Source File: my_matrix_functions.F90)

INTERFACE:

```
subroutine row_third_moment( M, N, A, third_moment )

  implicit none
```

ARGUMENTS:

```
  integer, intent(in) :: M, N
  real, intent(in), dimension(M,N) :: A
  real, intent(out), dimension(M)  :: third_moment
```

DESCRIPTION:

compute third moment of each row of an M-by-N matrix A

$$third_moment = 1/N * sum_{i=1}^{N}(x_i - mean(x)) * *3 \tag{1}$$

C.4.4 five_number_summary (Source File: my_matrix_functions.F90)

INTERFACE:

```
subroutine five_number_summary( M, N, A, five_numbers )

  implicit none
```

ARGUMENTS:

```
integer, intent(in) :: M, N
real, intent(in), dimension(M,N)  :: A
real, intent(out), dimension(M,5)  :: five_numbers
```

DESCRIPTION:

get five number summary (median, lower and upper quartiles, min and max) of each row of an M-by-N data matrix A
inputs: M : number of rows of data = number of different data types N : number of columns of data = number of ensemble members A : M-by-N matrix, left unchanged by this function
outputs: five_numbers : M-by-5 matrix containing statistical summary: column 1: min column 2: lower quartile column 3: median column 4: upper quartile column 5: max
Type: f90 Author: Rolf Reichle Date: 2 May 2001

C.4.5 adjust_mean (Source File: my_matrix_functions.F90)

INTERFACE:

```
subroutine adjust_mean( N_row, N_col, A, M )

  implicit none
```

ARGUMENTS:

```
integer, intent(in) :: N_row, N_col
real, intent(inout), dimension(N_row,N_col)  :: A
real, intent(in), optional, dimension(N_row) :: M
```

DESCRIPTION:

adjust N_row by N_col matrix A such that mean over columns for each row is given by the corresponding element in vector M of length N_row
vector of mean values M is optional input, if not present zero mean is assumed

---

C.4.6 adjust_std (Source File: my_matrix_functions.F90)

INTERFACE:

```
subroutine adjust_std( N_row, N_col, A, std )

  implicit none
```

ARGUMENTS:

```
integer, intent(in) :: N_row, N_col
real, intent(inout), dimension(N_row,N_col)  :: A
real, intent(in), optional :: std
```

DESCRIPTION:

adjust N_row by N_col matrix A such that (sample) standard deviation of all elements is exactly equal to std
std is optional input, if not present std=1 is assumed

C.4.7 matrix_std (Source File: my_matrix_functions.F90)

INTERFACE:

```
subroutine matrix_std( N_row, N_col, A, std )

  implicit none
```

ARGUMENTS:

```
  integer, intent(in) :: N_row, N_col
  real, intent(inout), dimension(N_row,N_col)  :: A
  real, intent(out) :: std
```

DESCRIPTION:

compute std of all elements of N_row by N_col matrix A

C.4.8 ludcmp (Source File: my_lu_decomp.F90)

LU decomposition after Numerical Recipes
Given a matrix a[1..n][1..n], this routine replaces it by the LU decomposition of a rowwise permutation of itself. a and n are inputs. a is output, arranged as in equation (2.3.14) (see book). indx[1..n] is an output vector that records the row permutation effected by the partial pivoting; This routine is used in combination with lubksb to solve linear equations or invert a matrix.

REVISION HISTORY:

```
edited 20 Apr 01, reichle
- eliminated d (for computation of determinant)
- eliminated np (can use automatic arrays nowadays...)
edited 03 Jun 02, reichle

- eliminated parameter NMAX, using dynamic allocation
```

INTERFACE:

```
SUBROUTINE ludcmp(a,n,indx)
```

C.4.9 lubksb (Source File: my_lu_decomp.F90)

LU backsubstitution after Numerical Recipes
Solves the set of n linear equations A X = B. Here a[1..n][1..n] is input, not as the matrix A but rather as its LU decomposition, determined by the routine ludcmp. indx[1..n] is input as the permutation vector returned by ludcmp. b[1..n] is input as the right-hand side vector B, and returns the solution vector X. a,n, and indx ar not modified by this routine and can be left in place for successive calls with different right-hand sides b. This routine takes into account the possibility that b will begin with many zero elements, so it is efficient for use in matrix inversion.

REVISION HISTORY:

```
edited 20 Apr 01, reichle
```

INTERFACE:

```
SUBROUTINE lubksb(a,n,indx,b)
```

C.4.10 nr_sort (Source File: nr_sort.F90)

QUICKSORT algorithm after Numerical Recipes
Sorts an array arr[1..n] into ascending numerical order using the Quicksort algorithm. n is input; arr is replaced on output by its sorted rearrangement.

INTERFACE:

```
SUBROUTINE nr_sort(n,arr)
```

D Perturbation Algorithms in LIS

This section contains the description of the implementations of perturbation algorithms in LIS. The algorithms are used to perturb forcing, observation and prognostic varibles for data assimilation.

E Static Perturbation Algorithm

E.1 Fortran: Module Interface staticPerturbMod (Source File: staticPerturbMod.F90)

This module contains routines to perturb the specified input using either random gaussian noise or random lognormal noise.

REVISION HISTORY:

```
08Jul05    Sujay Kumar; Initial Specification
```

E.1.1 staticPerturb_init (Source File: staticPerturbMod.F90)

INTERFACE:

```
subroutine staticPerturb_init (Base_State, Pert_State)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod,    only : logunit
```

ARGUMENTS:

```
type(ESMF_State)        :: Base_State(lis%nnest)
type(ESMF_State)        :: Pert_State(lis%nnest)
```

DESCRIPTION:

Initialization for perturbation setup. This routine is simply a placeholder

---

E.1.2 staticPerturburb (Source File: staticPerturbMod.F90)

INTERFACE:

```
subroutine staticPerturb(n, Base_State, Pert_State)
```

USES:

```
use lis_logmod,    only : logunit
```

ARGUMENTS:

```
integer, intent(in)       :: n
type(ESMF_State)          :: Base_State
type(ESMF_State)          :: Pert_State
```

DESCRIPTION:

This routine computes the perturbations (as increments) and packages them into an ESMF state. The perturbations are created based on a random gaussian noise (additive) or a random lognormal noise (multiplicative).

F GMAO Perturbation Algorithm

F.1 Fortran: Module Interface gmaopert_Mod (Source File: gmaopert_Mod.F90)

This module contains routines to peturb either the forcing, prognostic variables, or observations based on the algorithmic implementations of Rolf Reichle (NASA GMAO).

REVISION HISTORY:

```
08Jul05    Sujay Kumar; Initial Specification
```

F.1.1 gmaopert_init (Source File: gmaopert_Mod.F90)

Initializes memory structures for specified perturbation options.

INTERFACE:

```
subroutine gmaoPert_init(Base_State, Pert_State)
```

USES:

```
use landpert_routines, only : assemble_pert_param,&
     get_init_pert_rseed, get_pert
use lisdrv_module, only : lis
use nr_ran2_gasdev, only : NRANDSEED
```

F.1.2 gmaoperturb (Source File: gmaopert_Mod.F90)

This routine computes the perturbations (as increments) and packages them into an ESMF state.

INTERFACE:

```
subroutine gmaoperturb(n, Base_State, Pert_State)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use landpert_routines, only : get_pert

implicit none
```

ARGUMENTS:

```
integer, intent(in)       :: n
type(ESMF_State)          :: Base_State
type(ESMF_State)          :: Pert_State
```

F.2 Fortran: Module Interface landpert_module.F90 (Source File: landpert_module.F90)

This module contains routines that can be used to perturb forcing fields such as precipitation and radiation or model prognostic variables such as soil moisture or soil temperature
MUST initialize random seed and Pert_ntrmdt by calling get_pert() with initialize=.true. at the start of the driver program (otherwise set initialize=.false.)

REVISION HISTORY:

```
24Jan05 Rolf Reichle  Initial Specification
07Jun05 Rolf Reichle  more init options
07Jul05 Sujay Kumar   Initial Specification
```

F.2.1 allocate_pert_param (Source File: landpert_module.F90)

INTERFACE:

```
subroutine allocate_pert_param(N_pert, N_x, N_y, pert_param)

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: N_pert, N_x, N_y
type(pert_param_type), dimension(:), pointer :: pert_param
```

DESCRIPTION:

allocates memory for the perturbation data structures.

F.3 Fortran: Module Interface landpert_routines.F90 (Source File: landpert_routines.F90)

This module contains routines that can be used to perturb forcing fields such as precipitation and radiation or model prognostic variables such as soil moisture or soil temperature
MUST initialize random seed and Pert_ntrmdt by calling get_pert() with initialize=.true. at the start of the driver program (otherwise set initialize=.false.)

REVISION HISTORY:

```
24Jan05 Rolf Reichle  Initial Specification
07Jun05 Rolf Reichle  more init options
07Jul05 Sujay Kumar   Specification in LIS
```

F.3.1 assemble_pert_param (Source File: landpert_routines.F90)

INTERFACE:

```
subroutine assemble_pert_param( Pert_State, N_x, N_y,               &
     N_pert, pert_param )
```

DESCRIPTION:

sample subroutine that demonstrates how pert_param can be assembled for forcing perturbations
return N_force_pert, allocate and assemble structure pert_param
make sure order of fields is compatible with your driver
forcing field 1 = precip forcing field 2 = shortwave forcing field 3 = longwave forcing field 4 = air temperature

REVISION HISTORY:

```
reichle, 11 Feb 2005
reichle,  8 Jun 2005
```

USES:

```
use lisdrv_module, only : lis
use ESMF_Mod

implicit none
```

ARGUMENTS:

```
type(ESMF_State), intent(in) :: Pert_State
integer, intent(in) :: N_x, N_y
integer, intent(in) :: N_pert
type(pert_param_type), dimension(:), pointer :: pert_param ! out
```

F.3.2 get_sqrt_corr_matrix (Source File: landpert_routines.F90)

INTERFACE:

```
subroutine get_sqrt_corr_matrix( N, A, S )
```

USES:

```
use nr_jacobi

implicit none
```

ARGUMENTS:

```
integer, intent(in)                  :: N
real,    intent(in),  dimension(N,N) :: A
real,    intent(out), dimension(N,N) :: S
```

DESCRIPTION:

get sqrt S of real, symmetric (correlation) matrix A
NOTE: there is no check that A is indeed symmetric!
A = S*transpose(S)
reichle, 7 Jun 2005

F.3.3 get_init_Pert_rseed (Source File: landpert_routines.F90)

INTERFACE:

```
subroutine get_init_Pert_rseed( N_ens, N_domain, ens_id, domain_id, &
     init_Pert_rseed )

  implicit none
```

ARGUMENTS:

```
integer, intent(in) :: N_ens, N_domain
integer, dimension(N_ens),    intent(in) :: ens_id
integer, dimension(N_domain), intent(in) :: domain_id
integer, dimension(N_ens,N_domain), intent(out) :: init_Pert_rseed
```

DESCRIPTION:

get initial random seed "init_Pert_rseed" for initializing Pert_rseed within get_pert()
A different random seed is necessary for each ensemble member. In parallel applications, the random seed must also be different for each subdomain.
This subroutine is meant as a sample for how the initial Pert_rseed can be set. In this example, ens_id and domain_id are meant to be nonnegative small integers.

F.3.4 get_pert (Source File: landpert_routines.F90)

INTERFACE:

```
subroutine get_pert(                                &
     N_pert, N_ens, N_x, N_y,                       &
     dx, dy, dtstep,                                &
     pert_param,                                    &
     Pert_rseed,                                    &
     Pert_ntrmdt,                                   &
     Pert,                                          &
```

```
   initialize_rseed,                              &
   initialize_ntrmdt               )


implicit none
```

ARGUMENTS:

```
integer, intent(in) :: N_pert   ! # different perturbations

integer, intent(in) :: N_ens  ! # ensemble members

integer, intent(in) :: N_x    ! # grid cells in longitudinal direction
integer, intent(in) :: N_y    ! # grid cells in latitudinal direction

real, intent(in) :: dx   ! grid cell size in longitudinal direction [deg]
real, intent(in) :: dy   ! grid cell size in latitudinal direction [deg]

real, intent(in) :: dtstep        ! perturbation time step in seconds

! Parameter structure for perturbations (see type definition for details).

type(pert_param_type), dimension(:), pointer :: pert_param

! Pert_ntrmdt are intermediate perturbation fields
! that need to be remembered between calls to this subroutine.
! In essence, they store N_pert mutually uncorrelated
! perturbation fields of standard-normal distribution.
! Pert_rseed is the random seed for the generation of
! Pert_ntrmdt and is treated similarly to a prognostic variable.
! Each ensemble member has its own random seed.

integer, dimension(NRANDSEED,N_ens), intent(inout) :: Pert_rseed

real, dimension(N_pert,N_x,N_y,N_ens), intent(inout) :: Pert_ntrmdt

! Pert are N_pert cross-correlated perturbation
! fields that are rotated and scaled versions of Pert_ntrmdt
! so that Pert has the the mean values, standard deviations and
! cross-correlations specified in pert_param.
! The distribution is lognormal for multiplicative perturbations.
! Pert should be used as follows for field F (eg. large-scale
! precip, convective precip, lw radiation, ...)
! Fprime = F+Pert   for additive perturbations
! Fprime = F*Pert   for multiplicative perturbations
! Note that this subroutine does NOT ensure physically meaningful
! perturbed fields.  This is best done outside this subroutine
! after the perturbations have been applied.

real, dimension(N_pert,N_x,N_y,N_ens), intent(out) :: Pert

! If initialize_rseed==.true., set initial random seed vector.
! If initialize_rseed==.true., the first row of Pert_rseed must be
! filled with a different negative integer for each ensemble member.
! Note that in parallel applications initial Pert_rseed must also differ
! for each subdomain.  See sample subroutine get_init_Pert_rseed().
! If initialize_ntrmdt==.true., generate initial Pert_ntrmdt (must be
```

```
! allocated!!!).
! Note that when get_pert() is used for generating independent
! perturbations to forcings and prognostic variables, only one
! common Pert_rseed should be used, so one of the "ntrmdt" fields
! must be initialized without initializing "rseed" again.

! If initialize_*==.false. or absent, Pert_rseed and
! Pert_ntrmdt must be what was obtained as output from the last call
! to get_pert().  There is only one exception: if there are no temporal
! correlations, it is not necessary to remember Pert_ntrmdt.

logical, intent(in), optional :: initialize_rseed
logical, intent(in), optional :: initialize_ntrmdt
```

DESCRIPTION:

get perturbations
reichle, 11 Feb 2005
N_pert is the number of *perturbation* fields and is not necessarily equal to the number of forcing fields. E.g. generate N_pert=3 perturbations for precip, shortwave radiation, and longwave radiation. These can then be applied to N_force forcing fields, possibly various precip fields (incl large-scale & convective precip and snow) and to radiation fields.

F.3.5 propagate_pert (Source File: landpert_routines.F90)

INTERFACE:

```
subroutine propagate_pert(                      &
     N_pert, N_ens, loc_grid, dtstep,           &
     Pert_rseed,                                &
     pert_param,                                &
     Pert_ntrmdt,                               &
     initialize              )

  implicit none
```

ARGUMENTS:

```
integer, intent(in) :: N_ens, N_pert

type(grid_def_type), intent(in) :: loc_grid   ! local grid definition

type(pert_param_type), dimension(N_pert), intent(in) :: pert_param

real, intent(in) :: dtstep  ! time step of generation of error fields [s]

integer, dimension(NRANDSEED,N_ens), intent(inout) :: Pert_rseed

real, dimension(N_pert,loc_grid%N_x,loc_grid%N_y,N_ens), &
     intent(inout) :: Pert_ntrmdt

logical, intent(in) :: initialize   ! switch
```

DESCRIPTION:

generate zero-mean, unit-variance (!!) time series of N_pert 2d perturbation fields
can also be used just to get a set of 2d random fields (set dtstep to arbitrary number and "initialize" to .true.)
on input, Pert_ntrmdt must contain the corresponding perturbations from the previous time step
accounts for temporal correlation with AR(1) approach (if pert_param%tcorr==0 then error is white in time)
adapted from off-line EnKF, subroutine propagate_err() from NCAT_59124_tskin in enkf_catchment.f90
reichle, 14 Feb 2005

F.3.6 truncate_std_normal (Source File: landpert_routines.F90)

INTERFACE:

```
subroutine truncate_std_normal( N_x, N_y, std_normal_max, grid_data )

  implicit none
```

ARGUMENTS:

```
  integer, intent(in) :: N_x, N_y

  real, intent(in) :: std_normal_max

  real, dimension(N_x,N_y), intent(inout) :: grid_data
```

DESCRIPTION:

truncate a realization of standard normal variables (scale back freak outliers)

F.3.7 adjust_mean (Source File: landpert_routines.F90)

INTERFACE:

```
subroutine adjust_mean( N_row, N_col, A, M )

  implicit none
```

ARGUMENTS:

```
  integer, intent(in) :: N_row, N_col
  real, intent(inout), dimension(N_row,N_col)  :: A
  real, intent(in), optional, dimension(N_row) :: M
```

DESCRIPTION:

adjust N_row by N_col matrix A such that mean over columns for each row is given by the corresponding element in vector M of length N_row
vector of mean values M is optional input, if not present zero mean is assumed

---

F.4 Fortran: Module Interface nr_jacobi.F90 (Source File: nr_jacobi.F90)

This module performs the Jacobi transformation. Adopted from f77 Numerical Recipes.

REVISION HISTORY:

```
07Jun05 Rolf Reichle  Initial Specification
07Jul05 Sujay Kumar   Specification in LIS
```

F.4.1 jacobi (Source File: nr_jacobi.F90)

INTERFACE:

```
subroutine jacobi(amat,n,d,v,nrot)
```

ARGUMENTS:

```
integer,                  intent(in)               :: n
real,    dimension(n,n),  intent(in)               :: amat
real,    dimension(n,n),  intent(out)              :: v
real,    dimension(n),    intent(out)              :: d
integer,                  intent(out), optional    :: nrot
```

DESCRIPTION:

Computes all eigenvalues and eigenvectors of a real symmetric matrix amat, which is of size n by n. The vector d returns the eigenvalues of amat in its first n elements. v is a matrix with the same dimensions as amat, whose columns contain, on output, the normalized eigenvectors of amat. nrot returns the number of Jacobi rotations that were required.
The Jacobi method is absolutely foolproof for all real symmetric matrices. For matrices of order greater than about 10, say, the algorithm is slower, by a significant constant factor, than the QR method. However, the Jacobi algorithm is much simpler than the more efficient methods. We thus recommend it for matrices of moderate order, where expense is not a major consideration.
Eigenvector decomposition:
Amat = V*diag(D)*transpose(V)

F.5 Fortran: Module Interface nr_ran2_gasdev.F90 (Source File: nr_ran2_gasdev.F90)

adapted Numerical Recipes random number generator ran2() and gasdev()
use ran2() instead of rand() and/or ran1() for longer period (1e18)
gasdev() uses ran2() instead of ran1()
eliminate all save attributes, convert functions into subroutines, pass seed state vector into subroutines, enclose in module

REVISION HISTORY:

```
16Feb05 Rolf Reichle  Initial Specification
18Feb05 Rolf Reichle  make nr_ran2() public, change call statement
07Jul05 Sujay Kumar   Specification in LIS
```

F.6 Fortran: Module Interface random_fields.F90 (Source File: random_fields.F90)

This module contains a random field generator in 2d to generate a pair of random fields in 2d with zero mean
subroutines rfg2d_fft() and sqrt_gauss_spectrum are translated from C++ code rfg2d.C written for MIT EnKF work by reichle (see janus: reichle/nasa/EnKF)
covariance is specified through its spectrum, so far only Gaussian
IMPORTANT: read comments for function rfg2d_fft()
written for NSIPP - EnKF Type: f90 Author: Rolf Reichle Date: 2 Nov 2001

REVISION HISTORY:

```
18Feb05 Rolf Reichle  Initial Specification
                      updated for use with module nr\_ran2\_gasdev
                      deleted use of module select\_kinds
07Jul05 Sujay Kumar   Specification in LIS
```

F.6.1 sqrt_gauss_spectrum_2d (Source File: random_fields.F90)

INTERFACE:

```
subroutine sqrt_gauss_spectrum_2d( &
     variance, N_x, N_y, dkx, dky, lambda_x, lambda_y, &
     sqrt_spectrum_2d )

  implicit none
```

ARGUMENTS:

```
  real, intent(in) :: variance, dkx, dky, lambda_x, lambda_y
  integer, intent(in) :: N_x, N_y
  real, intent(out), dimension(N_x,N_y) :: sqrt_spectrum_2d
```

DESCRIPTION:

get SQUARE ROOT of 2d Gaussian spectrum (incl volume element)
2d Gaussian spectrum:

$$S(kx, ky) = variance * lambda_x * lambda_y/(2 * pi) * exp(-(kx^2 * lambda_x^2 + ky^2 * lambda_y^2)/2)$$

return: sqrt(S*dkx*dky)
that is return the SQUARE ROOT of the spectrum multiplied with the square root of the volume element d2k=dkx*dky of the ifft integral
spectrum is returned in "wrap-around" order compatible with CXML and matlab FFT algorithms
inputs: variance : variance desired for complex field, if pair of real fields is used each field must eventually be multiplied with sqrt(2) N_x : number of nodes in x direction N_y : number of nodes in y direction dkx : wave number spacing in x direction dky : wave number spacing in y direction lambda_x : decorrelation length in x direction lambda_y : decorrelation length in y direction

---

F.6.2 rfg2d_fft (Source File: random_fields.F90)

INTERFACE:

```
   subroutine rfg2d_fft( &
        rseed, variance, N_x, N_y, dx, dy, lambda_x, lambda_y, &
        field1, field2 )

     implicit none

     ! use Compaq Extended Math Library (CXML)
     ! (free with Compaq Fortran Compiler)

 #ifdef CXML_FFT_AVAILABLE
     include 'CXMLDEF.FOR'
 #endif
```

ARGUMENTS:

```
     integer, dimension(NRANDSEED), intent(inout) :: rseed
     real, intent(in) :: variance, dx, dy, lambda_x, lambda_y
     integer, intent(in) :: N_x, N_y
     real, intent(out), dimension(N_x,N_y) :: field1, field2
```

DESCRIPTION:

generate a pair of 2d zero-mean random fields using FFT method (so far only Gaussian covariance implemented)
NOTE: implemented with index counters of type int, must have
N_x*N_y ¡ maximum integer on given machine
on yama/alborz (at MIT) int varies from -2147483648...2147483647 -¿ can handle up to $N_x * N_y = (32768)^2$
if larger fields are needed, rewrite with type long int etc
NOTE: The fft method for generating random fields produces fields that are periodic with the size of the domain, that is the field is (almost) the same on each boundary. This introduces unwanted correlations at lags shorter than the domain size. Therefore, only a part of the generated field is usable. As a rule of thumb, the fields should be generated on a grid that is two correlation lenghts bigger than the field on which the grid is desired. Then cut out fields of the necessary size. This procedure is included in rfg2d_fft().
NOTE: The variance specified as input is the theoretical variance of the complex field that is obtained from the inverse fft of the realization dZ. The sample variance of this *complex* field is a FIXED (non-random) number which depends on the size of the domain, the grid spacing, and the correlation length (but not on the random seed!!). (This number is non-random because the variance is the integral of the absolute value of the spectrum. In this integral the randomness disappears because we only choose a random phase angle.) For vanishing discretization and spectral truncation error, this number converges to the theoretical (input) value.
This function is set up to use the pair of two real fields, where
field1 = sqrt(2)*real(ifft(dZ)) field2 = sqrt(2)*imag(ifft(dZ)).
The factor sqrt(2) re-scales the variance of field1 and field2 such that for vanishing discretization error and spectral truncation error each field converges to the specified theoretical (input) variance. NOTE: The sum of the sample variances of the two real fields is equal to the (FIXED) sample variance of the complex field (before re-scaling with sqrt(2)). The individual sample variances within each pair vary from realization to realization.

