```
package TEXTIO is

  -- Type definitions for text I/O

      type LINE is access STRING;   -- a LINE is a pointer to a STRING value
      type TEXT is file of STRING;  -- A file of variable-length ASCII records
      type SIDE is (RIGHT, LEFT);   -- For justifying output data w/in fields
      subtype WIDTH is NATURAL;     -- For specifying widths of output fields

  -- Standard Text Files

      file  INPUT:      TEXT open READ_MODE  is "STD_INPUT";
      file  OUTPUT:     TEXT open WRITE_MODE is "STD_OUTPUT";

  -- Input Routines for Standard Types

      procedure READLINE (file F: TEXT; L: inout LINE);
      procedure READ (L: inout LINE; VALUE: out BIT; GOOD: out BOOLEAN);
      procedure READ (L: inout LINE; VALUE: out BIT);
      procedure READ (L: inout LINE; VALUE: out BIT_VECTOR; GOOD: out BOOLEAN);
      procedure READ (L: inout LINE; VALUE: out BIT_VECTOR);
      procedure READ (L: inout LINE; VALUE: out BOOLEAN; GOOD: out BOOLEAN);
      procedure READ (L: inout LINE; VALUE: out BOOLEAN);
      procedure READ (L: inout LINE; VALUE: out CHARACTER; GOOD: out BOOLEAN);
      procedure READ (L: inout LINE; VALUE: out CHARACTER);
      procedure READ (L: inout LINE; VALUE: out INTEGER; GOOD: out BOOLEAN);
      procedure READ (L: inout LINE; VALUE: out INTEGER);
      procedure READ (L: inout LINE; VALUE: out REAL; GOOD: out BOOLEAN);
      procedure READ (L: inout LINE; VALUE: out REAL);
      procedure READ (L: inout LINE; VALUE: out STRING; GOOD: out BOOLEAN);
      procedure READ (L: inout LINE; VALUE: out STRING);
      procedure READ (L: inout LINE; VALUE: out TIME; GOOD: out BOOLEAN);
      procedure READ (L: inout LINE; VALUE: out TIME);

  -- Output Routines for Standard Types

      procedure WRITELINE (FILE F:  TEXT; L:  inout LINE);
      procedure WRITE (L: inout LINE;    VALUE: in BIT;
                      JUSTIFIED: in SIDE:= RIGHT; FIELD: in WIDTH := 0);
      procedure WRITE (L: inout LINE;    VALUE: in BIT_VECTOR;
                      JUSTIFIED: in SIDE:= RIGHT; FIELD: in WIDTH := 0);
      procedure WRITE (L: inout LINE;    VALUE: in BOOLEAN;
                      JUSTIFIED: in SIDE:= RIGHT; FIELD: in WIDTH := 0);
      procedure WRITE (L: inout LINE;    VALUE: in CHARACTER;
                      JUSTIFIED: in SIDE:= RIGHT; FIELD: in WIDTH := 0);
      procedure WRITE (L: inout LINE;    VALUE: in INTEGER;
                      JUSTIFIED: in SIDE:= RIGHT; FIELD: in WIDTH := 0);
      procedure WRITE (L: inout LINE;    VALUE: in REAL;
                      JUSTIFIED: in SIDE:= RIGHT; FIELD: in WIDTH := 0;
                      DIGITS: in NATURAL:= 0);
      procedure WRITE (L: inout LINE;    VALUE: in STRING;
                      JUSTIFIED: in SIDE:= RIGHT; FIELD: in WIDTH := 0);
      procedure WRITE (L: inout LINE;    VALUE: in TIME;
                      JUSTIFIED: in SIDE:= RIGHT; FIELD: in WIDTH := 0;
                      UNIT: in TIME:= ns);

  -- File Position Predicates

--    function ENDFILE (file F: TEXT) return BOOLEAN;
end TEXTIO;
```

```
package STANDARD is

-- Predefined enumeration types
    type BOOLEAN is (FALSE, TRUE);
    type BIT is ('0', '1');
    type CHARACTER IS (
      NUL,  SOH,  STX,  ETX,  EOT,  ENQ,  ACK,  BEL,
      BS,   HT,   LF,   VT,   FF,   CR,   SO,   SI,
      DLE,  DC1,  DC2,  DC3,  DC4,  NAK,  SYN,  ETB,
      CAN,  EM,   SUB,  ESC,  FSP,  GSP,  RSP,  USP,

      ' ',  '!',  '"',  '#',  '$',  '%',  '&',  ''',
      '(',  ')',  '*',  '+',  ',',  '-',  '.',  '/',
