

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 *i_32* and *i_64*, 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 *i_word* and *r_word* which are the kinds of default **INTEGER** and **REAL** numbers. But really most important to this library are *i_wp*, *r_wp* and *l_wp* 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 **LOGICAL** on Pentiums is a whole word, which is wasteful on memory, but usually faster).

```
"Precision.f90" 1 ≡
```

```

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 :: i_byte = byte_k, i_short = short_k, i_sp = int_k, i_dp = long_k,
    i_word = KIND(0)    // Integers
  INTEGER, PARAMETER, PUBLIC :: i_32 = i_sp, i_64 = i_dp    // 2's complement integers
  INTEGER, PARAMETER, PUBLIC :: r_sp = single_k, r_dp = double_k, r_qp = quad_k,
    r_word = KIND(0.0)  // Real numbers
  INTEGER, PARAMETER, PUBLIC :: l_short = l_byte_k, l_word = l_int_k    // Boolean values
  INTEGER, PARAMETER, PUBLIC :: r_wp = r_sp, i_wp = i_sp, l_wp = l_short
    // Working precisions
END MODULE Precision

```

```
[HPF2Formatting.hweb]
```

2 Formatting rules for HPF/F90 files

[HPF2Formatting.hweb] These are just same auxilliary 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 _DECLARE_LWORD(...)
    INTEGER :: #.
@m _DECLARE_LWP(...)
    INTEGER (KIND = i_wp) :: #.
@m _DECLARE_RWP(...)
    REAL (KIND = r_wp) :: #.
@m _DECLARE_RSP(...)
    REAL (KIND = r_sp) :: #.
@m _DECLARE_RDP(...)
    REAL (KIND = r_dp) :: #.
@m _FULLEXTENT(_rank) : $DO (DIM, 2, _rank) { , : }
@m _VARSEQUENCE(_variable, _start, _end)
    _variable##_start$DO (DIM, $EVAL(_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(...)

```