

FILEID**SRMDEF

L 2

SSSSSSSS	RRRRRRRR	MM	MM	DDDDDDDD	EEEEEEEEE	FFFFFFFFF		
SSSSSSSS	RRRRRRRR	MM	MM	DDDDDDDD	EEEEEEEEE	FFFFFFFFF		
SS	RR	RR	MMMM	MMMM	DD	EE	FF	
SS	RR	RR	MMMM	MMMM	DD	EE	FF	
SS	RR	RR	MM	MM	DD	DD	EE	FF
SS	RR	RR	MM	MM	DD	DD	EE	FF
SSSSSS	RRRRRRRR	MM	MM	DD	DD	EEEEEEE	FFFFFFF	
SSSSSS	RRRRRRRR	MM	MM	DD	DD	EEEEEEE	FFFFFFF	
SS	RR	RR	MM	MM	DD	DD	EE	FF
SS	RR	RR	MM	MM	DD	DD	EE	FF
SS	RR	RR	MM	MM	DD	DD	EE	FF
SS	RR	RR	MM	MM	DD	DD	EE	FF
SSSSSSSS	RR	RR	MM	MM	DDDDDDDD	EEEEEEEEE	FF	
SSSSSSSS	RR	RR	MM	MM	DDDDDDDD	EEEEEEEEE	FF	

.....
.....
....

SSSSSSSS	DDDDDDDD	LL	
SSSSSSSS	DDDDDDDD	LL	
SS	DD	DD	LL
SS	DD	DD	LL
SS	DD	DD	LL
SS	DD	DD	LL
SSSSSS	DD	DD	LL
SSSSSS	DD	DD	LL
SS	DD	DD	LL
SS	DD	DD	LL
SS	DD	DD	LL
SS	DD	DD	LL
SSSSSSSS	DDDDDDDD	LLLLLLLLL	
SSSSSSSS	DDDDDDDD	LLLLLLLLL	

SRI
CO
en
mo
/*
/*
/*
age
en
en
mo
/*
/*
/*
age

{ SRMDEF.MDL - system definitions for System Reference Manual

{ Version: 'V04-000'

{*****

{* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
{* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
{* ALL RIGHTS RESERVED.

{* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
{* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
{* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
{* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
{* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
{* TRANSFERRED.

{* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
{* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
{* CORPORATION.

{* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
{* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.

{******

{++
{ FACILITY: VAX/VMS System Macro Libraries

{ ABSTRACT:

{ This file contains the SDL source for the System Reference
{ Manual (SRM) structure definitions.

{ ENVIRONMENT:

{ n/a

{--

{ AUTHOR: The VMS Group CREATION DATE: 1-Aug-1976

{ MODIFICATION HISTORY:

3-003	DG0303	Debess Grabatz	27-Feb-1984
		Change DSC\$K_CLASS_SO and DSC\$K_CLASS_UBSO to	
		DSC\$K_CLASS_SB and DSC\$K_CLASS_OBSB (ECO 9.4).	
3-002	DG0302	Debess Grabatz	26-Oct-1983
		Add DSC\$K_CLASS_SO and DSC\$K_CLASS_UBSO.	
3-001	ACG0303	Andrew C. Goldstein	9-Dec-1982
		Add FILL attribute to extraneous field names	
2-010	FM2010	Farokh Morshed	1-DEC-1981
		Add DSC\$K_DTYPE_VT.	