      '0',  '1',  '2',  '3',  '4',  '5',  '6',  '7',
      '8',  '9',  ':',  ';',  '<',  '=',  '>',  '?',

      '@',  'A',  'B',  'C',  'D',  'E',  'F',  'G',
      'H',  'I',  'J',  'K',  'L',  'M',  'N',  'O',
      'P',  'Q',  'R',  'S',  'T',  'U',  'V',  'W',
      'X',  'Y',  'Z',' '[',  '\',  ']',  '^',  '_',

      '`',  'a',  'b',  'c',  'd',  'e',  'f',  'g',
      'h',  'i',  'j',  'k',  'l',  'm',  'n',  'o',
      'p',  'q',  'r',  's',  't',  'u',  'v',  'w',
      'x',  'y',  'z',' '{',  '|',  '}',  '~',  DEL,

      C128, C129, C130, C131, C132, C133, C134, C135,
      C136, C137, C138, C139, C140, C141, C142, C143,
      C144, C145, C146, C147, C148, C149, C150, C151,
      C152, C153, C154, C155, C156, C157, C158, C159,

      '?',  '?',  '?',  '?',  '?',  '?',  '?',  '?',
      '?',  '?',  '?',  '?',  '?',  '?',  '?',  '?',
      '?',  '?',  '?',  '?',  '?',  '?',  '?',  '?',
      '?',  '?',  '?',  '?',  '?',  '?',  '?',  '?',

      '?',  '?',  '?',  '?',  '?',  '?',  '?',  '?',
      '?',  '?',  '?',  '?',  '?',  '?',  '?',  '?',
      '?',  '?',  '?',  '?',  '?',  '?',  '?',  '?',
      '?',  '?',  '?',  '?',  '?',  '?',  '?',  '?',

      '?',  '?',  '?',  '?',  '?',  '?',  '?',  '?',
      '?',  '?',  '?',  '?',  '?',  '?',  '?',  '?',
      '?',  '?',  '?',  '?',  '?',  '?',  '?',  '?',
      '?',  '?',  '?',  '?',  '?',  '?',  '?',  '?');
    type SEVERITY_LEVEL is (NOTE, WARNING, ERROR, FAILURE);

-- Predefined numeric types
    type INTEGER is range -2147483648 to 2147483647;
    type REAL is range -1.7e38 to 1.7e38;

-- Predefined type TIME
    type TIME is range -9223372036854775808 to 9223372036854775807
        units
          fs;                 -- femtosecond
          ps      = 1000 fs;  -- picosecond
          ns      = 1000 ns;  -- nanosecond
          us      = 1000 ns;  -- microsecond
          ms      = 1000 us;  -- millisecond
          sec     = 1000 ms;  -- second
          min     = 60 sec;   -- minute
          hr      = 60 min;   -- hour
      end units;

    subtype DELAY_LENGTH is TIME range 0 fs to TIME'HIGH;
```

```
 -- A function that returns the current simulation time, Tc
    impure function NOW return DELAY_LENGTH;

-- Predefined numeric subtypes:
    subtype NATURAL is INTEGER range 0 to INTEGER'HIGH;
    subtype POSITIVE is INTEGER range 1 to INTEGER'HIGH;

-- Predefined array types:
    type STRING is array (POSITIVE range <>) of CHARACTER;
    type BIT_VECTOR is array (NATURAL range <>) of BIT;

-- The predefined types for opening files:
    type FILE_OPEN_KIND is (
      READ_MODE,  -- Resulting access mode is read-only.
      WRITE_MODE, -- Resulting access mode is write-only.
      APPEND_MODE); -- Resulting access mode is write-only; information
                   -- is appended to the end of the existing file.
    type FILE_OPEN_STATUS is (
      OPEN_OK,    -- File open was successful.
      STATUS_ERROR, -- File object was already open.
      NAME_ERROR, -- External file not found or inaccessible.
      MODE_ERROR); -- Could not open file with requested access mode.

-- The 'FOREIGN attribute:
    attribute FOREIGN: STRING;

end STANDARD;
```

Title : std_logic_1164 multi-value logic system
-- Library : This package shall be compiled into a library symbolically named IEEE.
-- Developers: IEEE model standards group (par 1164)
-- Purpose : This packages defines a standard for designers to use in describing the interconnection data types
-- : used in vhdl modeling.
