```
 SSSS  TTTTT   AAA    CCCC  K   K   SSSS
S        T    A   A  C      K   K  S
         T    A   A  C      K  K   S
 SS      T    A   A  C      KKK     SSS
   S     T    AAAAA  C      K  K       S
   S     T    A   A  C      K   K      S
SSSS     T    A   A   CCCC  K   K  SSSS
```

```
 AAA   DDDD    AAA          4   4    333    888
A   A  D   D  A   A         4   4   3   3  8   8
A   A  D   D  A   A         4   4       3  8   8
A   A  D   D  A   A         44444     3    888
AAAAA  D   D  AAAAA            4       3  8   8
A   A  D   D  A   A     ..     4   3   3  8   8
A   A  DDDD   A   A     ..     4    333    888
```

```
*START* Job STACKS Req #185 for DHB     Date 27-Feb-82 15:35:25 Monitor: Rational
File PS:<SIM.DEF>STACKS.ADA.438, created: 26-Feb-82 17:36:00
        printed: 27-Feb-82 15:35:26
Job parameters: Request created:27-Feb-82 15:35:25   Page limit:36   Forms:NORMAL
File parameters: Copy: 1 of 1   Spacing:SINGLE   File format:ASCII   Print mode:A
```

```
with BASE,
     CODE,
     TASKS,
     SEGMENTS,
     EXCEPTIONS,
     PROCESSORS;

package STACKS is

    type ID  is new INTEGER  range 0 .. 2 ** 24 - 1;

    subtype MODULE_ID     is ID range ID'FIRST .. 2 ** 23 - 1;
    subtype COLLECTION_ID is ID range 2 ** 23  .. ID'LAST;


    type NAME is
        record
            PROCESSOR : PROCESSORS.ID;
            NUMBER    : ID;
        end record;


    type CONTROL_DISPLACEMENT is new INTEGER range 0 .. 2 ** 20 - 1;

    type CONTROL_REFERENCE is
        record
            STACK  : NAME;
            OFFSET : CONTROL_DISPLACEMENT;
        end record;

    type CONTROL_DELTA     is new INTEGER range - 2 ** 20 + 1 .. 2 ** 20 - 1;


    STATE_OFFSET     : constant CONTROL_DISPLACEMENT :=  0;
    ALLOC_OFFSET     : constant CONTROL_DISPLACEMENT :=  1;
    RESOURCE_OFFSET  : constant CONTROL_DISPLACEMENT :=  2;
    MICRO_OFFSET     : constant CONTROL_DISPLACEMENT :=  3;
    SCHEDULE_OFFSET  : constant CONTROL_DISPLACEMENT :=  4;
    DEBUG_OFFSET     : constant CONTROL_DISPLACEMENT :=  5;
    SUBPROG_OFFSET   : constant CONTROL_DISPLACEMENT :=  6;
    STATIC_OFFSET    : constant CONTROL_DISPLACEMENT :=  7;
    DEPEND_OFFSET    : constant CONTROL_DISPLACEMENT :=  8;
    ACCEPT_OFFSET    : constant CONTROL_DISPLACEMENT :=  9;
    OUTER_STATE      : constant CONTROL_DISPLACEMENT := 10;
    OUTER_LINK       : constant CONTROL_DISPLACEMENT := 11;
    BEGIN_OFFSET     : constant CONTROL_DISPLACEMENT := 12;
```

```
type TYPE_DISPLACEMENT is new INTEGER range 0 .. 2 ** 20 - 1;

type TYPE_REFERENCE is
    record
        STACK  : NAME;
        OFFSET : TYPE_DISPLACEMENT;
    end record;


type DATA_DISPLACEMENT is new INTEGER range 0 .. 2 ** 32 - 1;

type WORD_DISPLACEMENT is new INTEGER range 0 .. 2 ** 25 - 1;

type DATA_REFERENCE is
    record
        STACK  : NAME;
        OFFSET : DATA_DISPLACEMENT;
    end record;


type WORD_INDEX is new INTEGER range 0 .. 2 **  2 - 1;

type CHILD_POINTER is
    record
        OFFSET : TYPE_DISPLACEMENT;
        INDEX  : WORD_INDEX;
    end record;


type IMPORT_ID is new INTEGER range 0 .. 2 ** 24 - 1;

type IMPORT_NAME is
    record
       PROCESSOR : PROCESSORS.ID;
       NUMBER    : IMPORT_ID;
    end record;
```

```
type WORD_KIND is  (DISCRETE_VAR,          FLOAT_VAR,          ACCESS_VAR,
