```
 CCCC  N    N  V    V  EEEEE  N    N  TTTTT
C      N    N  V    V  E      N    N    T
C      NN   N  V    V  E      NN   N    T
C      N N  N  V    V  EEEE   N N  N    T
C      N   NN  V    V  E      N   NN    T
C      N    N   V V   E      N    N    T
 CCCC  N    N    V    EEEEE  N    N    T
```

```
 AAA   DDDD    AAA          1     222    333
A   A  D   D  A   A        11    2   2  3   3
A   A  D   D  A   A         1        2      3
A   A  D   D  A   A         1       2      3
AAAAA  D   D  AAAAA        1       2        3
A   A  D   D  A   A  ..     1      2     3   3
A   A  DDDD   A   A  ..    111    22222   333
```

```
*START* Job CNVENT Req #204 for DHB    Date 27-Feb-82 16:12:04 Monitor: Rational
File PS:<SIM.DEF>CNVENT.ADA.123, created: 25-Feb-82 17:36:20
        printed: 27-Feb-82 16:12:04
Job parameters: Request created:27-Feb-82 16:12:02   Page limit:27   Forms:NORMAL
File parameters: Copy: 1 of 1   Spacing:SINGLE   File format:ASCII   Print mode:AS
```

```
with BASE,
     CODE,
     TASKS,
     STACKS,
     OFFSET,
     SEGMENTS,
     EXCEPTIONS;

package CONVENTION is

    NULL_SEGMENT : constant SEGMENTS.NAME := 0;

    NULL_ADDRESS : constant SEGMENTS.ADDRESS := SEGMENTS.ADDRESS'(0, 0, 0);

    NULL_STACK   : constant STACKS.NAME := STACKS.NAME'(PROCESSOR => 0,
                                                        NUMBER    => 0);

    NULL_IMPORT_SPACE  : constant STACKS.IMPORT_NAME
                               := STACKS.IMPORT_NAME'(PROCESSOR => 0,
                                                      NUMBER    => 0);

    NULL_COLLECTION_ID : constant STACKS.COLLECTION_ID
                               := STACKS.COLLECTION_ID'FIRST;

    NULL_COLLECTION    : constant STACKS.NAME
                               := STACKS.NAME'
                                   (PROCESSOR => 0,
                                    NUMBER    => NULL_COLLECTION_ID);

    NULL_CONTROL_REF   : constant STACKS.CONTROL_REFERENCE
                               := STACKS.CONTROL_REFERENCE'
                                       (STACK  => NULL_STACK,
                                        OFFSET => 0);

    NULL_TYPE_REF      : constant STACKS.TYPE_REFERENCE
                               := STACKS.TYPE_REFERENCE'
                                       (STACK  => NULL_STACK,
                                        OFFSET => 0);

    NULL_DATA_REF      : constant STACKS.DATA_REFERENCE
                               := STACKS.DATA_REFERENCE'
                                       (STACK  => NULL_STACK,
                                        OFFSET => 0);

    NULL_DISCRETE      : constant STACKS.VALUE
                               := STACKS.VALUE'
                                       (OF_KIND      => STACKS.DISCRETE_VAR,
                                        DISCRETE_VAL => BASE.DISCRETE'FIRST);

    NULL_FLOAT         : constant STACKS.VALUE
                               := STACKS.VALUE'
                                       (OF_KIND  => STACKS.FLOAT_VAR,
                                        REAL_VAL => BASE.REAL'FIRST);

    NULL_ACCESS_VALUE : constant STACKS.VALUE
                               := STACKS.VALUE'
                                       (OF_KIND   => STACKS.ACCESS_VAR,
                                        POINTER   => 0,
                                        ALLOCATOR => 0);
```

```
NULL_MODULE_VALUE : constant STACKS.VALUE
                          := STACKS.VALUE'
                                  (OF_KIND => STACKS.MODULE_VAR,
                                   STACK   => NULL_STACK);
```

```
NO_CHILDREN    : constant STACKS.CHILD_POINTER
                              := STACKS.CHILD_POINTER'(0,0);

OVERFLOW_SIZE  : constant STACKS.BIT_COUNT := STACKS.BIT_COUNT'LAST;