-- Limitation: The logic system defined in this package may be insufficient for modeling switched transistors,
-- : since such a requirement is out of the scope of this effort. Furthermore, mathematics, primitives,
-- : timing standards, etc. are considered orthogonal issues as it relates to this package and are therefore
-- : beyond the scope of this effort.
-- Note : No declarations or definitions shall be included in, or excluded from this package. The "package
-- declaration” defines the types, subtypes and declarations of std_logic_1164. The std_logic_1164 package body
-- shall be considered the formal definition of the semantics of this package. Tool developers may choose to
-- implement the package body in the most efficient manner available to them.
-- ---
-- modification history :
-- version | mod. date:|
-- v4.200 | 01/02/92 |

PACKAGE std_logic_1164 IS

```
    -- logic state system  (unresolved)
    -------------------------------------------------------------------
    TYPE std_ulogic IS ( 'U',  -- Uninitialized
                         'X',  -- Forcing  Unknown
                         '0',  -- Forcing  0
                         '1',  -- Forcing  1
                         'Z',  -- High Impedance
                         'W',  -- Weak     Unknown
                         'L',  -- Weak     0
                         'H',  -- Weak     1
                         '-'   -- Don't care
                       );
    -------------------------------------------------------------------
    -- unconstrained array of std_ulogic for use with the resolution function
    -------------------------------------------------------------------
    TYPE std_ulogic_vector IS ARRAY ( NATURAL RANGE <> ) OF std_ulogic;
    -------------------------------------------------------------------
    -- resolution function
    -------------------------------------------------------------------
    FUNCTION resolved ( s : std_ulogic_vector ) RETURN std_ulogic;
    -------------------------------------------------------------------
    -- *** industry standard logic type ***
    -------------------------------------------------------------------
    SUBTYPE std_logic IS resolved std_ulogic;
    -------------------------------------------------------------------
    -- unconstrained array of std_logic for use in declaring signal arrays
    -------------------------------------------------------------------
    TYPE std_logic_vector IS ARRAY ( NATURAL RANGE <>) OF std_logic;
    -------------------------------------------------------------------
    -- common subtypes
    -------------------------------------------------------------------
SUBTYPE X01  IS resolved std_ulogic RANGE 'X' TO '1'; -- ('X','0','1')
SUBTYPE X01Z IS resolved std_ulogic RANGE 'X' TO 'Z'; --('X','0','1','Z')
SUBTYPE UX01 IS resolved std_ulogic RANGE 'U' TO '1'; --('U','X','0','1')
SUBTYPE UX01Z IS resolved std_ulogic RANGE 'U' TO 'Z';--('U','X','0','1','Z')
    -------------------------------------------------------------------
    -- overloaded logical operators
    -------------------------------------------------------------------
    FUNCTION "and"  ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
    FUNCTION "nand" ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
    FUNCTION "or"   ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
    FUNCTION "nor"  ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
    FUNCTION "xor"  ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
    FUNCTION "xnor" ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
    FUNCTION "not"  ( l : std_ulogic                 ) RETURN UX01;
     -------------------------------------------------------------------
    -- vectorized overloaded logical operators
    -------------------------------------------------------------------
    FUNCTION "and"  ( l, r : std_logic_vector  ) RETURN std_logic_vector;
    FUNCTION "and"  ( l, r : std_ulogic_vector ) RETURN std_ulogic_vector;
    FUNCTION "nand" ( l, r : std_logic_vector  ) RETURN std_logic_vector;
    FUNCTION "nand" ( l, r : std_ulogic_vector ) RETURN std_ulogic_vector;
    FUNCTION "or"   ( l, r : std_logic_vector  ) RETURN std_logic_vector;