end
end

{ 2-009 SBL2009 Steven B. Lionel 26-Aug-1981
{} Update to Rev. 9.
{} 2-008 SBL2008 Steven B. Lionel 20-Aug-1981
{} Add missing DSC\$L POS.
{} 2-007 SBL2007 Steven B. Lionel 14-Jan-1981
{} Update to Rev. 8.0 (7-Oct-80).
{} 2-006 SBL2006 Steven B. Lionel 8-Sept-1980
{} Update to Rev. 7.5 (20-June-80).
{} 2-005 - SBL2005 - Update to Rev. 6 (30-Oct-1979). SBL 31-Dec-1979
{} 2-004 - SBL2004 - Make DSC\$B SCALE signed. SBL 10-August-1979
{} 2-003 - SBL2003 - Add SRMSK Symbols for floating faults. SBL 20-July-1979
{} 2-002 - SBL2002 - Update to Rev. 5.2 (25-Apr-79). SBL 4-Jun-1979
{} 2-001 - Added new descriptor datatypes from Appendix C Rev 5.
{} Steven B. Lionel 1-Feb-79

{ VAX Procedure Calling symbols.
{ These symbols are taken from Appendix C of the VAX-11
{ System Reference Manual which is under ECO control. No
{ additions to this file cannot be made without first getting
{ formal ECO approval to Appendix C of the SRM.
{ No symbols should be removed or changed without careful
{ evaluation of the effects of such changes on existing software.
{ In case of disagreement, SRM Appendix C takes precedence
{ over this file.
{ These symbols are taken from Appendix C Rev 9.0

module \$DSCDEF;

/* Define Procedure argument data types

/* C.9 ARGUMENT DATA TYPES

/* The following encoding is used for atomic data elements:

/* Mnemonic Code Description

constant DTYPE_Z	equals 0 prefix DSC tag \$K;	/* Unspecified. The calling program has /* specified no data type/* the called /* procedure should assume the argument is of /* the correct type.
constant DTYPE_V	equals 1 prefix DSC tag \$K;	/* Bit. An aligned bit string.
constant DTYPE_BU	equals 2 prefix DSC tag \$K;	/* Byte Logical. 8-bit unsigned quantity.
constant DTYPE_WU	equals 3 prefix DSC tag \$K;	/* Word Logical. 16-bit unsigned quantity.
constant DTYPE_LU	equals 4 prefix DSC tag \$K;	/* Longword Logical. 32-bit unsigned /* quantity.
constant DTYPE_QU	equals 5 prefix DSC tag \$K;	/* Quadword Logical. 64-bit unsigned /* quantity.
constant DTYPE_OU	equals 25 prefix DSC tag \$K;	/* Octaword Logical. 128-bit unsigned /* quantity.
constant DTYPE_B	equals 6 prefix DSC tag \$K;	/* Byte Integer. 8-bit signed 2's-complement /* integer.
constant DTYPE_W	equals 7 prefix DSC tag \$K;	/* Word Integer. 16-bit signed 2's-complement /* integer.
constant DTYPE_L	equals 8 prefix DSC tag \$K;	/* Longword Integer. 32-bit signed /* 2's-complement integer.
constant DTYPE_Q	equals 9 prefix DSC tag \$K;	/* Quadword Integer. 64-bit signed /* 2's-complement integer.

constant DTTYPE_0 equals 26 prefix DSC tag \$K; /* Octaword Integer. 128-bit signed
/* 2's-complement integer.

constant DTTYPE_F equals 10 prefix DSC tag \$K; /* F_floating. 32-bit F_floating quantity representing
/* a single-precision number.

constant DTTYPE_D equals 11 prefix DSC tag \$K; /* D_floating. 64-bit D_floating quantity representing
/* a double-precision number.

constant DTTYPE_G equals 27 prefix DSC tag \$K; /* G_floating. 64-bit G_floating quantity representing
/* a double-precision number.

constant DTTYPE_H equals 28 prefix DSC tag \$K; /* H_floating. 128-bit H_floating quantity representing
/* a quadruple-precision number.

constant DTTYPE_FC equals 12 prefix DSC tag \$K; /* F_floating complex. Ordered pair of F_floating
/* quantities representing a single-precision complex
/* number. The lower addressed quantity is the
/* real part, the higher addressed quantity is the
/* imaginary part.

constant DTTYPE_DC equals 13 prefix DSC tag \$K; /* D_floating complex. Ordered pair of D_floating
/* quantities representing a double-precision complex
/* number. The lower addressed quantity is the
/* real part, the higher addressed quantity is the
/* imaginary part.