F.6.3 get_fft_grid (Source File: random_fields.F90)

INTERFACE:

```
   subroutine get_fft_grid( N_x, N_y, dx, dy, lambda_x, lambda_y, &
        N_x_fft, N_y_fft )
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: N_x, N_y
real, intent(in) :: dx, dy, lambda_x, lambda_y
integer, intent(out) :: N_x_fft, N_y_fft
```

DESCRIPTION:

get a grid that can be used for 2d fft random number generator (this fft grid must extend beyond the desired random field by about two correlations lengths, it must also have N_x_fft and N_y_fft that are powers of two)

F.6.4 generate_white_field (Source File: random_fields.F90)

INTERFACE:

```
subroutine generate_white_field(N_x, N_y, rseed, rfield )

  implicit none
```

ARGUMENTS:

```
integer, intent(in) :: N_x, N_y
integer, dimension(NRANDSEED), intent(inout) :: rseed
real, dimension(N_x,N_y), intent(out) :: rfield
```

DESCRIPTION:

generate standard-normal random field that is white in space
note that nr_gasdev always produces a pair of random numbers
do not store random numbers between subsequent calls to the random field generator - works best if fields are large (ie. avoid using this subroutine with N_x=N_y=1)

Part XIII
Observation data handling for assimilation in LIS

G Observation data handling

This section contains the description of observational data handling for assimilation in LIS.

H Synthetic Soil Moisture from Noah

H.1 Fortran: Module Interface syntheticsm1obs_module (Source File: syntheticsm1obs_module.F90)

This module contains interfaces and subroutines to handle synthetic soil moisture observations generated from a previous LIS-Noah simulation

REVISION HISTORY:

```
    27Feb05    Sujay Kumar;   Initial Specification

 module syntheticsm1obs_module
```

USES:

```
   use ESMF_Mod
   use lisdrv_module, only : lis
```

H.1.1 syntheticsm1obs_setup (Source File: syntheticsm1obs_module.F90)

INTERFACE:

```
   subroutine syntheticsm1obs_setup(OBS_State, OBS_Pert_State)
```

USES:

```
     use ESMF_Mod
     use lisdrv_module, only : lis, config_lis
     use lis_logmod, only : logunit

     implicit none
```

ARGUMENTS:

```
     type(ESMF_State)       ::  OBS_State(lis%nnest)
     type(ESMF_State)       ::  OBS_Pert_State(lis%nnest)
```

DESCRIPTION:

This routine completes the runtime initializations and creation of data strctures required for soil moisture assimilation
The arguments are:

OBS_State observation state

OBS_Pert_State observation perturbations state

H.1.2 read_syntheticsm1obs (Source File: read_syntheticsm1obs.F90)

REVISION HISTORY:

```
21Jun2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_syntheticsm1obs(n, OBS_State)
```

USES:

```
use ESMF_Mod
use spmdMod
use drv_output_mod, only : drv_readvar_gridded
use lisdrv_module,  only : lis, lisdom
use lis_logmod,     only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
type(ESMF_State)    :: OBS_State
```

DESCRIPTION:

reads the synthetic soil moisture observations produced from a LIS control run and packages it into an ESMF State with certain predefined attributes
The arguments are:

n index of the nest

OBS_State observations state

H.1.3 getNSO_synsm1.F90 (Source File: getNSO_synsm1.F90)

Returns the number of selected observations (NSO) corresponding to the grid point specified

REVISION HISTORY:

```
21Jun2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine getNSO_synsm1(n, gid,st_id,en_id)

implicit none
```

DESCRIPTION:

Returns the number of selected soil moisture observations (NSO) corresponding to the selected modeling point
The arguments are:

n index of the nest

gid index of the modeling point

st_id starting index of the selected observations

en_id ending index of the selected observations

I Synthetic Soil Moisture from Catchment

I.1 Fortran: Module Interface syntheticsm2obs_module (Source File: syntheticsm2obs_module.F90)

This module contains interfaces and subroutines to handle synthetic soil moisture observations produced from a LIS catchment simulation.

REVISION HISTORY:

```
27Feb05    Sujay Kumar;   Initial Specification
```

```
module syntheticsm2obs_module
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
```

I.1.1 syntheticsm2obs_setup (Source File: syntheticsm2obs_module.F90)

INTERFACE:

```
subroutine syntheticsm2obs_setup(OBS_State, OBS_Pert_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis, config_lis
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
type(ESMF_State)       ::  OBS_State(lis%nnest)
type(ESMF_State)       ::  OBS_Pert_State(lis%nnest)
```

DESCRIPTION:

This routine completes the runtime initializations and creation of data strctures required for soil moisture assimilation
The arguments are:

OBS_State observation state

OBS_Pert_State observation perturbations state

I.1.2 read_syntheticsm2obs (Source File: read_syntheticsm2obs.F90)

REVISION HISTORY:

```
21Jun2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_syntheticsm2obs(n, OBS_State)
```

USES:

```
use ESMF_Mod
use spmdMod
use drv_output_mod, only : drv_readvar_gridded
use lisdrv_module,  only : lis, lisdom
use lis_logmod,     only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
type(ESMF_State)    :: OBS_State
```

DESCRIPTION:

reads the synthetic soil moisture observations produced from a LIS catchment control run.
The arguments are:

n index of the nest

OBS_State observations state

I.1.3 getNSO_synsm2.F90 (Source File: getNSO_synsm2.F90)

Returns the number of selected observations (NSO) corresponding to the grid point specified

REVISION HISTORY:

```
21Jun2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine getNSO_synsm2(n, gid,st_id,en_id)

implicit none
```

DESCRIPTION:

Returns the number of selected soil moisture observations (NSO) corresponding to the selected modeling point
The arguments are:

n index of the nest

gid index of the modeling point

st_id starting index of the selected observations

en_id ending index of the selected observations

J Synthetic Snow Cover Area

J.1 Fortran: Module Interface syntheticscaobs_module (Source File: syntheticscaobs_module.F90)

This module contains interfaces and subroutines to handle synthetic snow cover observations.

REVISION HISTORY:

```
27Feb05    Sujay Kumar;   Initial Specification
```

```
module syntheticscaobs_module
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
```

J.1.1 syntheticscaobs_setup (Source File: syntheticscaobs_module.F90)

INTERFACE:

```
subroutine syntheticscaobs_setup(OBS_State, OBS_Pert_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis, config_lis
use lis_logmod, only : logunit
```

ARGUMENTS:

```
type(ESMF_State)       ::  OBS_State(lis%nnest)
type(ESMF_State)       ::  OBS_Pert_State(lis%nnest)
```

DESCRIPTION:

This routine completes the runtime initializations and creation of data strctures required for SWE assimilation
The arguments are:

OBS_State observation state

OBS_Pert_State observation perturbations state

J.1.2 read_syntheticscaobs (Source File: read_syntheticscaobs.F90)

REVISION HISTORY:

```
21Jun2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_syntheticscaobs(n, OBS_State)
```

USES:

```
use ESMF_Mod
use spmdMod
use drv_output_mod, only : drv_readvar_gridded
use lisdrv_module,  only : lis, lisdom
use lis_logmod,     only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
type(ESMF_State)    :: OBS_State
```

DESCRIPTION:

reads the synthetic SWE observations produced from a LIS control run.
The arguments are:

n index of the nest

OBS_State observations state

J.1.3 getNSO_synsca (Source File: getNSO_synsca.F90)

REVISION HISTORY:

```
21Jun2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine getNSO_synsca(n, gid,st_id,en_id)

implicit none
```

ARGUMENTS:

```
integer,   intent(IN)    :: n
integer,   intent(IN)    :: gid
integer,   intent(OUT)   :: st_id
integer,   intent(OUT)   :: en_id
```

DESCRIPTION:

Returns the number of selected SWE observations (NSO) corresponding to the selected modeling point
The arguments are:

n index of the nest

gid index of the modeling point

st_id starting index of the selected observations

en_id ending index of the selected observations

K Synthetic Snow Water Equivalent

K.1 Fortran: Module Interface syntheticswe1obs_module (Source File: syntheticswe1obs_module.F90)

This module contains interfaces and subroutines to handle synthetic soil moisture observations.

REVISION HISTORY:

```
27Feb05    Sujay Kumar;   Initial Specification
```

```
module syntheticswe1obs_module
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
```

K.1.1 syntheticswe1obs_setup (Source File: syntheticswe1obs_module.F90)

INTERFACE:

```
subroutine syntheticswe1obs_setup(OBS_State, OBS_Pert_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis, config_lis
use lis_logmod, only : logunit
```

ARGUMENTS:

```
type(ESMF_State)       ::  OBS_State(lis%nnest)
type(ESMF_State)       ::  OBS_Pert_State(lis%nnest)
```

DESCRIPTION:

This routine completes the runtime initializations and creation of data strctures required for SWE assimilation
The arguments are:

OBS_State observation state

OBS_Pert_State observation perturbations state

K.1.2 read_syntheticswe1obs (Source File: read_syntheticswe1obs.F90)

REVISION HISTORY:

```
21Jun2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine read_syntheticswe1obs(n, OBS_State)
```

USES:

```
use ESMF_Mod
use spmdMod
use drv_output_mod, only : drv_readvar_gridded
use lisdrv_module,  only : lis, lisdom
use lis_logmod,     only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
type(ESMF_State)    :: OBS_State
```

DESCRIPTION:

reads the synthetic SWE observations produced from a LIS control run.
The arguments are:

n index of the nest

OBS_State observations state

K.1.3 getNSO_synswe1 (Source File: getNSO_synswe1.F90)

REVISION HISTORY:

```
21Jun2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine getNSO_synswe1(n, gid,st_id,en_id)

implicit none
```

ARGUMENTS:

```
integer,   intent(IN)    :: n
integer,   intent(IN)    :: gid
integer,   intent(OUT)   :: st_id
integer,   intent(OUT)   :: en_id
```

DESCRIPTION:

Returns the number of selected SWE observations (NSO) corresponding to the selected modeling point
The arguments are:

n index of the nest

gid index of the modeling point

st_id starting index of the selected observations

en_id ending index of the selected observations

Part XIV
Land Surface Models in LIS

L Land Surface Models in LIS

This section contains the description of interface implementations specific to different land surface models. The implementations include Noah LSM from NCEP, the community land model (CLM), Mosaic, HySSIB and catchment from NASA GSFC, etc. It is to be noted that the LIS reference manual documents only the interface hooks into LIS and it does not document the land surface physics routines. Please contact the developers of the individual model for same.

M Noah land surface model version 2.7.1

This section describes the interface implementations for the Noah land surface model. The community Noah Land Surface Model is a stand-alone, uncoupled, 1-D column model freely available at the National Centers for Environmental Prediction (NCEP; [2]). The name is an acronym representing the various developers of the model (N: NCEP; O: Oregon State University, Dept. of Atmospheric Sciences; A: Air Force (both AFWA and AFRL - formerly AFGL, PL); and H: Hydrologic Research Lab - NWS (now Office of Hydrologic Development – OHD)). Noah can be executed in either coupled or uncoupled mode. It has been coupled with the operational NCEP mesoscale Eta model [7] and its companion Eta Data Assimilation System (EDAS) [17], and the NCEP Global Forecast System (GFS) and its companion Global Data Assimilation System (GDAS). When Noah is executed in uncoupled mode, near-surface atmospheric forcing data (e.g., precipitation, radiation, wind speed, temperature, humidity) is required as input. Noah simulates soil moisture (both liquid and frozen), soil temperature, skin temperature, snowpack depth, snowpack water equivalent, canopy water content, and the energy flux and water flux terms of the surface energy balance and surface water balance. The model applies finite-difference spatial discretization methods and a Crank-Nicholson time-integration scheme to numerically integrate the governing equations of the physical processes of the soil vegetation-snowpack medium, including the surface energy balance equation, Richards' [15] equation for soil hydraulics, the diffusion equation for soil heat transfer, the energy-mass balance equation for the snowpack, and the Jarvis [9] equation for the conductance of canopy transpiration.

M.1 Fortran: Module Interface noah_module (Source File: noah_module.F90)

The code in this file provides a description of the data structure containing the noah 1-d variables. The variables specified in the data structure include:

soiltype soil type (integer index)

nroot Number of root layers, depends on the vegetation type

rsmin Minimum canopy resistance (s/m)

rgl Solar radiation term of canopy resistance function

hs vapor pressure deficit term of canopy resistance function

snup threshold snow depth (in water equivalent m) that implies 100% snow cover

z0 roughness length

lai leaf area index

vegip interpolated vegetation fraction

vegmp1 month 1 greenness fraction value

vegmp2 month 2 greenness fraction value

albsf1 date 1 snow free albedo value

albsf2 date 2 snow free albedo value

albsf interpolated snow free albedo value

smcmax maximum soil moisture content (porosity)

psisat saturated matric potential

dksat saturated hydraulic conductivity

bexp b parameter value

quartz quartz fraction

smcwlt wilting point (volumetric)

smcref reference soil moisture (onset of soil moisture stress in transpiration, volumetric)

dwsat saturated soil diffusivity

mxsnoalb maximum albedo expected over deep snow

tempbot bottom boundary temperature

tempbot1 month 1 bottom boundary temperature

tempbot2 month 2 bottom boundary temperature

t1 skin temperature (K)

cmc canopy water content

snowh actual snow depth (m)

sneqv snow water equivalent (m)

stc soil temperature for different layers

smc soil moisture for different layers

sh2o liquid-only soil moisture for different layers

ch heat/moisture exchange coefficient

cm momentum exchange coefficient

vegt vegetation type of tile

swnet net shortwave radiation (W/m2)

lwnet net longwave radiation (W/m2)

qle latent heat flux (W/m2)

qh sensible heat flux (W/m2)

qg ground heat flux (W/m2)

snowfall snowfall (kg/m2s)

pcp rainfall (kg/m2s)

evap evapotranspiration (kg/m2s)

qs surface runoff (kg/m2s)

qsb subsurface runoff (kg/m2s)

qsm snow melt (kg/m2s)

avgsurft average surface temperature (K)

albedo surface albedo (-)

swe snow water equivalent (kg/m2)

soilmoist soil moisture for different layers (kg/m2)

soilwet total column soil wetness (-)

ecanop interception evaporation (kg/m2s)

tveg vegetation transpiration (kg/m2s)

canopint total canopy water storage (kg/m2s)

rootmoist root zone soil moisture (kg/m2)

soilm_prev soil moisture from the previous model output

swe_prev snow water equivalent from the previous model output

tair 2m air temperature forcing

qair 2m specific humidity forcing

swdown downward shortwave forcing

lwdown downward longwave forcing

uwind u-wind component forcing

vwind v-wind component forcing

psurf surface pressure forcing

rainf total rainfall forcing

rainf_c convective rainfall forcing

snowf total snowfall forcing

REVISION HISTORY:

```
28 Apr 2002: K. Arsenault added NOAH LSM 2.5 code to LDAS.
14 Nov 2002: Sujay Kumar Optimized version for LIS
```

M.2 Fortran: Module Interface noah_varder (Source File: noah_varder.F90)

This module provides the definition of derived data type used to control the operation of Noah LSM. It also provides the entry method for the initialization of Noah-specific variables. The derived data type `noah_struc` includes the variables that specify the runtime options and other control variables as described below:

rfile name of the noah restart file

vfile name of the static vegetation parameter table

sfile name of the soil parameter table

pfile name of the general parameter table

count variable to keep track of the number of timesteps before an output

noahopen variable to keep track of opened files

soilscheme type of soil mapping scheme usd (1-zobler,2-statsgo)

numout number of output times

nslay number of soil layers

nvegp number of static vegetation parameters in the table

nsoilp number of soil parameters in the table

nstxts number of soil texture classes in the classification scheme

npr number of prognostic state variables used in data assimilation

varid NETCDF ids for variables (used for netcdf output)

tbotAlarmTime alarm time to keep track of TBOT data reading frequency

tbotInterval data interval of the TBOT data

initsm initial soil moisture for a cold start run

initTemp initial soil temperature for a cold start run

outInterval output writing interval

rstInterval restart writing interval

obsz height of meteorological observations

lyrthk thickness of soil layers

noah noah LSM specific variables

REVISION HISTORY:

```
Apr 2003; Sujay Kumar, Initial Code
```

USES:

```
use noah_module

implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
---------------------------------------------------------------------------
 public :: noah_varder_ini
---------------------------------------------------------------------------
```

PUBLIC TYPES:

```
---------------------------------------------------------------------------
 public :: noah_struc
```

M.2.1 noah_varder_ini (Source File: noah_varder.F90)

INTERFACE:

```
subroutine noah_varder_ini()
```

USES:

```
  use lisdrv_module, only : lis
```

DESCRIPTION:

This routine creates the datatypes and allocates memory for noah-specific variables. It also invokes the routine to read the runtime specific options for noah from the configuration file.
The routines invoked are:

readnoahcrd (M.2.2)
reads the runtime options for Noah LSM

M.2.2 readnoahcrd (Source File: readnoahcrd.F90)

REVISION HISTORY:

```
14 Oct 2003; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine readnoahcrd()
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis, config_lis
use spmdMod, only : masterproc
use noah_varder, only : noah_struc
use lis_logmod, only :logunit
```

DESCRIPTION:

This routine reads the options specific to Noah LSM from the LIS configuration file.

M.2.3 noah_setvegparms (Source File: noah_setvegparms.F90)

REVISION HISTORY:

```
28 Apr 2002: Kristi Arsenault;  Added NOAH LSM, Initial Code
13 Oct 2003: Sujay Kumar; Domain independent modifications
25 Jul 2005: Sujay Kumar; Removed the dependency to SIB classes
04 Oct 2005: Matthew Garcia; additions for Noah Unified
04 Oct 2005: Matthew Garcia; additions for Distributed Noah Router
```

INTERFACE:

```
subroutine noah_setvegparms
```

USES:

```
use lisdrv_module, only : lis,lisdom
use noah_varder
use spmdMod, only : iam,masterproc
```

DESCRIPTION:

This subroutine retrieves NOAH vegetation parameters. The current implementation uses a table-based lookup based on SIB classes to initialize the following parameters

```
nroot  - number of root zones
rsmin  - minimum canopy resistance
rgl    - solar radiation term in canopy resistance function
hs     - vapor pressure deficit term
snup   - threshold snow depth
z0     - roughness length
lai    - leaf area index
```

M.2.4 noah_setup (Source File: noah_setup.F90)

REVISION HISTORY:

```
28 Apr. 2002: Sujay Kumar; Initial version
```

INTERFACE:

```
subroutine noah_setup()
```

USES:

```
use lisdrv_module, only: lis
use noah_varder
use spmdMod, only : masterproc, npes
use listime_mgr, only : setMonthlyAlarm
```

DESCRIPTION:

This routine is the entry point to set up the parameters required for Noah LSM. These include the soils, greenness, albedo, bottom temperature and the initialization of state variables in Noah.
The routines invoked are:

noah_setvegparms (M.2.3)
initializes the vegetation-related parameters in Noah

noah_settbot (M.2.6)
initializes the bottom temperature fields

noah_setsoils (M.2.5)
initializes the soil parameters

noah_coldstart (M.2.7)
initializes the noah state variables

setMonthlyAlarm (2.4.13)
sets a monthly alarm if a climatology of bottom temperature is being read

---

M.2.5 noah_setsoils (Source File: noah_setsoils.F90)

REVISION HISTORY:

```
28 Apr 2002: Kristi Arsenault;  Added NOAH LSM, Initial Code
13 Oct 2003: Sujay Kumar; Domain independent modifications
15 Sep 2005: Matthew Garcia; update for ARMS project (Windows capability)
04 Oct 2005: Matthew Garcia; additions for Noah Unified
```

INTERFACE:

```
subroutine noah_setsoils
```

USES:

```
use lisdrv_module, only : lis, lisdom
use noah_varder
use spmdMod, only : iam,masterproc
use soils_module, only : lis_soils, soils_finalize
use lis_logmod, only :logunit
```

DESCRIPTION:

This subroutine sets the soil parameters in Noah. Noah uses a lookup table based on the input soil texture to derive these parameters.
The routines invoked are:

soiltype (2.29.3)
method to derive the soil texture type from the sand, silt, and clay fractions.

M.2.6 noah_settbot (Source File: noah_settbot.F90)

REVISION HISTORY:

```
28 Apr 2002: Kristi Arsenault;  Added NOAH LSM, Initial Code
13 Oct 2003: Sujay Kumar; Domain independent modifications
```

INTERFACE:

```
subroutine noah_settbot
```

USES:

```
use lisdrv_module, only : lis, lisdom
use noah_varder
use spmdMod, only : iam
use listime_mgr, only : isMonthlyAlarmRinging, computeMonthlyWeights
```

DESCRIPTION:

This subroutine retrieves bottom boundary temperature for Noah LSM. Currently a static data is being used. If a climatology is available, it can be used by setting the appropriate frequency of the climatology The routines invoked are:

readtbot (2.44.2)
retrieves the bottom temperature data

M.2.7 noah_coldstart (Source File: noah_coldstart.F90)

INTERFACE:

```
subroutine noah_coldstart()
```

USES:

```
use lisdrv_module, only: lis
use noah_varder
use listime_mgr
use spmdMod, only: iam,masterproc
use lis_logmod, only : logunit
```

DESCRIPTION:

This routine initializes the noah state variables with some predefined values uniformly for the entire domain. These initial values will be overwritten by the values read from the supplied Noah model restart file.

