ESP-r Common Coding Resources

by Jon Hand
Energy Systems Research Unit
of the University of Strathclyde
Version of 26 March 2009

Summary

The source code distribution of ESP-r contains many resources for the development community. Those wishing to understand the existing functions and subroutines within the Fortran, C and C++ code base can browse this document for the API which their code must interact with. It should also be possible to use this reference to understand the structure of the code and the dependencies within the code distribution.

Documentation of source code tends to lag the evolution of the code and readers are advised to also review the source code. As the ESP-r distribution is under source code control (via subversion), it is hoped that this document will be updated as the source code evolves.

This document should be considered work-in-progress. API details are still being added so check for new versions of this document regularly.

The source code of ESP-r is subdivided into a number of topic-focused folders. Each of the folders separates the source files by functional task, data structure and/or analysis domain. There is a rough order within the roughly million lines of code. There are chaotic corners which have yet to be consolidated and there is some duplication which awaits consolidation.

Below is the overall layout of the source distribution along with notes on the contents of each folder:

```
|-- src
|   |-- archive     documents for developers
|   |-- bin         scripts for developers and users
|   |-- bitmaps     images used by X11 version
|   |-- cetc        code from Natural Resources Canada and XML
|   |-- climate     climate data sets (ASCII versions)
|   |-- databases   materials, optics, constructions, plant components
|   |-- env         example Unix/Linux dot files
|   |-- esruXXX     ESP-r module specific source code
|   |-- esrucom     ESP-r common source code
   ...
|   |-- include     header files used for Fortran and C code
|   |-- lib         library code and user interface code
|   |-- manual      documentation about ESP-r and operating system variants
|   |-- shocc       library of occupant preferences
|   |-- training    example models for use in workshops and training
|   |-- tutorial    holds additional contextual help text for interfaces
|   `-- validation  models for use in formal validation e.g. BESTEST
`-- tester
    |-- additional_tests   infrequently used automated test models
    |-- scripts            scripts for running automated tests
    `-- test_suite         test models for automated tests
```

One key aspect of source code evolution is testing of changes and the tester folder includes a number of scripts to automate the process and over 170 test models to be run as part of the formal testing process. The tester folder are listed below:

```
tester
```

```
|-- additional_tests
|   |-- A42_combustion_cogen_comp_tests  notes to be added
|   |-- A42_fuel_cell_comp_tests         notes to be added
|   |-- ASHRAE-140                       notes to be added
|   `-- HOT3000_test_cases               notes to be added
|-- scripts
`-- test_suite
    |-- Annex42_fuel_cell                notes to be added
    |-- alberta_infil_model              notes to be added
    |-- ascii_dbs                        notes to be added
    |-- basesimp                         notes to be added
    |-- bld_PV                           notes to be added
    |-- bld_ground_reflectivity          notes to be added
    |-- bld_hc_ISO15099                  notes to be added
    |-- ccht_benchmark                   notes to be added
    |-- cellular_offices                 model with cellular offices
    |-- cetc_battery_model               notes to be added
    |-- elec_gain_into_zone              notes to be added
    |-- esru_benchmark_model             similar to validation/benchmark/QA
    |-- h3kreports                       notes to be added
    |-- idealized_hvac                   notes to be added
    |-- multi_year_simulations           notes to be added
    |-- plt_SDHW                         notes to be added
    |-- plt_adsorption_storage           notes to be added
    |-- plt_boundary_conditions          notes to be added
    |-- plt_elec_net                     notes to be added
    |-- plt_electric_HWT                 notes to be added
    |-- plt_lookup_table                 notes to be added
    |-- plt_pre_A42_PEMFC_model          notes to be added
    |-- plt_pre_A42_SOFC_model           notes to be added
    |-- plt_radiant_floor                notes to be added
    |-- plt_solar_collector              notes to be added
    |-- plt_trnsys_wrapper               notes to be added
    |-- plt_zone_heat_gain_coupling      notes to be added
    `-- pv_example                       notes to be added
```

The validation of ESP-r is also supported by a number of standard test models and scripts in the validation folder.

```
src
`-- validation
    |-- BESTEST
    |   |-- 195    test solid conduction
    |   |-- 200    test long wave radiation exchange at windows
    |   |-- 210    test long wave radiation external exchange on
    |   |-- 215    test long wave radiation external exchange off
    |   |-- 220    test long wave radiation internal exchange
    |   |-- 230    test infiltration
    |   |-- 240    test internal gains
    |   |-- 250    test exterior solar
    |   |-- 270    test south exterior solar
    |   |-- 280    test cavity albedo
    |   |-- 290    test south horizontal overhang
    |   |-- 300    test east and west external solar
    |   |-- 310    test east and west overhang and fins
    |   |-- 320    test thermostat deadband
    |   |-- 395    test solid conduction
    |   |-- 400    test surface convection and long-wave exchange
    |   |-- 410    test infiltration
    |   |-- 420    test internal heat generation
    |   |-- 430    test external solar incident
    |   |-- 440    test internal solar absorptance
    |   |-- 600    test south solar transmission
    |   |-- 600FF
    |   |-- 610    test south overhang
    |   |-- 620    test east and west solar transmission
    |   |-- 630    test east and west overhangs and fins
    |   |-- 640    test night setback
    |   |-- 650    test venting
    |   |-- 650FF
```

```
    |   |-- 800    test thermal mass with no solar
    |   |-- 810    test thermal mass with solar
    |   |-- 900    test thermal mass and solar interaction
    |   |-- 900FF
    |   |-- 910    test south overhang and thermal mass
    |   |-- 920    test east and west mass and solar interaction
    |   |-- 930    test east and west shading and mass interaction
    |   |-- 940    test night setback and mass interaction
    |   |-- 950    test venting and mass interaction
    |   |-- 950FF
    |   |-- 960    test test passive inter-zone transfer
    |   |-- 990
    |   |-- climate     climate files for use with BESTEST
    |   `-- dbs         databases for use with BESTEST
    |-- CEN
    |   |-- 13791
    |   `-- 15265
    |-- analytical
    |   `-- conduction01
    `-- benchmark
        |-- CFD
        |   |-- Archive_Feb2004    archive of standard assessments
        |   `-- Models             models for testing CFD
        `-- QA
            |-- Archive_linux_X11_040309
            |-- Archive_linux_X11_230209   archive of earlier predictions
            |-- benchmark_model     a set of archaic test models
            |-- model         test models with older geometry format
            `-- model_1.1     test models with new geometry format
```

The documentation associated with ESp-r is found in the manual and the archive folders.

```
src
|-- manual
|   |-- Adding_features
|   |-- Data_model           formal description of the ESP-r data model
|   |-- ESRUlib              notes about library subroutines (out-of-date)
|   |-- Implement
|   |-- Manual               text for the manual (out-of-date)
|   |   `-- Figs             files for figures to the manual
|   `-- OS
|       |-- Apple            instructions for OSX install
|       |-- Cygwin           instructions for Cygwin install
|       |-- Linux            instructions for Linux install
|       `-- Native_windows   instructions for Native Windows install
```

There are a number of example models for use with training workshops and courses and these are located in the training folder structure.

```
src
|-- training
|   |-- 3_windows         model with different flow network window representations
|   |-- CFD_room          model with CFD domain
|   |-- acoustic
|   |   `-- EOS_atrium    model including acoustic calculations
|   |-- basic             a simple model with numerous variants
|   |-- burdie            a house with moisture issues
|   |-- cellular_bc       base case version of two cellular offices
|   |-- cellular_bound    cellular offices with upper and lower bounding zones
|   |-- cellular_contam   cellular offices with contaminate tracking
|   |-- cellular_cvvt     cellular offices with idealized CV air supply
|   |-- cellular_earth    cellular offices with earth tube air supply
|   |-- cellular_flh      cellular offices with floor heating
|   |-- cellular_furn     cellular offices with furniture and internal mass
|   |-- cellular_hires    cellular offices with higher resolution geometry
|   |-- cellular_hvac     cellular offices with HVAC
|   |-- cellular_hybrid   cellular offices with hybrid ventilation
|   |-- cellular_natv     cellular offices with operated windows
|   |-- cellular_pv       cellular offices with PV embedded in facade
```

```
|   |-- cellular_shd      cellular offices with shading obstructions
|   |-- cfd
|   |   |-- IEA_A20       IEA Annex 20 models
|   |   |-- M_Age         model including mean age of air
|   |   |-- RoomVent98    models used for RoomVent 1998 paper
|   |   |-- displ_vent    models demonstrating displacement ventilation
|   |   `-- rad_htg       model with radiant heating
|   |-- cg_ctl
|   |   |-- coupling      coupling of ESp-r and Radiance
|   |   |-- daylit_coef   model using Radiance daylight coefficients
|   |   |-- el_chrom      model with electro chromic optical controls
|   |   `-- static
|   |-- chp
|   |   |-- sport_cen     model of sports centre with co-generation
|   |   `-- unit
|   |-- constr
|   |   |-- adapt         model with adaptive thermophysical properties
|   |   `-- tp_sub
|   |-- el_chr_ctl        model with electro chromic optical controls
|   |-- flow              model demonstrating network flow
|   |-- gridding          model with 2D conduction gridding
|   |-- house
|   |   |-- sun_space     house with a sun space
|   |   `-- svph          house with a sun space and solar ventilation preheating
|   |-- mould             model which includes mico-toxin parameters
|   |-- network           model with network flows
|   |-- office            a portion of an office building
|   |-- office_dfs        a portion of an office building with a double facade
|   |-- office_vent       a portion of an office building with controlled facade vent
|   |-- pattern           a folder with sample operation files
|   |-- pid               a model for demonstrating PID controls
|   |-- plant
|   |   |-- ac_pp           a model with primitive part representations of air conditioning
|   |   |-- ahu             a model with air handler plant components
|   |   |-- coil_pp         primitive part representation of a fan coil
|   |   |-- conv_ac_sys
|   |   |-- hvac_bas
|   |   |-- hvac_vav        VAV system represented by plant components
|   |   |-- mixed_ac_sys
|   |   |-- solar           a model with various solar components
|   |   |-- vent_detailed   a model with detailed mechanical ventilation
|   |   |-- vent_simple     a model with simple mechanical ventilation
|   |   `-- wch             a model with web central heating
|   |-- pv_facade           research facility with PV embedded facade
|   |-- simple              a single zone model with variants
|   `-- sunroom
|
```

High level goals

ESP-r is a mixed language environment. Essentially the data model and the decision about what to present to the user and how to interpret user actions is handled in the Fortran source. The C code in lib implements the directives about what to display and captures and passes back user actions (mouse movements and keystrokes). The goal is for the code in the application source folders to be substantially isolated from the underlying graphic API via the use of intermediate code in the lib folder. There are a few exceptions to this goal and the code relies on very few #ifdef statements and direct calls to the underlying graphic APIs.

The source code and databases are held in the src folder structure as shown below

```
src
|-- archive     documents for developers
|-- bin         scripts for developers and users
|-- bitmaps     images used by X11 version
|-- cetc        code from Natural Resources Canada
|   `-- h3kreports  XML report generation code
|       |-- Debug
|       `-- xsl
|-- climate     climate data sets (ASCII versions)
```

```
|-- databases   materials, optics, constructions, plant components
|   '-- UK_NCM  patterns of occupancy used in UK national calculation method
|-- env         example Unix/Linux dot files
|-- esruaco     source code specific to the acoustics module
|-- esrub2e     source code specific to BEMS import module
|-- esrubld     source code for zone solver
|-- esrubps     source code for the multi-domain solver
|-- esruc2e     source code for air flow pressure coefficients
|-- esruclm     source code for the clm module
|-- esrucnv     source code for importing and exporting to 3rd party tools
|-- esrucom     common code used by various modules
|-- esructl     source code for global controllers
|-- esrudbm     source code for generic databases
|   '-- db      sample generic databases
|-- esrudfs     source code for the CFD solver and gridding setup
|   '-- Info
|-- esrue2r     source code for interface to Radiance
|-- esrueco     source code for environmental impacts module
|-- esrugrd     source code for 2D and 3D gridding
|-- esruish     source code for shading and insolation pre-calculator
|-- esrumfs     source code for mass flow networks
|-- esrumld     source code for mico-toxin assessments
|-- esrumrt     source code for surface-to-surface viewfactors
|-- esrunet     source code for an iconic network sketch module
|-- esrupdb     source code for the plant template library manager
|-- esrupdf     source code for electrical network manager
|-- esrupfs     source code for electrical network solver
|-- esruplt     source code for plant component solver
|-- esruprj     source code for the project manager module
|-- esrures     source code for the results analysis module
|-- esrurun     source code for converting 3rd party descriptions into ESP-r models
|-- esruvew     source code for a hidden line wire-frame viewer
|-- esruvld     source code for validation and uncertainty studies
|-- esruvww     repository for archaic code
|-- include     header files for fortran and c code
|-- lib         library code and user interface code
|-- shocc       library of occupant preferences
|-- tutorial    holds additional contextual help text for interfaces
```

The ESP-r API

Having reviewed the overall layout of the ESP-r distribution we now turn to the details of the API. For each of the folders we will review the source files and the subroutines and/or functions they hold. This section will focus on high level descriptions and the Appendices will provide the full API.

The lib folder

The src/lib folder contains source code blocks that are so widely used that it makes sense to compile them into libraries. Included in this folder are a number of Fortran code blocks that act as wrappers around the calls to the low level user interaction API. The lib folder also includes the C source code related to the interfaces (X11 or GTK or pure-text) of ESP-r.