                    MODULE_VAR,            SEGMENT_VAR,        RECORD_VAR,
                    VARIANT_RECORD_VAR,    ARRAY_VAR,          DISCRETE_REF,
                    FLOAT_REF,             ACCESS_REF,         MODULE_REF,
                    SEGMENT_REF,           REFERENCE_VAR,      SUBPROGRAM_VAR,
                    SELECT_VAR,            ENTRY_VAR,          EXCEPTION_VAR,

                    MODULE_KEY,            DELETION_KEY,       BREAKPOINT_KEY,

                    INDEX_INFO_TEMP,

                    ACTIVATION_STATE,      ACTIVATION_LINK,
                    ACCEPT_LINK,           AUXILIARY_MARK,

                    MODULE_STATE,          ALLOCATION_STATE,   RESOURCE_LIMITS,
                    MICRO_STATE,           SCHEDULING_STATE,   DEBUGGING_STATE,
                    STATIC_CONNECTION,     DEPENDENCE_LINK);

subtype VARIABLE     is WORD_KIND range      DISCRETE_VAR .. EXCEPTION_VAR;
subtype CONTROL_KEY  is WORD_KIND range        MODULE_KEY .. BREAKPOINT_KEY;
subtype TEMPORARY    is WORD_KIND range  INDEX_INFO_TEMP .. INDEX_INFO_TEMP;
subtype MARK_WORD    is WORD_KIND range ACTIVATION_STATE .. AUXILIARY_MARK;
subtype STATE_WORD   is WORD_KIND range      MODULE_STATE .. DEPENDENCE_LINK;

subtype SCALAR_VAR      is WORD_KIND range  DISCRETE_VAR .. FLOAT_VAR;
subtype VALUE_VAR       is WORD_KIND range  DISCRETE_VAR .. SEGMENT_VAR;
subtype FIXED_SIZE_VAR  is VALUE_VAR range      FLOAT_VAR .. SEGMENT_VAR;
subtype INDIRECT_VAR    is WORD_KIND range     RECORD_VAR .. SEGMENT_REF;
subtype STRUCTURE_VAR   is WORD_KIND range     RECORD_VAR .. ARRAY_VAR;
subtype VALUE_REF       is WORD_KIND range  DISCRETE_REF .. SEGMENT_REF;
subtype FIXED_SIZE_REF  is VALUE_REF range      FLOAT_REF .. SEGMENT_REF;
subtype TYPED_VAR       is WORD_KIND range  DISCRETE_VAR .. SEGMENT_REF;
subtype IMPORT_VAR      is WORD_KIND range  DISCRETE_VAR .. REFERENCE_VAR;
subtype TASKING_VAR     is WORD_KIND range     SELECT_VAR .. ENTRY_VAR;
```

```
type VALUE (OF_KIND : VALUE_VAR := DISCRETE_VAR) is
    record
        case OF_KIND is
            when DISCRETE_VAR =>
                DISCRETE_VAL : BASE.DISCRETE;

            when FLOAT_VAR    =>
                REAL_VAL   : BASE.REAL;

            when ACCESS_VAR   =>
                POINTER    : DATA_DISPLACEMENT;
                ALLOCATOR  : PROCESSORS.ID;

            when MODULE_VAR   =>
                STACK      : NAME;

            when SEGMENT_VAR  =>
                SEGMENT    : SEGMENTS.NAME;

            when others =>
                null;
        end case;
    end record;
```

```
type SIGNED_BIT_COUNT is new INTEGER range
                        - DATA_DISPLACEMENT'LAST .. DATA_DISPLACEMENT'LAST;

subtype BIT_COUNT  is SIGNED_BIT_COUNT range 0 .. SIGNED_BIT_COUNT'LAST;

subtype VALUE_SIZE is SIGNED_BIT_COUNT range 0 .. 64;

FIELD_INDEX_SIZE : constant VALUE_SIZE
                                := VALUE_SIZE (CODE.FIELD_INDEX_SIZE);

FLOAT_SIZE   : constant VALUE_SIZE := 64;
ACCESS_SIZE  : constant VALUE_SIZE := 40;
MODULE_SIZE  : constant VALUE_SIZE := 32;
SEGMENT_SIZE : constant VALUE_SIZE := 20;

KEY_SIZE     : constant VALUE_SIZE := 64;


BOUND_SIZE   : constant VALUE_SIZE := 32;
OFFSET_SIZE  : constant VALUE_SIZE := 33;

subtype INDEX_VALUE is BASE.DISCRETE
                    range - 2 ** (INTEGER (BOUND_SIZE) - 1) ..