    FUNCTION "or"   ( l, r : std_ulogic_vector ) RETURN std_ulogic_vector;
    FUNCTION "nor"  ( l, r : std_logic_vector  ) RETURN std_logic_vector;
    FUNCTION "nor"  ( l, r : std_ulogic_vector ) RETURN std_ulogic_vector;
    FUNCTION "xor"  ( l, r : std_logic_vector  ) RETURN std_logic_vector;
    FUNCTION "xor"  ( l, r : std_ulogic_vector ) RETURN std_ulogic_vector;
    FUNCTION "xnor" ( l, r : std_logic_vector  ) RETURN std_logic_vector;
    FUNCTION "xnor" ( l, r : std_ulogic_vector ) RETURN std_ulogic_vector;
    FUNCTION "not"  ( l : std_logic_vector  ) RETURN std_logic_vector;
    FUNCTION "not"  ( l : std_ulogic_vector ) RETURN std_ulogic_vector;
    -------------------------------------------------------------------
    -- conversion functions
    -------------------------------------------------------------------
```

```
  FUNCTION To_bit (s: std_ulogic;        xmap : BIT := '0') RETURN BIT;
  FUNCTION To_bitvector (s: std_logic_vector; xmap: BIT := '0') RETURN BIT_VECTOR;
  FUNCTION To_bitvector (s: std_ulogic_vector; xmap: BIT := '0') RETURN BIT_VECTOR;
  FUNCTION To_StdULogic (b: BIT                ) RETURN std_ulogic;
  FUNCTION To_StdLogicVector (b: BIT_VECTOR     ) RETURN std_logic_vector;
  FUNCTION To_StdLogicVector (s: std_ulogic_vector) RETURN std_logic_vector;
  FUNCTION To_StdULogicVector (b: BIT_VECTOR     ) RETURN std_ulogic_vector;
  FUNCTION To_StdULogicVector (s: std_logic_vector) RETURN std_ulogic_vector;
    -------------------------------------------------------------------
    -- attributes for conversion functions
    -------------------------------------------------------------------
    attribute Synth_Conversion_Function: STRING;
    attribute Synth_Conversion_Function of To_bit: function is "BIT";
    attribute Synth_Conversion_Function of To_bitvector: function is "BIT_VECTOR";
    attribute Synth_Conversion_Function of To_StdULogic: function is "std_ulogic";
attribute Synth_Conversion_Function of To_StdLogicVector: function is "std_logic_vector";
attribute Synth_Conversion_Function of To_StdULogicVector: function is "std_ulogic_vector";
    -------------------------------------------------------------------
    -- strength strippers and type convertors
    -------------------------------------------------------------------
    FUNCTION To_X01  ( s : std_logic_vector  ) RETURN  std_logic_vector;
    FUNCTION To_X01  ( s : std_ulogic_vector ) RETURN  std_ulogic_vector;
    FUNCTION To_X01  ( s : std_ulogic        ) RETURN  X01;
    FUNCTION To_X01  ( b : BIT_VECTOR        ) RETURN  std_logic_vector;
    FUNCTION To_X01  ( b : BIT_VECTOR        ) RETURN  std_ulogic_vector;
    FUNCTION To_X01  ( b : BIT               ) RETURN  X01;
    FUNCTION To_X01Z ( s : std_logic_vector  ) RETURN  std_logic_vector;
    FUNCTION To_X01Z ( s : std_ulogic_vector ) RETURN  std_ulogic_vector;
    FUNCTION To_X01Z ( s : std_ulogic        ) RETURN  X01Z;
    FUNCTION To_X01Z ( b : BIT_VECTOR        ) RETURN  std_logic_vector;
    FUNCTION To_X01Z ( b : BIT_VECTOR        ) RETURN  std_ulogic_vector;
    FUNCTION To_X01Z ( b : BIT               ) RETURN  X01Z;
    FUNCTION To_UX01 ( s : std_logic_vector  ) RETURN  std_logic_vector;
    FUNCTION To_UX01 ( s : std_ulogic_vector ) RETURN  std_ulogic_vector;
    FUNCTION To_UX01 ( s : std_ulogic        ) RETURN  UX01;
    FUNCTION To_UX01 ( b : BIT_VECTOR        ) RETURN  std_logic_vector;
    FUNCTION To_UX01 ( b : BIT_VECTOR        ) RETURN  std_ulogic_vector;
    FUNCTION To_UX01 ( b : BIT               ) RETURN  UX01;
    -------------------------------------------------------------------
    -- edge detection
    -------------------------------------------------------------------
    FUNCTION rising_edge  (SIGNAL s : std_ulogic) RETURN BOOLEAN;
    FUNCTION falling_edge (SIGNAL s : std_ulogic) RETURN BOOLEAN;
    -------------------------------------------------------------------
    -- object contains an unknown
    -------------------------------------------------------------------
    FUNCTION Is_X ( s : std_ulogic_vector ) RETURN  BOOLEAN;
    FUNCTION Is_X ( s : std_logic_vector  ) RETURN  BOOLEAN;
    FUNCTION Is_X ( s : std_ulogic        ) RETURN  BOOLEAN;
```

END std_logic_1164;