Because ESP-r modules can be compiled for the GTK API, the X11 API and as a pure-text API there is a subdivision of the source code in the lib folder.

esru_lib.F
: Code which is independent of the choice of graphic library tends to be held in esru_lib.F and this is included in all variants of the library linked with the ESP-r application.

esru_libGTK.F
: Fortran code which provides an interface to the underlying GTK API is found in esru_libGTK.F. In some cases there are differences in the number of parameters of the C function, in other cases there are different C functions which are called to carry out a specific task. The C source used by esru_libGTK.F is esp-r.c esp_menu.c esp_text.c esp_ask.c esp_draw.c and exp_file.c

esru_libNonGTK.F
: Fortran code which provides an interface to either the pure-text or X11 API are found in esru_libNonGTK.F. There is a close match in the names of the Fortran subroutines in esru_libNonGTK.F and esru_libGTK.F but their details have been adjusted to the requirements of the specific API they are dealing with (esru_x.c or esru_nox.c).

esru_libX11.F
: This small source file provides a few wrappers which are needed for X11 specific API and primarily calls esru_x.c.

esru_ask.F
: ESP-r modules include context sensitive help and for each module there is a block of text which is held in esru_ask.F which provides a high level description of the module.

esru_blk.F
: ESP-r code makes use of Fortran common blocks and a number of these are initialized in the file esru_blk.F. This is roughly equivalent to the intent of the Fortran BLOCKDATA statement.

esru_fc.f
: Pure-text versions of ESP-r have no need for the graphic API calls and also have almost no need for C code. The file esru_fc.f provides dummy Fortran subroutines with the same names and parameter types as the X11 API wrappers.

libXML.a
: One option for ESP-r is to generate XML reports of building and system performance predictions. This is handled by C++ code in the cetc/h3kreports source folder.

Conceptually, if the ESP-r community decided to support another graphic API, this would be accomplished by creating another source file esru_libXXX.F to provide the intermediate wrappers.

Summary of functions

The list below is a summary of the contents of each of the source files in the lib folder. The full parameter list for these subroutines and functions can be found in Appendix A.

esru_ask.F

```
C   cpwpk()     Displays a copyright notice via help text data structure.
C   askabout()  Creates text for a pop-up notice for each module.
```

esru_blk.F

```
C ezero()       Loads the COMMONS used by libgtkesru.a, libxesru.a and libnoxesru.a.
```

esru_lib.F

The file esru_lib.f is a collection of low-level ESP-r fortran libraries that are applicable to all versions of ESP-r (that is, X11, GTK and noX). Code pertaining to version-specific configurations should be placed in the libGTK, libX11 and libNONGtk files.

```
C  st2name: Given 'string' swap blanks & wildcards to _ return as 'name'.
C  st2file: Given 'string' strip blanks & wildcards and return as 'name'.
C  backslashit: Given 'string' swap / to  : return as 'name'.
C  iprevblnk: Given a string, return position of blank just before ipos.
C  inextblnk: Given a string, return position of blank just after ipos.
C  icutstr: Given a string, cuts icut characters from position ipos.
C  EASKAB:  Generic A/B choice facility returning 1 or 2 according to which
C           of the choices has been chosen.
C  EASKABC: Generic A/B/C choice facility returning 1, 2 or 3 according
C           to which of the choices has been chosen.
C  EASKABCD: Generic A/B/C/D choice facility returning 1-4 according
C           to which of the choices has been chosen.

C  isadll:  Checks if module is being used as a dll (silent running).
C  isunix:  Checks if machine type is Unix or NT.
C  usrhome: Returns user's home directory.
C  usrname: Returns user's login name.
C  usrdir:  Returns user's current directory.
C  esppid:  Find current process number.
C  tstamp:  Date stamp with message.
C  dstamp:  Get date stamp in the form: Fri Jan 23 09:34:31 1998.
C  comparedate: is passed two date strings (generated by call to dstamp)
C    and returns 1 if first date is more current, 0 if the same, otherwise -1.
C  getsecs: Get computer clock seconds.

C  iEGetArrW: integer function that breaks a string into an array.

C  DAYCLK:  Print day, month, day no. and time based on the julian day & time.
C  FDROOT:  Given a file name see if it contains a path.
C  EFOPSEQ: Open a sequential file with existance flag & path check.
C  EFOPRAN: Open a random access file with existance flag & path check.
C  ADDPATH: Return file name appended onto the path and logical concat.
C  GETTOKENS checks a string, returning nb of tokens and array of tokens.
C  C2FSTR:  Convert c function returned string to fortran format.
C  EKPAGE:  Maps key characters, pages & array indexs in long display lists.
C  KEYIND:  Decodes EMENU index and returns the array index of the item.
C  EPMENSV: saves menu definitions (common block PMENU).
C  EPMENRC: recovers menu definitions (common block PMENU) from PMENUSV.
C  EPAGE:   Screen control: page without waiting.
C  EPAGEW:  Screen control: Wait before paging.
C  EPWAIT:  Screen control: Wait without paging.
C  EPAGEND: Screen control: Page then close window if open.

C  ang3vtx: Get angle between three vertex.
C  CLOSE3D: Calc min dist between two lines in 3D return dist and closest points.
C  CROW:    Function returns shortest dist between two points P(3) and Q(3).
C  CROWXYZ: Function returns shortest dist between two points Px,Py,Pz and Qx,Qy,Qz.
C  UVXYZ:   Returns Unit vector Ux,Uy,Uz from two points Px,Py,Pz and Qx,Qy,Qz.
C  UVAB:    Returns Unit vector U of vector A.
C  PLNDIS:  Finds distance DIST from a point (x,y,z) to a plane (eq EQN).
C  PLNDANG: Finds dihedral angel between two planes given their equations.
C  AVER:    Returns the centre of gravity of an polygon array.
C  CROSS:   Performs a cross-product on vectors A() & B() returning in C().
```

```
C  CROSS2:  Performs a cross-product on vectors passing parameters as ax,ay,az etc.
C  DOT3(a,b,product) Return dot product of two vectors a & b.
C  ZEROS:   Clear a 4x4 array prior to doing viewing transforms.
C  ECLOSE:  Checks tolerance between two real numbers.
C  ESIND:   Function returning SIN of angle where angle is given in degrees.
C  ECOSD:   Function returning COS of angle where angle is given in degrees.
C  ETAND:   Function returning TAN of angle where angle is given in degrees.
C  IFAX :   Integer function returning the integer part of it's argument.
C  EAZALT:  Computes the solar azimuth & altitude angles at current time.
C  AGNXYZ:  Given solar azi & elev return viewing coords @ 1000m.
C  ORTTRN:  Multiplies a point (XM,YM,ZM) by the transform matrix
C           TMAT to return the point XO,YO,Z0.
C  VECTRN:  Transforms a vector VECIN by the 4x4 (homogeneous) matrix
C           TMAT and returns the vector VECOUT.
C  VECPLN:  Returns the point of intersection X,Y,Z between a line
C           defined by X1,Y1,Z1 & X2,Y2,Z2 and a plane defined in PEQN.
C  HMATMUL: Multiplies the homogenous (4x4) matrices A by B returning C.
C  HREVMAT: Takes the homogenous perspective transformation PER and
C           returns it's inverse REP making use of CROUT.
C  CROUT:   Inverts a nonsymetric square matrix A (order N), returning
C           the matrix B and IERR =-1 if matrix is singular.
C  DPACC:   Provides double precision accumulation of inner products for
C           CROUT in the form SUM(+,-)SUM(+,-)AB.
C  EYEMAT:  Provides transform eye-point - viewpoint....

C  INTSTR: Converts integer into string (10 char) w/o leading blanks.
C  RELSTR: Converts a real into a string (12 char) w/o leading blanks.
C  REL16STR: Converts a real into a string (16 char) w/o leading blanks.
C  EXPSTR: Converts a exponential into a string (10 char) w/o leading blanks.
C  ARLIST: takes a real array (rlist) and builds a packed string.
C  AILIST: takes an int array (ilist) and builds a packed string.
C  ASLIST: takes an string array (list*24) and builds a packed string.
C  ASLIST2 takes the range (inst to inil) items of an string array (list)
C  ASFLIST: takes an string array (list*48) and builds a packed string.
C  STRIPC: Strips comments from a ASCII file str (124 char) & returns data.
C  LSTRIPC: Strips comments from a ASCII file str (248 char) & returns data.
C  STRIPC400 strips comments from a ASCII file (400 char long) string and returns the data.
C  CHITMS: Checks a character string & returns the number of data items.
C  LCHITMS Checks a (248) character string (A), returning the number of data
C  CHITMS400: Checks a (400) character string & returns the number of data items.
C  CHARCH: Routine to check a string for a specific number of data items.
C  NOYES:  INTEGER FUNCTION to read the answer Y,y,1,N,n,0 to a question.
C  IFIRST: Function returning ASCII value for 1st char in a string ISTR.

C  EDAY:   Returns the year day number when passed the day of month & month.
C  EDAYR: returns the day and month numbers from the day-of-year.
C  EWEEKD: returns the day of the week given the day of month, month
C          and year as integers.
C  EDAYCH: Checks for errors in the users specification of the day & month.
C  DATTIM: returns UNIX time via a string in the form : 16 Sep 73 14:23.
C  STDATE: Takes the day of year and returns two descriptive strings:
C          DESCR takes the form '12 Jan' & DESCR1 takes the form 'Fri 12 Jan'.
C  ESTIME: Takes an integer time-step and returns two string descriptions:
C          DESCRH in the form '12h28' & DESCRD which takes the form of 12.46.
C  EDTIME: takes an real time and returns two string descriptions:
C          DESCRH in the form '12h28' and DESCRD which takes the form of 12.46,
C  EPERSTR: creates three strings representing the start and
C           stop time of a diary period.

C  EGETW:  Finds first word after pos k in a string (' ' ',' '|' or tab separated).
C  EGETP:  Finds first phrase after pos k in a string (delimiter separated).
C  EGETDQ:  Finds first quoted phrase after pos k in a string (delimeter separated).
C  EGETWI: As EGETW for an integer with range checking & error messages.
C  EGETWR: As EGETW for a real with range checking & error messages.
C  EGETWRA: Recovers (IRVA) reals of real array (RVA) from an ASCII file.
C  EGETWIA: Recovers (IRVA) int of array (IVA) from an ASCII file.
C EGETAGWIA recovers (IRVA) integers of integer array (IVA) from a string
C   TSTR (from position K) and if TSTR does not hold all of the array then
C   it continues reading from an ASCII file (unit IAF) reading as
C   many lines as necessary to recover the data.
C EGETAGWRA recovers (IRVA) reals of real array (RVA) from a string
C   TSTR (from position K) and if TSTR does not hold all of the array then
```

```
C   it continues reading from an ASCII file (unit IAF) reading as
C   many lines as necessary to recover the data.
C EGETAGWSA recovers (ISVA) words into string array (SVA) from a string
C   TSTR (from position K) and if TSTR does not hold all of the array then
C   it continues reading from an ASCII file (unit IAF) reading as
C   many lines as necessary to recover the data
C EGETAGWPA recovers (ISVA) phrases into string array (SVA) from a string
C   TSTR (from position K) and if TSTR does not hold all of the array then
C   it continues reading from an ASCII file (unit IAF) reading as
C   many lines as necessary to recover the data
C  EGETRM: Returns the remainder (RSTR with no leading blanks) from
C          a text string (TSTR) after position k.

C  ERPFREE: Is used to close any file.
C  EFDELET: Delete the current file opened under IUN and return ISTAT.
C  FPOPEN:  is used to open a file with a name.
C  FPRAND:  is used to open a file with a name for random access.

C  SITELL2S: Takes site lat & long and returns descriptive string.
C  SIGFIG:  Returns number to specified number of significant figures.
C  SIpre:   Returns suitable SI prefix for number supplied.
C  pronam:  Returns the characters of a string after the last
C           occurrence of '/' or '\'
C DNOTZERO: Function returns non zero value with the same sign (dbl precision).
C ANOTZERO: Function returns non zero value with the same sign (sngl precision).

C  ASKTIM:  Enquire month and day and time (real for view).
C  SOLAIR:  Returns solair temperature.
C  LISTAS:  General read & display of an ascii file.
C  SHOWMEH: Extracts additional help text matching the key OTHER from
C           an external file
C  SDELIM:  Replaces blanks in a string A with alternative delimiter.
C  EDDISP:  As edisp with text separated with current delimiter.
C EDISP248 Displays a 248 char block of text (text or graphic).
C clrtextbuf: Clears the graphic text buffer common blocks.

C  UPDVIEW: Called from C to pass back updates to common VIEWPX & GFONT
C  WIREPK:  Called from 'C' upon a wireframe control button pick.
C  EPROMPT: Does nothing, for compatibility only.
```

esru_libGTK.F and esru_libNonGTK.F

The files esru_libGTK.f and esru_libNonGTK.F are a matched collection of mid-level ESP-r fortran wrappers around the GTK API and X11 and/or pure-text API. Unless otherwise noted, the API for each subroutine is the same for both of these files. Details of the API are included in Appendix B.

```
C  EASKI:   Ask user for an integer with prompt, error messages & range
C           checking as passed parameters.
C  EASKR:   Ask user for a real number with prompt, error messages & range
C           checking as passed parameters.
C  EASKE:   Ask user for a real in exponential format, otherwise as easkr.
C  EASKF:   Ask user for a file name with prompt, error messages & help.
C  EASKS:   Ask user for a string with prompt, error messages & help.
C  EASKS248: Edit a long (248 char) string in sections.
C  EASKSCMD Ask user for a string with prompt, alt, error messages & help.
C  EASKSCNCL Asking the user for a text string with prompt cancel error & help.
C  EASKS2CMD Ask user for a string with prompt, 2 alts, error messages & help.

C  EPICKS:  Allows several selections to be made from an array of strings.
C  MENUATOL Single item menu pick with items passed via parameters.
C  ASKOK:   Generic yes/no/default facility returning OK as a logical parameter.
C  EASKAB:  Generic A/B choice facility returning 1 or 2 according to which
C           of the choices has been chosen.
C  EASKABC: Generic A/B/C choice facility returning 1, 2 or 3 according
C           to which of the choices has been chosen.
C EASKABCD: Generic A/B/C/D choice facility returning 1-4 according
C           to which of the choices has been chosen.
C  EASKPER: Provides interface to specification of a period of days.
```

```
C eAskPerYear: Provides interface to specification of a multiyear period of days.
C easkPerGen: Handles specification of both single-year and multi-year
C           simulation periods.

C  EMPAGE:  Low level screen control for paging based on terminal model.
C  ELINC:   Controls scratch pad output for text screens.
C  EPAGES:  Initialise terminal, set up a scratch pad & line count.
C  SETLINC: Allows the user to change the length of the text page.
C  EMENU:   Control variable width menu display on various terminals.
C  EWMENU:  Is the binding to C function for menu dialogue.
C  VWMENU:  Is the binding to C function for variable width menu.
C  EMKEY:   Returns key (a-z) for a menu item based on data array index.
C  USRMSG:  Generic message/prompt facility for all terminal types.
C  LUSRMSG:  Generic long message/prompt facility for all terminal types.
C  EDISP:   Generic send text to scrolling display (text or graphic).
C  PHELPD:  Displays the current contents of common pophelp.
C  PHELPW:  Returns the width IWH of the longest popup help string.
```

