Precision (kind) constants for Pentium machines

Aleksandar Donev

January 2000

Contents
1 Module Precision

The module Precision contains definitions of certain kind parameters for integer, real and boolean numbers. It is of course very platform dependent. Here I use the module Standard_Types from the Portability Project by Dan Nagle to achieve some portability.

The most important parameters here are \texttt{i32} and \texttt{i64}, which are the kinds for 32- and 64-bit integers (used in certain codes where bit-based operations are performed, such as random-number generation or Hilbert-curve generation), the kinds \texttt{iword} and \texttt{rword} which are the kinds of default \texttt{integer} and \texttt{real} numbers. But really most important to this library are \texttt{iwp}, \texttt{rwp} and \texttt{lwp} which represent the working precisions for integers (only for numbers that may grow too large), reals (single or double—very important for optimization) and boolean (default \texttt{logical} on Pentiums is a whole word, which is wasteful on memory, but usually faster).

"Precision.f90" 1 ≡

\begin{verbatim}
MODULE Precision
USE Standard_Types     // From portability project
PRIVATE
    ! These values need to be changed when switching platforms, especially to a 64-bit machine such as an Alpha: */
    INTEGER, PARAMETER, PUBLIC :: byte = byte_k, short = short_k, int = int_k, long = long_k,
                                   iword = KIND(0)     // Integers
    INTEGER, PARAMETER, PUBLIC :: byte2 = byte, byte4 = byte        // 2's complement integers
    INTEGER, PARAMETER, PUBLIC :: single = single_k, double = double_k, quad = quad_k,
                                   rword = KIND(0,0)    // Real numbers
    INTEGER, PARAMETER, PUBLIC :: short = byte, word = int_k        // Boolean values
    INTEGER, PARAMETER, PUBLIC :: rwp = rwp, iwp = iwp, lwp = lwp   // Working precisions
END MODULE Precision
\end{verbatim}

[HPF2Formatting.hweb]
2 Formatting rules for HPF/F90 files

These are just same auxiliary formatting rules and useful macros I use from time to time.

@m _GenericInterface(generic_name, ...)
  INTERFACE generic_name
  MODULE PROCEDURE #.
  END INTERFACE generic_name
@m _DeclareIntWord(...)
  INTEGER :: #.
@m _DeclareIntWP(...)
  INTEGER (KIND = i_wp) :: #.
@m _DeclareRealWP(...)
  REAL (KIND = r_wp) :: #.
@m _DeclareRealSP(...)
  REAL (KIND = r_sp) :: #.
@m _DeclareRealDP(...)
  REAL (KIND = r_dp) :: #.
@m _FullExtent(rank) : DO (DIM, 2, rank) { , }
@m _VarSequence(variable, start, end)
  variable##start DO (DIM, SEVAL(start + 1), end) { , variable@DIM }
@m _NestedLoopStart(variable, array, rank)
  DO (DIM, rank, 1, -1) { DO variable@DIM = LBOUND(array, DIM), UBOUND(array, DIM) }
@m _NestedLoopEnd(rank) DO (DIM, 1, rank) { END DO }
@m _Dummy(...)