constant DTTYPE_GC equals 29 prefix DSC tag \$K; /* G_floating complex. Ordered pair of G_floating
/* quantities representing a double-precision complex
/* number. The lower addressed quantity is the
/* real part, the higher addressed quantity is the
/* imaginary part.

constant DTTYPE_HC equals 30 prefix DSC tag \$K; /* H_floating complex. Ordered pair of H_floating
/* quantities representing a quadruple-precision complex
/* number. The lower addressed quantity is the
/* real part, the higher addressed quantity is the
/* imaginary part.

constant DTTYPE_CIT equals 31 prefix DSC tag \$K; /* COBOL Intermediate Temporary. Floating point
/* datum with an 18 digit normalized decimal
/* fraction and a 2 digit decimal exponent. The
/* fraction is represented as a packed decimal
/* number. The exponent is represented as a
/* 16-bit 2's complement integer. A detailed
/* description of this data type can be found in
/* Section 7.4 of SRM Appendix C.

constant DTTYPE_VU equals 34 prefix DSC tag \$K; /* Bit Unaligned.

/* The following string types are ordinarily described by a string
/* descriptor. The data type codes below occur in those descriptors:

constant DTTYPE_T equals 14 prefix DSC tag \$K; /* Character-coded text. A single 8-bit character
/* (atomic data type) or a sequence of 0 to
/* 2**16-1 8-bit characters (string data type).

constant DTTYPE_VT equals 37 prefix DSC tag \$K; /* Varying Character-coded Text Data Type.
constant DTTYPE_NU equals 15 prefix DSC tag \$K; /* Numeric string, unsigned.
constant DTTYPE_NL equals 16 prefix DSC tag \$K; /* Numeric string, left separate sign.
constant DTTYPE_NLO equals 17 prefix DSC tag \$K; /* Numeric string, left overpunched sign.
constant DTTYPE_NR equals 18 prefix DSC tag \$K; /* Numeric string, right separate sign.
constant DTTYPE_NRO equals 19 prefix DSC tag \$K; /* Numeric string, right overpunched sign.
constant DTTYPE_NZ equals 20 prefix DSC tag \$K; /* Numeric string, zoned sign.
constant DTTYPE_P equals 21 prefix DSC tag \$K; /* Packed decimal string.
/* The following encodings are used for miscellaneous data types:
constant DTTYPE_ZI equals 22 prefix DSC tag \$K; /* Sequence of instructions.
constant DTTYPE_ZEM equals 23 prefix DSC tag \$K; /* Procedure entry mask.
constant DTTYPE_DSC equals 24 prefix DSC tag \$K; /* Descriptor. This data type allows a descriptor
/* to be an argument data type, thereby allowing
/* the use of levels of descriptors.
constant DTTYPE_BPV equals 32 prefix DSC tag \$K; /* Bound Procedure Value. A two longword entity
/* in which the first longword contains the address
/* of a procedure entry mask and the second longword
/* contains the environment value. The environment
/* value is determined in a language specific
/* fashion when the original Bound Procedure Value
/* is generated. When the bound procedure is
/* invoked, the calling program loads the second
/* longword into R1. This data type can be passed
/* using the immediate value mechanism when the
/* environment value is not needed. In this case
/* the argument list entry contains the address of
/* the procedure entry mask and the second longword
/* is omitted.
constant DTTYPE_BLV equals 33 prefix DSC tag \$K; /* Bound label value.
constant DTTYPE_ADT equals 35 prefix DSC tag \$K; /* Absolute Date and Time
/* K DTTYPE_???,36 /* DTTYPE code 36 is reserved for future definition
/* The following types are used by the VAX/VMS Debug Facility: \FOR
/* INTERNAL DOCUMENTATION ONLY. For details, see VAX/VMS Debug Facility
/* Documentation.\
constant DTTYPE_CAD equals 178 prefix DSC tag \$K; /* Address calculation command
constant DTTYPE_ENT equals 179 prefix DSC tag \$K; /* Entry global definition
constant DTTYPE_GBL equals 180 prefix DSC tag \$K; /* Global symbol definition
constant DTTYPE_EPT equals 181 prefix DSC tag \$K; /* Entry point to routine.

```
constant DTTYPE_R11      equals 182  prefix DSC tag $K; /* Line number relative to R11 correlation
constant DTTYPE_FLD      equals 183  prefix DSC tag $K; /* BLISS FIELD name.
constant DTTYPE_PCT      equals 184  prefix DSC tag $K; /* PSECT information.
constant DTTYPE_DPC      equals 185  prefix DSC tag $K; /* PC correlation table for FORTRAN IV+
constant DTTYPE_LBL      equals 186  prefix DSC tag $K; /* LITERAL or LABEL
constant DTTYPE_SLB      equals 187  prefix DSC tag $K; /* Label in non-assembly language modules
constant DTTYPE_MOD      equals 188  prefix DSC tag $K; /* Beginning of new module
constant DTTYPE_EOM      equals 189  prefix DSC tag $K; /* End of module
constant DTTYPE_RTN      equals 190  prefix DSC tag $K; /* Beginning of new routine
constant DTTYPE_EOR      equals 191  prefix DSC tag $K; /* End of routine
```