The common code folder

One goal of the ESP-r development community is to have one subroutine or function for each task in order to avoid duplication. Facilities that are needed by multiple modules tend to be placed in the esrucom folder and the relevant Makefile for each module links to source files in esrucom as required. For example, there is one subroutine that reads a zone geometry file and this is found in esrucom/egeometry.F and this is referenced by all the other modules. Updating egeometry.F and recompiling spreads new functionality to all dependent modules. The following files are found in the esrucom folder:

BC_data.F

This file contains procedures for managing user-specified boundary conditions. Used by NRCan staff.

```
C process_BC_data_file: Read a boundry condition data definition
C    file, and write out a binary database containing the necessary
C    records.

C get_BC_data_by_col: Collect boundry condition data for the current
C    timestep, and return the data corresponding to a given
C    column number.

C get_BC_data_by_name: Collect boundry condition data for the current
C    timestep, and return the data corresponding to a given
C    column header name.

C report_BC_data: transport boundry condition data to the H3K
C    xml reporting facility.

C cleanup_BC_data: Delete temporary files and free I/O channels
```

Lookup_data.F

This file contains subroutines and functions for managing a user-specified data table.

```
C process_Lookup_data_file: Reads a file that contains the
C data characteristics and data in ASCII format. Writes out a
C binary database containing the data records (binary file will
C be accessed by get* subroutines.

C iGet_Lookup_col_num: This function returns the column number that
C corresponds to a column's header.

C fGet_Lookup_value_by_col_num: This function returns data
C from a lookup table that corresponds to a user-specified
C reference value, reference column number and value column number.

C fGet_Lookup_value_by_col_name: This function returns data
C from a lookup table that corresponds
C to a user-specified reference value, reference column name
C and value column name. This procedure is an interface to the
C function fGet_Lookup_value_by_col_num which allows the column names
```

```
C (versus column numbers) to identify the reference and value columns.

C cleanup_Lookup_data: Deletes the temporary file created by the
C lookup table facility.
```

MultiYear_climate.F

This file contains procedures for handling climate data during multi-year simulations.

```
C Parse_MY_clm_db: Parses the multi-year climate index file
C
C Check_MY_clm_db: Performs error checking on a multi-climate db.
C
C MY_clm_db_menu:  Displays a menu allowing configuration of
C                  the multi-year database
C
C Spec_MY_clm_db:  Asks a user for a multi-year database file
C
C Save_MY_clm_db:  Controls saving a multi-year database file
C
C Sort_MY_clm_db:  Sorts the records within a multi-year db.
C
C Fill_MY_clm_db:  Checks the records within a multi-year db
C                  for continuity, and fills in missing records
C
C Prune_MY_clm_db: Removes duplicate records from the database
C
C Write_MY_clm_db: Writes the database to disk
```

SiteUtilities.F

This file contains routines used to collate fuel used on site.

```
C SiteUtilities:           Supervisory routine

C TotalPurchasedEnergy:    Totals fuel consumed in the building

C ConvertEnergyToFuel:     Converts purchased energy into an
C                          equivlent amount of fuel use.

C H3KReportsFuelUse:       Transports data to H3KReports.
```

bsimsel.F

This source file provides an interface to the selection of basement types based on the BASECALC foundation loss model from Natural Resources Canada.

```
C bsimtype is based on data from BASESIMP, A Simplified Foundation
C Energy-Loss Model Derived from BASECALC (TM) Simultions
C Ian Beausoleil-Morrison, CANMET Energy Technology Centre,
C Natural Resources Canada, 1996, 1998 1999.
```

c2fdum.F

This file holds dummy routines for modules that have been compiled in pure-text mode.

```
C  netwdrw    dummy routine for esrunet/network.F

C  nwkslctc() dummy routine for esrunet/netwkc2f.F

C  gnwkquery() dummy routine for esrunet/netwkc2f.F

C  netwsnap() dummy routine for esrunet/netwkc2f.F

C  netwmic()  dummy routine for esrunet/network.F

C  gconad()   dummy routine for esrunet/netwkc2f.F
```

```
C  gridupdt() dummy routine for esrunet/netwkc2f.F
C  nwkupdtpos() dummy routine for esrunet/netwkc2f.F
C  icntfm()   dummy routine for esrunet/netutil.F
C  netwmco()  dummy routine for esrunet/network.F
```

cfdrw.F

Read and write new (version 2,2.1 and 2.2) cfd description.

```
C DFDSV:     Saves the room air movement description file (version 2).
C DFDREAD:   Reads the CFD configuration file (version 2,2.1 and 2.2)
C NEW2OLD:   Copies new data structure to old.
C CFMFSU:    Set up CFD to mass flow connections/ components.
C MFLISTemp: List common block contents for debugging only.
C CFDDEFLT: Sets default values for all data items.
C SCHCHK:   Checks the turbulent schmidt number of all containments.
C BCEKCK:   Checks the boundary conditions of all boundary cells
```

X.F xxx xxx xxx

```
XXX XXX XXX
```

Appendix A: Function API

The list below includes the current (revision 4055) of the API for each of the subroutines and function in the lib folder.

esru_ask.F

```
C cpwpk displays a copyright notice via help text data structure.
C It saves and recover any current contextual help messages.
C The integer parameter il is not used.
      subroutine cpwpk(il)


C The intent of askabout is to have one place for the opening help
C message of each application (and thus support an application independent
C static help menu under GTK).
C If initonly is 1 then only assign the h() array and do not display
C If initonly is 0 save prior h(), update h(), display and re-establish prior.
C module (char *4) identifies the ESP-r module.
      subroutine askabout(module,initonly)
```

esru_blk.F

```
C ezero loads the COMMONS used by libgtkesru.a, libxesru.a and libnoxesru.a.
      subroutine EZERO
```