M.2.8 noah_dynsetup (Source File: noah_dynsetup.F90)

REVISION HISTORY:

```
28 Jan 2002: Sujay Kumar, Initial Specification
```

INTERFACE:

```
subroutine noah_dynsetup(n)
```

DESCRIPTION:

This routine sets up the time-dependent variables in Noah. Currently this routine is a placeholder

---

M.2.9 noah_f2t (Source File: noah_f2t.F90)

REVISION HISTORY:

```
15 Oct 2002: Sujay Kumar; Initial Code
```

INTERFACE:

```
subroutine noah_f2t(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module , only : lis
use baseforcing_module, only : FORC_State
use noah_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine transfers the LIS provided forcing onto the Noah model tiles.
The arguments are:

n index of the nest

M.2.10 noah_main (Source File: noah_main.F90)

INTERFACE:

```
subroutine noah_main(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use gfrac_module,  only : lis_gfrac
use albedo_module, only : lis_alb
use constantsMod,  only : CONST_RHOFW, CONST_TKFRZ
use noah_varder
use lis_logmod,    only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)  :: n
```

DESCRIPTION:

This is the entry point for calling the Noah LSM physics. This routine calls the `SFLX` routine that performs the land surface computations, to solve for water and energy equations. For documentation of the `SFLX` and other noah routines, please see the "NOAH LSM USERS GUIDE" maintained at the public server at NOAA NCEP.
The arguments are:

n index of the nest

M.2.11 noah_output (Source File: noah_output.F90)

INTERFACE:

```
subroutine noah_output(n)
```

USES:

```
#if ( defined USE_NETCDF )
  use netcdf
#endif
  use lisdrv_module, only : lis, metadata_output
  use noah_varder
  use spmdMod
  use lis_fileIOMod, only : create_output_directory, &
       create_output_filename,  &
       create_stats_filename

  implicit none

  integer, intent(in) :: n
```

DESCRIPTION:

This subroutines sets up methods to write model output from Noah LSM A timestamped filename and directory are created first. The routine supports model output in three data formats: Binary, Grib and NETCDF.
The arguments are:

n index of the nest

The routines invoked are:

create_output_directory (2.25.2)
creates a time stamped output directory

create_output_filename (2.25.3)
creates a time stamped output filename

create_stats_filename (2.25.5)
creates a stats filename

noah_binout (M.2.12)
routine to write Noah model output in binary format

noah_gribout (M.2.13)
routine to write Noah model output in grib format

noah_netcdfout (M.2.14)
routine to write Noah model output in netcdf format

noah_totint (M.2.15)
resets the time averaged variables

M.2.12 noah_binout (Source File: noah_binout.F90)

REVISION HISTORY:

```
02 Dec 2003; Sujay Kumar, Initial Version
```

INTERFACE:

```
subroutine noah_binout(ftn,ftn_stats,n)
```

USES:

```
use lisdrv_module, only : lis,metadata_output
use drv_output_mod, only : drv_writevar_bin
use constantsMod,   only : CONST_TKFRZ, CONST_LATVAP
use noah_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn,ftn_stats
```

DESCRIPTION:

This routine writes the specified noah variables into a binary output file.
The arguments are:

n index of the nest

ftn unit number for the model output file

ftn_stats unit number for the stats file

The routines invoked are:

drv_writevar_bin (2.22.2)
routine to write a variable to a binary output file

M.2.13 noah_gribout (Source File: noah_gribout.F90)

REVISION HISTORY:

```
02 Dec 2003; Sujay Kumar, Initial Version
29 Dec 2006; Charles Alonge, Added Parameter Output and additional
             grib information (projection, time range and intervals)
```

INTERFACE:

```
subroutine noah_gribout(ftn,ftn_stats,n)
```

USES:

```
use lisdrv_module, only : lis,lisdom,metadata_output
use drv_output_mod, only : drv_writevar_grib
use noah_varder
use constantsMod,   only : CONST_TKFRZ, CONST_LATVAP
use spmdMod, only : masterproc
use grib_module

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn,ftn_stats
```

DESCRIPTION:

This routine writes the specified noah variables into a grib output file.
The arguments are:

n index of the nest

ftn unit number for the model output file

ftn_stats unit number for the stats file

The routines invoked are:

drv_writevar_grib (2.22.6)
routine to write a variable to a grib output file

grib_init (2.24.1)
routine to initialize grib data structures

grib1_setup (2.24.3)
routine to initialize Section 1 (PDS) of grib record

grib2_setup (2.24.4)
routine to initialize Section 2 (GDS) of grib record

grib3_setup (2.24.5)
routine to initialize Section 3 (BMS) of grib record

grib1_finalize (2.24.2)
routine to complete Section 1 (PDS) of grib record

M.2.14 noah_netcdfout (Source File: noah_netcdfout.F90)

REVISION HISTORY:

```
02 Dec 2003; Sujay Kumar, Initial Version
```

INTERFACE:

```
subroutine noah_netcdfout(check, ftn,n)
```

USES:

```
#if (USE_NETCDF)
  use netcdf
#endif
  use lisdrv_module, only : lis
  use drv_output_mod, only : drv_writevar_netcdf
  use noah_varder
  use lis_logmod, only :logunit

  implicit none
```

ARGUMENTS:

```
  integer, intent(in) :: n
  integer, intent(in) :: ftn
  logical, intent(in) :: check
```

DESCRIPTION:

This routine writes the specified noah variables into a netcdf output file.
The arguments are:

n index of the nest

ftn unit number for the model output file

check flag to determine whether to write the header information

The routines invoked are:

drv_writevar_netcdf (2.22.7)
routine to write a variable to a netcdf output file

M.2.15 noah_totinit (Source File: noah_totinit.F90)

REVISION HISTORY:

```
14 Jun 2002  Sujay Kumar  Initial Specification
```

INTERFACE:

```
subroutine noah_totinit(n)
```

USES:

```
  use noah_varder
  use lisdrv_module, only : lis

  implicit none

  integer, intent(in) :: n
```

DESCRIPTION:

This routine resets the time-averaged Noah variables after a model output. The routine is called after a model output call.

M.2.16 noah_writerst (Source File: noah_writerst.F90)

REVISION HISTORY:

```
05 Sep 2001: Brian Cosgrove; Modified code to use Dag Lohmann's NOAA
             initial conditions if necessary.  This is controlled with
             local variable NOAAIC.  Normally set to 0 in this subroutine
             but set to 1 if want to use Dag's NOAA IC's.  Changed output
             directory structure, and commented out if-then check so that
             directory is always made.
28 Apr 2002: Kristi Arsenault; Added NOAH LSM into LDAS
28 May 2002: Kristi Arsenault; For STARTCODE=4, corrected SNEQV values
              and put SMC, SH2O, STC limit for GDAS and GEOS forcing.
14 Jun 2003: Sujay Kumar , Separated the write restart from the original
              code
```

INTERFACE:

```
subroutine noah_writerst(n)
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr
use noah_varder
use lis_logmod, only :logunit
use lis_fileIOMod, only : create_output_directory, &
                          create_restart_filename
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This program writes restart files for Noah. This includes all relevant water/energy storage and tile information
The routines invoked are:

create_output_directory (2.25.2)
creates a timestamped directory for the restart files

create_restart_filename (2.25.4)
generates a timestamped restart filename

noah_dump_restart (M.2.17)
writes the noah variables into the restart file

M.2.17 noah_dump_restart (Source File: noah_writerst.F90)

REVISION HISTORY:

```
24 Aug 2004: James Geiger, Initial Specification
```

INTERFACE:

```
subroutine noah_dump_restart(n, ftn)
```

USES:

```
use lisdrv_module, only : lis
use noah_varder
use listime_mgr
use drv_output_mod, only : drv_writevar_restart

implicit none

integer, intent(in) :: ftn
integer, intent(in) :: n
```

DESCRIPTION:

This routine gathers the necessary restart variables and performs the actual write statements to create the restart files.
The arguments are:

n index of the nest

ftn unit number for the restart file

The following is the list of variables written in the Noah restart file:

```
nc,nr,nch    - grid and tile space dimensions
t1           - noah skin temperature
cmc          - noah canopy moisture storage
snowh        - noah snow depth
sneqv        - noah snow water equivalent
stc          - noah soil temperature (for each layer)
smc          - noah soil moisture (for each layer)
sh2o         - noah liquid only soil moisture (for each layer)
ch           - noah heat/moisture exchange coefficient
cm           - noah momentum exchange coefficient
```

The routines invoked are:

drv_writevar_restart (2.22.3)
writes a variable to the restart file

M.2.18 noahrst (Source File: noahrst.F90)

REVISION HISTORY:

```
05 Sep 2001: Brian Cosgrove; Modified code to use Dag Lohmann's NOAA
             initial conditions if necessary.  This is controlled with
             local variable NOAAIC.  Normally set to 0 in this subroutine
             but set to 1 if want to use Dag's NOAA IC's.  Changed output
             directory structure, and commented out if-then check so that
             directory is always made.
28 Apr 2002: Kristi Arsenault; Added NOAH LSM into LDAS
28 May 2002: Kristi Arsenault; For STARTCODE=4, corrected SNEQV values
              and put SMC, SH2O, STC limit for GDAS and GEOS forcing.
```

INTERFACE:

```
subroutine noahrst
```

USES:

```
use spmdMod, only : iam,masterproc
use lisdrv_module, only : lis
use noah_varder
use listime_mgr
use drv_output_mod, only : drv_readvar_restart
use lis_logmod, only : logunit
```

DESCRIPTION:

This program reads restart files for Noah. This includes all relevant water/energy storages and tile information. The following is the list of variables specified in the Noah restart file:

```
nc,nr,nch    - grid and tile space dimensions
t1           - noah skin temperature
cmc          - noah canopy moisture storage
snowh        - noah snow depth
sneqv        - noah snow water equivalent
stc          - noah soil temperature (for each layer)
smc          - noah soil moisture (for each layer)
sh2o         - noah liquid only soil moisture (for each layer)
ch           - noah heat/moisture exchange coefficient
cm           - noah momentum exchange coefficient
```

The routines invoked are:

drv_readvar_restart (2.22.4)
reads a variable from the restart file

M.2.19 noah_transform_synsca (Source File: noah_transform_synsca.F90)

REVISION HISTORY:

```
25Jun2006: Sujay Kumar: Initial Specification
```

INTERFACE:

```
subroutine noah_transform_synsca(n,OBS_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis,lisdom
use noah_varder
```

M.2.20 noah_getscavars (Source File: noah_getscavars.F90)

REVISION HISTORY:

```
27Feb2005: Sujay Kumar; Initial Specification
25Jun2006: Sujay Kumar: Updated for the ESMF design
```

INTERFACE:

```
subroutine noah_getscavars(n, LSM_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
use noah_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
type(ESMF_State)       :: LSM_State
```

DESCRIPTION:

Returns the snow related state prognostic variables for data assimilation
The arguments are:

n index of the nest

LSM_State ESMF State container for LSM state variables

M.2.21 noah_map_sca (Source File: noah_map_sca.F90)

REVISION HISTORY:

```
27Feb2005: Sujay Kumar; Initial Specification
25Jun2006: Sujay Kumar: Updated for the ESMF design
```

INTERFACE:

```
subroutine noah_map_sca(n,OBS_State,LSM_Incr_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis, lisdom
use constantsMod, only  : CONST_TKFRZ
use lis_logmod,   only  : logunit
use noah_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: n
type(ESMF_State)         :: OBS_State
type(ESMF_State)         :: LSM_Incr_State
```

DESCRIPTION:

This subroutine directly maps the observation state to the corresponding variables in the LSM state for SCA data assimilation.
The arguments are:

n index of the nest

OBS_State ESMF State for observations

LSM_State ESMF State for LSM state variables

M.2.22 noah_setscavars (Source File: noah_setscavars.F90)

REVISION HISTORY:

```
27Feb2005: Sujay Kumar; Initial Specification
25Jun2006: Sujay Kumar: Updated for the ESMF design
```

INTERFACE:

```
subroutine noah_setscavars(n, LSM_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
use noah_varder
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
```

DESCRIPTION:

This routine assigns the snow progognostic variables to noah's model space.

M.2.23 noah_updatescavars (Source File: noah_updatescavars.F90)

REVISION HISTORY:

```
27Feb2005: Sujay Kumar; Initial Specification
25Jun2006: Sujay Kumar: Updated for the ESMF design
```

INTERFACE:

```
subroutine noah_updatescavars(n, LSM_State, LSM_Incr_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
use noah_varder
use lis_logmod,   only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
type(ESMF_State)       :: LSM_State
type(ESMF_State)       :: LSM_Incr_State
```

DESCRIPTION:

Returns the snow related state prognostic variables for data assimilation
The arguments are:

n index of the nest

LSM_State ESMF State container for LSM state variables

LSM_Incr_State ESMF State container for LSM state increments

M.2.24 noah_getsoilm (Source File: noah_getsoilm.F90)

REVISION HISTORY:

```
27Feb2005: Sujay Kumar; Initial Specification
25Jun2006: Sujay Kumar: Updated for the ESMF design
```

INTERFACE:

```
subroutine noah_getsoilm(n, LSM_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
use noah_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
type(ESMF_State)       :: LSM_State
```

DESCRIPTION:

Returns the soilmoisture related state prognostic variables for data assimilation
The arguments are:

n index of the nest

LSM_State ESMF State container for LSM state variables

M.2.25 noah_getsynsmpred (Source File: noah_getsynsmpred.F90)

REVISION HISTORY:

```
27Feb2005: Sujay Kumar; Initial Specification
25Jun2006: Sujay Kumar: Updated for the ESMF design
```

INTERFACE:

```
subroutine noah_getsynsmpred(n, obs_pred)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis,lisdom
use noah_varder
```

M.2.26 noah_setsoilm (Source File: noah_setsoilm.F90)

REVISION HISTORY:

```
27Feb2005: Sujay Kumar; Initial Specification
25Jun2006: Sujay Kumar: Updated for the ESMF design
```

INTERFACE:

```
subroutine noah_setsoilm(n, LSM_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
use noah_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
type(ESMF_State)       :: LSM_State
```

DESCRIPTION:

This routine assigns the soil moisture prognostic variables to noah's model space.

M.2.27 noah_descale_soilm (Source File: noah_descale_soilm.F90)

REVISION HISTORY:

```
27Feb2005: Sujay Kumar; Initial Specification
24Mar2005: Xiwu Zhan; modified for Noah model
25Jun2006: Sujay Kumar: Updated for the ESMF design
```

INTERFACE:

```
subroutine noah_descale_soilm(n, LSM_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
use noah_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
type(ESMF_State)       :: LSM_State
```

DESCRIPTION:

Descales soilmoisture related state prognostic variables for data assimilation
The arguments are:

n index of the nest

LSM_State ESMF State container for LSM state variables

M.2.28 noah_qcsoilm (Source File: noah_qcsoilm.F90)

REVISION HISTORY:

```
27Feb2005: Sujay Kumar; Initial Specification
24Mar2005: Xiwu Zhan; modified for Noah model
25Jun2006: Sujay Kumar: Updated for the ESMF design
```

INTERFACE:

```
subroutine noah_qcsoilm(n, LSM_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
use noah_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
type(ESMF_State)       :: LSM_State
```

DESCRIPTION:

Returns the soilmoisture related state prognostic variables for data assimilation
The arguments are:

n index of the nest

LSM_State ESMF State container for LSM state variables

M.2.29 noah_scale_soilm (Source File: noah_scale_soilm.F90)

REVISION HISTORY:

```
27Feb2005: Sujay Kumar; Initial Specification
25Jun2006: Sujay Kumar: Updated for the ESMF design
```

INTERFACE:

```
subroutine noah_scale_soilm(n, LSM_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
use noah_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
type(ESMF_State)       :: LSM_State
```

DESCRIPTION:

Scales soilmoisture related state prognostic variables for data assimilation
The arguments are:

n index of the nest

LSM_State ESMF State container for LSM state variables

M.2.30 noah_updatesoilm (Source File: noah_updatesoilm.F90)

REVISION HISTORY:

```
27Feb2005: Sujay Kumar; Initial Specification
25Jun2006: Sujay Kumar: Updated for the ESMF design
```

INTERFACE:

```
subroutine noah_updatesoilm(n, LSM_State, LSM_Incr_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
use noah_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
type(ESMF_State)       :: LSM_State
type(ESMF_State)       :: LSM_Incr_State
```

DESCRIPTION:

This routine assigns the soil moisture prognostic variables to noah's model space.

M.2.31 noah_updatesnowvars (Source File: noah_updatesnowvars.F90)

REVISION HISTORY:

```
27Feb2005: Sujay Kumar; Initial Specification
25Jun2006: Sujay Kumar: Updated for the ESMF design
```

INTERFACE:

```
subroutine noah_updatesnowvars(n, LSM_State, LSM_Incr_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
use noah_varder
use lis_logmod,   only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
type(ESMF_State)       :: LSM_State
type(ESMF_State)       :: LSM_Incr_State
```

DESCRIPTION:

Returns the snow related state prognostic variables for data assimilation
The arguments are:

n index of the nest

LSM_State ESMF State container for LSM state variables

LSM_Incr_State ESMF State container for LSM state increments

M.2.32 noah_getsnowvars (Source File: noah_getsnowvars.F90)

REVISION HISTORY:

```
27Feb2005: Sujay Kumar; Initial Specification
25Jun2006: Sujay Kumar: Updated for the ESMF design
```

INTERFACE:

```
subroutine noah_getsnowvars(n, LSM_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
use noah_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
type(ESMF_State)       :: LSM_State
```

DESCRIPTION:

Returns the snow related state prognostic variables for data assimilation
The arguments are:

n index of the nest

LSM_State ESMF State container for LSM state variables

M.2.33 noah_getsynswepred (Source File: noah_getsynswepred.F90)

REVISION HISTORY:

```
27Feb2005: Sujay Kumar; Initial Specification
25Jun2006: Sujay Kumar: Updated for the ESMF design
```

INTERFACE:

```
subroutine noah_getsynswepred(n, obs_pred)
```

USES:

```
   use ESMF_Mod
   use lisdrv_module, only : lis,lisdom
   use noah_varder

   implicit none
```

ARGUMENTS:

```
   integer, intent(in)    :: n
   real                   :: obs_pred(lis%ngrid(n),lis%nensem(n))

   integer                :: i,t,m,gid

   do i=1,lis%nch(n),lis%nensem(n)
      do m=1,lis%nensem(n)
         t = i+m-1
         gid = lisdom(n)%tile(t)%index
         obs_pred(gid,m)= noah_struc(n)%noah(t)%sneqv*1000.0
      enddo
   enddo

 end subroutine noah_getsynswepred

\markboth{Left}{Source File: noah\_setsnowvars.F90,  Date: Sat Jan  5 17:35:40 EST 2008
}


  ------------------------------------------------------------------------
   NASA Goddard Space Flight Center Land Information System  V5.0
   Released August 2006
   See SOFTWARE DISTRIBUTION POLICY for software distribution policies
   The LIS source code and documentation are in the public domain,
   available without fee for educational, research, non-commercial and
   commercial purposes.  Users may distribute the binary or source
   code to third parties provided this statement appears on all copies and
   that no charge is made for such copies.
   NASA GSFC MAKES NO REPRESENTATIONS ABOUT THE SUITABILITY OF THE
   SOFTWARE FOR ANY PURPOSE.  IT IS PROVIDED AS IS WITHOUT EXPRESS OR
   IMPLIED WARRANTY.  NEITHER NASA GSFC NOR THE US GOVERNMENT SHALL BE
   LIABLE FOR ANY DAMAGES SUFFERED BY THE USER OF THIS SOFTWARE.
   See COPYRIGHT.TXT for copyright details.
  ------------------------------------------------------------------------
%//////////////////////////////////////////////////////////////

\mbox{}\hrulefill\

\subsubsection{noah\_setsnowvars (Source File: noah\_setsnowvars.F90)}


   \label{noah_setsnowvars}

\bigskip{\sf REVISION HISTORY:}
\begin{verbatim}   27Feb2005: Sujay Kumar; Initial Specification
   25Jun2006: Sujay Kumar: Updated for the ESMF design
```

INTERFACE:

```
subroutine noah_setsnowvars(n, LSM_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
use noah_varder
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
```

DESCRIPTION:

This routine assigns the snow progognostic variables to noah's model space.

M.2.34 noah_descale_snow (Source File: noah_descale_snow.F90)

REVISION HISTORY:

```
27Feb2005: Sujay Kumar; Initial Specification
24Mar2005: Xiwu Zhan; modified for Noah model
25Jun2006: Sujay Kumar: Updated for the ESMF design
```

INTERFACE:

```
subroutine noah_descale_snow(n, LSM_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
use noah_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
type(ESMF_State)       :: LSM_State
```

DESCRIPTION:

Returns the snow related state prognostic variables for data assimilation
The arguments are:

n index of the nest

LSM_State ESMF State container for LSM state variables

M.2.35 noah_qcsnow (Source File: noah_qcsnow.F90)

REVISION HISTORY:

```
27Feb2005: Sujay Kumar; Initial Specification
25Jun2006: Sujay Kumar: Updated for the ESMF design
```

INTERFACE:

```
subroutine noah_qcsnow(n, LSM_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
use noah_varder
use lis_logmod

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
type(ESMF_State)       :: LSM_State
```

DESCRIPTION:

Returns the soilmoisture related state prognostic variables for data assimilation
The arguments are:

n index of the nest

LSM_State ESMF State container for LSM state variables

M.2.36 noah_scale_snow (Source File: noah_scale_snow.F90)

REVISION HISTORY:

```
27Feb2005: Sujay Kumar; Initial Specification
24Mar2005: Xiwu Zhan; modified for Noah model
25Jun2006: Sujay Kumar: Updated for the ESMF design
```

INTERFACE:

```
subroutine noah_scale_snow(n, LSM_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
use noah_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
type(ESMF_State)       :: LSM_State
```

DESCRIPTION:

Returns the snow related state prognostic variables for data assimilation
The arguments are:

n index of the nest

LSM_State ESMF State container for LSM state variables

!ROUTINE : noah_physics.F90
DESCRIPTION: SUB-DRIVER FOR "NOAH/OSU LSM" FAMILY OF PHYSICS SUBROUTINES FOR A SOIL/VEG/SNOWPACK LAND-SURFACE MODEL TO UPDATE SOIL MOISTURE, SOIL ICE, SOIL TEMPERATURE, SKIN TEMPERATURE, SNOWPACK WATER CONTENT, SNOWDEPTH, AND ALL TERMS OF THE SURFACE ENERGY BALANCE AND SURFACE WATER BALANCE (EXCLUDING INPUT ATMOSPHERIC FORCINGS OF DOWNWARD RADIATION AND PRECIP)

REVISION HISTORY:

```
 28 Apr 2002: Kristi Arsenault; Added NOAH LSM 2.5 code to LDAS
 15 May 2002: Urszula Jambor; Changed LOGICAL to LOGICAL*1 to match new
               GRIB libraries
 28 May 2002: Kristi Arsenault; Updated NOAH code to 2.5.1, and
              corrected problem with FX in DEVAP function.
 12 Jun 2002: Kristi Arsenault; Updated NOAH code to 2.5.2
 04 Nov 2002: Kristi Arsenault; Incorporated new TBOT fields
 24 Jun 2003: Kristi Arsenault; Updated Noah LSM to 2.6 version
 31 May 2004: Kristi Arsenault; Updated Noah LSM to 2.7 version
 24 Nov 2004: Kristi Arsenault; Updated Noah LSM to 2.7.1
 06 Apr 2005: Jesse Meng; Fix FDOWN, RTDIS, ETA, EVP
 16 Oct 2006: Jesse Meng; Complete Noah 2.7.1
-----------------------------------------------------------------------

-- 1. MAIN PHYSICS SUBROUTINE ==>  SUBROUTINE SFLX  ------------------
```

INTERFACE:

```
    SUBROUTINE SFLX (ntl, &
     FFROZP,ICE,DT,ZLVL,NSOIL,SLDPTH, &
     LWDN,SOLDN,SOLNET,SFCPRS,PRCP,SFCTMP,Q2,SFCSPD, &
     TH2,Q2SAT,DQSDT2, &
     SLOPE,SHDFAC,SHDMIN,PTU,ALB,SNOALB, &
     RSMIN,RGL,HS,SNUP,ZO,EMISSI, XLAI,NROOT, &
     PSISAT,BEXP,DKSAT,SMCMAX,QUARTZ,DWSAT, &
     SMCWLT,SMCREF,SMCDRY,F1,KDT,FRZX,FRZFACT,TBOT, &
     CMC,T1,STC,SMC,SH2O,SNOWH,SNEQV,ALBEDO,CH,CM, &
     EVP,ETA,SHEAT, ETA_KINEMATIC, &
     EC,EDIR,ET,ETT,ESNOW,DRIP,DEW, &
     BETA,ETP,SSOIL, &
     FLX1,FLX2,FLX3, &
```

```
SNOMLT,SNCOVR, &
RUNOFF1,RUNOFF2,RUNOFF3, &
RC,PC,RCS,RCT,RCQ,RCSOIL, &
SOILW,SOILT,SOILM)
```

M.2.37 noah_setwrfexport.F90 (Source File: noah_setwrfexport.F90)

Defines the export states from Noah to WRF in coupled mode

REVISION HISTORY:

```
02 Dec 2003; Sujay Kumar, Initial Version
```

INTERFACE:

```
subroutine noah_setwrfexport(n)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use noah_varder
use lisWRFGridCompMod, only : liswrf_export
use drv_output_mod, only : drv_tile2grid

implicit none
integer, intent(in) :: n
integer :: i,j,k
integer :: c,r
real :: temp(lis%nch(n))
```

M.2.38 noah_wrf_f2t (Source File: noah_wrf_f2t.F90)

REVISION HISTORY:

```
15 Oct 2002: Sujay Kumar; Initial Code
```

INTERFACE:

```
subroutine noah_wrf_f2t(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module , only : lis
use baseforcing_module, only : FORC_State
use noah_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine transfers the LIS provided forcing onto the Noah model tiles in a coupled mode to WRF
The arguments are:

n index of the nest

M.2.39 sh2o_init.F90 (Source File: sh2o_init.F90)

To do a 'cold start' initialization of soil liquid water SH2O for NOAH LSM, though also adaptable for other land-surface models, using either GDAS or Eta forcing data.

REVISION HISTORY:

```
    NCEP;  Original code developed by NCEP for Eta model
             subroutines to initialize soil liquid water, SH2O
    04 Nov 2002: Kristi Arsenault; Modified code to be used with noahrst.f
                                   to initialize NOAH with NCEP forcing data
```

INTERFACE:

```
 subroutine sh2oinit(smc,stc,smcmax,psis,beta,sh2o)

   implicit none
```

ARGUMENTS:

```
   REAL STC      ! NOAH Soil Layer Temperature (K)
   REAL SMC      ! NOAH Soil Layer Total Moisture (liq+frzn)
   REAL SH2O     ! NOAH Soil Layer Liquid Moisture

   REAL  PSIS                    ! Saturated soil potential
   REAL  BETA                    ! B-parameter
   REAL  SMCMAX                  ! Max soil moisture content (porosity)
   REAL  BX

! ----------------------------------------------------------------------
! COLD START:  determine liquid soil water content (SH2O)
! NSOIL number of soil layers
! ----------------------------------------------------------------------
!  SH2O <= SMC for T < 273.149K (-0.001C)
  IF (STC .LT. 273.149) THEN
! ----------------------------------------------------------------------
! first guess following explicit solution for Flerchinger Eqn from Koren
! et al, JGR, 1999, Eqn 17 (KCOUNT=0 in FUNCTION FRH2O).
! ----------------------------------------------------------------------
     BX = BETA
     IF ( BETA .GT. BLIM )  BX = BLIM
     FK=(((HLICE/(GRAV*(-PSIS)))* &
          ((STC-T0)/STC))**(-1/BX))*SMCMAX
     IF (FK .LT. 0.02) FK = 0.02
     SH2O = MIN ( FK, SMC )
! ----------------------------------------------------------------------
! now use iterative solution for liquid soil water content using
! FUNCTION FRH2O with the initial guess for SH2O from above explicit
! first guess.
```

