List of the macros of precompilation

Summary:

This document lists the macros present in the code and, for the majority, given at the time of the phase of precompilation.

Contents

1 Classification

The use of the macros of the precompiler makes it possible to associate a text with an identifier.
In Code_Aster, one uses the macros to adapt the code to the environment of machine (architecture, libraries available, etc).

One presents thereafter the macros gathered by category.

Notice

One does not describe the macros placed here in a conventional way in the file design of heading which makes it possible to avoid recursive inclusions. These macros must take again the name of the file (ASTER_CONSTANT_H for aster_constant.h) and are used only in the file in question.
One does not detail either some macros which are used only locally in a module by convenience of programming.

2 Configuration

One gathers in this category the macros related to the configuration preliminary to compilation of a version. One finds them in the headings asterc_config.h (and almost all in asterf_config.h) who are produced automatically at the time of the stage waf configures.
Macros preceded by (*) are given at the time of the phase of configuration but are not added in the files of heading asterc_config.h and asterf_config.h They passed on command line with -D<nom-macro>.

- _POSIX / _WINDOWS : exclusive, known as if the platform is of type Unix or Windows.
- _USE_64_BITS : known as if the platform is 64 bits
- LINUX, LINUX64, SOLARIS, SOLARIS64 : to use to distinguish certain systems among different Unix.
- _NO_UNDERSCORE : used for calls C-FORTRAN, not to add the '_' at the end of the routine.
- _USE_MPI : indicate that one uses parallelism MPI.
- _USE_OPENMP : indicate that one uses the OpenMP directives.
- _DISABLE_HDF5 : indicate that the library hdf5 is not available. By default, one supposes that hdf5 is available (like if HAVE_HDF5 was defined).
- _DISABLE_MED : indicate that the library med is not available. By default, one supposes that med is available (like if HAVE_MED was defined).
- _HAVE_MUMPS : indicate that the library mumps is available.
- MUMPS_VERSION : version of the library mumps.
- _HAVE_MONGREL : indicate that the library mongrel is available.
- _DISABLE_SCOTCH : indicate that the library Scotch tape is not available. By default, one supposes that Scotch tape is available (like if HAVE_SCOTCH was defined).
- _HAVE_PETSC : indicate that the PETSc library is available.
- (*) _USE_INTEL_IFORT : indicate that one uses the Intel compiler. The features specific to Intel must use _USE_INTEL_IFORT && HAVE_xxx because aslint does not have to use it, because independent of the compiler. This macro if need be passed on command line with -D_USE_INTEL_IFORT.
- HAVE_TRACEBACKQQ : indicate that the function tracebackqq is available (requires _USE_INTEL_IFORT).
- HAVE_BACKTRACE : indicate that the function backtrace is available.
- _WITHOUT_PYMOD : to use for the separate construction of a library containing the whole of the modules objects and modules python in the form of dynamic libraries.
- _MAIN_ : name expected for the main program (can be hand, or HAND preceded and/or followed or not of '_').
- STRINGIFY_USE_QUOTES, STRINGIFY_USE_OPERATOR : indicate how preprocessor FORTRAN converts code into character string.

Definition of the types:

- ASTER_INT4_SIZE : size of type FORTRAN integer (kind=4).
- ASTERC_FORTRAN_INT4 : type C corresponding to entirety FORTRAN integer (kind=4).
- ASTER_INT_SIZE : size of type FORTRAN integer.
- ASTERC_FORTRAN_INT : type C corresponding to entirety FORTRAN integer.
- ASTER_REAL4_SIZE : size of type FORTRAN real (kind=4).
- ASTERC_FORTRAN_REAL4 : type C corresponding to entirety FORTRAN real (kind=4).
- ASTER_REAL8_SIZE : size of type FORTRAN real (kind=8).
- ASTERC_FORTRAN_REAL8 : type C corresponding to entirety FORTRAN real (kind=8).
- ASTER_COMPLEX_SIZE : size of type FORTRAN complex (kind=8) (complex*16 fortran77).
- ASTERC_STRING_SIZE : type of the size of the character strings out of C, exchanged with FORTRAN.
- MED_INT_SIZE : size of the whole type used in med.

3 Features

One finds in this category the macros related to activation, the desactivation of certain features and their parameter setting.