                                    2 ** (INTEGER (BOUND_SIZE) - 1) - 1;
```

```
type INDEX_DESCRIPTOR is
    record
        MINIMUM  : INDEX_VALUE;
        MAXIMUM  : INDEX_VALUE;
        CONSPART : SIGNED_BIT_COUNT;
    end record;
```

```
type BOUNDS_LOCATION  is (WITH_TYPE, WITH_OBJECT);

type SUBPROGRAM_STATE is (IS_LOCAL, IS_ACTIVE, IS_INACTIVE);


type TYPE_PROTECTION is
    record
        PRIVACY : CODE.DECL_PRIVACY;
        DERIVES_PRIVACY : BOOLEAN;
    end record;


type VARIABLE_PROTECTION is
    record
        VISIBILITY  : CODE.VAR_VISIBILITY;
        IS_CONSTANT : BOOLEAN;
    end record;


type TYPE_LINK (OF_KIND : TYPED_VAR := DISCRETE_VAR) is
    record
        PATH     : TYPE_REFERENCE;
        FOR_TYPE : TYPE_PROTECTION;
        FOR_VAR  : VARIABLE_PROTECTION;
        case OF_KIND is
            when DISCRETE_VAR | DISCRETE_REF =>
                IS_SIGNED   : BOOLEAN;

            when ARRAY_VAR     =>
                BOUNDS_SITE : BOUNDS_LOCATION;

            when VARIANT_RECORD_VAR =>
                IS_CONSTRAINED : BOOLEAN;

            when others =>
                null;
        end case;
    end record;
```

```
package ENTRYS is

    subtype NAME          is CODE.SCOPE_DELTA;

    type CHOICE_NUMBER    is new INTEGER range 0 .. 255;


    type INFO (IS_FAMILY : BOOLEAN := FALSE) is
        record
            case IS_FAMILY is
                when FALSE =>
                    QUEUE_HEAD    : TASKS.QUEUE_DISPLACEMENT;
                    QUEUE_TAIL    : TASKS.QUEUE_DISPLACEMENT;
                    COUNT         : TASKS.COUNTER;

               when TRUE  =>
                    FAMILY_SITE   : WORD_DISPLACEMENT;
                    FAMILY_FIRST  : INDEX_VALUE;
                    FAMILY_LAST   : INDEX_VALUE;
            end case;
        end record;

end ENTRYS;
```

```
type CONTROL_WORD (OF_KIND : WORD_KIND := DISCRETE_VAR) is
    record
        case OF_KIND is
            when VALUE_VAR      =>
                VALUE_ITEM : VALUE (OF_KIND);
                VALUE_TYPE : TYPE_LINK (OF_KIND);

            when INDIRECT_VAR   =>
                REF_ITEM   : DATA_REFERENCE;
                REF_TYPE   : TYPE_LINK (OF_KIND);

            when REFERENCE_VAR =>
                REF_TO_VALUE : CONTROL_REFERENCE;
                REF_IN_SCOPE : NAME;
                REF_SUBPROG  : CODE.SUBPROG_SORT;

            when SUBPROGRAM_VAR =>
                SUBPROG_STATE : SUBPROGRAM_STATE;
                SUBPROG_SORT  : CODE.SUBPROG_SORT;
                DECLARE_FRAME : CONTROL_REFERENCE;
                LEX_LEVEL     : CODE.LEXICAL_LEVEL;
                FIRST_INSERT  : CODE.INSTRUCTION;
                CODE_SEGMENT  : SEGMENTS.NAME;
                BLOCK_BEGIN   : SEGMENTS.DISPLACEMENT;
                ACCEPT_ENTRY  : ENTRYS.NAME;

            when ENTRY_VAR      =>
                ENTRY_NAME    : ENTRYS.NAME;
                ENTRY_PARAMS  : TASKS.PARAM_COUNT;
                FIRST_FREE    : TASKS.QUEUE_DISPLACEMENT;
                FOR_ENTRY     : ENTRYS.INFO;