esru_lib.F

The file esru_lib.f is a collection of low-level ESP-r fortran libraries that are applicable to all versions of ESP-r (that is, X11, GTK and noX). Code pertaining to version-specific configurations should be placed in the libGTK, libX11 and libNONGtk files.

```
C st2name takes ‘string’ swap blanks & wildcards to _ : return as ‘name’.
      SUBROUTINE st2name(string,name)
      CHARACTER*(*) string,name


C st2file: Given ‘string’ strip blanks & wildcards and return as ‘name’.
C Useful to check file names.
      SUBROUTINE st2file(string,name)
      CHARACTER*(*) string,name


C backslashit: Given ‘string’ swap / to  : return as ‘name’.
      SUBROUTINE backslashit(string,name)
      CHARACTER*(*) string,name


C prevblnk: given a string, return position of blank just before ipos.
      function iprevblnk(string,ipos)
      character*(*) string


C inextblnk: given a string, return position of blank just after ipos
C (or the end of the string if that happens first).
      function inextblnk(string,ipos)
      character*(*) string


C icutstr: given a string, cuts icut characters from position ipos
C and shifts latter portion of the string down icut chars. Returns
C zero if no error.
      function icutstr(string,ipos,icut)
      character*(*) string
```

```
C FPOPEN Opens the file with standard parameter conventions for
C IUN, ISTAT, MODE, and IXIST;  FNARG is the file name.
        SUBROUTINE FPOPEN(IUN,ISTAT,MODE,IXIST,FNARG)
        CHARACTER*(*) FNARG


C FPRAND Opens a random access file with name FNARG.
C  LENG determines the record length and type as follows:-
C       <0      -( LENG ) characters per record, ASCII records.
C       =0      128 words per record,   IMAGE   MODE RECORDS
C               (BUFFER COUNT IS FORCED TO 1 TO ENSURE WRITES
C               ARE PERFORMED IN ORDER REQUESTED)
C       >0      (LENG  ) words per record, ff BINARY   records
C IUN,ISTAT, and IXIST are the same as standard conventions.
        SUBROUTINE FPRAND(IUN,ISTAT,LENG,IXIST,FNARG)
        CHARACTER*(*) FNARG


C ERPFREE is used to close any file.
C IUN (integer) is the file unit number
C ISTAT (integer) is its status 0=OK, 1=if IUN was zero.
      SUBROUTINE ERPFREE(IUN,ISTAT)


C EFDELET: Delete file IUN and return ISTAT for compatibility.
C IUN (integer) is the file unit number
C ISTAT (integer) is its status 0=OK, 1=if IUN was zero.
      SUBROUTINE EFDELET(IUN,ISTAT)


C EASKAB Generic choice facility returning 1 or 2 according to which
C of the choices has been chosen.
C MSG1 and MSG2 are prompts (upto 124 char) to be printed.
C AOPT & BOPT are the text strings describing the
C options available, (these should not be too long).
      SUBROUTINE EASKAB(MSG1,MSG2,AOPT,BOPT,IWHICH,NHELP)
      CHARACTER*(*) MSG1,MSG2,AOPT,BOPT


C EASKABC Generic choice facility returning 1, 2 or 3 according to which
C of the choices has been chosen. See EASKATOG for parameter documentation.
      SUBROUTINE EASKABC(MSG1,MSG2,AOPT,BOPT,COPT,IWHICH,NHELP)
      CHARACTER*(*) MSG1,MSG2,AOPT,BOPT,COPT


C EASKABCD Generic choice facility returning 1, 2, 3 or 4 according to which
C of the choices has been chosen. See EASKATOG for parameter documentation.
      SUBROUTINE EASKABCD(MSG1,MSG2,AOPT,BOPT,COPT,DOPT,IWHICH,NHELP)
      CHARACTER*(*) MSG1,MSG2,AOPT,BOPT,COPT,DOPT


C bSPEqual determines if two SP numbers are within machine round-off error.
      logical function bSPEqual(fFloat1, fFloat2)
      real fFloat1, fFloat2


C DAYCLK will print the day, month, day no. and time based on the day
C of the year IYD (integer) and clock time in hours TIME (real) on the
C output channel ITU (integer).
      SUBROUTINE DAYCLK(IYD,TIME,ITU)


C FDROOT: Given a file name (fstring) see if it contains a (path) and a (filen).
C Note: mingw compilers require fstring to be declared length (not *(*).
      SUBROUTINE fdroot(fstring,path,filen)
      COMMON/OUTIN/IUOUT,IUIN
      CHARACTER*(*) path,filen,fstring


C EFOPSEQ: Open a sequential ASCII file (SFILE) and return appropriate
C messages and error status (IEXIST).
C IUN is the file unit number, SFILE is the file name.
```

```
C IEXIST is a flag denoting:
C   0 - should exist, no message but error -301 if not.
C   1 - should exist, message & error -301 if not.
C   2 - should not exist, error 300 & message if exists.
C   3 - if exists use otherwise create.
C   4 - if exists ask user before overwriting otherwise create.
C IER is the error status (0 is OK).
C If the path is other than ‘./’ but is not ‘/usr/esru’ then
C concatenate path with sfile.
      SUBROUTINE EFOPSEQ(IUN,SFILE,IEXIST,IER)
      CHARACTER*(*) SFILE


C EFOPRAN: Open a random access file (SFILE) and return appropriate
C messages and error status (IEXIST).
C IUN is the file unit number, SFILE is the file name, IRW is
C record width.  IEXIST is a flag denoting:
C   0 - should exist, no message but error -301 if not.
C   1 - should exist, message & error -301 if not.
C   2 - should not exist, error 300 & message if exists.
C   3 - if exists use otherwise create.
C   4 - if exists ask user before overwriting otherwise create.
C IRW is the record width to use when opening the file.
C IER is the error status (0 is OK).
C If the path is other than ‘./’ but is not ‘/usr/esru’ then
C concatenate path with sfile.
      SUBROUTINE EFOPRAN(IUN,SFILE,IRW,IEXIST,IER)
      CHARACTER*(*) SFILE


C FINDFIL: Check for existance of a file SFILE (with path) & return XST.
C If the path is other than ‘./’ but is not ‘/usr/esru’ then
C concatenate path with sfile before looking.  If file name is
C blank or UNKNOWN then return XST=.FALSE.
      SUBROUTINE FINDFIL(SFILE,XST)
      LOGICAL xst
      CHARACTER SFILE*72


C ADDPATH: Return file name appended onto the path and logical concat.
C IUOUT is the message channel, SFILE is the file name.
C If the path does not begin with ’/’ or ’?:’ then
C concatenate path with sfile.
C If path is ’  ’ or ’./’ do not concatenate
C This version includes cross-platform logic.
      SUBROUTINE addpath(SFILE,tfile,concat)
      LOGICAL concat
      CHARACTER*(*) sfile,tfile


C GETTOKENS checks a character string (A), returning the number of data
C items (IW) separated by a file separator and an array of tokens/words
C (up to 12 32character words returned) that were in the string.
C Useful for parsing file paths.  Note to keep from
C overwriting the string passed, deal with a copy.
      SUBROUTINE GETTOKENS(A,IW,WORDS)
      CHARACTER*(*) A
      CHARACTER*32 WORDS(12)


C c2fstr: converts c function returned string to fortran format.
C Strip off the last character (end of line mark) as well as any
C leading blanks from string returned.
      SUBROUTINE c2fstr(cstr,fstr)
      CHARACTER*(*) cstr,fstr


C termode: given an index ‘mode‘ return a terminal text string.
      SUBROUTINE termode(mode,tmode)
      CHARACTER*(*) tmode
```

```
C EKPAGE maps key characters, pages & array indexs in long display lists.
C No menu should have more than 26 data selections (plus heading and
C control lines) so that key characters (a-z) for arrays will not be
C repeated. Notes on variables:
C IPACT  : page option, if IPACT= CREATE then only update the common
C          block PMENU, otherwise ask for action to be taken.
      SUBROUTINE EKPAGE(IPACT)


C KEYIND decodes the index INO and returns the array index IA of the item
C displayed and whether INO is within the list.
C MENUL (integer) is the length of the menu.
C INO (integer) is the index passed from the calling code.
C IA (integer) is array index of the data
C INOUT=0 if outside the current menu, =1 if within the menu.
      SUBROUTINE KEYIND(MENUL,INO,IA,INOUT)


C EPMENSV pushes the current state of common block PMENU into common block
C PMENUSV.  Up to 5 levels are maintained.
C This is useful if a second menu is about to be called and
C knowledge of the first is required.  See also EPMENRC which pops the
C information back into common PMENU.
      SUBROUTINE EPMENSV


C EPMENRC recovers menu definitions (common block PMENU) from common
C block PMENUSV.
C This is useful if returning from a second menu.  See also EPMENSV.
      SUBROUTINE EPMENRC


C EPAGE: Screen control: page without waiting.
      SUBROUTINE EPAGE

C EPAGEW: Screen control: Wait before paging.
      SUBROUTINE EPAGEW

C EPWAIT: Screen control: Wait without paging.
      SUBROUTINE EPWAIT

C EPAGEND: Screen control: Page then close window if open.
      SUBROUTINE EPAGEND


C ang3vtx: recover angle between three vertex (i.e. between two lines).
C Angel between lines: finds angle A between two lines'
C given 3 vertex as in:    * 1
C                           a C                    3 *------* 2
      SUBROUTINE ang3vtx(x1,y1,z1,x2,y2,z2,x3,y3,z3,ang)


C CLOSE3D - calc min dist between two lines in 3D return dist and
C closest points.
C PA,VA define line A (point and vector), SA is parametric variable at closest point
C PB,VB define line B (point and vector), SB is parametric variable at closest point
C DIST minimum distance between lines - if negative then error
C CA,CB are closest points on two lines (if intersect DIST=0 and CA=CB)
C Theory:
C  At intersection the values of the parametric variables will
C  produce the same x,y,z values.  Two equations are formed and
C  solved (these are easily derived from the parametric form of
C  the line equations but the variable names are used below).
C    SA.VA.VB - SB.VBsq = RHSa
C    SA.VAsq - SB.VA.VB = RHSb
      SUBROUTINE CLOSE3D(PA,VA,SA,PB,VB,SB,DIST,CA,CB)
      DIMENSION PA(3),VA(3),PB(3),VB(3),CA(3),CB(3)


C CROW: Function returning the distance 'as the crow flies'
C between the two points P and Q in X Y Z space.
      FUNCTION CROW(P,Q)
```

```
      DIMENSION  P(3), Q(3)


C CROWXYZ: Function returning the distance 'as the crow flies'
C between the two X Y Z points.
      FUNCTION CROWXYZ(PX,PY,PZ,QX,QY,QZ)


C UVXYZ: Subroutine returning Unit vector from two points along a
C line expressed as X Y Z points.
      subroutine UVXYZ(PX,PY,PZ,QX,QY,QZ,UX,UY,UZ)


C UVAB: Subroutine returning Unit normal vector U of vector A.
C ierr (integer) = -1 if vector is very short.
      subroutine UVAB(A,U,ierr)
      dimension A(3),U(3)


C PLNDANG: Finds dihedral angel between two planes given their equations.
      subroutine plndang(aeqn,beqn,dang)
      dimension aeqn(4),beqn(4)


C PLNDIS finds distance DIST from a point (x,y,z) to a plane (eq EQN).
C If DIST=0 then one the plane, if DIST > 0 then point on the side
C of the normal vector, if DIST < 0 then on the opposite side.
      subroutine PLNDIS(EQN,X,Y,Z,DIST)


C AVER returns the centre of gravity CG for an array
C of vertex points PNT with NP points in it.
      SUBROUTINE AVER(MNV,NP,PNT,CG)
      DIMENSION PNT(MNV,3),CG(3)


C CROSS performs a cross-product of two vectors A(X,Y,Z) and B(X,Y,Z),
C and returns the result in C(X,Y,Z).
      SUBROUTINE CROSS(A,B,C)
      DIMENSION A(3),B(3),C(3)


C CROSS2 performs a cross-product of two vectors AX,AY,AZ and BX,BY,BZ,
C and returns the result in CX,CY,CZ.
      subroutine cross2(ax,ay,az, bx,by,bz, cx,cy,cz)


C DOT3: Return dot product of two vectors a & b.
      subroutine dot3(a,b,product)
      dimension a(3),b(3)


C ZEROS Clear a 4x4 array prior to doing viewing transforms.
      SUBROUTINE ZEROS(A)
      DIMENSION A(4,4)


C ECLOSE allows two real numbers R1 & R2 to be checked for closeness
C to a given tolerance TOL and returns CLOSE = .TRUE. or .FALSE.
      SUBROUTINE ECLOSE(R1,R2,TOL,CLOSE)
      LOGICAL CLOSE


C ESIND: Returns SIN of angle where angle is given in degrees.
      FUNCTION ESIND (DEG)


C ECOSD: Returns COS of angel where angle is given in degrees.
      FUNCTION ECOSD (DEG)


C ETAND: Returns TAN of angel where angle is given in degrees.
```

```
      FUNCTION ETAND (DEG)


C IFAX is an integer function returning the integer part of it's
C argument truncated towards negative infinity as follows:
C     V        IFAX(V)
C    1.0        1
C     .9        0
C     .1        0
C      0        0
C    -.1       -1
C    -.9       -1
C   -1         -1
C   -1.1       -2
      FUNCTION IFAX(V)


C EAZALT computes the solar azimuth and altitude angles at the current
C time (REAL). The 'ISUNUP' variable determines whether the sun
C is up (=1) or down (=0). The solar angles are computed relative
C to local mean time (e.g. Greenwich is the reference time zone for
C the UK).
      SUBROUTINE EAZALTS(TIMEH,ID,SLAT,SLON,ISUNUP,SAZI1,SALT1)


C AGNXYZ: Given the solar azimuth and elevation return viewing coords
C         at a distance of 1000 metres from origin.
      SUBROUTINE ANGXYZ(YAZI,SALT,X,Y,Z)


C ORTTRN multiplies a point (XM,YM,ZM) by the transform matrix
C TMAT to return the point XO,YO,Z0.  If the points are to close
C together then IER=-1.
      SUBROUTINE ORTTRN(XM,YM,ZM,TMAT,XO,YO,ZO,IERR)
      DIMENSION TMAT(4,4)


C VECTRN transforms a vector VECIN by the 4x4 (homogeneous) matrix TMAT
C and returns the vector VECOUT.  If IERR is < 0 then a fatal error.
      SUBROUTINE VECTRN(VECIN,TMAT,VECOUT,IERR)