```
! ------------------------------------------------------------------------
     SH2O = FRH2O(STC,SMC,SH2O,SMCMAX,BETA,PSIS)

  ELSE
! ------------------------------------------------------------------------
!  SH2O = SMC for T => 273.149K (-0.001C)
     SH2O=SMC
! ------------------------------------------------------------------------
  ENDIF

  RETURN
```

M.2.40 noah_finalize (Source File: noah_finalize.F90)

REVISION HISTORY:

```
28 Apr 2002: Sujay Kumar; Initial code
```

INTERFACE:

```
subroutine noah_finalize()
```

USES:

```
use lisdrv_module, only : lis
use noah_varder
```

DESCRIPTION:

This routine cleans up the allocated memory structures in Noah

N Community Land Model version 2.0

This section describes the interface implementations for the CLM land surface model. CLM (Community Land Model) is a 1-D land surface model, written in Fortran 90, developed by a grass-roots collaboration of scientists who have an interest in making a general land model available for public use. LIS currently uses CLM version 2.0. CLM version 2.0 was released in May 2002. The source code for CLM 2.0 is freely available from the National Center for Atmospheric Research (NCAR) [1]. The CLM is used as the land model for the Community Climate System Model (CCSM) (http://www.ccsm.ucar.edu/), which includes the Community Atmosphere Model (CAM) (http://www.cgd.ucar.edu/cms/). CLM is executed with all forcing, parameters, dimensioning, output routines, and coupling performed by an external driver of the user's design (in this case done by LIS). CLM requires pre-processed data such as the land surface type, soil and vegetation parameters, model initialization, and atmospheric boundary conditions as input. The model applies finite-difference spatial discretization methods and a fully implicit time-integration scheme to numerically integrate the governing equations. The model subroutines apply the governing equations of the physical processes of the soil-vegetation-snowpack medium, including the surface energy balance equation, Richards' [15] equation for soil hydraulics, the diffusion equation for soil heat transfer, the energy-mass balance equation for the snowpack, and the Collatz et al. [4] formulation for the conductance of canopy transpiration.

N.1 Fortran: Module Interface clmtype (Source File: clmtype.F90)

The code in this file provides a description of the data structure containing the CLM 1-d variables. The variables specified in the data structure include:

nstep time step number

kpatch tile index

itypveg vegetation type

itypwat water type

itypprc precipitation type

isoicol color class for soil albedos

snl number of snow layers

frac_veg_nosno fraction of vegetation not covered by snow

frac_veg_nosno_alb fraction of vegetation not covered by snow

imelt flag for meltiing (=1), freezing (=2), not=0(new)

lakpoi flag for lakpoint (true=lake point)

do_capsnow flag to indicate snow capping (true=do sno caping)

present whether PFT is present in the current patch

lat latitude of the patch

lon longitude of the patch

dtime model timestep

zi interface level below a "z" level (m)

dz layer depth (m)

z layer thickness (m)

bsw Clapp and Hornberger "b"

watsat volumetric soil water at saturation (porosity)

hksat hydraulic conductivity at saturation (mm H2O /s)

sucsat minimum soil suction (mm)

csol heat capacity, soil solids (J/m**3/Kelvin)

tkmg thermal conductivity, soil minerals [W/m-K] (new)

tkdry thermal conductivity, dry soil (W/m/Kelvin)

tksatu thermal conductivity, saturated soil [W/m-K] (new)

rootfr fraction of roots in each soil layer

rootr effective fraction of roots in each layer

begwb water mass at the beginning of the time step

endwb water mass at the end of the time step

forc_t atmospheric temperature (K)

forc_u atmospheric wind speed in east direction (m/s)

forc_v atmospheric wind speed in north direction (m/s)

forc_q atmospheric specific humidity (kg/kg)

forc_hgt atmospheric reference height (m)

forc_hgt_u observational height of wind [m]

forc_hgt_v observational height of temperature [m]

forc_hgt_q observational height of humidity [m] (new)

forc_pbot atmospheric pressure (Pa)

forc_th atmospheric potential temperature (Kelvin)

forc_vp atmospheric vapor pressure (Pa)

forc_rho density (kg/m**3)

forc_lwrad downward infrared (longwave) radiation (W/m**2)

forc_solad direct beam radiation (vis=forc_sols , nir=forc_soll)

forc_solai diffuse radiation (vis=forc_solsd, nir=forc_solld)

forc_ch heat/moisture exchange coefficient

forc_rain rain rate [mm/s]

forc_snow snow rate [mm/s]

rssun sunlit stomatal resistance (s/m)

rssha shaded stomatal resistance (s/m)

psnsun sunlit leaf photosynthesis (umol CO2 /m**2/ s)

psnsha shaded leaf photosynthesis (umol CO2 /m**2/ s)

laisun sunlit leaf area

laisha shaded leaf area

sabg solar radiation absorbed by ground (W/m**2)

sabv solar radiation absorbed by vegetation (W/m**2)

fsa solar radiation absorbed (total) (W/m**2)

taux wind stress: e-w (kg/m/s**2)

tauy wind stress: n-s (kg/m/s**2)

eflx_lwrad_out emitted infrared (longwave) radiation (W/m**2)

eflx_lwrad_net net infrared (longwave) rad (W/m**2) [+ = to atm]

eflx_sh_tot total sensible heat flux (W/m**2) [+ to atm]

eflx_sh_veg sensible heat flux from leaves (W/m**2) [+ to atm]

eflx_sh_grnd sensible heat flux from ground (W/m**2) [+ to atm]

eflx_lh_tot total latent heat flux (W/m8*2) [+ to atm]

eflx_soil_grnd soil heat flux (W/m**2) [+ = into soil]

t_veg vegetation temperature (Kelvin)

t_grnd ground temperature (Kelvin)

t_rad radiative temperature (Kelvin)

t_ref2m 2 m height surface air temperature (Kelvin)

t_soisno soil temperature (Kelvin)

qflx_infl infiltration (mm H2O /s)

qflx_surf surface runoff (mm H2O /s)

qflx_drain sub-surface runoff (mm H2O /s)

qflx_top_soil net water input into soil from top (mm/s)

qflx_evap_soi soil evaporation (mm H2O/s) (+ = to atm)

qflx_evap_veg vegetation evaporation (mm H2O/s) (+ = to atm)

qflx_tran_veg vegetation transpiration (mm H2O/s) (+ = to atm)

qflx_snomelt snow melt (mm H2O /s)

qflx_evap_tot qflx_evap_soi + qflx_evap_veg + qflx_tran_veg

qflx_rain_grnd rain on ground after interception (mm H2O/s) [+]

qflx_evap_grnd ground surface evaporation rate (mm H2O/s) [+]

qflx_dew_grnd ground surface dew formation (mm H2O /s) [+]

qflx_sub_snow sublimation rate from snow pack (mm H2O /s) [+]

qflx_dew_snow surface dew added to snow pack (mm H2O /s) [+]

qflx_snowcap excess precipitation due to snow capping (mm H2O /s) [+]

qflx_qrgwl qflx_surf at glaciers, wetlands, lakes

h2osno snow water (mm H2O)

h2ocan canopy water (mm H2O)

h2osoi_liq liquid water (kg/m2)

h2osoi_ice ice lens (kg/m2)

h2osoi_vol volumetric soil water (0¡=h2osoi_vol¡=watsat) [m3/m3]

snowdp snow height (m)

snowage non dimensional snow age [-]

h2osno_old snow mass for previous time step (kg/m2) (new)

frac_sno fraction of ground covered by snow (0 to 1)

frac_iceold fraction of ice relative to the total water (new)

eff_porosity effective porosity = porosity - vol_ice

parsun average absorbed PAR for sunlit leaves (W/m**2)

albgrd ground albedo (direct)

albgri ground albedo (diffuse)

fabd flux absorbed by veg per unit direct flux

fabi flux absorbed by veg per unit diffuse flux

ftdd down direct flux below veg per unit dir flx

ftid down diffuse flux below veg per unit dir flx

ftii down diffuse flux below veg per unit dif flx

fsun sunlit fraction of canopy

surfalb instantaneous all-wave surface albedo

snoalb instantaneous all-wave snow albedo

hbot canopy bottom (m)

htop canopy top (m)

tlai one-sided leaf area index, no burying by snow

tsai one-sided stem area index, no burying by snow

elai one-sided leaf area index with burying by snow

esai one-sided stem area index with burying by snow

fwet fraction of canopy that is wet (0 to 1)

fdry fraction of foliage that is green and dry [-] (new)

annpsn annual photosynthesis (umol CO2 /m**2)

annpsnpot annual potential photosynthesis (same units)

wf soil water as frac. of whc for top 0.5 m

z0mr ratio of momentum roughness length to canopy top height [-]

z0m momentum roughness length [m]

displar ratio of displacement height to canopy top height [-]

displa displacement height [m]

dleaf leaf dimension [m]

xl pft_varcon leaf/stem orientation index

rhol pft_varcon leaf reflectance : 1=vis, 2=nir

rhos pft_varcon stem reflectance : 1=vis, 2=nir

taul pft_varcon leaf transmittance: 1=vis, 2=nir

taus pft_varcon stem transmittance: 1=vis, 2=nir

qe25 quantum efficiency at 25c (umol co2 / umol photon)

vcmx25 maximum rate of carboxylation at 25c (umol co2/m**2/s)

mp slope for conductance-to-photosynthesis relationship

c3psn photosynthetic pathway: 0. = c4, 1. = c3

totfsa solar absorbed solar radiation [W/m2]

toteflx_lwrad_net net longwave radiation [W/m2]

toteflx_lh_tot total latent heat flux [W/m2]

toteflx_sh_tot total sensible heat flux [W/m2]

toteflx_soil_grnd ground heat flux [W/m2]

toqflx_snomelt snowmelt heat flux [W/m2]

totrain accumulation of rain [mm]

totsnow accumulation of snow [mm]

totqflx_evap total evaporation [mm]

totqflx_surf surface runoff [mm]

totqflx_drain subsurface runoff [mm]

totqflx_ecanop interception evaporation [W/m2]

totqflx_tran_veg Total vegetation transpiration

totqflx_evap_grnd Total ground surface evaporation

totqflx_sub_snow Total sublimation rate from snow pack

acond aerodyamic conductance

soilmtc_prev Total column soil moisture for the prev.timestep

h2osno_prev Total column snow water equivalent for the prev.timestep

USES:

```
use precision, only : r8
use clm_varpar, only : nlevsoi, nlevsno, nlevlak, numrad, nvoc, ndst
```

N.2 Fortran: Module Interface clm_varder (Source File: clm_varder.F90)

This module provides the definition of derived data type used to control the operation of CLM. It also provides the entry method for the initialization of CLM-specific variables. The derived data type `clm_struc` includes the variables that specify the runtime options and other control variables as described below:

clm_rfile name of the CLM restart file

clm_vfile name of the CLM vegetation parameter lookup table

clm_chtfile name of the CLM canopy heights lookup table

count variable to keep track of the number of timesteps before an output

clmopen variable to keep track of opened files

numout number of output times

clm_ism initial soil moisture for a cold start run

clm_it initial soil temperature for a cold start run

clm_iscv initial snow mass for a cold start run

outInterval output writing interval

rstInterval restart writing interval

clm CLM specific variables

USES:

```
use clmtype

implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
---------------------------------------------------------------------------
public :: clm_varder_ini
---------------------------------------------------------------------------
```

PUBLIC TYPES:

```
---------------------------------------------------------------------------
public :: clm_struc
```

N.2.1 clm_varder_ini (Source File: clm_varder.F90)

INTERFACE:

```
subroutine clm_varder_ini()
```

USES:

```
use spmdMod, only : iam, masterproc
use infnan
use clm_varcon
use pft_varcon
use shr_orb_mod
use lis_logmod, only : logunit
use lisdrv_module, only : lis
```

DESCRIPTION:

This routine creates the datatypes and allocates memory for CLM-specific variables. It also invokes the routine to read the runtime specific options for CLM from the configuration file.
The routines invoked are:

readclmcrd (N.3.1) reads the runtime options for CLM

clm_varder_init (N.2.2) set initial values to CLM variables

iniTimeConst (N.3.3) initialize time invariant parameters

N.2.2 clm_varder_init (Source File: clm_varder.F90)

INTERFACE:

```
subroutine clm_varder_init(n)
```

USES:

```
use lisdrv_module, only : lis
use infnan
```

DESCRIPTION:

Initializes clm variables
The arguments are:

n index of the nest

N.3 Fortran: Module Interface atmdrvMod (Source File: atmdrvMod.F90)

This module defines the variables and routines to enable the interaction with an atmospheric component

USES:

```
use precision
use shr_const_mod, only : SHR_CONST_TKFRZ,SHR_CONST_PSTD
use spmdMod      , only : masterproc
```

N.3.1 atmdrv (Source File: atmdrvMod.F90)

INTERFACE:

```
subroutine atmdrv(n)
```

USES:

```
use ESMF_Mod
use infnan
use clm_varder
use clm_varcon  , only : rair, cpair, po2, pco2, tcrit, tfrz
use lisdrv_module, only : lis
use baseforcing_module, only : FORC_State

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine transfers the LIS provided forcing onto the CLM2 model tiles.
The arguments are:

n index of the nest

!ROUTINE : readclmcrd

REVISION HISTORY:

```
14 Oct 2003; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine readclmcrd()
```

USES:

```
use lis_logmod, only : logunit
use lisdrv_module, only : lis, config_lis
use ESMF_Mod
use clm_varder, only : clm_struc
```

DESCRIPTION:

This routine reads the options specific to CLM from the LIS configuration file.

N.3.2 clm_setup (Source File: clm_setup.F90)

REVISION HISTORY:

```
20 Jan 2003  Sujay Kumar Initial Specification
```

INTERFACE:

```
subroutine clm_setup()
```

USES:

```
use lisdrv_module, only: lis
use spmdMod
use listime_mgr
use clm_varder
use clm_varcon, only: eccen, obliqr, lambm0 , mvelpp
use lis_logmod, only :logunit
```

DESCRIPTION:

This routine is the entry point to set up the parameters required for CLM. The method invokes the routine to call time variant parameters.
The routines invoked are:

canhtset (N.3.4) reads the canopy height information

iniTimeVar (N.3.4) initialize time variant parameters

N.3.3 iniTimeConst (Source File: iniTimeConst.F90)

INTERFACE:

```
subroutine iniTimeConst (n)
```

USES:

```
use precision
use infnan
use clm_varder
use clm_varpar   , only : nlevsoi, nlevlak
use clm_varcon   , only : istsoil, istice, istdlak, istslak, istwet, spval
use pft_varcon   , only : ncorn, nwheat, roota_par, rootb_par,  &
                          z0mr, displar, dleaf, rhol, rhos, taul, taus, xl, &
                          qe25, vcmx25, mp, c3psn
use clm_varsur   , only : zlak, dzlak, zsoi, dzsoi, zisoi
use shr_const_mod, only : SHR_CONST_PI
use spmdMod      , only : masterproc
use lisdrv_module, only : lis, lisdom
use soils_module, only  : lis_soils
use lis_logmod, only    : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine initializes the time invariant CLM variables
The arguments are:

n index of the nest

N.3.4 iniTimeVar (Source File: iniTimeVar.F90)

INTERFACE:

```
subroutine iniTimeVar (n, eccen, obliqr, lambm0 , mvelpp)
```

USES:

```
use precision
use lisdrv_module, only : lis
use clm_varder
use clm_varcon  , only : bdsno, istice, istwet, &
     istsoil, denice, denh2o, tfrz, spval, doalb
use shr_sys_mod , only : shr_sys_abort
use spmdMod     , only : masterproc
use listime_mgr, only : get_nstep, get_curr_calday
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer , intent(in) :: n
real(r8), intent(in) :: eccen
real(r8), intent(in) :: obliqr
real(r8), intent(in) :: lambm0
real(r8), intent(in) :: mvelpp
```

DESCRIPTION:

Initialize the following time varying variables:
water : h2osno, h2ocan, h2osoi_liq, h2osoi_ice, h2osoi_vol
snow : snowdp, snowage, snl, dz, z, zi
temperature: t_soisno, t_veg, t_grnd

Note - h2osoi_vol is needed by clm_soilalb -this is not needed on restart since it is computed before the soil albedo computation is called
Note - remaining variables are initialized by calls to ecosystem dynamics and albedo subroutines.
The arguments are:

n index of the nest

eccen Earth's orbital eccentricity

obliqr Earth's obliquity in radians

lambm0 Mean longitude of perihelion at the vernal equinox (radians)

mvelpp Earth's moving vernal equinox long. of perihelion + pi (radians)

!ROUTINE:canhtset

REVISION HISTORY:

```
15 Nov 2002: Jon Gottschalck; Initial code
```

INTERFACE:

```
subroutine canhtset (n)
```

USES:

```
use clm_varder          ! CLM tile variables
use clm_varpar     , only : maxpatch
use lisdrv_module, only  : lis

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This subroutine reads in canopy height information into CLM. Expects a file with the top and bottom canopy heights for each veg type.

N.3.5 clm_dynsetup (Source File: clm_dynsetup.F90)

REVISION HISTORY:

```
20 Jan 2003  Sujay Kumar Initial Specification
```

INTERFACE:

```
subroutine clm_dynsetup(n)

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine sets up the time-dependent variables in CLM The routine invokes methods to read the LAI data

---

N.3.6 clm_main (Source File: clm_main.F90)

INTERFACE:

```
subroutine clm_main (n)
```

USES:

```
use precision
use clm_varder
use clm_varcon     , only : doalb, eccen, obliqr, lambm0, mvelpp, &
     denh2o, denice, hvap, hsub, hfus, istwet
use lisdrv_module, only : lis
use listime_mgr  , only : get_step_size, get_curr_calday, get_curr_date, get_nstep
```

```
#if (defined RTM)
  use RtmMod        , only : Rtmriverflux
#endif
#if (defined SPMD)
  use spmdMod       , only : masterproc, npes
  use mpishorthand  , only : mpir8, mpilog, mpicom
#else
  use spmdMod       , only : masterproc
#endif
#if (defined COUP_CSM)
  use clm_csmMod    , only : csm_dosndrcv, csm_recv, csm_send, csm_flxave, &
                             dorecv, dosend, csmstop_now
#endif
  use shr_sys_mod   , only : shr_sys_flush

  implicit none
```

ARGUMENTS:

```
  integer, intent(in) :: n
```

DESCRIPTION:

This is the entry point for calling the CLM physics. This routine calls various physics routines for CLM that performs the land surface computations, to solve for water and energy equations. For a more detailed documentation of CLM, please contact the NCAR repository
Calling sequence:

```
  -> loop over patch points calling for each patch point:
     -> Hydrology1          canopy interception and precip on ground
     -> Biogeophysics1      leaf temperature and surface fluxes
     -> Biogeophysics_Lake  lake temperature and surface fluxes
     -> Biogeophysics2      soil/snow and ground temp and update surface fluxes
     -> Hydrology2          surface and soil hydrology
     -> Hydrology_Lake      lake hydrology
     -> Biogeochemistry     surface biogeochemical fluxes (LSM)
     -> EcosystemDyn:       ecosystem dynamics: phenology, vegetation,
                            soil carbon
     -> SurfaceAlbedo:      albedos for next time step
       -> SnowAlbedo:       snow albedos: direct beam
       -> SnowAlbedo:       snow albedos: diffuse
       -> SoilAlbedo:       soil/lake albedos
       -> TwoStream:        absorbed, reflected, transmitted
                            solar fluxes (vis dir)
       -> TwoStream:        absorbed, reflected, transmitted
                            solar fluxes (vis dif)
       -> TwoStream:        absorbed, reflected, transmitted
                            solar fluxes (nir dir)
       -> TwoStream:        absorbed, reflected, transmitted
                            solar fluxes (nir dif)
     -> BalanceCheck        check for errors in energy and water balances
```

The arguments are:

n index of the nest

N.3.7 clm_output (Source File: clm_output.F90)

INTERFACE:

```
subroutine clm_output(n)
```

USES:

```
use lisdrv_module, only : lis
use clm_varder, only : clm_struc
use spmdMod, only : masterproc, npes
use lis_fileIOMod, only : create_output_directory, &
                          create_output_filename,  &
                          create_stats_filename
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This subroutines sets up methods to write model output from CLM A timestamped filename and directory are created first. The routine supports model output in three data formats: Binary, Grib and NETCDF. The arguments are:

n index of the nest

The routines invoked are:

create_output_directory (2.25.2) creates a time stamped output directory

create_output_filename (2.25.3) creates a time stamped output filename

create_stats_filename (2.25.5) creates a stats filename

clm_binout (N.3.8) routine to write CLM model output in binary format

clm_gribout (N.3.9) routine to write CLM model output in grib format

clm_totint (N.3.10) resets the time averaged variables

N.3.8 clm_binout (Source File: clm_binout.F90)

REVISION HISTORY:

```
02 Dec  2003: Sujay Kumar; Initial Version
```

INTERFACE:

```
subroutine clm_binout(ftn,ftn_stats,n)
```

USES:

```
use lisdrv_module, only : lis,metadata_output
use drv_output_mod, only : drv_writevar_bin
use clm_varcon, only : denh2o, denice, hvap, hsub, hfus, istwet
use clm_varpar, only : nlevsoi
use clm_varder, only : clm_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine writes the specified CLM variables into a binary output file.
The arguments are:

n index of the nest

ftn unit number for the model output file

ftn_stats unit number for the stats file

The routines invoked are:

drv_writevar_bin (2.22.2) routine to write a variable to a binary output file

---

N.3.9 clm_gribout (Source File: clm_gribout.F90)

REVISION HISTORY:

```
02 Dec  2003: Sujay Kumar; Initial Version
```

INTERFACE:

```
subroutine clm_gribout(ftn,n)
```

USES:

```
use lis_module
use lisdrv_module, only : lis,lisdom
use drv_output_mod, only : drv_writevar_grib
use clm_varcon, only : denh2o, denice, hvap, hsub, hfus, istwet
use clm_varpar, only : nlevsoi
use clm_varder, only : clm_struc
use listime_mgr, only : tick

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine writes the specified CLM variables into a grib output file.
The arguments are:

n index of the nest

ftn unit number for the model output file

The routines invoked are:

drv_writevar_grib (2.22.6) routine to write a variable to a grib output file

---

N.3.10 clm_totinit (Source File: clm_totinit.F90)

REVISION HISTORY:

```
14 Jun 2002  Sujay Kumar  Initial Specification
```

INTERFACE:

```
subroutine clm_totinit(n)
```

USES:

```
use lisdrv_module, only : lis
use clm_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine resets the time-averaged CLM variables after a model output. The routine is called after a model output call.

N.3.11 clm_writerestart (Source File: clm_writerestart.F90)

REVISION HISTORY:

```
20 Jan 2003; Sujay Kumar Initial Specification
26 Aug 2004; James Geiger, Added support for GrADS-DODS based and MPI based
                           parallel simulations.
```

INTERFACE:

```
subroutine clm_writerestart(n)
```

USES:

```
use spmdMod, only : masterproc, npes
use lisdrv_module, only : lis
use lis_logmod, only : logunit
use lis_fileIOMod, only : create_output_directory, &
                          create_restart_filename
use clm_varder, only : clm_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This program writes restart files for CLM. This includes all relevant water/energy storage and tile information The routines invoked are:

create_output_directory (2.25.2) creates a timestamped directory for the restart files

create_restart_filename (2.25.4) generates a timestamped restart filename

clm_dump_restart (N.3.12) writes the CLM variables into the restart file reads a variable from the restart file

---

N.3.12 clm_dump_restart (Source File: clm_writerestart.F90)

INTERFACE:

```
subroutine clm_dump_restart(n, ftn)
```

USES:

```
use clm_varder
use listime_mgr
use lisdrv_module,  only : lis
use drv_output_mod, only : drv_writevar_restart

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn
```

DESCRIPTION:

This routine gathers the necessary restart variables and performs the actual write statements to create the restart files.
The arguments are:

n index of the nest

ftn unit number for the restart file

The routines invoked are:

drv_writevar_restart (2.22.3) writes a variable to the restart file

---

N.3.13 clm_readrestart (Source File: clm_readrestart.F90)