            when SELECT_VAR =>
                SELECT_SITE     : WORD_DISPLACEMENT;
                SELECT_REF      : CONTROL_DISPLACEMENT;
                LAST_CHOICE     : ENTRYS.CHOICE_NUMBER;
                ACCEPT_COUNT    : ENTRYS.CHOICE_NUMBER;
                DELAY_COUNT     : ENTRYS.CHOICE_NUMBER;
                FIELD_COUNT     : CODE.FIELD_INDEX;
                HAS_TERMINATE   : BOOLEAN;
                TERMINATE_OPEN  : BOOLEAN;
                HAS_ELSE        : BOOLEAN;

            when EXCEPTION_VAR  =>
                EXCEPTION_VAL   : EXCEPTIONS.NUMBER;
                RAISED_ADDRESS  : SEGMENTS.ADDRESS;
                RAISED_SCOPE    : NAME;
```

```
when MODULE_KEY      =>
     FOR_MODULE      : NAME;
     PERMITTED       : TASKS.OPTION;

when DELETION_KEY    =>
     IS_LOCKED       : BOOLEAN;
    SETS_VISIBILITY  : BOOLEAN;

when BREAKPOINT_KEY  =>
     BREAK_ADDRESS   : SEGMENTS.ADDRESS;
     DEBUGGING_SCOPE : NAME;


when INDEX_INFO_TEMP  =>
     FOR_INDEX : INDEX_DESCRIPTOR;


when ACTIVATION_STATE =>
     OUTER_FRAME      : NAME;
     ENCLOSING_FRAME  : CONTROL_REFERENCE;
     CONTROL_PRED     : CONTROL_DISPLACEMENT;
     CURRENT_LEX      : CODE.LEXICAL_LEVEL;
     RETURN_INSERT    : CODE.INSTRUCTION;
     IN_ACCEPT_BODY   : BOOLEAN;
     IN_UTILITY       : BOOLEAN;
     HAS_CHILDREN     : BOOLEAN;

when ACTIVATION_LINK  =>
     TYPE_FRAME       : TYPE_DISPLACEMENT;
     DATA_FRAME       : DATA_REFERENCE;
     BLOCK_START      : SEGMENTS.DISPLACEMENT;
     RETURN_ADDRESS   : SEGMENTS.ADDRESS;
     FRAME_MARKED     : BOOLEAN;

when ACCEPT_LINK       =>
     IN_RENDEZVOUS    : NAME;
     FAMILY_INDEX     : INDEX_VALUE;
     RENDEZVOUS_REF   : CONTROL_DISPLACEMENT;
     ENTRY_REF        : CONTROL_DISPLACEMENT;
     PRIORITY         : TASKS.PRIORITY;
     IN_SELECT        : BOOLEAN;
     SELECT_CHOICE    : ENTRYS.CHOICE_NUMBER;

when AUXILIARY_MARK   =>
     TYPE_STACK_MARK  : TYPE_DISPLACEMENT;
     TYPE_MARK_USED   : BOOLEAN;
     DATA_STACK_MARK  : DATA_REFERENCE;
     DATA_MARK_USED   : BOOLEAN;
     HAS_PRIOR_MARK   : BOOLEAN;
```

```
        when MODULE_STATE       =>
            ELABORATION         : TASKS.MODE;
            BLOCKED_STATE       : TASKS.CONDITION;
            QUEUE_NEIGHBOR      : NAME;
            CONTROL_TOS         : CONTROL_DISPLACEMENT;
            CONTROL_EXTENT      : CONTROL_DISPLACEMENT;
            PROGRAM_COUNTER     : SEGMENTS.ADDRESS;
            NEXT_INSTRUCTION    : CODE.INSTRUCTION;

        when ALLOCATION_STATE   =>
            INNER_FRAME         : CONTROL_DISPLACEMENT;
            TYPE_TOS            : TYPE_DISPLACEMENT;
            TYPE_EXTENT         : TYPE_DISPLACEMENT;
            DATA_TOS            : DATA_REFERENCE;
            HAS_AUXILIARY_MARK  : BOOLEAN;

        when RESOURCE_LIMITS    =>
            DATA_EXTENT         : WORD_DISPLACEMENT;
            MAX_CONTROL         : CONTROL_DISPLACEMENT;
            MAX_TYPE            : TYPE_DISPLACEMENT;
            MAX_DATA            : WORD_DISPLACEMENT;