      DIMENSION VECIN(3),VECOUT(3),TMAT(4,4)


C VECPLN returns the point of intersection X,Y,Z between a line defined
C by X1,Y1,Z1 & X2,Y2,Z2 and a plane defined in PEQN.
C IERR is -1 if denominator is close to zero or negative.
C IERR is 0 if no problem found.
      SUBROUTINE  VECPLN(X1,Y1,Z1,X2,Y2,Z2,PEQN,X,Y,Z,IERR)
      DIMENSION  PEQN(4)


C HMATMUL multiplies the homogenous (4x4) matrices A by B returning C.
C Matrix A is premultiplied and matrix B is postmultiplied.
      SUBROUTINE HMATMUL(A,B,C)
      DIMENSION A(4,4),B(4,4),C(4,4)


C HREVMAT takes the homogenous perspective transformation PER and
C returns it's inverse REP making use of CROUT.
      SUBROUTINE HREVMAT(PER,REP,IERR)
      DIMENSION PER(4,4),REP(4,4),A(4,4),WS(4)


C ****************** CROUT
C CROUT inverts a nonsymetric square matrix A (order N), returning
C the matrix B and IERR =-1 if matrix is singular.
C WS is a workspace vector of dimension N, IA is the
C first dimension of A and IB is the second dimension.
C based on crout factorization based on code from
C Alan Bridges and Wilkinson and Reinsch 'Linear Algebra'
C Springer Verlag, 1971, PP 93-110
      SUBROUTINE CROUT(A,N,WS,B,IA,IB,IER)
```

```
      DIMENSION A(IA,N),B(IB,N),WS(N)


C DPACC provides double precision accumulation of inner products for
C CROUT in the form SUM(+,-)SUM(+,-)AB.
C A is the vector on left, I & J numerical identifiers of first two
C elements of A in the multiplication and IJ is the dimension of A.
C For vector B the parameters K,L,KL are analogous to I,J,IJ.
C X is the quantity to be added to the product of the specified
C elements of vectors A & B.
C SUM is the result, N is a counter, IND is an indicator as follows:
C   IF IND=1 SUM=AB+X
C   IF IND=2 SUM=X-AB
C   IF IND=3 SUM=AB-X
C   IF IND=4 SUM=-AB-X = -(AB+X)
      SUBROUTINE DPACC(A,I,J,IJ,B,K,L,KL,X,SUM,N,IND)
      DIMENSION A(IJ),B(KL)


C EYEMAT provides transform eye-point - viewpoint....
      SUBROUTINE EYEMAT(EP,VP,SCALE,TMAT,RMAT)
      DIMENSION  EP(3),VP(3),TMAT(4,4),RMAT(4,4)


C INTSTR converts an integer into a string (10 char long) with no
C leading blanks. ISWD is the length of the resulting string.
      SUBROUTINE INTSTR(INTIN,FSTR,ISWD,IFLAG)
      CHARACTER*10 CSTR, FSTR


C RELSTR converts a real into a string (12 char) with no leading
C blanks. ISWD is the actual length of the resulting string. Takes
C the magnitude of the number into account.
      SUBROUTINE RELSTR(RELIN,FSTR,ISWD,IFLAG)
      CHARACTER*12 CSTR, FSTR


C REL16STR converts a real into a string (16 char) with no leading
C blanks. ISWD is the actual length of the resulting string. Takes
C the magnitude of the number into account.
      SUBROUTINE REL16STR(RELIN,FSTR,ISWD,IFLAG)
      CHARACTER*16 CSTR, FSTR


C EXPSTR converts a exponential into a string (10 char) with no leading
C blanks. ISWD is the actual length of the resulting string.
      SUBROUTINE EXPSTR(RELIN,FSTR,ISWD,IFLAG)
      CHARACTER*10 CSTR, FSTR


C ARLIST takes the first (inst to inrl) items of a real array (rlist)
C of array size (inrs) and builds a packed string (pckstr)
C and returns pckstr and the actual character width (length). If itrunc
C is zero then all items written, else the index of the last item
C which was written. Delm is delimeter between items.
      SUBROUTINE ARLIST(inst,inrl,rlist,inrs,delm,pckstr,length,itrunc)
      dimension rlist(inrs)
      character*(*) pckstr


C AILIST takes the range (inst to inil) items of an integer array (ilist)
C of array size (inisz) and builds a packed string (pckstr) and
C returns pckstr and written character width (length). If itrunc
C is zero then all items written, else the index of the last item
C which was written. Delm is delimeter between items.
      SUBROUTINE AILIST(inst,inil,ilist,inisz,delm,pckstr,length,itrunc)
      dimension ilist(inisz)
      character*(*) pckstr


C ASLIST takes the range (inst to inil) items of an string array (list)
C of array size (inisz) and builds a packed string (pckstr) and
```

```
C returns pckstr and written character width (length). If itrunc
C is zero then all items written, else the index of the last item
C which was written. Delm is delimeter between items. It is assumed
C that each string array item is less than 32 characters wide.
      SUBROUTINE ASLIST(inst,inil,list,inisz,delm,pckstr,length,itrunc)
      dimension list(inisz)
      character*(*) pckstr,list
      CHARACTER delm*1


C ASLIST2 takes the range (inst to inil) items of an string array (list)
C of 2_dimensional array size (inisz,inisz2) and builds a packed string
C (pckstr) of the row (inirw) and
C returns pckstr and written character width (length). If itrunc
C is zero then all items written, else the index of the last item
C which was written. Delm is delimeter between items. It is assumed
C that each string array item is less than 24 characters wide.
      SUBROUTINE ASLIST2(inst,inil,list,inisz,inisz2,inirw,
     &                   delm,pckstr,length,itrunc)
      dimension list(inisz,inisz2)
      character*(*) pckstr,list
      CHARACTER delm*1


C ASFLIST takes the range (inst to inil) items of an string array (list)
C of array size (inisz) and builds a packed string (pckstr) and
C returns pckstr and written character width (length). If itrunc
C is zero then all items written, else the index of the last item
C which was written. Delm is delimeter between items. It is assumed
C that each string array item is less than 48 characters wide.
      SUBROUTINE ASFLIST(inst,inil,list,inisz,delm,pckstr,length,itrunc)
      dimension list(inisz)
      character*(*) pckstr,list
      CHARACTER delm*1


C STRIPC strips comments from a ASCII file string and returns the data.
C It assumes that if a string begins with a '#' then the whole line is
C a comment an the next line is read.  If a ' #' is discovered within
C a line the rest of the line is removed.
C IER=0 if ok. MSG is a text string used in error messages. If
C IR=0 then acts silently, otherwise notes when EOF found.
C IEXP is the number of expected items in the line:
C   IEXP = 0 means don't care or already know no. items - don't check
C   IEXP >0  means a specific number of items expected (error if not)
C   IEXP = 99 check number of items and return in ITEMS
      SUBROUTINE STRIPC(INPCH,OUTSTR,IEXP,ITEMS,IR,MSG,IER)
      CHARACTER*124 OUTSTR
      CHARACTER*(*) MSG


C LSTRIPC strips comments from a ASCII file (long) string and returns the data.
C It assumes that if a string begins with a '#' then the whole line is
C a comment an the next line is read.  If a ' #' is discovered within
C a line the rest of the line is removed.
C IER=0 if ok. MSG is a text string used in error messages. If
C IR=0 then acts silently, otherwise notes when EOF found.
C IEXP is the number of expected items in the line:
C   IEXP = 0 means don't care or already know no. items - don't check
C   IEXP >0  means a specific number of items expected (error if not)
C   IEXP = 99 check number of items and return in ITEMS
      SUBROUTINE LSTRIPC(INPCH,OUTSTR,IEXP,ITEMS,IR,MSG,IER)
      CHARACTER*248 OUTSTR
      CHARACTER*(*) MSG


C STRIPC400 strips comments from a ASCII file (400 char long) string and returns the data.
C It assumes that if a string begins with a '#' then the whole line is
C a comment an the next line is read.  If a ' #' is discovered within
C a line the rest of the line is removed.
C IER=0 if ok. MSG is a text string used in error messages. If
C IR=0 then acts silently, otherwise notes when EOF found.
```

```
C IEXP is the number of expected items in the line:
C   IEXP = 0 means don't care or already know no. items - don't check
C   IEXP >0  means a specific number of items expected (error if not)
C   IEXP = 99 check number of items and return in ITEMS
      SUBROUTINE STRIPC400(INPCH,OUTSTR,IEXP,ITEMS,IR,MSG,IER)
      CHARACTER*400 OUTSTR
      CHARACTER*(*) MSG


C CHITMS checks a character string (A), returning the number of data
C items (IW) separated by ' ' tab or ','.  Note to keep from
C overwriting the string passed, deal with a copy.
      SUBROUTINE CHITMS(A,IW)
      CHARACTER*(*) A


C LCHITMS checks a long character string (A), returning the number of data
C items (IW) separated by ' ' tab or ',' or '|'.  Note to keep from
C overwriting the string passed, deal with a copy.
      SUBROUTINE LCHITMS(A,IW)
      CHARACTER*(*) A


C CHITMS400 checks a 400 long char string (A), returning the number of data
C items (IW) separated by ' ' tab or ',' or '|'.  Note to keep from
C overwriting the string passed, deal with a copy.
      SUBROUTINE CHITMS400(A,IW)
      CHARACTER*(*) A


C CHARCH: Older routine to check a string for a number of data items.
C This is an older version of CHITMS, however it modifies the string A.
      SUBROUTINE CHARCH(A,ND,IERR)
      CHARACTER*72 A


C NOYES is an INTEGER FUNCTION which is used to read the answer to a
C question.  A line is read from the user, and the value of the
C function is as follows:-
C   1 The answer was '1', ' Y', or '  YES'
C   0 The answer was '0', ' N', or '  NO'
C  -1 otherwise.
      FUNCTION NOYES(J)


C IFIRST: Returns the ASCII value for the first character in
C a string ISTR.
      FUNCTION IFIRST(ISTR)
      CHARACTER*(*) ISTR


C EDAY Returns year day number IYDN when passed the day of the month
C IDAYN and the month number IMTHN. 1st Jan= 1, 31st Dec=365, no leap
C years considered.
      SUBROUTINE EDAY(IDAYN,IMTHN,IYDN)


C 'EDAYR' returns the day and month numbers from the day-of-year where:
C day-of-year 1 = 1st January and day-of-year 365 = 31st December.
C NO LEAP YEARS ARE CONSIDERED!
      SUBROUTINE EDAYR(IYDN,IDAYN,IMTHN)


C 'EWEEKD' returns the day of the week (IDAY) given the day of month, month
C and year (ND,NM,NY) as integers.
C MON=1, TUE=2, WED=3, THU=4, FRI=5, SAT=6 AND SUN=7.
      SUBROUTINE EWEEKD(ND,NM,NY,IDAY)


C EDAYCH will check for errors in the users specification of the day and
C month under consideration.
C IERR set to 1 if ID or IM are outwith the allowable range.
```

```
      SUBROUTINE EDAYCH(ID,IM,IERR)


C DATTIM returns UNIX time via a string in the form : 16 Sep 73 14:23.
      SUBROUTINE DATTIM(DT)
      CHARACTER*(*) DT


C STDATE takes the day of year and returns two descriptive strings.
C DESCR takes the form '12-Jan' and DESCR1 takes the form 'Fri-12-Jan'.
      SUBROUTINE STDATE(IYEAR,IDOY,DESCR,DESCR1)
      CHARACTER DESCR*7,DESCR1*10


C ESTIME takes an integer timestep and returns two string descriptions:
C DESCRH in the form '12h28' and DESCRD which takes the form of 12.46,
C As well as the time as a fraction of a day TIMER.
C STIME takes timestep averaging IDAVER into account ie. 0=averaging,
C 1=no averaging. NTS is the number of timesteps per hour.
      SUBROUTINE ESTIME(NTS,IDAVER,ITIME,DESCRH,DESCRD,TIMER)
      CHARACTER*5 DESCRH, DESCRD


C EDTIME takes an real time and returns two string descriptions:
C DESCRH in the form '12h28' and DESCRD which takes the form of 12.46,
C As well as the time as a fraction of a day TIMER.
      SUBROUTINE EDTIME(TIME,DESCRH,DESCRD,TIMER)
      CHARACTER*5 DESCRH, DESCRD


C EPERSTR creates three strings representing the start and stop
C time of a diary period based on the preferred time & date
C display format.
C IFDAY  0 gives 'DOY 10', 1 gives '10 Jan', 2 gives 'Fri 10 Jan'
C IFTIME 0 gives '10h30', 1 gives '10.50', 2 gives '0.4375'
C PERST1 (14 char) is:' 10h00  15h30 ',' 10.00  15.50 ',' 0.4375 0.6458'
C PERST3 (44 char):
C if IFDAY=0 then it is: 'period: DOY 100 to DOY 112, 1990'
C if IFDAY=1 then it is: 'period: 10 Jan to 31 Jan, 1990'
C if IFDAY=2 then it is: 'period: Mon 10 Jan to Mon 17 Jan, 1990'
C PERST2 (44 char) includes the time of day but not the year.
C IER=0 OK, IER=1 problem.
C BTIM and PETIM are in terms of decimal fractions of a day.
      SUBROUTINE EPERSTR(IYEAR,IBDOY,IBTIM,IEDOY,IETIM,NTSPH,
     &           IFDAY,IFTIME,PERST1,PERST2,PERST3,IER)
      CHARACTER PERST1*14,PERST2*44,PERST3*44


C EGETW gets first WORD after position K from the STRING of
C characters. Words are separated by blanks, commas, |, or tab: WORD,WORD,WORD
C or WORD WORD WORD or WORD, WORD, WORD  are all valid.  Provides a warning
C message if ACT='W', a failure message if ACT='F' and does
C no message if ACT='-'.  Modified after:
C G.N. Walton, US Nat. Institute of Standards and Technology
C     LS     - maximum length of STRING
C     L      - current position in WORD
C     LW     - maximum length of WORD
      SUBROUTINE EGETW(STRING,K,WORD,ACT,MSG,IER)
      CHARACTER*(*) WORD, STRING, MSG
      CHARACTER ACT*1


C EGETP gets first PHRASE after position K from the STRING of
C characters. Phrases are separated by tabs or commas. Provides a warning
C message if ACT='W', a failure message if ACT='F' and does
C no message if ACT='-'.
      SUBROUTINE EGETP(STRING,K,PHRASE,ACT,MSG,ier)
      CHARACTER*(*) PHRASE, STRING, MSG
      CHARACTER ACT*1


C EGETDQ gets first quoted PHRASE after position K from the STRING of
```

```
