The pop_in namelist
For the ITAMOC climatologic run with closed ITF
and the meaning of important parameters (in red)

```
&domain_nml
 nprocs_clinic = 3840            < - number of processors to be used for most of the code
 nprocs_tropic = 3840            <- number of processors to be used for barotropic solver
 clinic_distribution_type = 'cartesian'  <- method for distributing blocks across processors
 tropic_distribution_type = 'cartesian' <- method for distributing blocks across processors
 ew_boundary_type = 'cyclic'  <-type of boundary in the logical east-west direction for global domain
 ns_boundary_type = 'tripole' <-type of boundary in the logical north-south direction for gl. domain
/

&context_nml
/

&io_nml
 num_iotasks = 20                <- number of I/O processes for parallel binary I/O
 lredirect_stdout = .false. <- flag to write stdout to log file
 log_filename = 'pop.out' <- root filename (with path) of optional output log file
 luse_pointer_files = .true.   <- flag to turn on use of pointer files
 pointer_filename = 'pointer'
/

&time_manager_nml
 runid        = 'clim_closeditf'
 stop_option    = 'eom'                    <- units of time for 'stop count', eom = end of month
 stop_count     = 1 <- how long in above units to run this segment (use yyyymmdd for date)
 time_mix_opt   = 'avgfit'                 <- Method to suppress leapfrog computational mode
 fit_freq       = 1    <-When using 'avgfit', the intervals per day into which full and half steps must fit
 time_mix_freq  = 17  <- Requested frequency (in steps) for taking mixing steps
 dt_option      = 'steps_per_day' <- units for determining timestep (combined with dt count)
 dt_count       = 170 <- number of timesteps in above units to compute timestep
 impcor         = .true.   <- If .true., the Coriolis terms treated implicitly
 laccel         = .false.    <- if .true., tracer timesteps increase with depth
 accel_file     = 'unknown_accel_file'  <- file containing vertical profile of timestep acceleration factor
 dtuxcel        = 1.0 <- factor to multiply momentum timestep for different momentum and tracer tsteps
 allow_leapyear = .false.  <- use leap years in calendar
 iyear0         = 75          <- year (yyyy) at start of full run sequence
 imonth0        = 1           <- Month at start of sequence
 iday0          = 1           <- day at start of sequence
 ihour0         = 0           <- etc.
 iminute0       = 0
 isecond0       = 0
 date_separator = ' '     <- Character to separate yyyy mm dd in date (' ' means no separator)
/
```

```
&grid_nml
  horiz_grid_opt      = 'file'        read horizontal grid from a file OR create simple lat/lon grid
  horiz_grid_file     = ‘path_to_file/grid.3600x2400.fob.da'
  sfc_layer_opt       = 'varthick' <- surface layer is variable thickness OR rigid lid OR old free
                                      surface formulation
  vert_grid_opt       = 'file' <- read vertical grid structure from file OR compute vertical grid
                                  internally
  vert_grid_file      = 'path_to_file/in_depths.42.dat'
  topography_opt      = 'file'
  topography_file     = 'path_to_file /kmt_noITF.big_endian'
  partial_bottom_cells = .true.   <- use partial bottom cells
  bottom_cell_file    = 'path_to_file/dzbc_pbc.p1_tripole.s2.0-og.20060315.no_caspian_or_black'
  region_mask_file    = 'unknown_region_mask'<-file containing region number @ each gridpoint
  topo_smooth         = .false. <- if .true., smooth topography using 9-point averaging stencil
  flat_bottom         = .false. <- if .true., flat bottom is used
  lremove_points      = .false. <- if .true., remove iso- lated or disconnected ocean points
/

&init_ts_nml
  init_ts_option   = 'restart' <- start from restart OR read initial ocean conditions from a file OR
                                  create conditions from an input mean ocean profile OR create
                                  initial conditions based on 1992 Levitus mean ocean profile
                                  computed internally
  init_ts_file     = 'path_to_file/r.t0.1_42l_nccs01.00750101_fixedU' <- restart file OR file
                                                                         containing 3D potential
                                                                         temperature and salinity
                                                                         at grid points OR file
                                                                         containing depth pro-
                                                                         file of potential tempera-
                                                                         ture and salinity OR (ig-
                                                                         nored for ‘internal’ or
                                                                         when luse pointer files
                                                                         is enabled)
  init_ts_file_fmt = 'bin' <- data format (binary or netCDF) for input init ts file (‘file’ and ‘restart’
                              options only)
/

&diagnostics_nml
  diag_global_freq_opt  = 'nday'
  diag_global_freq      = 1   <-  how often (in above units) to compute and print global diagnostics
  diag_cfl_freq_opt     = 'nday'
  diag_cfl_freq         = 1 <- how often (in above units) to compute and print CFL stability
                               diagnostics
  diag_transp_freq_opt  = 'nday'
  diag_transp_freq      = 1  <- how often (in above units) to compute and print transport
                                diagnostics
  diag_transport_file   = 'transport_file_141lines'
  diag_outfile          = 'diag'
  diag_transport_outfile = 'transp'
  diag_all_levels       = .false. <- if true, tracer mean diagnostics at all vertical levels are output
  cfl_all_levels        = .false.