/* The following type codes are RESERVED for future use:

```
/*
/*
```

37-177 RESERVED to DEC

192-255 RESERVED to CSS and customers

/* C.10 ARGUMENT DESCRIPTORS

/* A uniform descriptor mechanism is defined for use by all procedures
/* which conform to this standard. Descriptors are uniformly typed and
/* the mechanism is extensible. As new varieties of descriptor become
/* necessary, they will be added to this catalogue.

/* Note:

/* All fields represent unsigned quantities unless explicitly stated
/* otherwise.

/* C.10.1 Descriptor Prototype

/* Each class of descriptor consists of at least 2 longwords in the
/* following format:

```
/*      +-----+-----+-----+ :Descriptor
/*      | CLASS | DTTYPE | LENGTH   |
/*      +-----+-----+-----+
/*                  | POINTER           |
/*      +-----+-----+
```

/* DSC\$W_LENGTH A one-word field specific to the descriptor
/* <0,15:0> class/* typically a 16-bit (unsigned) length.

/* DSC\$B_DTTYPE A one-byte atomic data type code (see C.9)
/* <0,23:16>

/* DSC\$B_CLASS A one-byte descriptor class code (see below)

```

/*
/* <0,31:24>
/*
/* DSC$A_POINTER A longword pointing to the first byte of the
/* <1,31:0> data element described.

/* Note that the descriptor can be placed in a pair of registers with a
/* MOVQ instruction and then the length and address used directly. This
/* gives a word length, so the class and type are placed as bytes in the
/* rest of that longword. Class 0 is unspecified and hence no more than
/* the above information can be assumed.

/* Define the descriptor class codes*/

constant CLASS_Z      equals 0  prefix DSC tag $K;      /* Unspecified
constant CLASS_S      equals 1  prefix DSC tag $K;      /* Scalar, String Descriptor
constant CLASS_D      equals 2  prefix DSC tag $K;      /* Dynamic String Descriptor
constant CLASS_V      equals 3  prefix DSC tag $K;      /* Reserved for use by Digital
constant CLASS_A      equals 4  prefix DSC tag $K;      /* Array Descriptor
constant CLASS_P      equals 5  prefix DSC tag $K;      /* Procedure Descriptor
constant CLASS_PI     equals 6  prefix DSC tag $K;      /* Procedure Incarnation Descriptor
constant CLASS_J      equals 7  prefix DSC tag $K;      /* Reserved for use by Digital
constant CLASS_JI     equals 8  prefix DSC tag $K;      /* Obsolete
constant CLASS_SD     equals 9  prefix DSC tag $K;      /* Decimal Scalar String Descriptor
constant CLASS_NCA    equals 10 prefix DSC tag $K;     /* Non-contiguous Array Descriptor
constant CLASS_VS     equals 11 prefix DSC tag $K;     /* Varying String Descriptor
constant CLASS_VSA    equals 12 prefix DSC tag $K;     /* Varying String Array Descriptor
constant CLASS_UBS    equals 13 prefix DSC tag $K;     /* Unaligned Bit String Descriptor
constant CLASS_UBA    equals 14 prefix DSC tag $K;     /* Unaligned Bit Array Descriptor
constant CLASS_SB     equals 15 prefix DSC tag $K;     /* String with Bounds Descriptor
constant CLASS_UBSB   equals 16 prefix DSC tag $K;     /* Unaligned Bit String with Bounds Descriptor

/* The following descriptor class is FOR INTERNAL USE ONLY by the VAX
/* Common Run-Time Library and Debugger. This descriptor is not passed
/* between separately compiled modules. For further information, see
/* VAX-11 BASIC Description of Generated Code, Software Document
/* Retrieval Number JBS-79-004.

constant CLASS_BFA    equals 191 prefix DSC tag $K; /* BASIC File Array Descriptor

/* Descriptor classes 17-190 are RESERVED to DEC. Classes
/* 192 through 255 are RESERVED to CSS and customers.

/* Array Descriptor (DSC$K_CLASS_A)

/* An array descriptor consists of 3 contiguous blocks. The first block
/* contains the descriptor prototype information and is part of every
/* array descriptor. The second and third blocks are optional. If the
/* third block is present then so is the second.

/* A complete array descriptor has the form:

/*
+-----+-----+-----+
| 4 : DTYPe : LENGTH | :Descriptor
+-----+-----+-----+
|           POINTER          |