REVISION HISTORY:

```
20 Jan 2003; Sujay Kumar Initial Specification
```

INTERFACE:

```
subroutine clm_readrestart()
```

USES:

```
use spmdMod, only : npes
use lisdrv_module, only : lis
use clm_varder
use listime_mgr
use lis_logmod, only : logunit
use drv_output_mod, only : drv_readvar_restart
```

DESCRIPTION:

This program reads restart files for CLM. This includes all relevant water/energy storages and tile information.
The routines invoked are:

drv_readvar_restart (2.22.4) reads a variable from the restart file

---

N.3.14 clm_finalize (Source File: clm_finalize.F90)

REVISION HISTORY:

```
26 Oct 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine clm_finalize()
```

USES:

```
use lisdrv_module, only : lis
use clm_varder
```

DESCRIPTION:

This routine cleans up the allocated memory structures in CLM

N.3.15 clm_setwrfexport.F90 (Source File: clm_setwrfexport.F90)

LIS CLM data writer: Writes clm output in binary format

REVISION HISTORY:

```
02 Dec 2003; Sujay Kumar, Initial Version
```

INTERFACE:

```
subroutine clm_setwrfexport(n)
```

USES:

```
use lisdrv_module
use clm_varder
use clm_varcon, only : denh2o, denice, hvap, hsub, hfus, istwet
use clm_varpar, only : nlevsoi
use infnan
use lisWRFGridCompMod, only : liswrf_export
use drv_output_mod, only : drv_tile2grid

implicit none
integer :: n

integer :: i,j,k,t,m
real :: temp(lis%nch(n))
real :: asurft(lis%nch(n))
real :: snowt(lis%nch(n))
real :: soilmr(lis%nch(n))
real :: soilmtc(lis%nch(n))
real :: soilmoist(lis%nch(n),1:nlevsoi)
real :: q1(lis%nch(n))
real :: RCH
```

N.3.16 clm2_wrf_f2t (Source File: clm2_wrf_f2t.F90)

INTERFACE:

```
subroutine clm2_wrf_f2t(n)
```

USES:

```
use ESMF_Mod
use infnan
use clm_varder
use clm_varcon  , only : rair, cpair, po2, pco2, tcrit, tfrz
use lisdrv_module, only : lis
use baseforcing_module, only : FORC_State

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine transfers the LIS provided forcing onto the CLM model tiles, when used in the mode coupled to WRF.
The arguments are:

n index of the nest

N.3.17 clmlairead.F90: (Source File: clmlairead.F90)

This program reads in AVHRR LAI data for CLM

REVISION HISTORY:

```
27 Nov 2001: Jon Gottschalck; Initial code
20 Feb 2002: Jon Gottschalck; Modified to use for 1/4 and 2x2.5 using 1/8 degree monthly data
01 Oct 2002: Jon Gottschalck; Modified to add MODIS LAI data
```

INTERFACE:

```
subroutine clmlairead (n)
```

USES:

```
use lisdrv_module, only : lis
use clm_varder
use lai_module, only : lis_lai
use sai_module, only : lis_sai
```

O Mosaic Land Surface Model

This section describes the interface implementations for the Mosaic land surface model.

O.1 Fortran: Module Interface mos_module (Source File: mos_module.F90)

The code in this file provides a description of the data structure containing the mosaic 1-d variables. The variables specified in the data structure include:

vegt vegetation type of tile

soiltype soil type of tile

vegp static vegetation parameter values

vegip interpolated monthly parameter values

soilp static soil parameter values

lai leaf area index of the tile

green greeness fraction value of the tile

dsai stem area index of the tile

ct canopy/soil temperature

qa canopy humidity

ics interception canopy storage

snow snow depth

sot deep soil temperature

sowet soil wetness

tair 2m air temperature forcing

qair 2m specific humidity forcing

swdown downward shortwave forcing

lwdown downward longwave forcing

uwind u-wind component forcing

vwind v-wind component forcing

psurf surface pressure forcing

rainf total rainfall forcing

rainf_c convective rainfall forcing

snowf total snowfall forcing

swnet net shortwave radiation (W/m2)

lwnet net longwave radiation (W/m2)

qle latent heat flux (W/m2)

qh sensible heat flux (W/m2)

qg ground heat flux (W/m2)

snowfall snowfall (kg/m2s)

pcp rainfall (kg/m2s)

evap evapotranspiration (kg/m2s)

qs surface runoff (kg/m2s)

qsb subsurface runoff (kg/m2s)

qsm snow melt (kg/m2s)

avgsurft average surface temperature (K)

albedo surface albedo (-)

swe snow water equivalent (kg/m2)

soilmoist1 soil moisture for layer1 (kg/m2)

soilmoist2 soil moisture for layer2 (kg/m2)

soilmoist3 soil moisture for layer3 (kg/m2)

soilwet total column soil wetness (-)

ecanop interception evaporation (kg/m2s)

tveg vegetation transpiration (kg/m2s)

canopint total canopy water storage (kg/m2s)

rootmoist root zone soil moisture (kg/m2)

soilm_prev soil moisture from the previous model output

swe_prev snow water equivalent from the previous model output

water1 ??

water2 ??

water3 ??

dtcanal change in temperature based on analysis

REVISION HISTORY:

```
11 Feb 2002: Jon Gottschalck; Added AVHRR derived variables
25 Sep 2007: Sujay Kumar; Upgraded for LIS5.0
09 Nov 2007: Chuck Alonge; Removed nslay - unused
```

INTERFACE:

```
module mos_module
```

O.1.1 mos_varder (Source File: mos_varder.F90)

This module provides the definition of derived data type used to control the operation of Mosaic LSM. It also provides the entry method for the initialization of Mosaic-specific variables. The derived data type `mos_struc` includes the variables that specify the runtime options and other control variables as described below:

mos_rfile name of the mosaic restart file

mos_vfile name of the static vegetation parameter table

mos_sfile name of the soil parameter table

mos_mvfile name of monthly vegetation parameter file

mos_pfile name of general parameter file (levs, depths)

count variable to keep track of the number of timesteps before an output

mosopen variable to keep track of opened files

numout number of output times

mos_nvegp number of static vegetation parameters in the table

mos_nmvegp number of monthly vegetation parameters in the table

mos_ism initial soil moisture for a cold start run

mos_it initial soil temperature for a cold start run

outInterval output writing interval

rstInterval restart writing interval

mos Mosaic LSM specific variables

REVISION HISTORY:

```
Jun 2003; Jon Gottschalck, Initial Code
Sept 2007: Sujay Kumar, Upgraded the data structures for LIS 5.0
```

INTERFACE:

USES:

```
use mos_module
```

PUBLIC MEMBER FUNCTIONS:

```
public :: mos_varder_ini
```

PUBLIC TYPES:

```
public :: mos_struc
```

O.1.2 mos_varder_ini (Source File: mos_varder.F90)

INTERFACE:

```
subroutine mos_varder_ini()
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

This routine creates the datatypes and allocates memory for mosaic-specific variables. It also invokes the routine to read the runtime specific options for mosaic from the configuration file.
The routines invoked are:

readmoscrd (O.1.17)
reads the runtime options for Mosaic LSM

O.1.3 mos_setup (Source File: mos_setup.F90)

REVISION HISTORY:

```
3 Jun  2003: Jon Gottschalck; Initial Code
25 Sep 2007: Sujay Kumar; Upgraded for LIS5.0
```

INTERFACE:

```
subroutine mos_setup()
```

USES:

```
use lisdrv_module, only : lis
use mos_varder
use spmdMod, only : masterproc, npes

implicit none
```

DESCRIPTION:

This routine is the entry point to set up the parameters required for Mosaic LSM. These include the soils and topography and the initialization of state variables in Mosaic.
The routines invoked are:

setmosp (O.1.18)
initializes the static soil and topography fields

mapsib2umd (O.1.8)
maps the SIB vegetation types to UMD classification

mos_coldstart (O.1.9)
initializes the noah state variables

O.1.4 mos_f2t (Source File: mos_f2t.F90)

REVISION HISTORY:

```
 28 Jan 2002: Jon Gottschalck; Added option for different number of forcing variables
25 Sep 2007: Sujay Kumar; Upgraded for LIS5.0
```

INTERFACE:

```
subroutine mos_f2t(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module ,     only : lis
use baseforcing_module, only : FORC_State
use mos_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine transfers the LIS provided forcing onto the mosaic model tiles.
The arguments are:

n index of the nest

O.1.5 mos_readrestart (Source File: mos_readrestart.F90)

REVISION HISTORY: 19 Jan 2001: Brian Cosgrove; Initial Code 12 Dec. 2002: Brian Cosgrove; Fixed usage of Wiltpoint variable. Before, Wiltpoint1 and Wiltpoint2 were used in calculation of root zone soil moisture availability...now, only Wiltpoint2 is used since wiltpoint1 is not the correct wilting point needed for the calculation. 23 Jan 2003: Urszula Jambor; Switch index order of GEOS forcing array. Snow is 12, soil wetness is 13. 25 Sep 2007: Sujay Kumar, Upgraded for LIS 5.0

INTERFACE:

```
subroutine mos_readrestart
```

USES:

```
use lisdrv_module, only : lis
use mos_varder
use drv_output_mod, only : drv_readvar_restart
use lis_logmod,     only : logunit
```

DESCRIPTION:

This program reads restart files for Mosaic. This includes all relevant water/energy storages and tile information. The following is the list of variables specified in the Mosaic restart file:

```
nc,nr,nch    - grid and tile space dimensions
ct           - mosaic canopy temperature
qa           - mosaic canopy humidity
ics          - mosaic interception canopy storage
snow         - mosaic snow depth
sot          - mosaic deep soil temperature
soWet        - mosaic soil wetness
```

The routines invoked are:

drv_readvar_restart (2.22.4)
reads a variable from the restart file

O.1.6 mos_main (Source File: mos_main.F90)

REVISION HISTORY: 11 Apr 2000: Brian Cosgrove; Initial Code 12 May 2000: Brian Cosgrove/Jared Entin; Added code to prevent numerical instablility due to small values of wind and humidity 22 Aug. 2000: Brian Cosgrove; Modified code for output of standard LDAS output variables. Added LDAS and MOS modules into call for MOSTILE and also added many new variables in the MOS25 Aug. 2000: Brian Cosgrove; Fixed snowcover fraction output variable 21 Sep. 2000: Brian Cosgrove; Fixed code so that rain output field is set to zero when snow output field is greater than zero 27 Sep. 2000: Brian Cosgrove; Fixed code so that 1 meter soil moisture is correctly calculated (FACTOR variable had been incorrectly computed before this fix) 23 Mar. 2001: Jon Radakovich; Updated for PSAS temperature assimilation 19 Apr. 2001 Updated scheme for calculating albedo, using sibalb module 05 Sep. 2001: Brian Cosgrove; Added in volumetric variables soilwm,soilww for use in LDAS output. Zthick changed from 50 to 10 sometime by someone else. Added volumetric output at 4 more levels corresponding to OK mesonet levels. Changed calculation of soil wetness output variables 05 Feb. 2002: Brian Cosgrove; Changed Zthick back to 50 from 10 after talk with Jon Radokovich. 10 is correct for observation height, but was causing fluxes that were too high, so changed back to 'incorrect' value of 50 22 Apr. 2002: Urszula Jambor; Added conditional to suppress calculation of dewpt temp. if using Aaron Berg's reanalysis ECMWF data. 13 Sep. 2002: Urszula Jambor; Reversed suppression of dew point temp. calculation if using Aaron Berg's reanalysis ECMWF data. 27 Nov. 2002: Urszula Jambor; Restricted mosto prevent problems in GRIB output snow fields. 12 Dec. 2002: Brian Cosgrove; Fixed usage of Wiltpoint variable. Before, Wiltpoint1 and Wiltpoint2 were used in calculation of root zone soil moisture availability...now, only Wiltpoint2 is used since wiltpoint1 is not the correct wilting point needed for the calculation. 25 Sep 2007: Sujay Kumar; Upgraded for LIS5.0

INTERFACE:

```
subroutine mos_main(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use mos_varder
use sibalb_module
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
DESCRIPTION:
 This is the entry point for calling the Mosaic LSM physics. This routine
 calls the {\tt MOSTILE} routine that performs the land surface
 computations, to solve for water and energy equations. For documentation
 of the {\tt MOSTILE} and other Mosaic routines, please see the reference:
 Koster, R. D., and M. J. Suarez, Energy and Water Balance
 Calculations in the MOSAIC LSM. NASA Technical Memorandum
 104606, 9, 76 pp., 1996.

 The arguments are:
 \begin{description}
 \item[n]
  index of the nest
 \end{description}
```

O.1.7 mos_binout (Source File: mos_binout.F90)

REVISION HISTORY:

```
02 Dec 2003; Sujay Kumar, Initial Version
25 Sep 2007: Sujay Kumar; Upgraded for LIS5.0
```

INTERFACE:

```
subroutine mos_binout(ftn,ftn_stats,n)
```

USES:

```
use lisdrv_module, only : lis,metadata_output
use drv_output_mod, only : drv_writevar_bin
use constantsMod,   only : CONST_TKFRZ, CONST_LATVAP
use mos_varder
use lis_logmod,     only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn,ftn_stats
```

DESCRIPTION:

This routine writes the specified mos variables into a binary output file.
The arguments are:

n index of the nest

ftn unit number for the model output file

ftn_stats unit number for the stats file

The routines invoked are:

drv_writevar_bin (2.22.2)
routine to write a variable to a binary output file

O.1.8 mapsib2umd (Source File: mapsib2umd.F90)

REVISION HISTORY:

```
13 Apr 2001: Urszula Jambor; Initial code, based on calc_albedo.f scheme
03 Feb 2002: Jon Gottschalck; Added sections to use Koster tilespace
             (may be better to put the tile space distinction in
             umd_sibalb.F90 at a later time)
```

INTERFACE:

```
subroutine mapsib2umd ()
```

USES:

```
use sibalb_module    ! SiB-based coefficients for albedo calculation.

implicit none
DESCRIPTION:
 This routine assigns then maps 7 SiB vegetation types to 13 UMD
 vegetation types, defining 4 coefficient arrays.
 **NOTE: current form of this routine only has mapping based on North
 America, and not also for the globe.**
```

O.1.9 mos_coldstart (Source File: mos_coldstart.F90)

REVISION HISTORY:

```
3 Jun  2003: Jon Gottschalck; Initial code
25 Sep 2007: Sujay Kumar; Upgraded for LIS5.0
```

INTERFACE:

```
subroutine mos_coldstart(n)
```

USES:

```
use lisdrv_module, only : lis
use mos_varder
use listime_mgr
use spmdMod, only : iam
use lis_logmod, only : logunit
```

DESCRIPTION:

This routine initializes the mosaic state variables with some predefined values uniformly for the entire domain. These initial values will be overwritten by the values read from the supplied Mosaic model restart file.

O.1.10 mosdynp (Source File: mosdynp.F90)

REVISION HISTORY:

```
6  Apr 2001: Matt Rodell; Initial Specification
11 Feb 2002: Jon Gottschalck; Added use of AVHRR derived LAI/Greenness
01 Oct 2002: Jon Gottschalck; Modified to allow for MODIS LAI
25 Sep 2007: Sujay Kumar; Upgraded for LIS5.0
```

INTERFACE:

```
subroutine mosdynp(n)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use mos_varder
use listime_mgr
use lai_module,    only : lis_lai
use sai_module,    only : lis_sai

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This subroutine take all the monthly varying parameters and the date and determine the actual value of the parameter for that date this actual value is returned to the main program The assumption is that the data point is valid for the 16th of the given month at 00hr

---

O.1.11 mos_gribout (Source File: mos_gribout.F90)

REVISION HISTORY:

```
02 Dec 2003; Sujay Kumar, Initial Version
29 Dec 2006; Charles Alonge, Added Parameter Output and additional
             grib information (projection, time range and intervals)
```

INTERFACE:

```
subroutine mos_gribout(ftn,ftn_stats,n)
```

USES:

```
use lisdrv_module, only : lis,lisdom,metadata_output
use drv_output_mod, only : drv_writevar_grib
use mos_varder
use constantsMod,   only : CONST_TKFRZ, CONST_LATVAP
use spmdMod, only : masterproc
use grib_module

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn,ftn_stats
```

DESCRIPTION:

This routine writes the specified mos variables into a grib output file.
The arguments are:

n index of the nest

ftn unit number for the model output file

ftn_stats unit number for the stats file

The routines invoked are:

drv_writevar_grib (2.22.6)
routine to write a variable to a grib output file

grib_init (2.24.1)
routine to initialize grib data structures

grib1_setup (2.24.3)
routine to initialize Section 1 (PDS) of grib record

grib2_setup (2.24.4)
routine to initialize Section 2 (GDS) of grib record

grib3_setup (2.24.5)
routine to initialize Section 3 (BMS) of grib record

grib1_finalize (2.24.2)
routine to complete Section 1 (PDS) of grib record

O.1.12 mos_output (Source File: mos_output.F90)

REVISION HISTORY:

```
25 Sep 2007: Sujay Kumar; Upgraded for LIS5.0
```

INTERFACE:

```
subroutine mos_output(n)
```

USES:

```
use lisdrv_module, only : lis, metadata_output
use mos_varder
use spmdMod
use lis_fileIOMod, only : create_output_directory, &
     create_output_filename,  &
     create_stats_filename
use lis_logmod,   only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This subroutines sets up methods to write model output from Mosaic LSM A timestamped filename and directory are created first. The routine supports model output binary format.
The arguments are:

n index of the nest

The routines invoked are:

create_output_directory (2.25.2)
creates a time stamped output directory

create_output_filename (2.25.3)
creates a time stamped output filename

create_stats_filename (2.25.5)
creates a stats filename

mos_binout (O.1.7)
routine to write Mosaic model output in binary format

mos_gribout (O.1.11)
routine to write Mosaic model output in grib format

mos_totint (O.1.13)
resets the time averaged variables

O.1.13 mos_totinit (Source File: mos_totinit.F90)

REVISION HISTORY:

```
1 Aug 2003  Sujay Kumar  Initial Specification
25 Sep 2007: Sujay Kumar; Upgraded for LIS5.0
```

INTERFACE:

```
subroutine mos_totinit(n)
```

USES:

```
use mos_varder      ! Mosaic LSM module
use lisdrv_module, only : lis

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine resets the time-averaged Mosaic variables after a model output. The routine is called after a model output call.

O.1.14 mos_writerst (Source File: mos_writerst.F90)

REVISION HISTORY: 19 Jan 2001: Brian Cosgrove; Initial Code 12 Dec. 2002: Brian Cosgrove; Fixed usage of Wiltpoint variable. Before, Wiltpoint1 and Wiltpoint2 were used in calculation of root zone soil moisture availability...now, only Wiltpoint2 is used since wiltpoint1 is not the correct wilting point needed for the calculation. 23 Jan 2003: Urszula Jambor; Switch index order of GEOS forcing array. Snow is 12, soil wetness is 13. 25 Sep 2007: Sujay Kumar, Upgraded for LIS 5.0 INTERFACE:

```
subroutine mos_writerst(n)
```

USES:

```
use spmdmod, only       : masterproc
use lisdrv_module, only : lis
use mos_varder, only    : mos_struc
use lis_logmod, only    : logunit
use lis_fileIOMod, only : create_output_directory, &
     create_restart_filename

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This program writes restart files for Mosaic. This includes all relevant water/energy storage and tile information
The routines invoked are:

create_output_directory (2.25.2)
creates a timestamped directory for the restart files

create_restart_filename (2.25.4)
generates a timestamped restart filename

mos_dump_restart (O.1.15)
writes the Mosaic variables into the restart file

O.1.15 mos_dump_restart (Source File: mos_writerst.F90)

REVISION HISTORY:

```
25 Sep 2007: Sujay Kumar, Initial Specification
```

INTERFACE:

```
subroutine mos_dump_restart(ftn,n)
```

USES:

```
use spmdmod, only        : masterproc
use lisdrv_module, only  : lis
use mos_varder
use drv_output_mod, only : drv_writevar_restart

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn
```

DESCRIPTION:

This routine gathers the necessary restart variables and performs the actual write statements to create the restart files.
The arguments are:

n index of the nest

ftn unit number for the restart file

The following is the list of variables written in the Mosaic restart file:

```
nc,nr,nch    - grid and tile space dimensions
ct           - mosaic canopy temperature
qa           - mosaic canopy humidity
ics          - mosaic interception canopy storage
snow         - mosaic snow depth
sot          - mosaic deep soil temperature
soWet        - mosaic soil wetness
```

The routines invoked are:

drv_writevar_restart (2.22.3)
writes a variable to the restart file

O.1.16 paramb (Source File: paramb.F90)

REVISION HISTORY:

```
11 Mar 2001 : Matt Rodell, Initial Specification
```

INTERFACE:

```
subroutine paramb (nch,tex,b)

implicit none
```

ARGUMENTS:

```
integer :: nch
integer :: tex(nch)
real   :: b(nch)
```

DESCRIPTION:

This subroutine assigns the dimensionless b parameter based on values modified from the soil texture class and the table of cosby et al. [1984].
The arguments are:

nch number of tiles

tex texture class

b b parameter

O.1.17 readmoscrd (Source File: readmoscrd.F90)

REVISION HISTORY:

```
15 Oct 2003; Sujay Kumar, Initial Code
25 Sep 2007: Sujay Kumar, Upgraded for LIS 5.0
12 Nov 2007: Chuck Alonge, Added entry # of soil textures
```

INTERFACE:

```
subroutine readmoscrd()
```

USES:

```
use ESMF_Mod
use lis_logmod, only : logunit
use lisdrv_module, only : lis, config_lis
use mos_varder, only : mos_struc
```

DESCRIPTION:

This routine reads the options specific to Mosaic LSM from the LIS configuration file.

O.1.18 setmosp (Source File: setmosp.F90)

REVISION HISTORY:

```
31 Jul 2001: Matt Rodell; Initial Specification
14 Feb 2002: Jon Gottschalck; Added allocated space for AVHRR LAI/DSAI
07 Mar 2002: Brian Cosgrove; Corrected declaration of TEX1 var from real to int
14 Jan 2003: Urszula Jambor; Added conditional to check if need exists
             to allocate for AVHRR LAI/DSAI variables.
25 Sep 2007: Sujay Kumar; Upgraded for LIS5.0
```

INTERFACE:

```
subroutine setmosp(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use mos_varder
use soils_module, only : lis_soils
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This subroutine sets the static mosaic parameters related to vegetation, soils and topography.

O.2 Fortran: Module Interface sibalb_module (Source File: sibalb_module.F90)

In order to use the MOSAIC-SiB albedo calculation, SiB vegetation types are mapped to UMD vegetation types. For both SiB and UMD, 4 coefficient arrays are defined for calculating the albedos of soil in the subroutine sibalb, representing the following bands: visible, direct solar radiation, infra-red, direct solar radiation, visible, diffuse solar radiation, and infra-red, diffuse solar radiation.
MOSAIC/SiB ITYP: Vegetation type as follows: 1: BROADLEAF EVERGREEN TREES 2: BROADLEAF DECIDUOUS TREES 3: NEEDLELEAF TREES 4: GROUND COVER 5: BROADLEAF SHRUBS 6: DWARF TREES (TUNDRA) 7: BARE SOIL
UMD ITYP: Vegetation type as follows: 1. Evergreen Needleleaf Forest 2. Evergreen Broadleaf Forest 3. Deciduous Needleleaf Forest 4. Deciduous Broadleaf Forest 5. Mixed Cover 6. Woodland 7. Wooded Grassland 8. Closed Shrubland 9. Open Shrubland 10. Grassland 11. Cropland 12. Bare Ground 13. Urban and Built-Up

REVISION HISTORY:

```
04 Apr 2001: Urszula Jambor; Initial code, using old calc_albedo.f scheme
```

O.2.1 texture (Source File: texture.F90)

REVISION HISTORY:

```
7 Mar 2001 : Matt Rodell, Initial Specification
```

INTERFACE:

```
subroutine texture (n, nc, nr, nch, sand,silt,clay,tex)
```

USES:

```
use lisdrv_module, only : lisdom, lis

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
integer, intent(in)    :: nc,nr,nch
real,    intent(in)    :: sand(nc,nr),silt(nc,nr),clay(nc,nr)
integer, intent(inout) :: tex(nch)
```

DESCRIPTION:

Finds the USDA texture class based on percentages of sand, silt, and clay.