NULL_VARIANT_INDEX   : constant STACKS.VARIANT_CLAUSE_INDEX := 0;


NULL_COMPONENT_INFO : constant STACKS.COMPONENT_INFO :=
                              STACKS.COMPONENT_INFO'
                                      (HAS_MODULE  => FALSE,
                                       HAS_ACCESS  => FALSE,
                                       HAS_SEGMENT => FALSE,
                                       HAS_VARIANT => FALSE);

NULL_TYPE_PROTECTION : constant STACKS.TYPE_PROTECTION
                              := STACKS.TYPE_PROTECTION'
                                      (PRIVACY => CODE.IS_LOCAL,
                                       DERIVES_PRIVACY => FALSE);

OPEN_TYPE_PROTECTION : constant STACKS.TYPE_PROTECTION
                              := STACKS.TYPE_PROTECTION'
                                      (PRIVACY => CODE.IS_PUBLIC,
                                       DERIVES_PRIVACY => FALSE);

NULL_VARIABLE_PROTECTION : constant STACKS.VARIABLE_PROTECTION
                              := STACKS.VARIABLE_PROTECTION'
                                      (VISIBILITY  => CODE.IS_HIDDEN,
                                       IS_CONSTANT => FALSE);

OPEN_VARIABLE_PROTECTION : constant STACKS.VARIABLE_PROTECTION
                              := STACKS.VARIABLE_PROTECTION'
                                      (VISIBILITY  => CODE.IS_VISIBLE,
                                       IS_CONSTANT => FALSE);

NULL_OBJECT_TYPE : constant STACKS.TYPE_LINK
                              := STACKS.TYPE_LINK'
                                      (OF_KIND  => STACKS.FLOAT_VAR,
                                       PATH     => NULL_TYPE_REF,
                                       FOR_TYPE => NULL_TYPE_PROTECTION,
                                       FOR_VAR  => NULL_VARIABLE_PROTECTION);


NULL_SUBPROG   : constant STACKS.CONTROL_WORD
                              := STACKS.CONTROL_WORD'
                                  (OF_KIND       => STACKS.SUBPROGRAM_VAR,
                                   SUBPROG_STATE => STACKS.IS_ACTIVE,
                                   SUBPROG_SORT  => CODE.FOR_OPERATION,
                                   DECLARE_FRAME => NULL_CONTROL_REF,
                                   LEX_LEVEL     => 0,
                                   FIRST_INSERT  => CODE.NO_OP,
                                   CODE_SEGMENT  => 0,
                                   BLOCK_BEGIN   => 0,
                                   ACCEPT_ENTRY  => 0);
```

```
NO_ERROR : constant STACKS.CONTROL_WORD := STACKS.CONTROL_WORD'
                                  (OF_KIND         => STACKS.EXCEPTION_VAR,
                                   EXCEPTION_VAL   => EXCEPTIONS.NO_ERROR,
                                   RAISED_ADDRESS  => NULL_ADDRESS,
                                   RAISED_SCOPE    => NULL_STACK);

EMPTY_SCOPE  : constant STACKS.NAME := STACKS.NAME'(PROCESSOR => 15,
                                                    NUMBER    =>  6);

ALL_SCOPE    : constant STACKS.NAME := STACKS.NAME'(PROCESSOR => 15,
                                                    NUMBER    =>  3);

MAIN_STACK   : constant STACKS.NAME := STACKS.NAME'(PROCESSOR =>  1,
                                                    NUMBER    =>  0);

IN_OUT_STACK : constant STACKS.NAME := STACKS.NAME'(PROCESSOR =>  1,
                                                    NUMBER    =>  2);

INITIAL_STATE       : constant STACKS.CONTROL_WORD
                         := STACKS.CONTROL_WORD'
                              (OF_KIND           => STACKS.MODULE_STATE,
                               ELABORATION       => TASKS.DECLARING,
                               BLOCKED_STATE     => TASKS.UNBLOCKED,
                               QUEUE_NEIGHBOR    => NULL_STACK,
                               CONTROL_TOS       => 0,