/
```

```
&restart_nml
  restart_freq_opt = 'nmonth'  <- units of time for 'restart freq'
  restart_freq     = 1 <- number of units between output of restart files
  restart_outfile  = 'path_to_file/restart/r' <- root filename (with path prepended, if necessary)
                                                 for restart files ('runid' and suf- fixes will be added)
  restart_fmt      = 'bin' <- data format (binary or netCDF) for restart output files
  leven_odd_on     = .false. <- create alternating even/odd restart outputs
                                  which over- write each other
  even_odd_freq    = 3840 <- frequency (in steps) for even/odd output
  pressure_correction = .false. <- if true, corrects surface pres- sure error due to (possible)
                                   different timestep. use .false. for exact restart
/

&tavg_nml
  tavg_freq_opt  = 'nmonth'
  tavg_freq      = 1 <- interval in above units for computation & output of time average history files
  tavg_start_opt = 'nstep'
  tavg_start     = 0 <- time in above units after which to start accumulating time average
  tavg_infile    = '' <- restart file for partial tavg sums if starting from restart (ignored if luse pointer
                         files is enabled)
  tavg_fmt_in    = 'bin' <- format for tavg restart file
  tavg_outfile   = 'path_to_file /tavg/t'
  tavg_fmt_out   = 'bin' <- format for tavg output files
  tavg_contents  = 'tavg_contents' <- file name for input file containing names of fields
                                      requested for tavg output
/

&history_nml
  history_freq_opt = 'never'  <- This namelist makes snapshot history files possible, we do not need
                                 this, we want monthly mean history files so it's set to never
  history_freq     = 100000
  history_outfile  = 'unknown_history'
  history_fmt      = 'nc'
  history_contents = 'sample_history_contents'
/

&movie_nml
  movie_freq_opt  = 'nday'
  movie_freq      = 1 <- number of units (movie_freq_opt) between output of movie files
  movie_outfile   = 'path_to_file'/movie/m'
  movie_fmt       = 'bin'
  movie_contents  = 'movie_contents' <- file containing names of fields requested for movie
                                        output
/

&solvers
  solverChoice       = 'ChronGear'
  convergenceCriterion = 1.e-12 <- convergence  criterion:
                                   |δX/X| < convergenceCriterion
  maxIterations      = 1000 <- upper limit on number of iterations allowed
  convergenceCheckFreq = 25 <- check for convergence every convergenceCheckFreq
                               iterations
  preconditionerChoice = 'diagonal'
  preconditionerFile   = 'unknownPrecondFile' <- file containing preconditioner coefficients
                                                 for solver
```

```
&vertical_mix_nml
  vmix_choice = 'kpp'    <- method of computing vertical diffusion
  aidif      = 1.0              <- time-centering parameter for implicit vertical mixing; use of the default
                                   value [1.0] is recommended
  bottom_drag = 1.0e-3 <- (dimensionless) coefficient used in quadratic bottom drag formula
  implicit_vertical_mix = .true.