```

```
/*
 *-----+-----+
 * DIMCT : AFLAGS:DIGITS : SCALE          Block 1 - Prototype
 *-----+-----+
 * ARSIZE
 *
 *-----+-----+
 * A0
 *-----+-----+
 * M1
 *-----+-----+
 * ...
 *-----+-----+
 * M(n-1)
 *-----+-----+
 * Mn
 *-----+-----+
 * L1
 *-----+-----+
 * U1
 *-----+-----+
 * ...
 *-----+-----+
 * Ln
 *-----+-----+
 * Un
 */
/*Define descriptor fields:
 * \The following three fields are only valid for descriptor
 * class DSC$K_CLASS_BFA (see above).*/
aggregate DSCDEF structure prefix DSC$ origin "LENGTH";
  HANDLE address;
/*           /* The address of a vector of data concerning the
   *           /* current state of processing of the array. This
   *           /* is set to zero by the creator of the
   *           /* descriptor, and maintained thereafter by the
   *           /* BASIC file array processing functions.
  BYTEOFF longword unsigned;           /* Byte offset in the file of element 0,0,...,0
   *           /* of the array. This need not be within the
   *           /* array, or even within the file if the array
   *           /* does not have zero origin.
  LOGUNIT longword unsigned;           /* The logical unit number (BASIC channel) on
   *           /* which the file containing the array has
   *           /* been opened. This is set by the creator of the
   *           /* descriptor before first referencing the array.
/* End of BASIC File Array specific descriptor fields. The remaining
 * fields are common to all array descriptors.
```

```
/*
  "LENGTH" word unsigned;
  /* A one-word field specific to the descriptor class;
   * typically a 16-bit (unsigned) length.

end DSCDEF;

aggregate DSCDEF1 structure prefix DSC$;
  MAXSTRLEN word unsigned;
  /* Max length of the BODY field of the varying string
   * in bytes in the range 0 to 2**16-1.
   * (Classes VS, VSA)

  DTTYPE byte unsigned;
  /* A one-byte atomic data type code (see C.9)
   * Symbols used in this filed have form: DSC$K_DTTYPE_t
   * where t is the data type mnemonic from above
   * and agrees with methodology manual.

  CLASS byte unsigned;
  /* A one-byte descriptor class code (see above).
   * Symbols used in this field have form: DSC$K_CLASS_f
   * where f is the argument form mnemonic from above
   * and agrees with the methodology manual.

  POINTER address;
  /* A longword pointing to the first byte of the data
   * element described.

end DSCDEF1;

aggregate DSCDEF2 structure prefix DSC$;
  FILL_3 byte dimension 4 fill prefix DSCDEF tag $$;
  BASE address;
  /* Base of address relative to which the signed relative
   * bit position, POS, is used to locate the bit string.
   * (Classes UBS, UBA and UBSB)

constant Z_BLN equals . prefix DSC$ tag K;
constant Z_BLN equals . prefix DSC$ tag C;