                               CONTROL_EXTENT    => 0,
                               PROGRAM_COUNTER   => NULL_ADDRESS,
                               NEXT_INSTRUCTION  => CODE.NO_OP);

INITIAL_ALLOCATION : constant STACKS.CONTROL_WORD
                         := STACKS.CONTROL_WORD'
                              (OF_KIND      => STACKS.ALLOCATION_STATE,
                               INNER_FRAME  => 0,
                               TYPE_TOS     => 0,
                               TYPE_EXTENT  => 0,
                               DATA_TOS     => NULL_DATA_REF,
                               HAS_AUXILIARY_MARK => FALSE);

INITIAL_RESOURCES  : constant STACKS.CONTROL_WORD
                   := STACKS.CONTROL_WORD'
                        (OF_KIND      => STACKS.RESOURCE_LIMITS,
                         DATA_EXTENT  => 0,
                         MAX_CONTROL  => STACKS.CONTROL_DISPLACEMENT'LAST,
                         MAX_DATA     => STACKS.WORD_DISPLACEMENT'LAST,
                         MAX_TYPE     => STACKS.TYPE_DISPLACEMENT'LAST);

INITIAL_MICRO_STATE : constant STACKS.CONTROL_WORD
                   := STACKS.CONTROL_WORD'(OF_KIND => STACKS.MICRO_STATE);

INITIAL_SCHEDULING  : constant STACKS.CONTROL_WORD
                   := STACKS.CONTROL_WORD'
                        (OF_KIND           => STACKS.SCHEDULING_STATE,
                         SCHEDULING_GROUP  => 0,
                         BASE_TIME_SLICE   => TASKS.SLICE_TIME'LAST,
                         CURRENT_SLICE     => TASKS.SLICE_TIME'LAST,
                         INTERSLICE_DELAY  => 0,
                         USED_SLICE_COUNT  => 0,
                         PAGE_FAULT_COUNT  => 0);
```

```
INITIAL_DEBUGGING  : constant STACKS.CONTROL_WORD
                   := STACKS.CONTROL_WORD'
                       (OF_KIND          => STACKS.DEBUGGING_STATE,
                        SCOPE_FOR_DEBUG  => EMPTY_SCOPE,
                        DEBUG_ENABLE     => TASKS.BREAK_ENABLE'
                                              (others => FALSE));

INITIAL_CONNECTION : constant STACKS.CONTROL_WORD
                   := STACKS.CONTROL_WORD'
                       (OF_KIND         => STACKS.STATIC_CONNECTION,
                        MODULE_IMPORTS  => NULL_IMPORT_SPACE,
                        MODULE_TYPE     => NULL_TYPE_REF);

INITIAL_DEPENDENCE : constant STACKS.CONTROL_WORD
                   := STACKS.CONTROL_WORD'
                       (OF_KIND             => STACKS.DEPENDENCE_LINK,
                        DECLARER            => NULL_STACK,
                        DEPEND_FRAME        => NULL_TYPE_REF,
                        LAST_ACTIVATE       => NO_CHILDREN,
                        HAS_VISIBLE_CHILD   => FALSE);


EMPTY_STATE  : constant STACKS.CONTROL_WORD
                        := STACKS.CONTROL_WORD'
                            (OF_KIND          => STACKS.ACTIVATION_STATE,
                             OUTER_FRAME      => MAIN_STACK,
                             ENCLOSING_FRAME  => (MAIN_STACK, 0),
                             CONTROL_PRED     => 0,
                             CURRENT_LEX      => 0,
                             RETURN_INSERT    => CODE.NO_OP,
                             IN_ACCEPT_BODY   => FALSE,
                             IN_UTILITY       => FALSE,
                             HAS_CHILDREN     => FALSE);

EMPTY_LINK   : constant STACKS.CONTROL_WORD
                        := STACKS.CONTROL_WORD'
                            (OF_KIND          => STACKS.ACTIVATION_LINK,
                             TYPE_FRAME       => 0,
                             DATA_FRAME       => NULL_DATA_REF,
                             BLOCK_START      => 0,
                             RETURN_ADDRESS   => NULL_ADDRESS,
                             FRAME_MARKED     => FALSE);