  convection_type = 'diffusion' <- Convection treated by adjustment or by large mixing
                                        coefficients
  nconvad = 2   <- number of passes through the convec- tive adjustment algorithm
  convect_diff = 1000.0   <- tracer mixing coefficient to use with diffusion option
  convect_visc = 1000.0  <- momentum mixing coefficient to use with diffusion option
  bottom_heat_flx = 0.0    <- constant (geothermal) heat flux (W/m2) to apply to bottom layers
  bottom_heat_flx_depth = 100000.00 <- depth (cm) below which to apply bot- tom heat flux
/

&vmix_const_nml   <- Constant vertical mixing namelist
  const_vvc   = 0.25 <- vertical viscosity coefficient (momentum mixing) (cm2/s)
  const_vdc   = 0.25 <- vertical diffusivity coefficient (tracer mix- ing) (cm2/s)
/

&vmix_rich_nml  <- Richardson-number vertical mixing namelist
  bckgrnd_vvc = 1.0 <- background vertical viscosity (cm2/s)
  bckgrnd_vdc = 0.1 <- background vertical diffusivity (cm2/s)
  rich_mix    = 50.0 <- Coefficient for Richardson-number function
/

&vmix_kpp_nml
  bckgrnd_vdc1    = 0.55  <- base background      vertical  diffusivity (cm2 /s)
  bckgrnd_vdc2    = 0.303615 <- variation in background vertical diffusivity (cm2 /s)
  bckgrnd_vdc_dpth= 2500.0e2 <- depth (cm) at which background vertical diffusivity is vdc1
  bckgrnd_vdc_linv= 4.5e-5 <- inverse of the length scale (1/L in cm-1) over which diffusivity
                                    transition takes place
  Prandtl        = 10.0 <- (unitless) ratio of background vertical vis- cosity and diffusivity
  rich_mix       = 50.0 <- Coefficient for Richardson-number function
  lrich          = .true. <- use Richardson-number for interior mixing
  ldbl_diff      = .true.  <- add double-diffusive parameterization
  lshort_wave     = .true. <- use penetrative shortwave forcing
  lcheckekmo      = .false.<- check whether boundary layer exceeds Ekman or Monin-Obukhov
                                  limit
  num_v_smooth_Ri = 1 <- Number of passes to smooth Richardson number
/

&advect_nml
  tadvect_ctype = 'centered' <- centered differences OR 3rd-order up- winding
/

&hmix_nml
  hmix_momentum_choice = 'del4' <- method for horizontal mixing of momentum (Laplacian,
                                        biharmonic or anisotropic)
  hmix_tracer_choice   = 'del4' <- method for horizontal mixing of tracers (Laplacian, biharmonic
                                        or Gent-McWilliams)
/
```

```
&hmix_del2u_nml
  lauto_hmix        = .true. <- computes mixing coefficient based on resolution
  lvariable_hmix    = .false. <- scales mixing coeff by grid cell area
  am                = 1.e8 <- momentum mixing coefficient (cm2/s)
/

&hmix_del2t_nml
  lauto_hmix        = .true.  <- computes mixing coefficient based on res- olution
  lvariable_hmix    = .false. <- scales mixing coeff by grid cell area
  ah                = 1.e8 <- tracer mixing coefficient (cm2/s)
/

&hmix_del4u_nml
  lauto_hmix        = .false. <- compute mixing coefficient based on resolution
  lvariable_hmix    = .true.  <- scale mixing coeff by grid cell area
  am                = -27.0e17 <- momentum mixing coeff (cm2/s)
/

&hmix_del4t_nml
  lauto_hmix        = .false. <- compute mixing coefficient based on resolution
  lvariable_hmix    = .true. <- scale mixing coeff by grid cell area
  ah                = -3.0e17 <-tracer mixing coefficient (cm2/s)
/

&hmix_gm_nml <- Gent-McWilliams horizontal mixing namelist
/

&hmix_aniso_nml <- Anisotropic viscosity namelist
/

&state_nml    <- Equation of state namelist
  state_choice = 'mwjf' <- McDougall et al. eos OR Jackett and McDougall eos OR polynomial fit
                           to UN- ESCO eos OR linear eos
  state_file = 'internal' <- compute polynomial coefficients inter- nally OR read from file filename
  state_range_opt = 'enforce' <- ignore (ignore) when T,S outside valid polynomial range OR
                                 check (check) and report OR compute (enforce)  eos as if T,S
                                 were in valid range (but don't alter T,S)
  state_range_freq = 100000 <- frequency (steps) for checking T,S range
/

&baroclinic_nml
  reset_to_freezing = .true. <- if .true. and Tsurf(i,j) < Tfreezing, Tsurf(i,j) is reset to Tfreezing
/

&ice_nml
  ice_freq_opt = 'never'  <- frequency units for computing ice formation
  ice_freq     = 100000  <- frequency in above units for com- puting ice formation
  kmxice   = 1   <- compute ice formation above this vertical level
/
```

```
&pressure_grad_nml
  lpressure_avg = .true. <- use pressure averaging to increase time step
  lbouss_correct = .false. <- applies depth-dependent factor to correct for assumed constant
                              density
/

&topostress_nml
  ltopostress = .false.  <- true if topographic stress enabled
  nsmooth_topo = 0     <- number of passes to smooth topography
/

&xdisplay_nml
  lxdisplay = .false. <- if .true., enable x-display
  nstep_xdisplay = 1 <- frequency (in steps) for updating x-display
/

&forcing_ws_nml    <- Windstress forcing namelist
  ws_data_type    = 'monthly'   <- type or periodicity of wind stress forcing
  ws_data_inc     = 1.e20   <- increment (in hours) between forcing times if ws data type='n-hour'
  ws_interp_freq  = 'every-timestep'  <- how often to temporally interpolate wind stress data to
                                         current time
  ws_interp_type  = 'linear'   <- type of temporal interpolation for wind stress data
  ws_interp_inc   = 1.e20   <- increment (in hours) be- tween interpolation times if
                               ws interp freq = 'n-hour'
  ws_filename     =
'/work/e24/sar00059/sar00059/itamoc/scripts/prod_run3_0.5Sv/files_mat/forcing/
ws.o_n_avg.mon' <- name of file containing wind stress, or root of filenames if
                   ws data type='n-hour'
  ws_file_fmt     = 'bin' <- format of wind stress file
  ws_data_renorm(1)  = 10. <- renormalization constants for the components in the wind stress
                              forcing file
  ws_data_renorm(2)  = 10.