        when MICRO_STATE        =>
            null;

        when SCHEDULING_STATE   =>
            SCHEDULING_GROUP    : TASKS.GROUP_ID;
            BASE_TIME_SLICE     : TASKS.SLICE_TIME;
            CURRENT_SLICE       : TASKS.SLICE_TIME;
            INTERSLICE_DELAY    : TASKS.SLICE_TIME;
            USED_SLICE_COUNT    : TASKS.SLICE_COUNT;
            PAGE_FAULT_COUNT    : TASKS.FAULT_COUNT;

        when DEBUGGING_STATE    =>
            SCOPE_FOR_DEBUG     : NAME;
            DEBUG_ENABLE        : TASKS.BREAK_ENABLE;

        when STATIC_CONNECTION  =>
            MODULE_IMPORTS      : IMPORT_NAME;
            MODULE_TYPE         : TYPE_REFERENCE;

        when DEPENDENCE_LINK    =>
            DECLARER            : NAME;
            DEPEND_FRAME        : TYPE_REFERENCE;
            LAST_ACTIVATE       : CHILD_POINTER;
            HAS_VISIBLE_CHILD   : BOOLEAN;
    end case;
end record;
```

```
package CHILD is

    type WORD_KIND is (MODULE,       COLLECTION,      IMPORT,
                       QUAD_LINK,    DEPENDENT_COUNT);

    type WORD (OF_KIND : WORD_KIND := DEPENDENT_COUNT) is
        record
            case OF_KIND is
                when MODULE | COLLECTION =>
                    ADDRESS_SPACE : NAME;

                when IMPORT     =>
                    IMPORT_SPACE  : IMPORT_NAME;

                when QUAD_LINK =>
                    PRIOR_QUAD    : TYPE_DISPLACEMENT;
                    LAST_ITEM     : WORD_INDEX;

                when DEPENDENT_COUNT =>
                    DEPENDENTS    : TASKS.COUNTER;
            end case;
        end record;

    type QUAD_WORD is array (WORD_INDEX) of WORD;

    type REFERENCE is
        record
            STACK : NAME;
            WORD  : CHILD_POINTER;
        end record;

end CHILD;


type MODULE_LEVEL is new INTEGER range 0 .. 2 ** 8 - 1;

type DIMENSIONALITY   is new INTEGER range 1 .. 63;

type VARIANT_CLAUSE_INDEX is new INTEGER range 0 .. 255;


type GROUND_TYPE_INFO is
    record
        TYPE_REF : TYPE_REFERENCE;
        DECLARES_PRIVACY : BOOLEAN;
        DERIVES_PRIVACY  : BOOLEAN;
    end record;


type COMPONENT_INFO is
    record
        HAS_MODULE  : BOOLEAN;
        HAS_ACCESS  : BOOLEAN;
        HAS_SEGMENT : BOOLEAN;
        HAS_VARIANT : BOOLEAN;
    end record;
```

```
type TYPE_KIND is (SCALAR_BOUNDS,      ACCESS_INFO,      ACCESS_FRAME,
                   MODULE_INFO,        VARIANT_INFO,     FIELD_INFO,
                   DEFAULT_DISCRIMS,   ARRAY_INFO,       ARRAY_INDEX_INFO,
                   TYPE_INFO,          TYPE_UTILITY,     CHILDREN);


type TYPE_WORD (OF_KIND : TYPE_KIND := SCALAR_BOUNDS) is
    record
        case OF_KIND is
            when SCALAR_BOUNDS =>
                MINIMUM          : VALUE; -- (OF_KIND => SCALAR_VAR);
                MAXIMUM          : VALUE; -- (OF_KIND => SCALAR_VAR);

            when ACCESS_INFO   =>
                OBJECT_TYPE      : TYPE_LINK;
                COLLECTION       : COLLECTION_ID;
                CREATOR          : PROCESSORS.ID;