EMPTY_ACCEPT : constant STACKS.CONTROL_WORD
                        := STACKS.CONTROL_WORD'
                            (OF_KIND          => STACKS.ACCEPT_LINK,
                             IN_RENDEZVOUS    => NULL_STACK,
                             FAMILY_INDEX     => 0,
                             RENDEZVOUS_REF   => 0,
                             ENTRY_REF        => 0,
                             PRIORITY         => 1,
                             IN_SELECT        => FALSE,
                             SELECT_CHOICE    => 0);
```

```
STANDARD_STACK    : constant STACKS.NAME := STACKS.NAME'(PROCESSOR => 1,
                                                         NUMBER    => 1);

STANDARD_OFFSET   : constant STACKS.TYPE_DISPLACEMENT := 2;

STANDARD_SEGMENT  : constant SEGMENTS.NAME := 0;

STANDARD_TYPE     : constant STACKS.TYPE_LINK
                     := STACKS.TYPE_LINK'
                             (OF_KIND  => STACKS.MODULE_VAR,
                              PATH     => STACKS.TYPE_REFERENCE'
                                           (STACK  => MAIN_STACK,
                                            OFFSET => STANDARD_OFFSET),
                              FOR_TYPE => OPEN_TYPE_PROTECTION,
                              FOR_VAR  => STACKS.VARIABLE_PROTECTION'
                                           (VISIBILITY  => CODE.IS_VISIBLE,
                                            IS_CONSTANT => TRUE));

PACKAGE_STANDARD : constant STACKS.CONTROL_WORD
                            := STACKS.CONTROL_WORD'
                                    (OF_KIND    => STACKS.MODULE_VAR,
                                     VALUE_ITEM => STACKS.VALUE'
                                             (OF_KIND => STACKS.MODULE_VAR,
                                              STACK   => STANDARD_STACK),
                                     VALUE_TYPE => STANDARD_TYPE);

STANDARD_BOOLEAN  : constant STACKS.TYPE_LINK
                         := STACKS.TYPE_LINK'
                             (OF_KIND  => STACKS.DISCRETE_VAR,
                              PATH     => STACKS.TYPE_REFERENCE'
                                     (STACK  => STANDARD_STACK,
                                      OFFSET => STACKS.TYPE_DISPLACEMENT
                                             (OFFSET.ON_TYPE.FOR_BOOLEAN)),
                              FOR_TYPE  => OPEN_TYPE_PROTECTION,
                              FOR_VAR   => OPEN_VARIABLE_PROTECTION,
                              IS_SIGNED => FALSE);

STANDARD_INTEGER : constant STACKS.TYPE_LINK
                         := STACKS.TYPE_LINK'
                             (OF_KIND   => STACKS.DISCRETE_VAR,
                              PATH      => STACKS.TYPE_REFERENCE'
                                     (STACK  => STANDARD_STACK,
                                      OFFSET => STACKS.TYPE_DISPLACEMENT
                                             (OFFSET.ON_TYPE.FOR_INTEGER)),
                              FOR_TYPE  => OPEN_TYPE_PROTECTION,
                              FOR_VAR   => OPEN_VARIABLE_PROTECTION,
                              IS_SIGNED => TRUE);
```

```
   STANDARD_FLOAT    : constant STACKS.TYPE_LINK
                            := STACKS.TYPE_LINK'
                                (OF_KIND  => STACKS.FLOAT_VAR,
                                 PATH     => STACKS.TYPE_REFERENCE'
                                        (STACK  => STANDARD_STACK,
                                         OFFSET => STACKS.TYPE_DISPLACEMENT
                                                   (OFFSET.ON_TYPE.FOR_FLOAT)),
                                 FOR_TYPE => OPEN_TYPE_PROTECTION,
                                 FOR_VAR  => OPEN_VARIABLE_PROTECTION);

   STANDARD_CHARACTER : constant STACKS.TYPE_LINK