/

&forcing_shf_nml   <- Surface heat flux forcing namelist
  shf_formulation    = 'normal-year'   <- surface heat flux formulation
  shf_data_type      = 'monthly' <- type or periodicity of surface heat flux forcing
  shf_data_inc       = 1.e20  <- increment (in hours) between forcing times if shf data type='n-hour'
  shf_interp_freq    = 'every-timestep'  <- how often to temporally in- terpolate surface heat flux
                                            data to current time
  shf_interp_type    = 'linear' <- type of temporal interpola- tion for surface heat flux data
  shf_interp_inc     = 1.e20 <- increment (in hours) be- tween interpolation times if
                                shf interp freq = 'n-hour'
  shf_restore_tau    = 1.e20 <- restoring timescale (days) if type restoring
  shf_weak_restore   = 0.0 <- restoring flux for weak restor- ing in bulk-NCEP
  shf_strong_restore = 15.8 <- restoring flux for strong restoring in bulk-NCEP
  shf_filename       =
'/work/e24/sar00059/sar00059/itamoc/scripts/run_clim_closeditf/files_mat/forcing/
shf.normal_year+Hurrell.monthly' <- name of file containing surface heat flux data, or root of
                                    filenames if shf data type='n-hour'
  shf_file_fmt       = 'bin' <- format (binary or netCDF) of shf file
  shf_data_renorm(3) = 1. <- renormalization constants for the components in the sur- face heat
                             flux forcing file
  shf_data_renorm(4) = 1.
```

```
&forcing_sfwf_nml   <- Surface fresh water flux forcing namelist
  sfwf_formulation    = 'bulk-NCEP' <- surface fresh water flux formulation. Bulk-NCEP means:
                                       calculate fluxes based on atmospheric state variables and
                                       radiation similar to a fully-coupled model (and using bulk
                                       flux formulations extracted from the NCAR flux coupler)
  sfwf_data_type      = 'monthly' <- type or periodicity of surface fresh water flux forcing
  sfwf_data_inc       = 1.e20 <- increment (hours) between forcing times if sfwf data type='n-hour'
  sfwf_interp_freq    = 'every-timestep' <- how often to temporally in- terpolate surface fresh
                                            water flux data to current time
  sfwf_interp_type    = 'linear' <- type of temporal interpola- tion for surface fresh water flux data
  sfwf_interp_inc     = 1.e20 <- increment (hours) between interpolation times if
                                 sfwf interp freq = 'n-hour'
  sfwf_restore_tau    = 1.e20 <- restoring timescale (days) if restoring
  sfwf_weak_restore   = 0.009   <- restoring flux for weak restor- ing in bulk-NCEP
  sfwf_strong_restore = 0.11 <- restoring flux for strong restoring in bulk-NCEP
  sfwf_filename       =
'/work/e24/sar00059/sar00059/itamoc/scripts/run_clim_closeditf/files_mat/forcing/
sfwf.CORE+runoff.monthly' <- name of file containing surface fresh water flux data, or root of
                             filenames if sfwf data type='n-hour'
  sfwf_file_fmt       = 'bin'  <- format (binary or netCDF) for sfwf file
  sfwf_data_renorm(1)    = 0.001 <- renormalization constants for components in
                                    sfwf forcing file
  sfwf_data_renorm(2)    = 1.
  ladjust_precip      = .true. <- adjust precipitation to balance water budget
  lfw_as_salt_flx     = .true. <- treat fresh water flux as virtual salt flux
                                  when using varthick sfc layer
  runoff            = .true.