C characters. Phrases can contain spaces and commas but not tabs.
C Provides a warning message if ACT='W', a failure message if ACT='F' and
C no message if ACT='-'.  The string returned has the quotes removed.
C Note it uses an internal string buffer which assumes that the
C phrase is less than 248 characters long.
      SUBROUTINE EGETDQ(STRING,K,PHRASE,ACT,MSG,ier)
      CHARACTER*(*) PHRASE, STRING, MSG
      CHARACTER ACT*1


C EGETWI gets first word after position K from the STRING of
C characters and converts it into an integer IV, tests it against
C the minimum MN and the maximum MX and provides a warning
C message if ACT='W', a failure message if ACT='F' and does
C no range checking if ACT='-'. Words may be separated by blanks,
C commas, or tab: WORD,WORD,WORD or WORD WORD WORD or WORD, WORD, WORD
C are all valid.
      SUBROUTINE EGETWI(STRING,K,IV,MN,MX,ACT,MSG,IER)
      CHARACTER*(*) STRING, MSG
      CHARACTER ACT*1


C EGETWR gets first word after position K from the STRING of
C characters and converts it into a real number RV, tests it against
C the minimum RMN and the maximum RMX and provides a warning
C message if RACT='W', a failure message if RACT='F' and does
C no range checking if RACT='-'. Words may be separated by blanks,
C commas, or tab: WORD,WORD,WORD or WORD WORD WORD or WORD, WORD, WORD
C are all valid.
      SUBROUTINE EGETWR(STRING,K,RV,RMN,RMX,RACT,MSG,IER)
      CHARACTER*(*) STRING, MSG
      CHARACTER RACT*1


C EGETWRA recovers (IRVA) reals of real array (RVA) from an ASCII file
C (unit IAF) reading as many lines as necessary to recover the data.
C IER=2 if EOF is reached before INUM items have been parsed. IER=1 if
C there was a problem reading it. Each value is tested against
C the minimum RMN and the maximum RMX and provides a warning
C message if RACT='W', a failure message if RACT='F' and does
C no range checking if RACT='-'. Words may be separated by blanks,
C commas, or tab: WORD,WORD,WORD or WORD WORD WORD or WORD, WORD, WORD
C are all valid.
C Note: all reads are able to accept commented files.
      SUBROUTINE EGETWRA(IAF,RVA,IRVA,RMN,RMX,RACT,MSG,IER)
      DIMENSION RVA(*)
      CHARACTER*(*) MSG
      CHARACTER RACT*1


C EGETWIA recovers (IRVA) integers of integer array (IVA) from an
C ASCII file (unit IAF) reading as many lines as necessary to
C recover the data.
C IER=2 if EOF is reached before IRVA items have been parsed. IER=1 if
C there was a problem reading it. Each value is tested against
C the minimum IRMN and the maximum IRMX and provides a warning
C message if IACT='W', a failure message if IACT='F' and does
C no range checking if IACT='-'. Words may be separated by blanks,
C commas, or tab: WORD,WORD,WORD or WORD WORD WORD or WORD, WORD, WORD
C are all valid.
C Note: all reads are able to accept commented files.
      SUBROUTINE EGETWIA(IAF,IVA,IRVA,IRMN,IRMX,IACT,MSG,IER)
      DIMENSION IVA(*)
      CHARACTER*(*) MSG
      CHARACTER IACT*1


C EGETAGWIA recovers (IRVA) integers of integer array (IVA) from a string
C TSTR (from position K) and if TSTR does not hold all of the array then
C it continues reading from an ASCII file (unit IAF) reading as
C many lines as necessary to recover the data.
C IER=2 if EOF is reached before IRVA items have been parsed. IER=1 if
```

```
C there was a problem reading it. Each value is tested against
C the minimum IRMN and the maximum IRMX and provides a warning
C message if IACT='W', a failure message if IACT='F' and does
C no range checking if IACT='-'. Words may be separated by blanks,
C commas, or tab: WORD,WORD,WORD or WORD WORD WORD or WORD, WORD, WORD
C are all valid.
C Note: all reads are able to accept commented files.
      SUBROUTINE EGETAGWIA(TSTR,K,IAF,IRVA,IVA,IRMN,IRMX,IACT,MSG,IER)
      DIMENSION IVA(*)
      CHARACTER*(*) TSTR,MSG
      CHARACTER IACT*1


C EGETAGWRA recovers (IRVA) reals of real array (RVA) from a string
C TSTR (from position K) and if TSTR does not hold all of the array then
C it continues reading from an ASCII file (unit IAF) reading as
C many lines as necessary to recover the data.
C IER=2 if EOF is reached before IRVA items have been parsed. IER=1 if
C there was a problem reading it. Each value is tested against
C the minimum RMN and the maximum RMX and provides a warning
C message if IACT='W', a failure message if IACT='F' and does
C no range checking if RACT='-'. Words may be separated by blanks,
C commas, or tab: WORD,WORD,WORD or WORD WORD WORD or WORD, WORD, WORD
C are all valid.
C Note: all reads are able to accept commented files.
      SUBROUTINE EGETAGWRA(TSTR,K,IAF,IRVA,RVA,RMN,RMX,RACT,MSG,IER)
      DIMENSION RVA(*)
      CHARACTER*(*) TSTR,MSG
      CHARACTER RACT*1


C EGETAGWSA recovers (ISVA) words into string array (SVA) from a string
C TSTR (from position K) and if TSTR does not hold all of the array then
C it continues reading from an ASCII file (unit IAF) reading as
C many lines as necessary to recover the data

C The string array SVN is an array of inisz words assumed to be less than
C or equal to *32 characters (that is the size of the buffer). This
C will work with space or comma or tab separations.

C IER=2 if EOF is reached before ISVA items have been parsed. IER=1 if
C there was a problem reading it. Each value is tested against
C a blank stringt and provides a warning message if SACT='W', a failure
C message if SACT='F' and does no blank checking if SACT='-'.
C Phrases may be separated by spaces commas, or tab: word,word,word or
C word, word, word, or word<tab>word<tab>word are all valid.
C Note: all reads are able to accept commented files.
      SUBROUTINE EGETAGWSA(TSTR,K,IAF,ISVA,SVA,inisz,SACT,MSG,IER)
      dimension SVA(inisz)
      CHARACTER*(*) SVA
      CHARACTER*(*) TSTR,MSG
      CHARACTER SACT*1


C EGETAGWPA recovers (ISVA) phrases into string array (SVA) from a string
C TSTR (from position K) and if TSTR does not hold all of the array then
C it continues reading from an ASCII file (unit IAF) reading as
C many lines as necessary to recover the data

C The string array SVN is an array of inisz items assumed to be less than
C or equal to *32 characters (that is the size of the buffer). This is
C flexible enough to work with phrases (and so it requires comma or
C tab separation in the file).

C IER=2 if EOF is reached before ISVA items have been parsed. IER=1 if
C there was a problem reading it. Each value is tested against
C a blank stringt and provides a warning message if SACT='W', a failure
C message if SACT='F' and does no blank checking if SACT='-'.
C Phrases may be separated by commas, or tab: phrase,phrase,phrase or
C phrase, phrase, phrase, or phrase<tab>phrase<tab>phrase are all valid.
C Note: all reads are able to accept commented files.
      SUBROUTINE EGETAGWPA(TSTR,K,IAF,ISVA,SVA,inisz,SACT,MSG,IER)
```

```
      dimension SVA(inisz)
      CHARACTER*(*) SVA
      CHARACTER*(*) TSTR,MSG
      CHARACTER SACT*1


C EGETRM returns the remainder of a text string TSTR after position k
C in RSTR where RSTR has no leading blanks.
      SUBROUTINE EGETRM(TSTR,K,RSTR,ACT,MSG,IER)
      CHARACTER*(*) TSTR,RSTR,MSG
      character ACT*1


C Find if module is being used as a dll (silent running).
      subroutine isadll(yes)
      logical yes


C Find if machine is unix.
C NOTE: uses compiler variable -DMINGW to signal .false.
      subroutine isunix(yes)
      logical yes


C usrhome: Find users home directory.
      subroutine usrhome(upath)
      COMMON/OUTIN/IUOUT,IUIN
      character*(*) upath


C usrname: Find users name.
      subroutine usrname(uname)
      character*(*) uname


C usrdir: Find current folder.
      subroutine usrdir(upwd)
      character*(*) upwd


C esppid: Find current process number.
      subroutine esppid(ipid)
      integer getpid


C tstamp: generates a time-stamped message.
C act (char*1) = '-' Current time, = 'm' Message plus time
C  = '>' append message to journal
      subroutine tstamp(act,msg)
      character*(*) msg
      character act*1


C ectime function as fixed length character*24 based on the
C the passed value of ictime
      character*24 FUNCTION ECTIME(ICTIME)
      integer ICTIME


C dstamp: Get date stamp in the form: Fri Jan 23 09:34:31 1998.
C Used to isolate code from system details.
      subroutine dstamp(date_str)
      character date_str*24


C comparedate is passed two date strings (generated by call to dstamp).
C act is requested action '?' is ??
C dif is positive if datea is more current than dateb.
      subroutine comparedate(datea,dateb,act,idif)
      character datea*24,dateb*24,act*1
```

```
C getsecs: Used to isolate fortran code from system specifics.
C Edit for machine type.
      subroutine getsecs(ictime)


C runit: Execute a command string, in text (runs in a new xterm) or graphics
C mode.  Assumes that command string terminates in a '&' if the
C user wishes to run in background mode. For minGW xterm is not used.
      subroutine runit(cmd,tg)
      character*(*) cmd,tg


C iEGetArrW reads all of the words in a 248 character string, and populates a character
C array with each word. Words are separated by blanks, commas, or tab: WORD,WORD,WORD
C or WORD WORD WORD or WORD, WORD, WORD  are all valid.
      integer function iEGetArrW(cString,cWORDS)
      character*248 cString, cWords(124)


C sitell2s takes latitude (clat) and longitude difference (clong) and returns
C descriptive string (descr).
      subroutine sitell2s(clat,clong,descr)
      character descr*16


C SIGFIG returns number to required significant figure level.
C R - real number
C NSIG - number of significant figures required.
C RNO - real number to NSIG figures
C STR - string version of RNO
C LSTR - length of STR
       subroutine sigfig(R,NSIG,RNO,STR,LSTR)
       character*12 STR


C SIpre returns suitable SI prefix for number supplied.  NOTE that
C   it is assumed that the value is supplied in the base SI unit, for
C   example supply a flux in W not kW.
C R - real number
C NSIG - number of significant figures required.
C RNO - real number to NSIG figures, including prefix
C STR - string version of RNO
C LSTR - length of STR
C PRE - prefix
C SYM - symbol
       subroutine SIpre(R,NSIG,RNO,STR,LSTR,PRE,SYM)
       character*12 STR
       character SYM*1
       character PRE*5


C pronam returns the characters of a string after the last
C occurrence of '/' or '´
      SUBROUTINE PRONAM(longstr,last)
      CHARACTER*(*) LONGSTR,LAST


C DNOTZERO returns a non zero value with the same sign.
      DOUBLE PRECISION FUNCTION DNOTZERO(A)
      double precision A


C ANOTZERO returns a non zero value with the same sign.
      REAL FUNCTION ANOTZERO(A)


C ASKTIM is a standard call to enquire which month and day and time
C (point in time for snapshot analysis. Returns IMO (month), IDO (day
C of month), IJDAY (day of year for output), TIME (real representation),
C IT (timestep). IFDAY is a toggle provided in setres.f to control
C the display and input of periods - 0 = julian day, 1 or 2 = day of
C month.
```

```
      SUBROUTINE ASKTIM(IFDAY,NTS,IMO,IDO,IJDAY,TIME,IT,IER)


C LISTAS: General read of an ascii file.
      SUBROUTINE LISTAS(iunit,LFIL,IER)
      character LFIL*72


C SHOWMEH extracts additional help text matching the key OTHER from
C an external file (LTUT) and returns it in exh along with the
C number of lines (nhi) and their maximum width (ihw).
      SUBROUTINE SHOWMEH(IFTUT,LTUT,OTHER,nhi,ihw,exh,IER)
      dimension exh(60)
      CHARACTER*(*) OTHER
      CHARACTER exh*72,ltut*72


C SDELIM replaces blanks in a string A with alternative delimiter and
C returns in B. If the last character in the string is the alternative
C delimiter then replace it with a blank.
      SUBROUTINE SDELIM(A,B,delm,IW)
      CHARACTER*(*) A,B
      CHARACTER delm*1


C EDDISP is a generic routine which displays lines of text passed to it
C in a format depending on the terminal type and the currently set
C delimiter:
      SUBROUTINE EDDISP(ITRU,MSG)
      CHARACTER*(*) MSG


C EDISP248 displays a 248 char block of text passed to it
C in a format depending on the terminal type. If it will not
C fit on one line, subsequent lines are used and breaks are
C set based on nearest width to iwid.
      SUBROUTINE EDISP248(ITRU,MSG,iwid)
      CHARACTER*(*) MSG


C clrtextbuf clears the text buffer common blocks.
      subroutine clrtextbuf()


C UPDVIEW is called from C code in esru_x.c or esp-r.c with values to
C update common blocks GFONT, VIEWPX and SPAD. Set REFRESH=true.
C Set MODIFY=true to force any wire-frame images to be redrawn.
      subroutine updview(ifsc,itfsc,imfsc,ilc,irc,itc,ibc,iwc,ihc,lttyc)
      integer ifs,itfs,imfs


C WIREPK is called from C code in esru_x.c or esp-r.c  with current
C number of zones which have been selected for display.
C Compilation of X11 or GTK version assumes this will be available in
C each modules code somewhere.  If not needed then provide a dummy.
      subroutine wirepk(inpk)


C EPROMPT: Does nothing, for compatibility only.
      SUBROUTINE EPROMPT
```