                            := STACKS.TYPE_LINK'
                                (OF_KIND   => STACKS.DISCRETE_VAR,
                                 PATH      => STACKS.TYPE_REFERENCE'
                                        (STACK  => STANDARD_STACK,
                                         OFFSET => STACKS.TYPE_DISPLACEMENT
                                              (OFFSET.ON_TYPE.FOR_CHARACTER)),
                                 FOR_TYPE  => OPEN_TYPE_PROTECTION,
                                 FOR_VAR   => OPEN_VARIABLE_PROTECTION,
                                 IS_SIGNED => FALSE);

   STANDARD_STRING   : constant STACKS.TYPE_LINK
                            := STACKS.TYPE_LINK'
                                (OF_KIND     => STACKS.ARRAY_VAR,
                                 PATH        => STACKS.TYPE_REFERENCE'
                                        (STACK  => STANDARD_STACK,
                                         OFFSET => STACKS.TYPE_DISPLACEMENT
                                                   (OFFSET.ON_TYPE.FOR_STRING)),
                                 FOR_TYPE    => OPEN_TYPE_PROTECTION,
                                 FOR_VAR     => OPEN_VARIABLE_PROTECTION,
                                 BOUNDS_SITE => STACKS.WITH_OBJECT);


   LITERAL_DISCRETE : constant STACKS.TYPE_LINK
                            := STACKS.TYPE_LINK'
                                (OF_KIND     => STACKS.DISCRETE_VAR,
                                 PATH        => NULL_TYPE_REF,
                                 FOR_TYPE    => NULL_TYPE_PROTECTION,
                                 FOR_VAR     => NULL_VARIABLE_PROTECTION,
                                 IS_SIGNED   => TRUE);

   LITERAL_FLOAT : constant STACKS.TYPE_LINK
                            := STACKS.TYPE_LINK'
                                (OF_KIND     => STACKS.FLOAT_VAR,
                                 PATH        => NULL_TYPE_REF,
                                 FOR_TYPE    => NULL_TYPE_PROTECTION,
                                 FOR_VAR     => NULL_VARIABLE_PROTECTION);

end CONVENTION;

package body CONVENTION is end CONVENTION;
```

```
DDDDDDDD        SSSSSSSS      CCCCCCCC    RRRRRRRR      PPPPPPPP
DDDDDDDD        SSSSSSSS      CCCCCCCC    RRRRRRRR      PPPPPPPP
DD      DD      SS            CC          RR      RR    PP      PP
DD      DD      SS            CC          RR      RR    PP      PP
DD      DD      SS            CC          RR      RR    PP      PP
DD      DD      SS            CC          RR      RR    PP      PP
DD      DD        SSSSSS      CC          RRRRRRRR      PPPPPPPP
DD      DD        SSSSSS      CC          RRRRRRRR      PPPPPPPP
DD      DD              SS    CC          RR  RR        PP
DD      DD              SS    CC          RR  RR        PP
DD      DD              SS    CC          RR    RR      PP
DD      DD              SS    CC          RR    RR      PP
DDDDDDDD        SSSSSSSS      CCCCCCCC    RR      RR    PP
DDDDDDDD        SSSSSSSS      CCCCCCCC    RR      RR    PP

  AAAAAA        DDDDDDDD        AAAAAA                    333333
  AAAAAA        DDDDDDDD        AAAAAA                    333333
AA      AA      DD      DD    AA      AA                33      33
AA      AA      DD      DD    AA      AA                33      33
AA      AA      DD      DD    AA      AA                        33
AA      AA      DD      DD    AA      AA                        33
AA      AA      DD      DD    AA      AA                    33
AA      AA      DD      DD    AA      AA                    33
AAAAAAAAAA      DD      DD    AAAAAAAAAA                      33
AAAAAAAAAA      DD      DD    AAAAAAAAAA                      33
AA      AA      DD      DD    AA      AA        ....    33      33
AA      AA      DD      DD    AA      AA        ....    33      33