O.2.2 mos_finalize (Source File: mos_finalize.F90)

REVISION HISTORY:

```
04 Oct 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine mos_finalize()
```

USES:

```
use lisdrv_module, only : lis
use mos_varder
```

DESCRIPTION:

This routine cleans up the allocated memory structures in Mosaic

P HySSIB Land Surface Model

This section describes the interface implementations for the HySSIB land surface model.

P.1 Fortran: Module Interface hyssib_module (Source File: hyssib_module.F90)

The code in this file provides a description of the data structure containing the hyssib 1-d variables. The variables specified in the data structure include:

ts time step in seconds (integer)

maxt maximum number of tiles per grid

vegt HYSSIB vegetation class index value

count number of timesteps between outputs

snowtcount number of timesteps with snow (for averaging)

albedocount number of timesteps for albedo averaging

sliqfraccount number of timesteps for snow cover averaging

vegp static monthly vegetation parameters

vegip monthly albedo vegetation lookup table values

albsf1 month 1 albedo value

albsf2 month 2 albedo value

albsf interpolated albedo value

tempbot bottom temperature value

tempbot1 month 1 bottom temp value

tempbot2 month 2 bottom temp value

tempstd standard deviation of topography (m)

sgfg flag which indicates if snow model is active

sdens snow density

tc canopy temperature

tg ground temperature

tsn temperature of the snow pack

www soil moisture (volumetric) for each soil layer (3)

capac canopy/ground water content (1/2)

snow canopy/ground snow water equivalent

swnet net shortwave radiation flux

lwnet net longwave radiation flux

qle latent heat flux

qh sensible heat flux

qg ground heat flux

qf energy of fusion

qv energy of sublimation

qtau momentum flux from/to surface

qa advective energy

delsurfheat change in surface heat storage

delcoldcont change in snow cold content

snowf snowfall

rainf rainfall

evap total evaporation

qs surface runoff

qrec recharge

qsb subsurface runoff

qsm snowmelt

qfz re-freezing of water in the snow

qst snow throughfall

delsoilmoist change in soil moisture

delswe change in snow water equivalent

delintercept change in interception storage

snowt snow surface temperature (K)

vegtc vegetation canopy temperature

radteff surface radiative temperature

albedo surface albedo

swe snow water equivalent

sweveg snow water equivalent intercepted by veg

soilmoist1 soil moisture layer 1 (kg/m2)

soilmoist2 soil moisture layer 2 (kg/m2)

soilmoist3 soil moisture layer 3 (kg/m2)

soiltemp soil temperature (K)

soilwet total column soil wetness

potevap potential evaporation

ecanop interception evaporation

tveg transpiration

esoil bare soil evaporation

rootmoist root zone soil moisture

canopint canopy interception

subsnow snow sublimation

subsurf sublimation of snow free area

acond areodynamic conductance

ccond canopy conductance

snowfrac percent of tile covered by snow

sliqfrac snow liquid fraction

ect transpiration from canopy

eci evaporation of canopy interepted water

egs evaporation of ground intercepted water

hc sensible heat flux canopy

hg sensible heat flux ground

radnvisdir net radiation

radnvisdif net radiation

radnnirdir net radiation

radnnifdif net radiation

salbvisdir visible component of surface albedo - direct

salbvisdif visible component of surface albedo - diffuse

salbnirdir near ir component of surface albedo - direct

salbnirdif near ir component of surface albedo - diffuse

radtc net radiation canopy

radtg net radiation ground

watcan canopy interception

watgrd ground interception

snocan swe in canopy

snogrd swe on ground

wet1 soil wetness layer 1

wet2 soil wetness layer 2

wet3 soil wetness layer 3

tair air temperature forcing

qair specific humidity forcing

swdown downward shortwave forcing

lwdown downward longwave forcing

uwind u-wind component forcing

vwind v-wind component forcing

psurf surface pressure forcing

rainf_in total rainfall forcing

rainf_cp convective rainfall forcing

snowf_in total snowfall forcing

REVISION HISTORY:

```
28 Apr 2002: K. Arsenault, added NOAH LSM 2.5 code to LDAS
14 Nov 2002: Sujay Kumar, Optimized version for LIS
21 Apr 2004: David Mocko, Conversion from NOAH to HY-SSiB
25 Aug 2007: Chuck Alonge, Updates for LIS 5.0 compliance
```

P.2 Fortran: Module Interface hyssib_varder (Source File: hyssib_varder.F90)

This module provides the definition of derived data type used to control the operation of Hyssib LSM. It also provides the entry method for the initialization of Hyssib-specific variables. The derived data type `hyssib_struc` includes the variables that specify the runtime options and other control variables as described below:

rfile name of the hyssib restart file

vfile name of the static vegetation parameter table

topostdfile name of the file for std. dev. of topography

count variable to keep track of the number of timesteps before an output

hyssibopen variable to keep track of opened files

numout number of output times

nvegp number of static vegetation parameters in the table

nvegip number of vegetation based albedo parameters in the table

npr number of prognostic state variables used in data assimilation

tbotAlarmTime alarm time to keep track of TBOT data reading frequency

tbotInterval data interval of the TBOT data

statevar_avg flag to average states for output or not 1/0

initsm initial soil moisture for a cold start run

initTemp initial soil temperature for a cold start run

outInterval output writing interval

rstInterval restart writing interval

hyssib hyssib LSM tile specific variables

REVISION HISTORY:

```
    Apr 2003: Sujay Kumar, Initial Code
 21 Feb 2004: David Mocko, Conversion from NOAH to HY-SSiB
 25 Aug 2007: Chuck Alonge, Updates for LIS 5.0 Compliance
```

USES:

```
use hyssib_module

implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: hyssib_varder_ini
```

PUBLIC TYPES:

```
public :: hyssib_struc
```

P.2.1 hyssib_varder_ini (Source File: hyssib_varder.F90)

INTERFACE:

```
subroutine hyssib_varder_ini()
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

Reads in runtime HY-SSiB parameters, allocates memory for variables
The routines invoked are:

hyssib_readcrd (P.4.11)
reads the runtime options for Hyssib LSM

P.3 Fortran: Module Interface hyssibalb_module (Source File: hyssibalb_module.F90)

In order to use the HySSiB albedo calculations, UMD vegetation types need to be mapped to SSiB vegetation types. The code in this file provides a description of the data structure used to contain the vegetation specific constants used in HySSiB's albedo calculations. The varaiables specified in the data structure include:

cedfu

cedir

cedfu1

cedir1

cedfu2

xmiu

cedir2

clefu

cledir

cether

xmiw

REVISION HISTORY:

```
    01 Dec 2007: Chuck Alonge; Initial Specification
   implicit none
  ------------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
  ------------------------------------------------------------------------------
   public :: hyssibalb_ini
  ------------------------------------------------------------------------------
```

PUBLIC TYPES:

```
  ------------------------------------------------------------------------------
   public :: hyssibalbdec
```

P.3.1 hyssibalb_ini (Source File: hyssibalb_module.F90)

INTERFACE:

```
   subroutine hyssibalb_ini()
```

USES:

```
     use lisdrv_module, only : lis
     use hyssib_varder ! HY-SSiB tile variables
```

DESCRIPTION:

Reads in runtime HY-SSiB albedo parameters to populate constants used in Hyssib albedo calculations

---

P.4 Fortran: Module Interface hyssibveg_module (Source File: hyssibveg_module.F90)

In order to use the HySSiB vegetation table data efficiently, SiB vegetation parameter data (static and monthly) are read into this module to reduce model i/o when updating monthly values. The varaiables specified in this data structure include:

rstpar par influence on stomatal resist. coefficients (rst)

chil leaf angle distribution factor

topt optimum temperature for rst calculation

tll bottom temperature for rst calculation

tu top temperature for rst calculation

defac dew factor for rst calculation

ph1 stome slope factor

ph2 point at which stomates close

rootd rooting depth for canopy/ground cover

bee Clapp-Hornberger empirical constant

phsat soil moisture potential at saturation

satco saturation hydraulic conductivity

poros soil porosity

zdepth exact independent depth of 3 soil layers

slope avg. topographic slope in

green green leaf fraction

vcover fraction of vegetation cover [fpar/greenness]

zlt leaf area index

z0 roughness height

z2 canopy top height

z1 canopy base height

rdc ground to canopy air space resistance coeff.

rbc bulk canopy boundary layer resistance coeff.

REVISION HISTORY:

```
    01 Dec 2007: Chuck Alonge; Initial Specification
   implicit none
  ------------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
  ------------------------------------------------------------------------------
   public :: hyssibveg_ini
  ------------------------------------------------------------------------------
```

PUBLIC TYPES:

```
  ------------------------------------------------------------------------------
   public :: hyssibvegdec
```

P.4.1 hyssibveg_ini (Source File: hyssibveg_module.F90)

INTERFACE:

```
subroutine hyssibveg_ini()
```

USES:

```
use lisdrv_module, only : lis
use hyssib_varder ! HY-SSiB model control variables
```

DESCRIPTION:

Reads in runtime HY-SSiB vegetation parameters to populate constant and monthly varying values Hyssib calculations

---

P.4.2 hyssib_setup (Source File: hyssib_setup.F90)

REVISION HISTORY:

```
21 Apr 2004: David Mocko, Conversion from NOAH to HY-SSiB
27 Sep 2007: Chuck Alonge, Updates for LIS 5.0
```

INTERFACE:

```
subroutine hyssib_setup()
```

USES:

```
use lisdrv_module, only : lis
use hyssib_varder
use hyssibveg_module
use hyssibalb_module
use listime_mgr, only : setMonthlyAlarm
```

DESCRIPTION:

This routine is the entry point to set up the parameters required for Hyssib LSM. These include the vegetation, albedo, bottom temperature and the initialization of state variables in Hyssib.
The routines invoked are:

hyssibveg_ini (P.4.1)
: initializes the Hyssib vegetation parameter module

hyssibalb_vegini (P.3.1)
: initializes the Hyssib albedo parameter module

hyssib_setvegparms (P.4.18)
: initializes the vegetation-related parameters in Hyssib

hyssib_settbot (P.4.16)
: initializes the bottom temperature fields

hyssib_settopostd (P.4.17)
: initializes the std. dev. of topography in hyssib

hyssib_gfrac (P.4.13)
: initializes monthly vegetation parameters in hyssib

hyssib_coldstart (P.4.12)
initializes the hyssib state variables

setMonthlyAlarm (2.4.13)
sets a monthly alarm if a climatology of bottom temperature is being read

---

P.4.3 hyssib_dynsetup (Source File: hyssib_dynsetup.F90)

REVISION HISTORY:

```
15 Apr 2002: Sujay Kumar, Initial Specification
23 Apr 2004: David Mocko, Conversion from NOAH to HY-SSiB
25 Aug 2007: Chuck Alonge, Updates for LIS 5.0 compliance
```

INTERFACE:

```
subroutine hyssib_dynsetup(n)
```

DESCRIPTION:

This routine sets up the time-dependent variables in Hyssib. Currently this routine is a placeholder

---

P.4.4 hyssib_f2t (Source File: hyssib_f2t.F90)

REVISION HISTORY:

```
21 Apr 2004: David Mocko, Conversion from NOAH to HY-SSiB
25 Aug 2007: Chuck Alonge, Updates for LIS 5.0 Compliance
```

INTERFACE:

```
subroutine hyssib_f2t(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module , only : lis
use baseforcing_module, only : FORC_State
use hyssib_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine transfers the LIS provided forcing onto the Hyssib model tiles.
The arguments are:

n index of the nest

P.4.5 hyssib_output (Source File: hyssib_output.F90)

REVISION HISTORY:

```
21 Apr 2004: David Mocko, Conversion from NOAH to HY-SSiB
25 Aug 2007: Chuck Alonge, Updates for LIS 5.0 compliance
```

INTERFACE:

```
subroutine hyssib_output(n)
```

USES:

```
#if ( defined USE_NETCDF )
  use netcdf
#endif
  use lisdrv_module, only : lis,metadata_output
  use grid_module           ! LIS non-model-specific grid variables
  use hyssib_varder         ! HY-SSiB-specific variables
  use spmdMod
  use lis_fileIOMod, only : create_output_directory, &
       create_output_filename,  &
       create_stats_filename
  use lis_logmod,  only   : logunit

  implicit none

  integer, intent(in) :: n
```

DESCRIPTION:

This subroutines sets up methods to write model output from Hyssib LSM A timestamped filename and directory are created first. The routine supports model output in three data formats: Binary, and Grib. The arguments are:

n index of the nest

The routines invoked are:

create_output_directory (2.25.2)
creates a time stamped output directory

create_output_filename (2.25.3)
creates a time stamped output filename

create_stats_filename (2.25.5)
creates a stats filename

hyssib_binout (P.4.6)
routine to write Hyssib model output in binary format

hyssib_totint (P.4.7)
resets the time averaged variables

P.4.6 hyssib_binout (Source File: hyssib_binout.F90)

REVISION HISTORY:

```
02 Dec 2003: Sujay Kumar, Initial Version
15 Feb 2004: David Mocko, Conversion from NOAH to HY-SSiB
27 Nov 2007: Sujay Kumar, Updated for LIS5.0
```

INTERFACE:

```
subroutine hyssib_binout(ftn, ftn_stats, n)
```

USES:

```
use lisdrv_module, only : lis, metadata_output
use drv_output_mod, only : drv_writevar_bin
use hyssib_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn,ftn_stats
```

DESCRIPTION:

This routine writes the specified hyssib variables into a binary output file.
The arguments are:

n index of the nest

ftn unit number for the model output file

ftn_stats unit number for the stats file

The routines invoked are:

drv_writevar_bin (2.22.2)
routine to write a variable to a binary output file

P.4.7 hyssib_totinit (Source File: hyssib_totinit.F90)

REVISION HISTORY:

```
14 Jun 2002: Sujay Kumar, Initial Specification
21 Apr 2004: David Mocko, Conversion from NOAH to HY-SSiB
28 Aug 2007: Chuck Alonge, Updates for LIS 5.0
```

INTERFACE:

```
subroutine hyssib_totinit(n)
```

USES:

```
use hyssib_varder
use lisdrv_module, only : lis

implicit none

integer, intent(in) :: n
```

DESCRIPTION:

This routine resets the time-averaged Hyssib variables after a model output. The routine is called after a model output call. The arguments are:

n index of the nest

P.4.8 hyssib_writerst (Source File: hyssib_writerst.F90)

REVISION HISTORY:

```
21 Apr 2004: David Mocko, Conversion from NOAH to HY-SSiB
25 Aug 2007: Chuck Alonge, Updates for LIS 5.0 Compliance
RESTART FILE FORMAT(fortran sequential binary):
 YR,MO,DA,HR,MN,SS,VCLASS,NCH !Restart time,Veg class,no.tiles, no.soil lay
 TILE(NCH)%COL        !Grid Col of Tile
 TILE(NCH)%ROW        !Grid Row of Tile
 TILE(NCH)%FGRD       !Fraction of Grid covered by Tile
 TILE(NCH)%VEGT       !Vegetation Type of Tile
 HYSSIB(NCH)%STATES   !Model States in Tile Space
```

INTERFACE:

```
subroutine hyssib_writerst(n)
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr
use hyssib_varder
use lis_logmod, only :logunit
use lis_fileIOMod, only : create_output_directory, &
                          create_restart_filename
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This program writes restart files for Hyssib. This includes all relevant water/energy storage and tile information
The routines invoked are:

create_output_directory (2.25.2)
creates a timestamped directory for the restart files

create_restart_filename (2.25.4)
generates a timestamped restart filename

hyssib_dump_restart (P.4.9)
writes the hyssib variables into the restart file

P.4.9 hyssib_dump_restart (Source File: hyssib_writerst.F90)

REVISION HISTORY:

```
24 Aug 2004: James Geiger, Initial Specification
27 Sep 2005: Chuck Alonge, Updates for LIS 5.0
```

INTERFACE:

```
subroutine hyssib_dump_restart(n, ftn)
```

USES:

```
use lisdrv_module, only : lis
use hyssib_varder
use listime_mgr
use drv_output_mod, only : drv_writevar_restart

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn
```

DESCRIPTION:

This routine gathers the necessary restart variables and performs the actual write statements to create the restart files.
The arguments are:

n index of the nest

ftn unit number for the restart file

The following is the list of variables written in the Hyssib restart file:

```
nc,nr,nch    - grid and tile space dimensions
tc           - hyssib canopy temperature
tg           - hyssib ground temperature
tsn          - hyssib snow on ground temperature
td           - hyssib deep soil temperature
www(3)       - hyssib soil wetness (for each layer)
capac(2)     - hyssib liq equiv. water&snow on canopy/ground
snow(2)      - hyssib snow storage (swe) on snow/ground
sgfg         - hyssib density of bulk snow layer
sdens        - hyssib momentum exchange coefficient
```

The routines invoked are:

drv_writevar_restart (2.22.3)
writes a variable to the restart file

P.4.10 hyssib_readrst (Source File: hyssib_readrst.F90)

REVISION HISTORY:

```
21 Apr 2004: David Mocko, Conversion from NOAH to HY-SSiB
25 Aug 2007: Chuck Alonge, Updates for LIS 5.0 Compliance
```

INTERFACE:

```
subroutine hyssib_readrst
```

USES:

```
use spmdMod, only : iam,masterproc
use lisdrv_module, only : lis
use hyssib_varder
use listime_mgr
use drv_output_mod, only : drv_readvar_restart
use lis_logmod, only : logunit
```

DESCRIPTION:

This program reads restart files for HY-SSiB. This includes all relevant water/energy storages, tile information, and time information. The following is the list of variables specified in the Hyssib restart file:

```
nc,nr,nch    - grid and tile space dimensions
tc           - hyssib canopy temperature
tg           - hyssib ground temperature
tsn          - hyssib snow on ground temperature
td           - hyssib deep soil temperature
www(3)       - hyssib soil wetness (for each layer)
capac(2)     - hyssib liq equiv. water&snow on canopy/ground
snow(2)      - hyssib snow storage (swe) on snow/ground
sgfg         - hyssib density of bulk snow layer
sdens        - hyssib momentum exchange coefficient
```

The routines invoked are:

drv_readvar_restart (2.22.4)
reads a variable from the restart file

P.4.11 hyssib_readcrd (Source File: hyssib_readcrd.F90)

REVISION HISTORY:

```
14 Oct 2003: Sujay Kumar, Initial Code
21 Apr 2004: David Mocko, Conversion from NOAH to HY-SSiB
 5 Sep 2007: Chuck Alonge, Updates for LIS 5.0
```

INTERFACE:

```
subroutine hyssib_readcrd()
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
use spmdMod, only : masterproc
use hyssib_varder, only : hyssib_struc
use lis_logmod, only :logunit
use lisdrv_module, only : config_lis
```

DESCRIPTION:

Routine to read HY-SSiB specific parameters from the LIS configuration file

---

P.4.12 hyssib_coldstart (Source File: hyssib_coldstart.F90)

REVISION HISTORY:

```
15 Dec 2003: Luis-Gustavo Goncalves, Initial version
29 Apr 2004: David Mocko, Conversion from NOAH to HY-SSiB
25 Aug 2007: Chuck Alonge, Updates for LIS 5.0 compliance
```

INTERFACE:

```
subroutine hyssib_coldstart()
```

USES:

```
use lisdrv_module, only: lis
use hyssib_varder    ! HY-SSiB tile variables
use listime_mgr
use spmdMod, only: iam,masterproc
use lis_logmod, only : logunit
```

DESCRIPTION:

This routine initializes the hyssib state variables with some predefined values uniformly for the entire domain. These initial values will be overwritten by the values read from the supplied HYSSIB model restart file.

P.4.13 hyssib_gfrac.F90 (Source File: hyssib_gfrac.F90)

REVISION HISTORY:

```
28 Apr 2002: Kristi Arsenault, Added SSiB LSM to LDAS, initial code
15 Feb 2004: David Mocko, Conversion from SSiB to HY-SSiB
28 Oct 2007: Chuck Alonge, Updates for LIS 5.0
```

INTERFACE:

```
subroutine hyssib_gfrac()
```

USES:

```
use lisdrv_module, only : lis, lisdom
use hyssib_varder
use hyssibveg_module
use listime_mgr
```

DESCRIPTION:

This subroutine takes vegetation greenness fraction data and the date to interpolate and determine the actual value of the greenness fraction for that date. This actual value is then returned to the main program. The assumption is that the data point is valid for the 16th of the given month, at 00Z. Gustavo will take advantage of this routine to read the other time varying parameters and interpolate them to the appropriate time frame

---

P.4.14 hyssib_main (Source File: hyssib_main.F90)

REVISION HISTORY:

```
21 Apr 2004: David Mocko, Initial Code
30 Apr 2004: David Mocko, Revisions to fix MPI version
25 November: Chuck Alonge, Updates for LIS 5.0
```

INTERFACE:

```
subroutine hyssib_main(n)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use albedo_module, only : lis_alb
use hyssib_varder  ! HY-SSiB tile variables
use hyssibalb_module
use lis_logmod,    only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)  :: n
```

DESCRIPTION:

********** *** This code is the offline HY-SSiB Model (Hydrology with Simple SiB) Authors: Y.C. Sud and David M. Mocko, NASA Goddard Space Flight Center Code Maintenance by: David Mocko ¡mocko@climate.gsfc.nasa.gov¿
Referred Papers:
SMP1: Sud, Y.C., and D.M. Mocko, 1999: New snow-physics to complement SSiB. Part I: Design and evaluation with ISLSCP Initiative I datasets. J. Meteor. Soc. Japan, Vol. 77, No. 1B, 335-348.
SMP2: Mocko, D.M., Walker, G.K., and Y.C. Sud, 1999: New snow-physics to complement SSiB. Part II: Effects on soil moisture initialization and simulated surface fluxes, precipitation and hydrology of GEOS II GCM. J. Meteor. Soc. Japan, Vol. 77, No. 1B, 349-366.
SMP3: Mocko, D.M., and Y.C. Sud, 2001: Refinements to SSiB with an emphasis on snow-physics: Evaluation and validation using GSWP and Valdai data. Earth Interactions, Vol. 5, (5-001), 31 pp.
Other papers in comments:
Sellers et al (two sets of authors), 1996: J. Climate, Vol. 6, No. 4, A revised land surface parameterization (SiB2) for atmospheric GCMs Part I - Model formulation. p. 676-705. Part II - The generation of global fields of terrestrial biophysical parameters from satellite data. p. 706-737.
GEWEX/GSWP, 1995: International GEWEX Project Office. Version 1.0, 1 Dec 1995. 46 pp.
Douville et al, 1995: A new snow parameterization for the Meteo-France climate model. Parts I and II. Clim. Dyn., Vol. 12, 21-52. Mahrt and Sun, 1995: The subgrid velocity scale in the bulk aerodynamic relationship for spatially averaged scalar fluxes. Mon. Wea. Rev., Vol. 123, No. 10, 3032-3041.
Xue et al, 1991: A simplified biosphere model for global climate studies. J. Climate, Vol. 4, No. 3, 345-364.

Sellers et al, 1986: A simple biosphere model (SiB) for use within general circulation models. J. Atmos Sci., Vol. 43, No. 6, 505-531.
Milly and Eagleson, 1982: Parameterization of moisture and heat fluxes across the land surface for use in atmospheric general circulation models. Dept. of Engineering, Massachusetts Inst. of Technology, Rep. 279, 159 pp.
Deardorff, 1972: Parameterization of planetary boundary-layer for use in general circulation models. Mon. Wea. Rev., Vol. 100, No. 2, 93-106.
The arguments are:

n index of the nest

P.4.15 hyssib_mapvegc.F90 (Source File: hyssib_mapvegc.F90)

REVISION HISTORY:

```
28 Apr 2002: K Arsenault, Added SSIB LSM to LDAS
   Feb 2004: David Mocko, Conversion from SSiB to HY-SSiB
25 Nov 2007: Chuck Alonge, Updates for LIS 5.0
```

INTERFACE:

```
SUBROUTINE HYSSIB_MAPVEGC(VEGT)

  implicit none
```

ARGUMENTS:

```
  INTEGER, INTENT(INOUT) :: VEGT
```

DESCRIPTION:

This subroutine converts the UMD classes to the SIB classes used by SSIB LSM (v 2.5). (Originally from Dag Lohmann at NCEP)
The arguments are:

vegt UMD Vegetation class number to be converted to SIB class

---

P.4.16 hyssib_settbot (Source File: hyssib_settbot.F90)

REVISION HISTORY:

```
03 Jun 2005: David Mocko, Conversion from NOAH to HY-SSiB
05 Sep 2007: Chuck Alonge, Updates for LIS 5.0 compliance
```

INTERFACE:

```
subroutine hyssib_settbot
```

USES:

```
use lisdrv_module, only : lis, lisdom
use hyssib_varder
use spmdMod, only : iam
use listime_mgr, only : isMonthlyAlarmRinging, computeMonthlyWeights
```

DESCRIPTION:

This subroutine retrieves HY-SSiB bottom soil temperature Currently a static data is being used. If a climatology is available, it can be used by setting the appropriate frequency of the climatology
The routines invoked are:

readtbot (2.44.2)
retrieves the bottom temperature data

P.4.17 hyssib_settopostd (Source File: hyssib_settopostd.F90)

REVISION HISTORY:

```
03 Jun 2005: David Mocko, Conversion from NOAH to HY-SSiB
27 Sep 2007: Chuck ALonge, Updates for LIS 5.0
```

INTERFACE:

```
subroutine hyssib_settopostd
```

USES:

```
use lisdrv_module, only : lis, lisdom
use hyssib_varder
use lis_logmod,    only : logunit
```

DESCRIPTION:

This subroutine retrieves standard deviation of topography data for the Hyssib LSM. Currently static data is being used.

P.4.18 hyssib_setvegparms (Source File: hyssib_setvegparms.F90)

This subroutine retrieves HY-SSiB vegetation parameters

REVISION HISTORY:

```
21 Apr 2004: David Mocko, Conversion from NOAH to HY-SSiB
05 Sep 2007: Chuck Alonge, Updates for LIS 5.0
```

INTERFACE:

```
subroutine hyssib_setvegparms
```

USES:

```
use lisdrv_module, only : lis,lisdom
use hyssib_varder
use hyssibveg_module
use lis_logmod, only : logunit
```

DESCRIPTION:

This subroutine retrieves Hyssib veg./soil parameters. The current implementation uses a table-based lookup based on SIB classes to initialize the following parameters:

```
chil    - leaf angle distribution factor
topt    - optimum temperature for rst calculation
tll     - low temp for rst calculation
tu      - top temp for rst calculation
defac   - dew factor for rst calculation
ph1     - stome slope factor
ph2     - point at which stomates close
rootd   - rooting depth - canopy
rootd   - rooting depth - ground
rstpar1 - PAR influence on stomatal resist. coefficients
rstpar2 - PAR influence on stomatal resist. coefficients
rstpar3 - PAR influence on stomatal resist. coefficients
phsat   - soil moisture potential at saturation
poros   - soil porosity
bee     - Clapp-Hornberger emperical const.
satco   - sat. hydraulic conductivity
slope   - average topographic slope %
zdepth1 - depth of soil layer 1
zdepth2 - depth of soil layer 2
zdepth3 - depth of soil layer 3
```

P.4.19 hyssib_finalize (Source File: hyssib_finalize.F90)

REVISION HISTORY:

```
28 Apr 2002: Sujay Kumar; Initial code
05 Sep 2007: Sujay Kumar; Addition of Hyssib to LIS 5.0
```

INTERFACE:

```
subroutine hyssib_finalize()
```

USES:

```
use lisdrv_module, only : lis
use hyssib_varder
use hyssibveg_module
use hyssibalb_module
```

DESCRIPTION:

This routine cleans up the allocated memory structures in Hyssib

Q Catchment Land Surface Model

This section describes the interface implementations for the Catchment land surface model.

Q.0.20 cat_varder.F90 (Source File: cat_varder.F90)

Module for 1-D catchment land model driver variable initialization

REVISION HISTORY:

```
15 Dec 2005; Sujay Kumar, Initial Code
```

INTERFACE:

```
module cat_varder
```

USES:

```
use catch_types
use drv_catch_types
use modis_alb_types

type cat_type_dec
   integer :: catopen       !Keeps track of opening files
   integer :: numout       !Counts number of output times for Cat
   integer :: count
   integer :: writeint         !CAT Output Interval (minutes)
   integer :: rstInterval       !CAT restart interval (minutes)
   integer :: varid(39)         !For netcdf output
   real    :: zol_lookup(9,12)
   real    :: vgd_lookup(9,12)
   real    :: initSM          ! initial soil moisture
   real    :: initST          ! initial soil temperature
   character*100 :: rfile     !restart file
   character*100 :: zolfile     !roughness length table
   character*100 :: vgdfile     !veg displacement ht table
   character*100 :: paramdir     !directory for parameter files
   character*100 :: modisdir     !sub-directory for MODIS files
   character*100 :: vegclassfile !vegetation classification file
   character*100 :: soilparamfile!soil parameters file
   character*100 :: sltsfile     !surface layer timescales file
   character*100 :: topo_ar_file !topography parameters file
   character*100 :: topo_bf_file !topography parameters file
   character*100 :: topo_ts_file !topography parameters file

   logical, pointer              :: good_forcing_mask(:)
   type(cat_param_type), pointer :: cat_param(:)
   type(met_force_type), pointer :: met_force(:)
   type(cat_progn_type), pointer :: cat_progn(:)
   type(cat_diagn_type), pointer :: cat_diagn(:)
   type(cat_output_type), pointer :: cat_output(:)
   type(modis_alb_type), pointer :: modis_param(:,:)
end type cat_type_dec

type(cat_type_dec), pointer :: cat_struc(:)
```

Q.0.21 cat_varalloc (Source File: cat_varder.F90)

Reads in runtime catchment parameters, allocates memory for variables

INTERFACE:

```
subroutine cat_varalloc()
```

USES:

```
use lisdrv_module, only : lis
```

Q.1 Fortran: Module Interface catch_types (Source File: catch_types.F90)

definition of types and associated operators for Catchment Model
IMPORTANT: When adding a field to any of the derived types, must also update the associated assignment and operator definitions. THERE IS NO WARNING/ERROR IF OPERATOR IS NOT DEFINED FOR ALL FIELDS!

REVISION HISTORY:

```
21 May 2003, Rolf Reichle: Initial Specification
25 Jan 2005, Rolf Reichle: - added cat_force_type
```

Q.2 Fortran: Module Interface cat_diagn_routines (Source File: catch_diagn_routines.F90)

This module contains subroutinte for computing various diagnostics in the Catchment model

REVISION HISTORY:

```
Feb 5,  2004, Rolf Reichle:  Initial Specification
Mar 19, 2004, Rolf Reichle:  Revised subroutine, calc_soil_moist
Aug 31, 2004, Rolf Reichle:  Added calc_tsurf
Jun 21, 2005, Rolf Reichle:  Added calc_arX
```

Q.2.1 calc_tsurf (Source File: catch_diagn_routines.F90)

INTERFACE:

```
subroutine calc_tsurf( N_cat, cat_param, cat_progn, tsurf )
```

DESCRIPTION:

Calculate diagnostic surface temperature "tsurf" from prognostics

---

Q.2.2 calc_arX (Source File: catch_diagn_routines.F90)

INTERFACE:

```
subroutine calc_arX( N_cat, cat_param, cat_progn, ar1, ar2, ar4 )
```

DESCRIPTION:

???

Q.2.3 calc_asnow (Source File: catch_diagn_routines.F90)

INTERFACE:

```
subroutine calc_asnow( N_cat, cat_progn, asnow )
```

DESCRIPTION:

Calculate diagnostic snow area from prognostic SWE

Q.2.4 cat_setup (Source File: cat_setup.F90)

REVISION HISTORY:

```
16 Dec 2005: Sujay kumar; Initial Code
```

INTERFACE:

```
subroutine cat_setup()
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_logmod,    only : logunit
use soils_module,  only : lis_soils
use cat_varder
use clsm_para_mapping
use spmdMod, only : masterproc
```

DESCRIPTION:

This routine is the entry point to set up the parameters required for Noah LSM. These include the soils, greenness, albedo, bottom temperature and the initialization of state variables in Noah.
The routines invoked are:

read_land_parameter1 (Q.2.14)

read_land_parameter2 (Q.2.15)

read_land_parameter3 (Q.2.16)

modis_scale_param (**??**)

soiltype (2.29.3)

read_rst (Q.3.3)

read_catchment_parameters (Q.3.1)

get_param (Q.3.5)

read_catchment_parameter_finalize (**??**)

cat_totinit (Q.2.12)

read

Q.2.5 cat_dynsetup.F90 (Source File: cat_dynsetup.F90)

Sets the time varying parameters for catchment

REVISION HISTORY:

```
28 Jan 2002: Sujay Kumar, Initial Specification
```

INTERFACE:

```
subroutine cat_dynsetup(n)
```

USES:

Q.2.6 cat_main (Source File: cat_main.F90)

REVISION HISTORY:

```
13 Jun 2005 Rolf Reichle, Initial Specification
17 Dec 2005 Sujay Kumar, Implementation in LIS
```

INTERFACE:

```
subroutine cat_main(nid)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use cat_varder, only : cat_struc
use catch_diagn_routines, only : calc_asnow
use catch_types
use gfrac_module, only : lis_gfrac
use albedo_module, only : lis_alb
use lai_module, only : lis_lai
use esat_qsat
use constantsMod, only : const_stebol
use catchment_model
use catch_constants
use lis_logmod,  only : logunit

implicit none

integer, intent(in) :: nid
```

DESCRIPTION:

This is the entry point for calling the Catchment LSM physics. This routine calls `catchment` routine that performs the land surface computations, to solve water and energy equations.
The arguments are:

nid index of the nest

Q.2.7 cat_output (Source File: cat_output.F90)

INTERFACE:

```
subroutine cat_output(n)
```

USES:

```
use lisdrv_module, only : lis, metadata_output
use cat_varder, only : cat_struc
use spmdMod, only : masterproc,npes
use lis_fileIOMod, only : create_output_directory, &
                          create_output_filename,  &
                          create_stats_filename

implicit none

integer, intent(in) :: n
```

DESCRIPTION:

This subroutines sets up methods to write model output from Catchment LSM A timestamped filename and directory are created first. The routine supports model output in three data formats: Binary, Grib and NETCDF.
The arguments are:

n index of the nest

The routines invoked are:

create_output_directory (2.25.2)
creates a time stamped output directory

create_output_filename (2.25.3)
creates a time stamped output filename

create_stats_filename (2.25.5)
creates a stats filename

cat_binout (Q.2.8)
routine to write Cat model output in binary format

cat_totint (Q.2.12)
resets the time averaged variables

Q.2.8 cat_binout (Source File: cat_binout.F90)

LIS Catchment data writer: Writes output in binary format

REVISION HISTORY:

```
14 Feb 2006; Sujay Kumar, Initial Version
```

INTERFACE:

```
subroutine cat_binout(ftn,ftn_stats,n)
```

USES:

```
use lisdrv_module, only : lis,metadata_output
use drv_output_mod, only : drv_writevar_bin
use constantsMod, only : CONST_TKFRZ
use cat_varder
use catch_constants
use lai_module, only : lis_lai
use catchment_model, only : calc_soil_moist
use lis_logmod, only      : logunit
```

DESCRIPTION:

This routine writes the specified catchment variables into a binary output file.
The arguments are:

n index of the nest

ftn unit number for the model output file

ftn_stats unit number for the stats file

The routines invoked are:

calc_soil_moist (**??**)
computes the catchment diagnostic variables for output

drv_writevar_bin (2.22.2)
routine to write a variable to a binary output file

Q.2.9 cat_readrestart (Source File: cat_readrestart.F90)

REVISION HISTORY:

```
16 Dec 2005: Sujay Kumar, Initial Specification
```

INTERFACE:

```
subroutine cat_readrestart
```

USES:

```
use spmdMod, only : iam,masterproc
use lisdrv_module, only : lis,lisdom
use cat_varder
use listime_mgr
use lis_logmod, only : logunit
use drv_output_mod, only : drv_readvar_restart
use catch_constants
```

DESCRIPTION:

This program reads restart files for Catchment. This includes all relevant water/energy storages, tile information.
The routines invoked are:

drv_readvar_restart (2.22.4)
reads a variable from the restart file

Q.2.10 cat_writerestart (Source File: cat_writerestart.F90)

REVISION HISTORY:

```
18 Dec 2005: Sujay Kumar, Initial Specification
```

INTERFACE:

```
subroutine cat_writerestart(n)
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr
use cat_varder
use lis_logmod, only :logunit
use lis_fileIOMod, only : create_output_directory, &
                          create_restart_filename
```

ARGUMENTS:

```
implicit none
integer, intent(in) :: n
```

DESCRIPTION:

This program writes restart files for catchment. This includes all relevant water/energy storage and tile information
The routines invoked are:

create_output_directory (2.25.2)
creates a timestamped directory for the restart files

create_restart_filename (2.25.4)
generates a timestamped restart filename

cat_dump_restart (Q.2.11)
writes the catchment variables into the restart file

Q.2.11 cat_dump_restart (Source File: cat_writerestart.F90)

REVISION HISTORY:

```
24 Aug 2004: James Geiger, Initial Specification
```

INTERFACE:

```
subroutine cat_dump_restart(n, ftn)
```

USES:

```
use lisdrv_module, only : lis
use cat_varder
use listime_mgr
use drv_output_mod, only : drv_writevar_restart
use catch_constants

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: ftn
integer, intent(in) :: n
```

DESCRIPTION:

This routine gathers the necessary restart variables and performs the actual write statements to create the restart files.
The arguments are:

n index of the nest

ftn unit number for the restart file

The routines invoked are:

drv_writevar_restart (2.22.3)
writes a variable to the restart file

Q.2.12 cat_totinit (Source File: cat_totinit.F90)

REVISION HISTORY:

```
14 Feb 2006  Sujay Kumar  Initial Specification
```

INTERFACE:

```
subroutine cat_totinit(n)
```

USES:

```
use cat_varder      ! NOAH LSM module
use lisdrv_module, only : lis

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine resets the time-averaged Catchment variables after a model output. The routine is called after a model output call.

Q.2.13 cat_f2t (Source File: cat_f2t.F90)

REVISION HISTORY:

```
16 Dec 2005: Sujay Kumar; Initial Specification
11 Oct 2006: Sujay Kumar; modified to work with ESMF_State design
```

INTERFACE:

```
subroutine cat_f2t(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module , only : lis
use cat_varder, only : cat_struc
use baseforcing_module, only : FORC_State
use constantsMod,       only : CONST_TKFRZ

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine transfers the LIS provided forcing onto the Catchment model tiles.
The arguments are:

n index of the nest

Q.2.14 read_land_parameters1 (Source File: read_land_parameters.F90)

REVISION HISTORY:

```
12 May 2003 Rolf Reichle, Initial Specification
06 Jun 2005 Rolf Reichle, adapted to read "SiB2_V2" parameters
10 Jul 2006 James Geiger, Implementation in LIS
```

INTERFACE:

```
subroutine read_land_parameters1(nest)
```

USES:

```
    use catch_constants, only : N_gndtmp
    use cat_varder
    use lisdrv_module, only : lis

    implicit none
```

ARGUMENTS:

```
    integer, intent(in) :: nest
```

DESCRIPTION:

Reads in the vegetation, soil properties and topographic parameters from global files and outputs parameters for the domain.
Additional parameters are derived from the ones that have been read from files.
The arguments are:

nest index of the nest

Q.2.15 read_land_parameters2 (Source File: read_land_parameters.F90)

REVISION HISTORY:

```
12 May 2003 Rolf Reichle, Initial Specification
06 Jun 2005 Rolf Reichle, adapted to read "SiB2_V2" parameters
10 Jul 2006 James Geiger, Implementation in LIS
```

INTERFACE:

```
subroutine read_land_parameters2(nest)
```

USES:

```
!use tile_coord_types
use catch_constants, only : N_gndtmp
use cat_varder
use lisdrv_module, only : lis

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: nest
```

DESCRIPTION:

Reads in the vegetation, soil properties and topographic parameters from global files and outputs parameters for the domain.
Additional parameters are derived from the ones that have been read from files.
The arguments are:

nest index of the nest

Q.2.16 read_land_parameters3 (Source File: read_land_parameters.F90)

REVISION HISTORY:

```
12 May 2003 Rolf Reichle, Initial Specification
06 Jun 2005 Rolf Reichle, adapted to read "SiB2_V2" parameters
10 Jul 2006 James Geiger, Implementation in LIS
```

INTERFACE:

```
subroutine read_land_parameters3(nest)
```

USES:

```
!use tile_coord_types
use catch_constants, only : N_gndtmp
use cat_varder
use lisdrv_module, only : lis

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: nest
```

DESCRIPTION:

Reads in the vegetation, soil properties and topographic parameters from global files and outputs parameters for the domain.
Additional parameters are derived from the ones that have been read from files.
The arguments are:

nest index of the nest

Q.2.17 d2g (Source File: read_land_parameters.F90)

REVISION HISTORY:

```
10 Jul 2006 James Geiger, Implementation in LIS
```

INTERFACE:

```
function d2g(n,k)
```

USES:

```
use lisdrv_module, only : lisdom
implicit none
```

ARGUMENTS:

```
integer :: d2g
integer, intent(in) :: n
integer, intent(in) :: k
```

DESCRIPTION:

This routine returns the global tile number (from the .til file) corresponding to the the given local tile number.
The arguments are:

n index of the nest

k local index of the tile

Q.3 Fortran: Module Interface clsm_para_mapping (Source File: clsm_para_mapping.F90)

REVISION HISTORY:

```
30 Nov 2006, Sarith Mahanama:  Initial Specification
```

DESCRIPTION:

??

Q.3.1 read_catchment_parameters (Source File: clsm_para_mapping.F90)

INTERFACE:

```
subroutine read_catchment_parameters(pathout)

   implicit none
```

ARGUMENTS:

```
   character(len=*) :: pathout
```

DESCRIPTION:

??

Q.3.2 read_catchment_parameters_finalize (Source File: clsm_para_mapping.F90)

INTERFACE:

```
subroutine read_catchment_parameters_finalize()

   implicit none
```

DESCRIPTION:

??

Q.3.3 read_rst (Source File: clsm_para_mapping.F90)

INTERFACE:

```
SUBROUTINE read_rst (filein1,filein2,tile_id)

  implicit none
```

ARGUMENTS:

```
  character(len=*) :: filein1,filein2
  integer, dimension (i_raster,j_raster) :: tile_id
```

DESCRIPTION:

??

Q.3.4 get_tile_index (Source File: clsm_para_mapping.F90)

INTERFACE:

```
SUBROUTINE get_tile_index (                &
     lat,lon,tile_id,                      &
     tile_index)

  implicit none
```

ARGUMENTS:

```
  real , intent(in) :: lat,lon
  integer,  intent(in), dimension (i_raster,j_raster) :: tile_id
  integer, intent(out) :: tile_index
```

DESCRIPTION:

??

Q.3.5 get_param (Source File: clsm_para_mapping.F90)

INTERFACE:

```
SUBROUTINE get_param (                     &
     tile_index,                           &
     soil_type_in,veg_type_in,             &
     veg_type_out,                         &
     BEE, PSIS,POROS,COND,WPWET,           &
     atau, btau, soil_depth,               &
     ARS1,ARS2,ARS3,ARA1,ARA2,ARA3,ARA4,   &
     ARW1,ARW2,ARW3,ARW4, bf1, bf2, bf3,   &
     tsa1, tsa2,tsb1, tsb2)

  implicit none
```

DESCRIPTION:

??

Q.3.6 cat_finalize (Source File: cat_finalize.F90)

Complete the finalize routines for catchment

REVISION HISTORY:

```
11 Feb 2006: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine cat_finalize()
```

USES:

```
use lisdrv_module, only : lis
use cat_varder
```

Q.3.7 cat_getsoilm (Source File: cat_getsoilm.F90)

REVISION HISTORY:

```
12 Feb 2006; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine cat_getsoilm(n,LSM_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only  : lis
use cat_varder

implicit none
```

ARGUMENTS:

```
integer, intent(IN)       :: n
type(ESMF_State)          :: LSM_State
```

DESCRIPTION:

Routine to retrieve soil moisture prognostic variables in catchment.
The arguments are:

n index of the nest

LSM_State ESMF State container for LSM state variables

Q.3.8 cat_setsoilm (Source File: cat_setsoilm.F90)

REVISION HISTORY:

```
12 Feb 2006; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine cat_setsoilm(n,LSM_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only  : lis
use cat_varder

implicit none
```

ARGUMENTS:

```
integer, intent(IN)       :: n
type(ESMF_State)          :: LSM_State
```

DESCRIPTION:

Routine to assign soil moisture prognostic variables in catchment.
The arguments are:

n index of the nest

LSM_State ESMF State container for LSM state variables

Q.3.9 cat_update_soilm (Source File: cat_update_soilm.F90)

REVISION HISTORY:

```
12 Feb 2006; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine cat_update_soilm(n,LSM_State, LSM_Incr_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only  : lis
use cat_varder

implicit none
```

ARGUMENTS:

```
integer, intent(IN)       :: n
type(ESMF_State)          :: LSM_State
type(ESMF_State)          :: LSM_Incr_State
```

DESCRIPTION:

Routine to assign soil moisture prognostic variables in catchment.
The arguments are:

n index of the nest

LSM_State ESMF State container for LSM state variables

LSM_Incr_State ESMF State container for LSM incr state variables

---

Q.3.10 cat_qcsoilm (Source File: cat_qcsoilm.F90)

REVISION HISTORY:

```
12 Feb 2006; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine cat_qcsoilm(n,LSM_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only  : lis
use cat_varder
use catchment_model, only : calc_soil_moist

implicit none
```

ARGUMENTS:

```
integer, intent(IN)       :: n
type(ESMF_State)          :: LSM_State
```

DESCRIPTION:

Routine to retrieve soil moisture prognostic variables in catchment.
The arguments are:

n index of the nest

LSM_State ESMF State container for LSM state variables

Q.3.11 cat_scale_soilm (Source File: cat_scale_soilm.F90)

REVISION HISTORY:

```
12 Feb 2006; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine cat_scale_soilm(n,LSM_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only  : lis
use cat_varder

implicit none
```

ARGUMENTS:

```
integer, intent(IN)       :: n
type(ESMF_State)          :: LSM_State
```

DESCRIPTION:

Routine to retrieve soil moisture prognostic variables in catchment.
The arguments are:

n index of the nest

LSM_State ESMF State container for LSM state variables

Q.3.12 cat_descale_soilm (Source File: cat_descale_soilm.F90)

REVISION HISTORY:

```
12 Feb 2006; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine cat_descale_soilm(n,LSM_State)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only  : lis
use cat_varder

implicit none
```

ARGUMENTS:

```
integer, intent(IN)       :: n
type(ESMF_State)          :: LSM_State
```

DESCRIPTION:

Routine to retrieve soil moisture prognostic variables in catchment.
The arguments are:

n index of the nest

LSM_State ESMF State container for LSM state variables

Q.3.13 cat_getsynsmpred (Source File: cat_getsynsmpred.F90)

REVISION HISTORY:

```
12 Feb 2006; Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine cat_getsynsmpred(n,obs_pred)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only  : lis,lisdom
use cat_varder
use catchment_model, only : calc_soil_moist

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
real                   :: obs_pred(lis%ngrid(n),lis%nensem(n))
```

DESCRIPTION:

Returns the Soil moisture obs pred (model's estimate of observations) for data assimilation
The arguments are:

n index of the nest

obs_pred model's estimate of observations

R SiB2 Land Surface Model

This section describes the interface implementations for the SiB2 land surface model.

R.1 Fortran: Module Interface sib2_module (Source File: sib2_module.F90)

Module for 1-D SiB2 land model driver variable specification

REVISION HISTORY:

```
28 Apr 2002: K. Arsenault, added NOAH LSM 2.5 code to LDAS
14 Nov 2002: Sujay Kumar, Optimized version for LIS
 1 Mar 2004: Luis Gustavo G de Goncalves, SiB2 in LIS
22 May 2004: David Mocko, made compatible with SiB-lings
```

INTERFACE:

```
module sib2_module
```

R.2 Fortran: Module Interface sib2_varder (Source File: sib2_varder.F90)

Module for 1-D SiB2 land model driver variable initialization

REVISION HISTORY:

```
   Apr 2003: Sujay Kumar, Initial Code
 1 Mar 2004: Luis Gustavo G de Goncalves, SiB2 in LIS
 6 May 2004: David Mocko, made compatible with SiB-lings
21 Dec 2007: Sujay Kumar, Updated for LIS 5.0
```

INTERFACE:

```
module sib2_varder
```

USES:

```
use sib2_module
```

R.2.1 sib2_varder_ini (Source File: sib2_varder.F90)

INTERFACE:

```
subroutine sib2_varder_ini()
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr,   only : setMonthlyAlarm
```

DESCRIPTION:

This routine creates the datatypes and allocates memory for SiB2-specific variables. It also invokes the routine to read the runtime specific options for noah from the configuration file.
The routines invoked are:

readcrdsib2 (R.3.2)
reads the runtime options for SiB2 LSM

R.2.2 sib2_setup (Source File: sib2_setup.F90)

REVISION HISTORY:

```
 1 Mar 2004: Luis Gustavo G de Goncalves, SiB2 in LIS
22 May 2004: David Mocko, made compatible with SiB-lings
```

INTERFACE:

```
subroutine sib2_setup
```

USES:

```
use lisdrv_module, only : lis
use sib2_varder
use sib2alb_module
```

DESCRIPTION:

This routine is the entry point to setup the parameters required for SiB2 LSM.

R.3 Fortran: Module Interface sib2alb_module (Source File: sib2alb_module.F90)

In order to use the Sib2 albedo calculations, UMD vegetation types need to be mapped to SSiB vegetation types. The code in this file provides a description of the data structure used to contain the vegetation specific constants used in Sib2's albedo calculations. The varaiables specified in the data structure include:

cedfu

cedir

cedfu1

cedir1

cedfu2

xmiu

cedir2

clefu

cledir

cether

xmiw

REVISION HISTORY:

```
    01 Dec 2007: Chuck Alonge; Initial Specification
   implicit none
  ------------------------------------------------------------------------------
```

PUBLIC MEMBER FUNCTIONS:

```
  ------------------------------------------------------------------------------
   public :: sib2alb_ini
  ------------------------------------------------------------------------------
```

PUBLIC TYPES:

```
  ------------------------------------------------------------------------------
   public :: sib2albdec
```

R.3.1 sib2alb_ini (Source File: sib2alb_module.F90)

INTERFACE:

```
   subroutine sib2alb_ini()
```

USES:

```
     use lisdrv_module, only : lis
     use sib2_varder
     use lis_logmod,    only : logunit
```

DESCRIPTION:

Reads in runtime SiB2 albedo parameters to populate constants used in Sib2 albedo calculations

---

R.3.2 readcrdsib2 (Source File: readcrdsib2.F90)

REVISION HISTORY:

```
   14 Oct 2003: Sujay Kumar, Initial Code
    1 Mar 2004: Luis Gustavo G de Goncalves, SiB2 in LIS
    5 May 2004: David Mocko, made compatible with SiB-lings
   21 Dec 2007: Sujay Kumar, Updated for LIS 5.0
```

INTERFACE:

```
subroutine readcrdsib2()
```

USES:

```
  use ESMF_Mod
  use sib2_varder,   only : sib2_struc
  use lis_logmod,    only : logunit
  use lisdrv_module, only : lis, config_lis
```

DESCRIPTION:

Routine to read SiB2 specific parameters from the card file

R.3.3 sib2_output (Source File: sib2_output.F90)

INTERFACE:

```
subroutine sib2_output(n)
```

USES:

```
#if ( defined USE_NETCDF )
  use netcdf
#endif
  use lisdrv_module, only : lis, metadata_output
  use sib2_varder
  use spmdMod
  use lis_fileIOMod, only : create_output_directory, &
       create_output_filename,  &
       create_stats_filename

  implicit none

  integer, intent(in) :: n
```

DESCRIPTION:

This subroutines sets up methods to write model output from SiB2 LSM A timestamped filename and directory are created first. The routine supports model output in three data formats: Binary, Grib and NETCDF.
The arguments are:

n index of the nest

The routines invoked are:

create_output_directory (2.25.2)
: creates a time stamped output directory

create_output_filename (2.25.3)
: creates a time stamped output filename

create_stats_filename (2.25.5)
: creates a stats filename

sib2_binout (R.3.4)
: routine to write Sib2 model output in binary format

sib2_totint (R.3.9)
: resets the time averaged variables

R.3.4 sib2_binout (Source File: sib2_binout.F90)

REVISION HISTORY:

```
02 Dec 2003: Sujay Kumar, Initial Version
   Feb 2004: David Mocko, Conversion from NOAH to SiB2
```

INTERFACE:

```
subroutine sib2_binout(ftn, ftn_stats, n)
```

USES:

```
use lisdrv_module, only : lis,metadata_output
use drv_output_mod, only : drv_writevar_bin
use constantsMod,   only : CONST_TKFRZ, CONST_LATVAP
use sib2_varder

implicit none
```

ARGUMENTS:

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn,ftn_stats
```

DESCRIPTION:

This routine writes the specified SiB2 variables into a binary output file.
The arguments are:

n index of the nest

ftn unit number for the model output file

ftn_stats unit number for the stats file

The routines invoked are:

drv_writevar_bin (2.22.2)
routine to write a variable to a binary output file

R.3.5 sib2_coldstart (Source File: sib2_coldstart.F90)

REVISION HISTORY:

```
 1 Mar 2004: Luis Gustavo G de Goncalves, SiB2 in LIS
18 May 2004: David Mocko, made compatible with SiB-lings
```

INTERFACE:

```
subroutine sib2_coldstart(n)
```

USES:

```
use lisdrv_module, only: lis
use sib2_varder
use listime_mgr
use lis_logmod, only : logunit
```

DESCRIPTION:

This routine initializes the SiB2 state variables with some predefined values uniformly for the entire domain. These initial values will be overwritten by the values read from the supplied SiB2 model restart file.
The arguments are:

n index of the nest

R.3.6 sib2_writerst (Source File: sib2_writerst.F90)

REVISION HISTORY:

```
1 Mar 2004: Luis Gustavo G de Goncalves, SiB2 in LIS
6 May 2004: David Mocko, made compatible with SiB-lings
```

INTERFACE:

```
subroutine sib2_writerst(n)
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr
use sib2_varder
use lis_logmod,    only : logunit
use lis_fileIOMod, only : create_output_directory, &
     create_restart_filename

implicit none

integer, intent(IN)    :: n
```

DESCRIPTION:

This program writes restart files for SiB2. This includes all relevant water/energy storages, and tile information.
The routines invoked are:

create_output_directory (2.25.2)
creates a timestamped directory for the restart files

create_restart_filename (2.25.4)
generates a timestamped restart filename

sib2_dump_restart (R.3.7)
writes the sib2 variables into the restart file

R.3.7 sib2_dump_restart (Source File: sib2_writerst.F90)

REVISION HISTORY:

```
24 Aug 2004: James Geiger, Initial Specification
```

INTERFACE:

```
subroutine sib2_dump_restart(n, ftn)
```

USES:

```
use lisdrv_module, only : lis
use sib2_varder
use listime_mgr
use drv_output_mod, only : drv_writevar_restart

implicit none

integer, intent(in) :: ftn
integer, intent(in) :: n
```

DESCRIPTION:

This routine gathers the necessary restart variables and performs the actual write statements to create the restart files.
The arguments are:

n index of the nest

ftn unit number for the restart file

The routines invoked are:

drv_writevar_restart (2.22.3)
writes a variable to the restart file

R.3.8 sib2_readrst (Source File: sib2_readrst.F90)

REVISION HISTORY:

```
1 Mar 2004: Luis Gustavo G de Goncalves, SiB2 in LIS
5 May 2004: David Mocko, made compatible with SiB-lings
```

INTERFACE:

```
subroutine sib2_readrst
```

USES:

```
use lisdrv_module, only : lis
use sib2_varder
use listime_mgr
use drv_output_mod, only : drv_readvar_restart
use lis_logmod, only : logunit
```

DESCRIPTION:

This program reads restart files for SiB2. This includes all relevant water/energy storages, tile information, and time information. It also rectifies changes in the tile space.

R.3.9 sib2_totinit (Source File: sib2_totinit.F90)

REVISION HISTORY:

```
22 May 2004: David Mocko, Conversion from HY-SSiB to SiB2
```

INTERFACE:

```
subroutine sib2_totinit(n)
```

USES:

```
use sib2_varder
use lisdrv_module, only : lis

implicit none
```

ARGUMENTS:

```
integer,  intent(IN)   :: n
```

DESCRIPTION:

Initialize SiB2 output arrays

R.3.10 sib2_dynsetup (Source File: sib2_dynsetup.F90)

REVISION HISTORY:

```
15 Apr 2002: Sujay Kumar, Initial Specification
 1 Mar 2004: Luis Gustavo G de Goncalves, SiB2 in LIS
 6 May 2004: David Mocko, made compatible with SiB-lings
```

INTERFACE:

```
subroutine sib2_dynsetup(n)
```

USES:

```
use lisdrv_module, only: lis
use sib2_varder

implicit none
```

ARGUMENTS:

```
integer, intent(IN)  :: n
```

DESCRIPTION:

Updates the time dependent SiB2 variables The arguments are:

n index of the nest

The routines invoked are:

sib2_monthly (R.3.14)
reads the parameter file containing monthly varying data.

R.3.11 sib2_f2t (Source File: sib2_f2t.F90)

REVISION HISTORY:

```
 1 Mar 2004: Luis Gustavo G de Goncalves, SiB2 in LIS
 5 May 2004: David Mocko, made compatible with SiB-lings
```

INTERFACE:

```
subroutine sib2_f2t(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis   ! LIS non-model-specific 1-D variables
use sib2_varder
use baseforcing_module, only : FORC_State

implicit none
```

ARGUMENTS:

```
integer, intent(IN)   :: n
```

DESCRIPTION:

This routine transfers the LIS provided forcing onto the SiB2 model tiles.
The arguments are:

n index of the nest

R.3.12 sib2_main (Source File: sib2_main.F90)

REVISION HISTORY:

```
 1 Mar 2004: Luis Gustavo G de Goncalves, SiB2 in LIS
25 May 2004: David Mocko, removed duplicate code and provided ALMA outputs
```

INTERFACE:

```
subroutine sib2_main(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use sib2_varder
use sib2alb_module, only : sib2alb
```

DESCRIPTION:

This is the entry point for calling the SiB2 LSM physics. This routine calls the `sib2_routines` routine that performs the land surface computations, to solve for water and energy equations.
references:

sato, n., p. j. sellers, d. a randall, e. k. schneider, j. shukla, j. l kinter iii, y-t, hou, and albertazzi (1989) "effects of implementing the simple biosphere model in a general circulation model" j. atmos. sci., 46, 2757-2782.

sellers, p. j., d. a. randall, c. j. collatz, j. a. berry, c.b.field, d.a.dazlich, c.zhang, g.collelo,l.bounoua (1996) "a revised land-surface parameterization(sib2) for atmospheric general circulation models. part 1: model formulation. (accepted by jcl ,april 1996)

The arguments are:

n index of the nest

R.3.13 sib2_mapvegc (Source File: sib2_mapvegc.F90)

This subroutine converts the UMD classes to the SiB classes used by NOAH LSM (v 2.5). (Originally from Dag Lohmann at NCEP)

REVISION HISTORY:

```
28 Apr 2002: Kristi Arsenault, Added SSiB LSM to LDAS
 1 Mar 2004: Luis Gustavo G de Goncalves, SiB2 in LIS
 5 May 2004: David Mocko, made compatible with SiB-lings
```

INTERFACE:

```
subroutine sib2_mapvegc(vegt)

  implicit none
```

ARGUMENTS:

```
  integer, intent(inout) :: vegt
```

R.3.14 sib2_monthly (Source File: sib2_monthly.F90)

REVISION HISTORY:

```
28 Apr 2002: Kristi Arsenault, Added NOAH LSM, Initial Code
 1 Mar 2004: Luis Gustavo G de Goncalves, SiB2 in LIS
18 May 2004: David Mocko, made compatible with SiB-lings
```

INTERFACE:

```
subroutine sib2_monthly(n)
```

USES:

```
  use lisdrv_module, only : lis, lisdom
  use listime_mgr,   only : isMonthlyAlarmRinging
  use sib2_varder
  use lis_logmod,    only : logunit
```

DESCRIPTION:

This subroutine retrieves monthly SiB2 parameters
The arguments are:

n index of the nest

R.3.15 sib2_setvegparms (Source File: sib2_setvegparms.F90)

REVISION HISTORY:

```
28 Apr 2002: Kristi Arsenault, Added NOAH LSM, Initial Code
13 Oct 2003: Sujay Kumar, Domain independent modifications
 1 Mar 2004: Luis Gustavo G de Goncalves, SiB2 in LIS
 5 May 2004: David Mocko, made compatible with SiB-lings
```

INTERFACE:

```
subroutine sib2_setvegparms(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_logmod,    only : logunit
use sib2_varder
```

DESCRIPTION:

This subroutine retrieves constant SiB2 parameters

R.3.16 sib2_finalize (Source File: sib2_finalize.F90)

REVISION HISTORY:

```
04 Oct 2005: Sujay Kumar; Initial Specification
```

INTERFACE:

```
subroutine sib2_finalize()
```

USES:

```
use lisdrv_module, only : lis
use sib2_varder
```

DESCRIPTION:

This routine cleans up the allocated memory structures in Sib2aic

S Template

This is a dummy placeholder implementation of a LSM, to be used as a template for new implementations.

S.1 Fortran: Module Interface template_varder (Source File: template_varder.F90)

Module for 1-D land model driver variable initialization

count variable to keep track of the number of timesteps before an output

numout number of output times

outInterval output writing interval

templateopen variable to keep track of opened files

template Template LSM specific variables

REVISION HISTORY:

```
    Apr 2003; Sujay Kumar, Initial Code
 23 Oct 2007; Kristi Arsenault, Updated for V5.0
```

USES:

```
use template_module

implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: template_varder_ini
```

PUBLIC TYPES:

```
public :: template_struc
```

S.1.1 template_varder_ini (Source File: template_varder.F90)

INTERFACE:

```
subroutine template_varder_ini()
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
```

DESCRIPTION:

S.1.2 readtemplatecrd (Source File: readtemplatecrd.F90)

REVISION HISTORY:

```
21 Jul 2004; Sujay Kumar, Initial Code
23 Oct 2007; Kristi Arsenault, Updated for use in LISv5.0
```

INTERFACE:

```
subroutine readtemplatecrd()
```

USES:

```
use ESMF_Mod
use lisdrv_module, only : lis
use spmdMod, only : masterproc
use template_varder, only : template_struc
use lis_logmod, only :logunit
use lisdrv_module, only : config_lis
```

DESCRIPTION:

This routine reads the options specific to template LSM option from the LIS configuration file.

S.1.3 template_setup (Source File: template_setup.F90)

REVISION HISTORY:

```
21 Jul 2004; Sujay Kumar: Initial Specification
23 Feb 2007: Kristi Arsenault; Updated for LISv5.0
```

INTERFACE:

```
subroutine template_setup()
```

USES:

ARGUMENTS:

DESCRIPTION:

Complete the setup routines for Template option (forcing-only)

S.2 Fortran: Module Interface template_module.F90 (Source File: template_module.F90)

Declare forcing-only option (template) variables

forcing Array of meteorological forcing

tair 2m air temperature forcing

qair 2m specific humidity forcing

swdown downward shortwave forcing

lwdown downward longwave forcing

uwind u-wind component forcing

vwind v-wind component forcing

psurf surface pressure forcing

rainf total rainfall forcing

rainf_c convective rainfall forcing

snowf total snowfall forcing

REVISION HISTORY:

```
21 Jul 2004: Sujay Kumar; Initial Specification
23 Oct 2007: Kristi Arsenault;  Added code for LISv5.0
```

S.2.1 template_output (Source File: template_output.F90)

INTERFACE:

```
subroutine template_output(n)
```

USES:

```
use lisdrv_module, only : lis, metadata_output
use template_varder
use spmdMod
use lis_logmod,    only : logunit
use lis_fileIOMod, only : create_output_directory, &
     create_output_filename,  &
     create_stats_filename

implicit none

integer, intent(in) :: n
```

DESCRIPTION:

Forcing-only (Template) option for calling the output routines.
The arguments are:

n index of the nest

The routines invoked are:

create_output_directory (2.25.2)
creates a time stamped output directory

create_output_filename (2.25.3)
creates a time stamped output filename

create_stats_filename (2.25.5)
creates a stats filename

template_binout (S.2.2)
routine to write Template model output in binary format

S.2.2 template_binout (Source File: template_binout.F90)

REVISION HISTORY:

```
02 Dec 2003; Sujay Kumar, Initial Version
23 Oct 2007; Kristi Arsenault, Updated for V5.0
```

INTERFACE:

```
subroutine template_binout(ftn,ftn_stats,n)
```

USES:

```
use lisdrv_module, only : lis,metadata_output
use drv_output_mod, only : drv_writevar_bin
use constantsMod,   only : CONST_TKFRZ, CONST_LATVAP
use template_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn,ftn_stats
```

DESCRIPTION:

This routine writes the specified template variables into a binary output file.
The arguments are:

n index of the nest

ftn unit number for the model output file

ftn_stats unit number for the stats file

The routines invoked are:

drv_writevar_bin (2.22.2)
routine to write a variable to a binary output file

S.2.3 template_dynsetup (Source File: template_dynsetup.F90)

INTERFACE:

```
subroutine template_dynsetup(n)
```

USES:

```
implicit none
```

ARGUMENTS:

```
integer, intent(in)  :: n
```

DESCRIPTION:

Currently this routine is a placeholder
The arguments are:

n index of the nest

S.2.4 template_f2t (Source File: template_f2t.F90)

REVISION HISTORY:

```
21 Jul 2004: Sujay Kumar   Initial Specification
23 Oct 2007: Kristi Arsenault, Implemented code for LISv5.0
```

INTERFACE:

```
subroutine template_f2t(n)
```

USES:

```
use ESMF_Mod
use lisdrv_module , only : lis
use baseforcing_module, only : FORC_State
use template_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in)  :: n
```

DESCRIPTION:

Forcing-only option (template) for calling the forcing transfer routines.
The arguments are:

n index of the nest

S.2.5 template_main (Source File: template_main.F90)

S.2.6 template_main.F90 (Source File: template_main.F90)

Apr 2003; Sujay Kumar, Initial Code 23 Oct 2007; Kristi Arsenault, Updated code for LISv50

INTERFACE:

```
subroutine template_main(n)

implicit none
```

ARGUMENTS:

```
integer, intent(in)  :: n
```

DESCRIPTION:

Calls the run routines for the forcing-only option (template)
The arguments are:

n index of the nest

S.2.7 template_writerst (Source File: template_writerst.F90)

INTERFACE:

```
subroutine template_writerst(n)
```

USES:

```
implicit none
```

ARGUMENTS:

```
integer, intent(in)  :: n
```

DESCRIPTION:

Forcing-only (template) option for calling the write restart routines.
The arguments are:

n index of the nest

S.2.8 templaterst (Source File: templaterst.F90)

INTERFACE:

```
subroutine templaterst()
```

USES:

```
implicit none
```

ARGUMENTS:

DESCRIPTION:

Calls the restart reading routines for the forcing-only option (template)
The arguments are:

n index of the nest

S.2.9 template_finalize (Source File: template_finalize.F90)

INTERFACE:

```
subroutine template_finalize()
```

USES:

```
use lisdrv_module, only : lis
use template_varder
```

ARGUMENTS:

DESCRIPTION:

This routine cleans up the allocated memory structures in the template (forcing-only option)
The arguments are:

n index of the nest

References

[1] CLM. http://www.cgd.ucar.edu/tss/clm.

[2] Noah. ftp://ftp.ncep.noaa.gov/pub/gcp/ldas/noahlsm/.

[3] M.-D. Chous. A solar radiation model for use in climate studies. *Journal of Atmospheric Science*, 49:762–772, 1992.

[4] G. J. Collatz, C Grivet, J. T. Ball, and J. A. Berry. Physiological and environmental regulation of stomatal conducatance: Photosynthesis and transpiration: A model that includes a laminar boundary layer. *Agric. For. Meteorol.*, 5:107–136, 1991.

[5] B. Cosgrove, D. Lohmann, K. E. Mitchell, P. R. Houser, E. F. Wood, J. C. Schaake, A. Robock, C. Marshall, J. Sheffield, Q. Duan, L. Luo, R. W. Higgins, R. T. Pinker, J. D. Tarpley, and J. Meng. Real-time and retrospective forcing in the north american land data assimilation system (NLDAS) project. *Journal of Geophysical Research*, 108, 2003.

[6] J. C. Derber, D.F. Parrish, and S.J. Lord. The new global operational analysis system at the national meteorological center. *Weather and Forecasting*, 6:538–547, 1991.

[7] Chen. F., Mitchell. K., Schaake. J, Xue. J, Pan. H, Koren. V., Ek. M Duan, and A. Betts. Modeling of land-surface evaporation by four schemes and comparison with fife observations. *J. Geophys. Res.*, 101(D3):7251–7268, 1996.

[8] T.M. Hamill, R.P.D. D'Entremon, and J.T. Bunting. A description of the air force real-time nephanalysis model. *Weather Forecasting*, 7:288–306, 1992.

[9] P. G. Jarvis. The interpretation of leaf water potential and stomatal conductance found in canopies of the field. *Phil. Trans. R. Soc.*, 273:593–610, 1976.

[10] S. V. Kumar, C. D. Peters-Lidard, Y. Tian, P. R. Houser, J. Geiger, S. Olden, L. Lighty, J. L. Eastman, B. Doty, P. Dirmeyer, J. Adams, K. Mitchell, E. F. Wood, , and J. Sheffield. Land information system-an interoperable framework for high resolution land surface modeling. *Environmental Modelinng and Software*, 2006.

[11] K. E. Mitchell, D. Lohmann, P. R. Houser, J. C. Wood, E. F. Schaake, A. Robock, B. Cosgrove, J. Sheffield, Q. Duan, L. Luo, W. R. Higgins, R. T. Pinker, J. D. Tarpley, D. P. Lettenmaier, C. H. Marshall, J. K. Entin, M. Pan, W. Shi, V. Koren, J. Meng, B. H. Ramsay, and A. A. Bailey. The Multi-institution North American Land Data Assimilation system (NLDAS): Utilization of multiple GCIP products and partners in a continental distributed hydrological modeling system. *Journal of Geophysical Research*, 109:DOI:10.1029/2003JD003823, 2004.

[12] J.-J. Morcrette, L. Smith, and Y. Fouquart. Pressure and temperature dependence of the absorption in longwave radiation parameterizations. *Beiträge zur Physik der Atmosphäre*, 59:455–469, 1986.

[13] J.J. Morcrette. Radiation and cloud radiative properties in the ECMWF opertational weather forecast model. *Journal of Geophysical Research*, 96D:9121–9132, 1991.

[14] C. D. Peters-Lidard, S. V. Kumar, Y. Tian, J. L. Eastman, and P. R. Houser. Global urban-scale land-atmosphere modeling with the land information system. In *Symposium on Planning, Nowcasting, and Forecasting in the Urban Zone, 84th AMS Annual Meeting*, Seattle, WA., January 2004.

[15] L. A. Richards. Capillary conduction of liquids in porous media. *Physics*, 1:318–333, 1931.

[16] M. Rodell, P. R. Houser, U. Jambor, J. Gottschalck, K. Mitchell, C.-J. Meng., K. Arsenault, B. Cosgrove, J. Radakovich, M. Bosilovich, J. K. Entin, J. P. Walker, D. Lohmann, and D. Toll. The Global Land Data Assimilation System. *Bulletin of the American Meteorological Society*, 85(3):381–394, 2004.

[17] E. Rogers, T. L. Black, D. G. Deaven, G. J. DiMego, Q. Zhao, M. Baldwin, N. W. Junker, and Y. Lin. Changes to the operational "early" eta analysis/forecast system at the national centers of environmental prediction. *Wea. Forecasting*, 11:391–413, 1996.

[18] L.S. Rothman, R.R. Gamache, A. Barbe, A. Goldman, J.R. Gillis, L.R. Brown, R.A. Roth, J.-M. Flaud, and C. Camy-Peyret. Afgl atmospheric absorption line parameters compilation. *Applied Optics*, 22:2247–2256, 1982.

[19] M. D. Schwarzkopf and S. B.Fels. The simplified exchange method revisited: An accurate rapid method for computation of infrared cooling rates and fluxes. *Journal of Geophysical Research*, 96:9075–9096, 1991.

[20] R. Shapiro. A simple model for the calculation of the flux of direct and diffuse solar radiation through the atmosphere. Technical Report 35, AFGL Scientific Report, 1987.