Appendix B: Function API for esru_libXXX.F

The list below includes the current (revision 4055) of the API for each of the subroutines and function in esru_libGTK.F. If these differ with the API in esru_libNonGTK.F this is noted.

esru_libGTK.F

```
C EASKI is a facility for asking the user for an integer which
C incorporates the prompt, error messages and range checking
C as passed parameters as follows:

C IVAL is the integer returned, PROMP1 & PROMP2 are the prompts to be
C given to the user (similar to the syntax of USRMSG which allows
C two lines or a leading blank line).
C MIN is the minimum value, MAX is the maximum value.
C MINACT and MAXACT are actions to take if the range is exceeded:
C   'W' means warn user but accept, 'F' means refuse/fail to
C   accept value and ask again, '-' means no range checking.
C IDEFLT is the default value if 'D','d' typed.
C If a space or carriage return is typed then IVAL is not changed.
C ERMSG is a string appended to the range checking or read error
C to identify the value.
C IER is the error state, if 0 then OK, if -3 then cancel pressed.
C NHELP is the number of help lines and H() is the array
C of text strings to be printed out if '?','H','h' is typed
C by the user.
      SUBROUTINE EASKI(IVAL,PROMP1,PROMP2,MINV,MINACT,
     &                 MAXV,MAXACT,IDEFLT,ERMSG,IER,NHELP)
      CHARACTER*(*) PROMP1,PROMP2,ERMSG
      CHARACTER MINACT*1,MAXACT*1


C EASKR is a facility for asking the user for an real which
C incorporates the prompt, error messages and range checking
C as passed parameters as follows:

C RVAL is the real returned, PROMP1 & PROMP2 are the prompts to be
C given to the user (similar to the syntax of USRMSG which allows
C two lines or a leading blank line).
C RMIN is the minimum value, RMAX is the maximum value.
C MINACT and MAXACT are actions to take if the range is exceeded:
C   'W' means warn user but accept, 'F' means refuse/fail to
C   accept value and ask again, '-' means no range checking.
C DEFLT is the default value if 'D','d', or return typed.
C ERMSG is a string appended to the range checking or read error
C to identify the value.
C IER is the error state, if 0 then OK, if -3 then cancel button pressed.
C NHELP is the number of help lines and H() is the array
C of text strings to be printed out if '?','H','h' is typed
C by the user.
      SUBROUTINE EASKR(RVAL,PROMP1,PROMP2,RMIN,MINACT,
     &                 RMAX,MAXACT,DEFLT,ERMSG,IER,NHELP)
      CHARACTER*(*) PROMP1,PROMP2,ERMSG
      CHARACTER MINACT*1,MAXACT*1


C EASKE is a facility for asking the user for an real number
C in exponential format which incorporates the prompt, error
C messages and range checking as passed parameters as follows:

C RVAL is the real returned, PROMP1 & PROMP2 are the prompts to be
C given to the user (similar to the syntax of USRMSG which allows
C two lines or a leading blank line).
C RMIN is the minimum value, RMAX is the maximum value.
C MINACT and MAXACT are actions to take if the range is exceeded:
C   'W' means warn user but accept, 'F' means refuse/fail to
C   accept value and ask again, '-' means no range checking.
C DEFLT is the default value if 'D','d', or return typed.
C ERMSG is a string appended to the range checking or read error
C to identify the value.
C IER is the error state, if 0 then OK, if -3 cancel button pressed.
```

```
C NHELP is the number of help lines and H() is the array
C of text strings to be printed out if '?','H','h' is typed
C by the user.
      SUBROUTINE EASKE(RVAL,PROMP1,PROMP2,RMIN,MINACT,
     &                 RMAX,MAXACT,DEFLT,ERMSG,IER,NHELP)
      CHARACTER*(*) PROMP1,PROMP2,ERMSG
      CHARACTER MINACT*1,MAXACT*1


C EASKF is a facility for asking the user for a file name which
C incorporates the prompt, error messages and help facilities
C as follows:

C FILEN is the string returned, ISTRW is its length.
C PROMP1 & PROMP2 are the prompts using the same syntax as USRMSG.
C If a space is typed then FILEN is not changed.
C ERMSG is a string placed at the top of the selection menu and to
C any range checking or read errors to identify the value.
C  IER is the error state, if 0 then OK if -3 then user requested CANCEL.
C NHELP is the number of help lines and H() is the array
C of text strings to be printed out if '?','H','h' is typed
C by the user.
C DSTR is a string to use as a default.
      SUBROUTINE EASKF(FILEN,PROMP1,PROMP2,ISTRW,DSTR,ERMSG,IER,NHELP)
      CHARACTER*(*) PROMP1,PROMP2,ERMSG,FILEN,DSTR


C EASKS is a facility for asking the user for a text string which
C incorporates the prompt, error messages and help facilities
C as follows:

C STRVAL is the string returned, ISTRW is its length.
C PROMP1 & PROMP2 are the prompts using the same syntax as USRMSG.
C If a space is typed then STRVAL is not changed.
C ERMSG is a string placed at the top of the selection menu and to
C any range checking or read errors to identify the value.
C IER is the error state, if 0 then OK, if -3 then cancel button pressed.
C NHELP is the number of help lines and H() is the array
C of text strings to be printed out if '?','H','h' is typed
C by the user.
C DSTR is a string to use as a default.
      SUBROUTINE EASKS(STRVAL,PROMP1,PROMP2,ISTRW,DSTR,ERMSG,IER,NHELP)
      CHARACTER*(*) PROMP1,PROMP2,ERMSG,STRVAL,DSTR


C EASKS248 edit a (248 char) string in a multi-line graphic box or via phrases.
C  STRVAL is the string returned, ISTRW is its length.
C  PROMP1 & PROMP2 are the prompts using the same syntax as USRMSG.
C  If a space is typed then STRVAL is not changed.
C  ERMSG is a string placed at the top of the selection menu and to
C  any range checking or read errors to identify the value.
C  IER is the error state, if 0 then OK, if -3 then cancel button pressed.
C  NHELP is the number of help lines and H() is the array
C  of text strings to be printed out if '?','H','h' is typed
C  by the user.
C  DSTR is a string to use as a default.
      SUBROUTINE EASKS248(STRVAL,PROMP1,PROMP2,ISTRW,DSTR,ERMSG,IER,
     &   NHELP)
      CHARACTER*(*) PROMP1,PROMP2,ERMSG,STRVAL,DSTR


C EASKSCMD is a facility for asking the user for a text string which
C incorporates the prompt, alternative command, error messages and
C help facilities as follows:
C  STRVAL is the string returned, ISTRW is its length.
C  PROMP1 & PROMP2 are the prompts using the same syntax as USRMSG.
C  CMD is an alternative command string, CMDACT is logical if the
C  command action is selected.
C  If a space is typed then STRVAL is not changed.
C  ERMSG is a string placed at the top of the selection menu and to
C  any range checking or read errors to identify the value.
C  IER is the error state, if 0 then OK, if -3 then cancel pressed.
```

```
C  NHELP is the number of help lines and H() is the array
C  of text strings to be printed out if '?','H','h' is typed
C  by the user.
C  DSTR is a string to use as a default.
      SUBROUTINE EASKSCMD(STRVAL,PROMP1,PROMP2,CMD,CMDACT,ISTRW,DSTR,
     &  ERMSG,IER,NHELP)
      CHARACTER*(*) PROMP1,PROMP2,CMD,ERMSG,STRVAL,DSTR
      LOGICAL CMDACT


C EASKSCNCL is a facility for asking the user for a text string which
C incorporates the prompt, calling code defined cancel string, error
C messages and help facilities as follows:
C  STRVAL is the string returned, ISTRW is its length.
C  PROMP1 & PROMP2 are the prompts using the same syntax as USRMSG.
C  CNCL is cancel command string, CNCLACT is logical if the
C  cancel action is selected.
C  If a space is typed then STRVAL is not changed.
C  ERMSG is a string placed at the top of the selection menu and to
C  any range checking or read errors to identify the value.
C  IER is the error state, if 0 then OK.
C  NHELP is the number of help lines and H() is the array
C  of text strings to be printed out if '?','H','h' is typed
C  by the user.
C  DSTR is a string to use as a default.
      SUBROUTINE EASKSCNCL(STRVAL,PROMP1,PROMP2,CNCL,CNCLACT,ISTRW,
     &  DSTR,ERMSG,IER,NHELP)
      CHARACTER*(*) PROMP1,PROMP2,CNCL,ERMSG,STRVAL,DSTR
      LOGICAL CNCLACT


C EASKS2CMD is a facility for asking the user for a text string which
C incorporates the prompt, two alternative commands, error messages and
C help facilities as follows:
C  STRVAL is the string returned, ISTRW is its length.
C  PROMP1 & PROMP2 are the prompts using the same syntax as USRMSG.
C  CMD & CMD2 are alternative command strings, if ICACT is non-zero
C  a command action is selected.
C  If a space is typed then STRVAL is not changed.
C  ERMSG is a string placed at the top of the selection menu and to
C  any range checking or read errors to identify the value.
C  IER is the error state, if 0 then OK, if -3 then cancel button pressed.
C  NHELP is the number of help lines and H() is the array
C  of text strings to be printed out if '?','H','h' is typed
C  by the user.
C  DSTR is a string to use as a default.
      SUBROUTINE EASKS2CMD(STRVAL,PROMP1,PROMP2,CMD,CMD2,ICACT,ISTRW,
     &  DSTR,ERMSG,IER,NHELP)
      CHARACTER*(*) PROMP1,PROMP2,CMD,CMD2,ERMSG,STRVAL,DSTR


c EPICKS is a facility allowing a number of selections to be made
C from an array of strings passed into the routine. EPICKS
C incorporates the prompt, error messages and returns an array of
C selected indexes as follows:

C PROMP1 & PROMP2 are the prompts using the same syntax as USRMSG.
C INPICK is passed as the number of items which are allowed to
C   be selected. If = NSTALT then the prompt will include the
C   phrase '* ALL' otherwise if INPICK < NSTALT then the prompt
C   will be in the form '* pick 2 items'. On return INPICK becomes
C   the actual number of items selected ( if 0 then none).
C IVALS is an array of NSTALT size such that:
C   IVALS(1) is the first index selected,
C   IVALS(2) is the second index ...
C   IVALS(INPICK) is the last index chosen.
C NSTALT is the number of selection strings STALT passed.
C ERMSG is a string appended to the range checking or read error
C to identify the value. IER is the error state, if 0 then OK.
C NHELP is the number of help lines and H() is the array
C of text strings to be printed out if '?','H','h' is typed
C by the user.
```

```
C Note that there is no particular limit on the number of alternate
C strings which can be passed to the subroutine as the selection menu
C allows paging.
C User-defined text strings and string alternatives
C will be truncated at the width ISTRW.
      SUBROUTINE EPICKS(INPICK,IVALS,PROMP1,PROMP2,
     &                ISTRW,NSTALT,STALT,ERMSG,IER,NHELP)
      DIMENSION IVALS(NSTALT)
      CHARACTER*(*) PROMP1,PROMP2,ERMSG,STALT(*)


C MENUATOL presents a list of up to a dozen items to select via a
C menu format but with the text for each selection passed (as in
C easkatog). It is assumed that the user will make one selection
C only and if nothing selected it returns an index of zero.
C
C It is passed a prompt, menu title, menu width (mw characters),
C default index (idindex) and current index (index) if zero then
C assumes no current selection. If user exits without selection then
C index is returned as zero if index was passed in as zero and
C is reset to index value if non-zero was passed.
C Automatic key characters are suppressed and are assumed to be
C passed within the parameter list text.
      SUBROUTINE MENUATOL(prompt,title,AOPT,BOPT,COPT,DOPT,EOPT,FOPT,
     &  GOPT,HOPT,IOPT,JOPT,KOPT,LOPT,index,idindex,nhelp)
      LOGICAL SELECT
      CHARACTER*(*) prompt,title,AOPT,BOPT,COPT,DOPT,EOPT,FOPT,GOPT
      CHARACTER*(*) HOPT,IOPT,JOPT,KOPT,LOPT


C ASKOK Generic choice facility returning logical variable
C from a yes no (with default indicated) prompt (supports help).
C In the case of a graphic menu the messages will appear in a
C dialogue box at the bottom of the graphic window.
C If NHELP=-1 there is no default option.
      SUBROUTINE ASKOK(MSG1,MSG2,OK,DOK,NHELP)
      CHARACTER*(*) MSG1,MSG2
      logical ok,dok


C EASKATOG choice of 3 - 7 items returning 1 to 7 according to
C which of the choices has been chosen.
C MSG1 and MSG2 are prompts (upto 124 char) to be printed.
C AOPT, BOPT & COPT, DOPT, EOPT, FOPT, GOPT are the text strings describing
C the options available, (these should not be too long).  If all optons
C are filled with text then all presneted.  If and option (beginning
C with FOPT and working backwards is a single blank space then fewer
C options are presented.
C In the case of a graphic menu us gtk functions will be called
C and the messages will appear in a pop-up dialogue box.
      SUBROUTINE EASKATOG(MSG1,MSG2,AOPT,BOPT,COPT,DOPT,EOPT,FOPT,GOPT,
     &  IWHICH,NHELP)
      CHARACTER*(*) MSG1,MSG2,AOPT,BOPT,COPT,DOPT,EOPT,FOPT,GOPT


C EASKDOZEN Generic choice of a dozen items returning 1 to 12 according to
C which of the choices has been chosen.
C MSG1 and MSG2 are prompts (upto 124 char) to be printed.
C AOPT, BOPT & COPT, DOPT, EOPT, FOPT, etc. are the text strings describing
C the options available, (these should not be too long).  If all optons
C are filled with text then all presneted.  If and option (beginning
C with FOPT and working backwards is a single blank space then fewer
C options are presented.
C In the case of a graphic menu us gtk functions will be called
C and the messages will appear in a pop-up dialogue box.
      SUBROUTINE EASKDOZEN(MSG1,MSG2,AOPT,BOPT,COPT,DOPT,EOPT,FOPT,
     &  GOPT,HOPT,IOPT,JOPT,KOPT,LOPT,IWHICH,NHELP)
      CHARACTER*(*) MSG1,MSG2,AOPT,BOPT,COPT,DOPT,EOPT,FOPT,GOPT
      CHARACTER*(*) HOPT,IOPT,JOPT,KOPT,LOPT


C EASKPER: Provides legacy interface to the more general eAskPerGen below.
```

```
      SUBROUTINE eAskPer(PROMP1,IBDOY,IEDOY,IFDAY,IER)
      CHARACTER*(*) PROMP1
      integer ibdoy,iedoy,ifday,iBYr,iEYr,ier


C EASKPER: Provides multi-year capable interface to the more general
C eAskPerGen below.
      SUBROUTINE eAskPerYear(PROMP1,ibdoy,iedoy,iBYr,iEYr,ifday,ier)
      CHARACTER*(*) PROMP1
      integer ibdoy,iedoy,iBYr,iEYr,ifday,ier


C eAskPerGen: Provides interface to specification of a multiyear period of days. It
C returns IBDOY and IEDOY based on the current settings of IFDAY.
C iBYr,iEYr are the begining year and ending year of the period.
C Promp1 gives the context of the request for a period.
      SUBROUTINE eAskPerGen(PROMP1,ibdoy,iedoy,iBYr,iEYr,ifday,
     &                      bMY_enabled,IER)
      integer ibdoy,iedoy,iBYr,iEYr,ifday,ier
      CHARACTER*(*) PROMP1


C EMPAGE: Low level screen control for paging based on terminal MMOD.
C The available terminal see EPAGES.
      SUBROUTINE EMPAGE(IPAG,IW,IEND)


C ELINC: Controls scratch pad output for text screens which returns:
C For TTY & LPT ELINC tests if N lines fit on the page, if yes then the
C line cout is updated to give lines left on page, if not the terminal is
C paged and a new limit is set.
      SUBROUTINE ELINC(N)


C EPAGES: Initialise terminal, set up a scratch pad counter depending
C on terminal type. The parameter TITLE will appear in the window heading.
C The method of page termination depends on the MMOD number.
C The available terminal types are:
C type -6 = batch/shell/function button mode.
C type -2 = teletype with waiting.
C type -1 = teletype.
C type  8 = bitmapped with dialogue box.
C iappwi is the requested pixel width, iapphi is the requested pixel height
C iappx & iappy are the upper-left position on the monitor <not yet implemented >
C menuchw is the initial request for menu width (in characters)
      SUBROUTINE EPAGES(MODEL,IIN,IOUT,iappwi,iapphi,iappx,iappy,
     &  menuchw,TITLE)
      CHARACTER*(*) TITLE


C EMENU: Control menu display on various terminals.  Name is a character
C string to form the heading of menu, ITEMS is an array of character
C strings making up the menu, NITMS is the number of items in the menu,
C INO is the number of the item chosen.
      SUBROUTINE EMENU(NAME,ITEMS,NITMS,INO)
      CHARACTER*(*) NAME,ITEMS(NITMS)


C EWMENU: Is the binding to C function for menu dialogue.  It
C allows the string widths to be variable widths. Uses width of
C items passed in call to vwmenu.
      SUBROUTINE EWMENU(name,items,nitms,impx,impy,irpx,irpy,ino)
      character*(*) name, items(*)


C VWMENU: Is the binding to C function for menu dialogue.  It
C allows the string widths to be variable widths.
      SUBROUTINE VWMENU(name,items,nitms,impx,impy,iw,irpx,irpy,ino)
      character*(*) name, items(*)


C EMKEY returns a key character for a menu item (a-z) based on the array
```

```
C index of the item.  Uses ICHPK(26), 'a','b'... from esrulib...
      SUBROUTINE EMKEY(IAI,KEY,IER)
      CHARACTER*1 KEY


c Generic error reporting facility.  MSG1
C and MSG2 are text strings (upto 124 char) to be printed.  LEVEL is a
C single character 'W' or 'w' for warning (in graphic mode followed by
C a clearing of dialog), 'F' or 'f' for Fatal error,
C '-' to only print the messages, '?' is a prompt in text mode. If
C LEVEL is 'P' or 'p' then pause breifly before continuing. In the case of a
C fatal error STOP will be called, otherwise execution will return to
C the calling point.
C In the case of a graphic menu the c function msg_box will be called
C and the messages will appear in a dialogue box at the bottom of the
C graphic window. The user must have previously called open_msg_box(2).
      SUBROUTINE USRMSG(MSG1,MSG2,LEVEL)
      CHARACTER*(*) MSG1,MSG2
      CHARACTER LEVEL*1


c Generic error reporting facility.  MSG1
C and MSG2 are text strings (upto 248 char) to be printed.  LEVEL is a
C single character 'W' or 'w' for warning (in graphic mode followed by
C a clearing of dialog), 'F' or 'f' for Fatal error,
C '-' to only print the messages, '?' is a prompt in text mode. If
C LEVEL is 'P' or 'p' then pause breifly before continuing. In the case of a
C fatal error STOP will be called, otherwise execution will return to
C the calling point.
C In the case of a graphic menu the c function msg_box will be called
C and the messages will appear in a dialogue box at the bottom of the
C graphic window. The user must have previously called open_msg_box(2).
      SUBROUTINE LUSRMSG(MSG1,MSG2,LEVEL)
      CHARACTER*(*) MSG1,MSG2
      CHARACTER LEVEL*1


C EDISP is a generic routine which displays lines of text passed to it
C in a format depending on the terminal type:
C For types -1 -2 9 does a fortran write to channel IUOUT,
C For types -6 writes to ICOUT,
C For type 8 manages the text which is passed to inserttext for
C treatment as a scrolling window.
      SUBROUTINE EDISP(ITRU,MSG)
      CHARACTER*(*) MSG


C PHELPD displays the current contents of common pophelp in a form
C appropriate to the current terminal type. MSG is a short descriptive
C string for the subject. NHELP is the number of lines to be displayed.
C IBX & IBY are the preferred coords of the lower left corner if in
C terminal type 8.
      SUBROUTINE PHELPD(MSG,NHELP,OTHER,IBX,IBY,IER)
      CHARACTER*(*) MSG,OTHER
```

Appendix C: ESRU esrucom folder

Each of the API for the files in folder esrucom are listed in the following sections.

BC_data.F

This file contains procedures for managing user-specified boundary conditions. Used by NRCan staff.

```
C process_BC_data_file parses a boundary condition definition file, checks
C  it for validity, and writes a binary database containing the
C  relevant records.
C  Inputs
C    iASCII_num: file number to use for ascii file input
C    iBC_TmpFile: file number to use for binary file output
C    bAutoOK: flag indicating simulator is in silent running mode
C  Outputs
C    bFatal_Error: flag indicating that error was encountered
      subroutine process_BC_data_file(
     &    iBase_File_num, bAutoOK, bFatal_Error )
      integer iBase_File_num              ! Basic I/O file number
      logical bAutoOK                     ! flag for silent-running mode
      logical bFatal_Error                ! fatal error flag


C bInquire_BC_Name_Exists determines if a requested boundary condition
C has been defined.
C Inputs:
C    - cCol_name: Name of desired boundary condition
C Outputs:
C    - bInquire_BC_Name_Exists: logical result.
      logical function bInquire_BC_Name_Exists ( cCol_name )
      character*248 cCol_name


C fGet_BC_data_by_name is a short-hand interface to  fGet_BC_data_by_col
C which allows the boundary condition's name to be used to identify
C the appropriate column.
C Inputs:
C   cCol_name: Desired BC column number
C   fDay: real number denoting day (& fraction thereof) for which
C      the boundry condition should be recovered.
C   iInterp_method: method for interpolation
C      1 -> step-wise
C      2 -> linear
C Outputs
C   Value of BC at future time row
      real function fGet_BC_data_by_name(
     &                                   cCol_name,
     &                                   fDay,
     &                                   iInterp_method,
     &                                   cContext       )
      character*248 cCol_name
      real fDay
      integer iInterp_method
      character*124 cContext


C fGet_BC_data_by_col returns boundry-condition data corresponding
C to the current time step and user-specified column number.
C Inputs:
C   iCol_number: Desired BC column number
C   fDay: real number denoting day (& fraction thereof) for which
C      the boundry condition should be recovered.
C   iInterp_method: method for interpolation
C      1 -> step-wise
C      2 -> linear
C Outputs
C   Value of BC at future time row
      real function fGet_BC_data_by_col(iCol_number,
     &                                  fDay,
     &                                  iInterp_method,
```

```
     &                                  cContext       )
      integer iCol_number     ! column number.
      integer iInterp_method  ! interpolation method
      real fDay               ! day number
      character*124 cContext  ! Contextual message


C get_BC_row returns an array containing boundary condition values
C corresponding to a given boundary condition row number.
C Inputs:
C   iRow: row number
C Outputs
C   fValues: array containing the BC values for the given row number.
      subroutine get_BC_row( iRow, fValues, cContext )
      integer iRow                    ! Row number
      real fValues(mBC_max_count)     ! Values from file


C report_BC_data reports the values of the boundary condition data to
C the XML reporting facility
C Inputs:
C   None
C Outputs
C   None
      subroutine report_BC_data()


C cleanup_BC_data  deletes the temporary file created by
C the boundary condition management facility.
C Inputs:
C   None
C Outputs
C   None
      subroutine cleanup_BC_data()
```