AA      AA      DDDDDDDD      AA      AA        ....      333333
AA      AA      DDDDDDDD      AA      AA        ....      333333
```

```
*START* Job DSCRPT Req #205 for DHB     Date 27-Feb-82 16:13:19 Monitor: Rational
File PS:<SIM.DEF>DSCRPT.ADA.38, created:  6-Feb-82  2:07:09
        printed: 27-Feb-82 16:13:29
Job parameters: Request created:27-Feb-82 16:12:37   Page limit:9   Forms:NORMAL
File parameters: Copy: 1 of 1   Spacing:SINGLE   File format:ASCII   Print mode:AS
```

```
with CODE,
     STACKS;

ckage DESCRIPTOR is

     use CODE;

     type FOR_SCALAR is
          record
               UTIL    : STACKS.TYPE_WORD (OF_KIND => STACKS.TYPE_UTILITY);    -- -2
               TYPE_INFO : STACKS.TYPE_WORD  (OF_KIND => STACKS.TYPE_INFO);    -- -1
               BOUNDS : STACKS.TYPE_WORD (OF_KIND => STACKS.SCALAR_BOUNDS);    --  0
          end record;


     type FOR_MODULE is
          record
               UTIL : STACKS.TYPE_WORD (OF_KIND => STACKS.TYPE_UTILITY);       -- -2
               TYPE_INFO : STACKS.TYPE_WORD (OF_KIND => STACKS.TYPE_INFO);     -- -1
               INFO : STACKS.TYPE_WORD (OF_KIND => STACKS.MODULE_INFO);        --  0
          end record;


     type FOR_ACCESS is
          record
               UTIL  : STACKS.TYPE_WORD (OF_KIND => STACKS.TYPE_UTILITY);      -- -2
               TYPE_INFO : STACKS.TYPE_WORD  (OF_KIND => STACKS.TYPE_INFO);    -- -1
               INFO  : STACKS.TYPE_WORD  (OF_KIND => STACKS.ACCESS_INFO);      -- 0
               FRAME : STACKS.TYPE_WORD  (OF_KIND => STACKS.ACCESS_FRAME);     -- 1
          end record;


     type FOR_ARRAY (DIMENSIONS : STACKS.DIMENSIONALITY) is
          record
               UTIL     : STACKS.TYPE_WORD (OF_KIND => STACKS.TYPE_UTILITY); -- -2
               TYPE_INFO : STACKS.TYPE_WORD  (OF_KIND => STACKS.TYPE_INFO);  -- -1
               INFO     : STACKS.TYPE_WORD  (OF_KIND => STACKS.ARRAY_INFO);  --  0
               INDEX_N : array (1 .. DIMENSIONS) of STACKS.TYPE_WORD
                                 (OF_KIND => STACKS.ARRAY_INDEX_INFO);       --1..N
          end record;


     type FOR_RECORD (FIELD_COUNT : CODE.FIELD_INDEX) is
          record
               UTIL  : STACKS.TYPE_WORD (OF_KIND => STACKS.TYPE_UTILITY);    -- -2
               TYPE_INFO : STACKS.TYPE_WORD (OF_KIND => STACKS.TYPE_INFO);   -- -1
               FIELD : array (0 .. FIELD_COUNT - 1) of STACKS.TYPE_WORD
                                        (OF_KIND => STACKS.FIELD_INFO);--0..N-1
          end record;
```

```
    type FOR_UNCONSTRAINED_VARIANT_RECORD (DISCRIM_COUNT : CODE.FIELD_INDEX;
                                           FIXED_COUNT    : CODE.FIELD_INDEX;
                                           VARIANT_COUNT : CODE.FIELD_INDEX) is
        record
            UTIL   : STACKS.TYPE_WORD (OF_KIND => STACKS.TYPE_UTILITY);    -- -2
            TYPE_INFO : STACKS.TYPE_WORD  (OF_KIND => STACKS.TYPE_INFO); -- -1
            INFO   : STACKS.TYPE_WORD  (OF_KIND => STACKS.VARIANT_INFO);   --0
            FIELD : array (1 .. DISCRIM_COUNT + FIXED_COUNT + VARIANT_COUNT)
                                    of STACKS.TYPE_WORD
                                        (OF_KIND => STACKS.FIELD_INFO);
        end record;


    type FOR_CONSTRAINED_VARIANT_RECORD (DISCRIM_COUNT : CODE.FIELD_INDEX;
                                         FIXED_COUNT    : CODE.FIELD_INDEX;
                                         VARIANT_COUNT : CODE.FIELD_INDEX) is
        record
            DEFAULTS : array (((-5) - DISCRIM_COUNT) / 2 .. -3)
                                of STACKS.TYPE_WORD
                                        (OF_KIND => STACKS.DEFAULT_DISCRIMS);
            TAIL     : FOR_UNCONSTRAINED_VARIANT_RECORD;
        end record;

end DESCRIPTOR;

package body DESCRIPTOR is end DESCRIPTOR;
```