- _DISABLE_MATHLIB_FPE : allows to be unaware of the error digital (floating not exceptions) at certain places of the code, between two calls to the function matfpe, in particular around the calls to the routines blas/lapack. It is the behavior by default.
- _ENABLE_MATHLIB_FPE : modify the behavior by default, does not intercept this kind of error.
- ASTER_DISABLE_MPI_CHECK : disable the checking of communications MPI.
- _NO_EXPIR : allows to remove alarm preventing that a version has more than 15 month.
- USE_ASSERT : allows to activate the assertions in the part C written with AS_ASSERT.
- TEST_STRICT : when this macro is activated, one does not take account of the value of the keyword TOLE_MACHINE in the functions of test (TEST_RESU and derived). A message of information points out it.

4 Interfaces C-FORTRAN

In definition.h, many macros is defined in order to avoid the errors at the time of the passage of argument between C and FORTRAN.

For example:

```
#define CALL_JEEXIN (has, b)  CALLSP (JEEXIN, jeexin, has, b)
```

This makes it possible to call since C the subroutine FORTRAN JEEXIN whose first argument is a character string (S) and the second a pointer (P) towards an entirety or a reality.

The interest is that according to the platform, it is necessary to add or not a '_' at the end of the name of the subroutine. In the same way, it can be necessary to pass the length of the character string right after the variable or into last position.
NB: following the passage in FORTRAN 90, one could improve the passage of the character strings.

5 System

One gathers in this category the macros related to the operating system and architecture of the machine. Macros preceded by (*) are activated automatically in aster_depend.h according to the stage of configuration or platform.
The objective is that they all are given with the configuration.

- (*) _STRLEN_AT_END : used for calls C-FORTRAN, indicates that the length of the character string must be deferred after the arguments and not just after the argument of the type chains.
- (*) GNU_LINUX : indicate that one will use later (in the management of the signals) _GNU_SOURCE=1 to reach functions except standard POSIX.
- _NON_BULL : a particular value for “real the nondefinite one defines”, R8UND in envima.c .

Types determined at the time of the configuration, redefined for practical reasons and of legibility:
- (*) STRING_SIZE : type of the size of the character strings out of C.
- (*) ASTERINTEGER4 : type C corresponding to entirety FORTRAN integer (kind=4).
- (*) ASTERINTEGER : type C corresponding to entirety FORTRAN integer.
- (*) ASTERREAL4 : type C corresponding to entirety FORTRAN real (kind=4).
- (*) ASTERDOUBLE : size of type FORTRAN real (kind=8).
- (*) ASTERC_FORTRAN_REAL8 : type C corresponding to entirety FORTRAN real (kind=8).
- REAL_NB_CHIFFRES_SIGNIFICATIFS : many significant figures for realities.
- INTEGER_NB_CHIFFRES_SIGNIFICATIFS : many significant figures for the entireties.
- OPT_TAILLE_BLOC_MULT_FRONT : optimal size of the blocks for the solvor mult-face.
- _USE_STDCALL : known as if one uses __stdcall for the calls of functions FORTRAN since C, for Windows.

6 Use by the developer

In the programming, one avoids using macros of very low level.
For example, out of C, one never uses ASTER_INT_SIZE (the number of bytes used for an entirety) but the associated type. And for this type, one does not use ASTERC_FORTRAN_INT but the “internal” name: INTEGER.
In the same way in FORTRAN, one does not use directly ASTER_INT_SIZE, but values defined in aster_types.h.

6.1 Types to be used out of C

One includes the file of heading general aster.h (which included aster_depend.h). One has access to the above mentioned types then: INTEGER, INTEGER4, DOUBLE, REAL4, STRING_SIZE.

6.2 Types to be used in FORTRAN

The file of heading is included aster_types.h who defines:
- aster_int_kind
- aster_int

as well as functions of conversion to_aster_int.
Types and similar functions of conversion are defined: med_int, mpi_int, mumps_int, blas_int and thus to_mpi_int, etc.

One takes care to use these types in the functions which exchange data with external libraries.

6.3 Management of error and exception out of C

In FORTRAN, one has macro ASSERT who checks a condition and stops in error when this one is not checked by indicating the place in the source.

To transmit an error message out of C, the macro one is used MYABORT who allows to raise an exception to stop calculation properly.

One emulates in the code C the mechanism of exception while being based on the functions system setjmp and longjmp.
The writing is simplified by the use of the macros: try, except, exceptAll, endTry. Like raiseException and raiseExceptionString to raise an exception.
See the module aster_exceptions.c for an example.

6.4 Macros utilities in FORTRAN

- `ASSERT (condition)` : an error message emits if the condition is not checked by indicating the name of the file and the line of code where the checking was made.
- `TO_STRING (code)` : replace `code` by a character string, usable in others macros (example in `ASSERT`).