/

&forcing_pt_interior_nml  <- Interior potential temperature forcing namelist
  pt_interior_formulation = 'restoring' <- interior pt formulation
  pt_interior_data_type   = 'none' <- type or periodicity of in- terior pt forcing
  pt_interior_data_inc    = 1.e20 <- increment (hours) between forcing times if data type 'n-hour'
  pt_interior_interp_freq = 'never' <- how often to temporally interpolate interior pt data to
                                       current time
  pt_interior_interp_type = 'nearest' <- type of temporal interpo- lation for interior pt data
  pt_interior_interp_inc  = 1.e20 <- increment (hours) between     interpolation times if
                                     interp freq    = 'n-hour'
  pt_interior_restore_tau = 1.e20  <- restoring    timescale (days) if restoring
  pt_interior_filename    = 'unknown-pt_interior' <- file containing interior pt data, or root of
                                                     filenames if data type='n-hour'
  pt_interior_file_fmt    = 'bin' <- file format (binary or netCDF)
  pt_interior_data_renorm = 1. <- renormalization constants for components in
                                  interior pt forcing file
  pt_interior_restore_max_level = 0  <- maximum level for inte- rior pt restoring
  pt_interior_variable_restore  = .false. <- enable variable interior pt restoring
  pt_interior_restore_filename  = 'unknown-pt_interior_restore' <- name of file contain- ing
                                                                   variable interior pt restoring data
  pt_interior_restore_file_fmt  = 'bin' <- file format (binary or netCDF)
/
```

```
&forcing_s_interior_nml <- Interior salinity restoring namelist
  s_interior_formulation = 'restoring' <- forcing formulation
  s_interior_data_type   = 'none' <- type or periodicity of interior salinity forcing
  s_interior_data_inc    = 1.e20 <- increment (hours) be- tween forcing times if data type 'n-hour'
  s_interior_interp_freq = 'never' <- how often to tempo- rally interpolate interior S data to
                                      current time
  s_interior_interp_type = 'nearest' <- type of temporal interpolation for interior S data
  s_interior_interp_inc  = 1.e20 <- increment (in hours) between interpolation times if
                                    interp freq 'n-hour'
  s_interior_restore_tau = 1.e20 <- restoring timescale (days) if restoring
  s_interior_filename    = 'unknown-s_interior' <- name of file containing interior S data, or
                                                   root of filenames if data type 'n- hour'
  s_interior_file_fmt    = 'bin' <- format   (binary   or netCDF) of s interior file
  s_interior_data_renorm = 1. <- renormalizationconstants for components in interior S forcing
                                 file
  s_interior_restore_max_level = 0 <- maximum level for inte- rior S restoring
  s_interior_variable_restore  = .false. <- enable variable interior S restoring
  s_interior_restore_filename  = 'unknown-s_interior_restore' <- name of file containing
                                                                variable interior S restoring data
  s_interior_restore_file_fmt  = 'bin'
/

&forcing_ap_nml    <- Atmospheric pressure forcing namelist
  ap_data_type   = 'none' <- type or periodicity of atmo- spheric forcing forcing
  ap_data_inc    = 1.e20 <- increment (in hours) between    forcing   times if ap data type='n-hour'
  ap_interp_freq = 'never'
  ap_interp_type = 'nearest'
  ap_interp_inc  = 1.e20
  ap_filename    = 'unknown-ap'
  ap_file_fmt    = 'bin'
  ap_data_renorm = 1.
/

&coupled_nml
  coupled_freq_opt = 'never' <- unit of time for coupled freq
  coupled_freq     = 100000
/

&tidal_nml
/

&passive_tracers_on_nml
  dye_on = .false.
  iage_on = .false.
/
```

```
&dye_nml
  init_dye_option = 'restart'
  init_dye_init_file = 'same_as_TS'
  dye_region_file = 'blablabla'
  dye_region_file_fmt = 'bin'
  tracer_init_ext(1)%mod_varname  = 'DYE'
  tracer_init_ext(1)%filename     = 'unknown'
  tracer_init_ext(1)%default_val  = 0.0
  dye_tadvect_ctype = 'lw_lim'
/

&sw_absorption_nml
/

&float_nml
/
```

For more information about parameters that are not described check:
http://climate.lanl.gov/Models/POP/UsersGuide.pdf