Lookup_data.F

This file contains subroutines and functions for managing a user-specified data table. The API follows.

```
C process_Lookup_data_file  parses a lookup data file, checks that the contents
C of the file are valid and writes a binary database containing the
C table of data.
C Inputs:
C iBase_file_num: Basic I/O file number; ASCII and binary files will
C                 be set relative to this number.
C Output:
C bFatal_Error: flag indicating that error was encountered
       subroutine process_Lookup_data_file(iBase_file_num,
     &                                     bAutook,
     &                                     bFatal_error)
       integer iBase_file_num        ! basic I/O file number
       logical bAutook               ! flag for silent-running mode
       logical bFatal_error          ! fatal error flag


C iGet_Lookup_col_num  returns the column number that corresponds to
C the column header specified by the user
C Inputs:
C cCol_name: Column name for which column number is to be determined
C Output:
C iCol_num: Columnn number corresponding to column name
       integer function iGet_Lookup_col_num( cCol_name,
     &                                       cContext   )
       character*248 cCol_name  ! column header/name
       character*124 cContext   ! contextual message


C fGet_Lookup_value_by_col_num  returns data from a lookup table that corresponds
C to a user-specified reference value, reference column number
C and value column number.
```

```
C Inputs:
C iCol_ref : Reference column number
C iCol_value: Value column number
C fRef_value: Reference value
C iInterp_method: method for interpolation
C                 1 -> step-wise
C                 2 -> linear
C Outputs
C Value from lookup table corresponding reference value
      real function fGet_Lookup_value_by_col_num(iCol_ref,
     &                                           iCol_value,
     &                                           fRef_value,
     &                                           iInterp_method,
     &                                           cContext )
      integer iCol_ref        ! reference column number
      integer iCol_value      ! value column number
      real fRef_value         ! reference value
      integer iInterp_method  ! interpolation method
      character*124 cContext  ! contextual message


C fGet_Lookup_value_by_col_name returns data from a lookup table that corresponds
C to a user-specified reference value, reference column name
C and value column name. This procedure is an interface to the
C function fGet_Lookup_value_by_col_num which allows the column names
C (versus column numbers) to identify the reference and value columns.
C Inputs:
C cCol_ref : Reference column name
C cCol_value: Value column name
C fRef_value: Reference value
C iInterp_method: method for interpolation
C                 1 -> step-wise
C                 2 -> linear
C Outputs
C Value from lookup table corresponding to reference value
      real function fGet_Lookup_value_by_col_name(cCol_ref,
     &                                            cCol_value,
     &                                            fRef_value,
     &                                            iInterp_method,
     &                                            cContext )
      character*248 cCol_ref     ! reference column name
      character*248 cCol_value   ! value column name
      character*124 ctext        ! context message
      real fRef_value            ! reference value
      integer iInterp_method     ! interpolation method
      character*124 cContext     ! contextual message


C cleanup_Lookup_data deletes the temporary file created by
C the lookup table facility.
       subroutine cleanup_Lookup_data()
```

MultiYear_climate.F

This file contains procedures for handling climate data during multi-year simulations.

```
C Parse_MY_clm_db parses the multi-year climate index file. It:
C  - Checks if the multiyear climate file has been properly
C    defined.
C  - Saves the multi-year climate database in memory
C Inputs:
C  iFile: current file index.
C  AutoOK: logical flag indicating that the the simulator is
C    running in silent running mode
C Outputs:
C    bCriticalErr     ! flag incidating a critical error has been encountered
      subroutine Parse_MY_clm_db( iFile_num,
     &                            bAutoOK,
     &                            bCriticalErr )
      integer iFile_num        ! climate file number used by bps
```

```
      logical bAutoOK          ! flag for silent-running mode
      logical bCriticalErr     ! flag incidating a critical
                               ! error has been encountered


C Check_MY_clm_db  checks that the multi-year climate database
C data stored in memory is correct.
C  - Checks if the listed year records are contigious and
C    in order, and
C  - Checks if the clm files exist.
C Inputs:
C  - bAutoOK: status of simulatior (silent/interactive modes)
C  - cAction: action to be taken if recoverable errors are
C    found.
C Outputs:
C  - bCriticalErr: flag indicating if a critical error was
C    encountered.
      subroutine Check_MY_clm_db(bAutoOK,cActionPass,bCriticalErr)
      logical bCriticalErr
      logical bAutoOK
      character*(*) cActionPass


C MY_clm_db_menu is used to control specification of a multi-year
C climate database
C Inputs - none; outputs - none
      subroutine MY_clm_db_menu ()


C Spec_MY_clm_db is used to specify an existing, or new
C multi-year climate file.
C Inputs
C  - cAction - Action to be taken if file not found
C Outputs:
C  - bFileExists - flag indicating that file was found
C  - bCreate  - flag indicating file should be created
      subroutine Spec_MY_clm_db(cActionPass,bFileExists,bCreate)
      logical bFileExists
      logical bCreate
      character*(*) cActionPass
      character*16 cAction


C Sort_MY_clm_db checks to see if the climate records are in order, and
C sorts them if necessary
C Inputs:
C  - cContext: contextual buffer
C  - cAction: flag indicating what actions should be taken
C             on out-of-order records
C Outputs:
C  - bOrderChanged: flag indicating if records were out of order
      subroutine Sort_MY_clm_db(cContext,cActionPass,bOrderChanged)
      logical bOrderChanged
      character*(*) cActionPass
      character*124 cContext      ! contextual buffer


C Fill_MY_clm_db checks for continuity in the multi-year climate
C database, and fills in missing years, if necessary
C Inputs:
C  - cContext: contextual buffer
C  - cAction: flag indicating what actions should be taken
C             on out-of-order records
C Outputs:
C  - bCriticalErr: flag indicating if a critical error was
C    encountered.
C  - bContigious: Flag indicating if file was contigious
      subroutine Fill_MY_clm_db(cContext,cActionPass,
     &                          bCriticalErr,bNonContigious)
      logical bCriticalErr
      logical bNonContigious
      logical bClmFileExists      ! Flag indicating climate
```

```
                                  ! file has been found
      character*124 cContext !msg buffers
      character*(*) cActionPass


C Prune_MY_clm_db checks for duplicate records, and optionally deletes
C them
C Inputs:
C  - cContext: contextual message
C  - cAction: flag indicating the action that should be undertaken
C    if duplicates are found
C Outputs:
C  - bDuplicates: Flag indicating if file was contigious
      subroutine Prune_MY_clm_db(cContext,cActionPass,bDuplicates)
      logical bDuplicates
      character*(*) cActionPass
      character*124 cContext      ! context


C Save_MY_clm_db controls saving a multi-climate database to
C disk
C Inputs:
C  - iFile_num: file number to use
C Outputs:
C  - bCriticalErr: Flag indicating if error was encountered.
C  - bSaveOk: flag indicating that file was saved successfully.
      subroutine Save_MY_clm_db ( iFile_num, bCriticalErr,bSaveOk )
      integer iFile_num
      logical bCriticalErr
      logical bSaveOk


C Write_MY_clm_db writes out a climate file database to disk.
C Inputs:
C  - iFile_num: file number to use
C Outputs:
C  - bCriticalErr: Flag indicating if error was encountered.
C  - bSaveOk: flag indicating that file was saved successfully.
      subroutine Write_MY_clm_db ( iFile_num, bCriticalErr,bSaveOk )
      integer iFile_num
      logical bCriticalErr
      logical bSaveOk
```

SiteUtilities.F

This file contains routines used to collate fuel used on site.

```
C SiteUtilities invokes the various subroutines needed to characterize
C purchased energy. Support for green-house-gas emissions may be
C added in the future.
      subroutine SiteUtilities()


C TotalPurchasedEnergy computes the total amount of energy use reported from all
C parts of the simulator.
      subroutine TotalEnergyConsumption()


C ConvertEnergyToFuel converts the total energy use associated with a given
C fuel into a standard unit of measure, based on the fuel's calorific
C value.
      subroutine ConvertEnergyToFuel()


C H3KReportsFuelUse reports transport data on fuel usage to the H3Kreports facility.
      subroutine H3KReportsFuelUse


C ZeroFuelEnergyStorageArray zeros the site energy usage storage array.
      subroutine ZeroFuelEnergyStorageArray()
```

```
C ZeroPltEnergyStorageArray zeros the site energy usage storage array.
      subroutine ZeroPltEnergyStorageArray()


C StoreSiteUtilityData takes the data stored in the passed fSUFuelEnergyUse
C array and stuffs it into the protected fSiteEnergyUse array
C Inputs:
C Parameters defined via include statements.
C   - iComponent: integer flag indicating where data originated
C   - fSUFuelEnergyUse: 2d array used transport fuel consumption
C     data into Site utilities facility. Array indicies correspond
C     to fuel, end-use.
      subroutine StoreSiteEnergyUse(iComponent, fSUFuelEnergyUse )


C StorePltCompEnergyUse stores fuel energy use computed by plant
C component models for later access in the site utilities facility.
C Inputs:
C Parameters defined via include statements.
C   - iComP: Plant component index
C   - fSUFuelEnergyUse: 2d array used transport fuel consumption
C     data into Site utilities facility. Array indicies correspond
C     to fuel, end-use.
      subroutine StorePltCompEnergyUse(iComponent, fSUFuelEnergyUse )


C AggregatePltCompEnergy aggregates energy consumption computed
C by various plant components, and appends it to the site-wide
C energy consumption array.
C Inputs:
C   - None.
      subroutine AggregatePltCompEnergyUse()


C AttributePltPfsEnergyUse analyses uncatagorized energy consumption reported in
C the plant/elec domains and attempts to attribute them according to
C end-use.
C Inputs: None.
      subroutine AttributePltPfsEnergyUse()


C fConvertEnergyToFuelAmount converts an energy input into an equivlent
C amount of fuel. It makes the OffsiteUtilities conversion factors
C available elsewhere in the simulator.
C Inputs:
C  -fEnergy:   Energy used (W)
C  -iFuel:     Integer indicating the type of fuel used.
C  Outputs:
C  -fConvertEnergyToFuelAmount: Equivlent amount of fuel (various units)
      real function fConvertEnergyToFuelAmount(fEnergy,iFuelIndex)
      real fEnergy
      integer iFuelIndex
```

bsimsel.F

This source file provides an interface to the selection of basement types based on the BASECALC foundation loss model from Natural Resources Canada.

```
C bsimtype is based on data from BASESIMP, A Simplified Foundation
C Energy-Loss Model Derived from BASECALC (TM) Simultions
C Ian Beausoleil-Morrison, CANMET Energy Technology Centre,
C Natural Resources Canada, 1996, 1998 1999.
C ibst (integer) is the index of the user selection to be returned
      subroutine bsimtype(ibst)
      integer ibst  ! index of user selection
```

c2fdum.F

Dummy routines for modules which do not require interactive graphics. See the files in esrunet for specific API.

```
C  netwdrw  dummy drawing routine for esrunet/network.F

C  nwkslctc() dummy routine for esrunet/netwkc2f.F

C  gnwkquery() dummy routine for esrunet/netwkc2f.F

C  netwsnap() dummy routine for esrunet/netwkc2f.F

C  netwmic() dummy routine for esrunet/network.F

C  gconad() dummy routine for esrunet/netwkc2f.F

C  gridupdt() dummy routine for esrunet/netwkc2f.F

C  nwkupdtpos() dummy routine for esrunet/netwkc2f.F

C  icntfm() dummy routine for esrunet/netutil.F

C  netwmco() dummy routine for esrunet/network.F
```

cfdrw.F

C Read and write new (version 2,2.1 and 2.2) cfd description.

```
C DFDSV saves the room air movement description file (version 2).
C IUF unit number for CFD input file.
C IZONE is the index of the zone with a CFD domain.
C IER=0 indicates no error.
      SUBROUTINE DFDSV(IUF,IZONE,IER)
      integer iuf,izone,ier


C NEW2OLD - Copies new data structure to old.
      SUBROUTINE NEW2OLD


C CFMFSU - set up CFD to mass flow connections/ components.
C All connections to the dom_nod (ICFDNOD), are deactivated.
C A new node (opn_nod) is created for this opening in the CFD domain.
C A new connection from this node is made to the node, ICFDNOD was
C connected to (ext_nod), using the same component.
C Consequentially the number of new nodes created will be equal to the
C number of openings in CFD domain. The nodes created here are internal
C and constant pressure (type 1) type nodes.
C IV is the index of the opening number.
C ITRC signals verbose reporting if greater than zero.
C IND is the index of the node.
C ICN is the index of the connection.
C OPENNAM (char 12) name of the opening.
      subroutine CFMFSU(IV,ITRC,IND,ICN,OPENNAM)
      integer iv,itrc,ind,icn
      character OPENNAM*12


C MFLISTemp - list common block contents for debugging only.
      SUBROUTINE MFLISTemp


C CFDDEFLT - Sets default values for all data items.
      SUBROUTINE CFDDEFLT


C SCHCHK checks the turbulent schmidt number of all containments
C defined in the CFD domain to be non - zero. If any
```

```
C contaminant is zero it is set to the default value unity
      SUBROUTINE SCHCHK


C BCEKCK checks the boundary conditions of all boundary cells
C (blockages are yet not considered) to be defined.
C itrc signals verbose reporting if greater than zero
      SUBROUTINE BCEKCK(ITRC)
```

X.F

xx

To be done