                GARBAGE_COLLECT  : BOOLEAN;
                IS_HOMOGENEOUS   : BOOLEAN;

            when ACCESS_FRAME  =>
                COUNT_FRAME      : TYPE_REFERENCE;

            when MODULE_INFO   =>
                CODE_SEGMENT     : SEGMENTS.NAME;
                DECLARE_LEVEL    : MODULE_LEVEL;
                GENERIC_COUNT    : CODE.SCOPE_DELTA;
                VISIBLE_COUNT    : CODE.SCOPE_DELTA;
                BODY_COUNT       : CODE.SCOPE_DELTA;
                MODULE_IMPORTS   : IMPORT_NAME;
                PROTECTION       : TYPE_PROTECTION;
                HAS_ENTRYS       : BOOLEAN;

            when VARIANT_INFO  =>
                DISCRIM_SIZE     : BIT_COUNT;
                FIXED_SIZE       : BIT_COUNT;
                DISCRIM_COUNT    : CODE.FIELD_INDEX;
                FIXED_COUNT      : CODE.FIELD_INDEX;
                VARIANT_COUNT    : CODE.FIELD_INDEX;
                VARIANT_INDEX    : VARIANT_CLAUSE_INDEX;
                INDEX_MAX        : VARIANT_CLAUSE_INDEX;
                MUST_CLEAR_FIXED_PART   : BOOLEAN;
                MUST_COPY_VARIANT_PART  : BOOLEAN;
                IS_CONSTRAINED_SUBTYPE  : BOOLEAN;

            when FIELD_INFO    =>
                FIELD_TYPE       : TYPE_LINK;
                FIELD_SIZE       : VALUE_SIZE;
                FIELD_PLACE      : BIT_COUNT;
                FIELD_NUMBER     : CODE.FIELD_INDEX;
                FIELD_KIND       : CODE.FIELD_SORT;
                CLAUSE_COUNT     : CODE.FIELD_INDEX;
                OF_VARIANT       : VARIANT_CLAUSE_INDEX;

            when DEFAULT_DISCRIMS =>
                FIRST_DISCRIM    : BASE.DISCRETE;
                SECOND_DISCRIM   : BASE.DISCRETE;
```

```
                when ARRAY_INFO     =>
                    DIMENSIONS      : DIMENSIONALITY;
                    ELEMENT_TYPE    : TYPE_LINK;
                    ELEMENT_SIZE    : BIT_COUNT;

                when ARRAY_INDEX_INFO  =>
                    FOR_INDEX       : INDEX_DESCRIPTOR;

                when TYPE_UTILITY   =>
                    UTILITY_SUBPROG : CONTROL_WORD (OF_KIND => SUBPROGRAM_VAR);

                when TYPE_INFO      =>
                    OBJECT_SIZE      : BIT_COUNT;
                    GROUND_TYPE      : GROUND_TYPE_INFO;
                    TYPE_COMPLETE    : BOOLEAN;
                    CONTENTS         : COMPONENT_INFO;    -- structures
                    OBJECTS_ALIGNED  : BOOLEAN;           -- structures
                    FIELD_COUNT      : CODE.FIELD_INDEX;  -- records

                when CHILDREN       =>
                    CHILD_QUAD       : CHILD.QUAD_WORD;
            end case;
        end record;


    type DESCRIPTOR_DELTA is new INTEGER range - 512 .. 511;

    DISCRIM_BEGIN  : constant DESCRIPTOR_DELTA := - 3;
    UTILITY_OFFSET : constant DESCRIPTOR_DELTA := - 2;
    INFO_OFFSET    : constant DESCRIPTOR_DELTA := - 1;

end STACKS;



package body STACKS is

    package body ENTRYS is end ENTRYS;
    package body CHILD  is end CHILD;

end STACKS;
```