constant S_BLN equals . prefix DSC$ tag K;
constant S_BLN equals . prefix DSC$ tag C;
constant D_BLN equals . prefix DSC$ tag K;
constant D_BLN equals . prefix DSC$ tag C;
constant P_BLN equals . prefix DSC$ tag K;
constant P_BLN equals . prefix DSC$ tag C;
constant J_BLN equals . prefix DSC$ tag K;
constant J_BLN equals . prefix DSC$ tag C;
constant VS_BLN equals . prefix DSC$ tag K;
constant VS_BLN equals . prefix DSC$ tag C;
  /* Block length in bytes for Z class desc.
  /* Block length in bytes for Z class desc.
  /* Block length in bytes for S class descr.
  /* Block length in bytes for S class descr.
  /* Block length in bytes for D class descr.
  /* Block length in bytes for D class descr.
  /* Block length in bytes for P class descr.
  /* Block length in bytes for P class descr.
  /* Block length in bytes for J class descr.
  /* Block length in bytes for J class descr.
  /* Block length in bytes for VS class descr.
  /* Block length in bytes for VS class descr.

*/
/* End of common definitions for all descriptors.

*/
/* Unaligned bit string definitions.
```

```
/*
end DSCDEF2;

aggregate DSCDEF3 structure prefix DSC$;
    FILL_4 byte dimension 8 fill prefix DSCDEF tag $$;
    POS longword unsigned;
    constant UBS_BLN equals . prefix DSC$ tag K;
    constant UBS_BLN equals . prefix DSC$ tag C;
    /* Signed longword relative bit position
     * with respect to BASE of the first bit
     * of unaligned bit string
     * WARNING!! Do not use this symbol to reference
     * class UBA descriptors! The DSCSL_POS in that
     * descriptor is at a variable location after the
     * bounds and multiplier blocks.
    /* Block length in bytes for UBS class descr.
    /* Block length in bytes for UBS class descr.

/**+
/* Varying type descriptor definitions
*/

end DSCDEF3;

aggregate DSCDEF4 structure prefix DSC$;
    FILL_5 byte dimension 8 fill prefix DSCDEF tag $$;
    MAXLEN word unsigned;
    FILL_1 word fill prefix DSCDEF tag $$;
    /* An unsigned word specifying the
     * maximum length of the data item
     * (i.e., the space allocated)
    /* reserved word

/**+
/* Array Descriptor definitions
*/

end DSCDEF4;

aggregate DSCDEF5 structure prefix DSC$;
    FILL_6 byte dimension 8 fill prefix DSCDEF tag $$;
    SCALE byte;
    DIGITS byte unsigned;
    AFLAGS OVERLAY union fill;
    AFLAGS byte unsigned;
    AFLAGS BITS structure fill;
    FILL_2 bitfield length 4 fill prefix DSCDEF tag $$; /* reserved to Digital
    FL_REDIM bitfield;
    /* Signed power of ten multiplier to convert the
     * internal form to external form. For example,
     * if internal number is 123 and scale is +1,
     * then the represented external number is 1230.
    /* If non-zero, unsigned number of decimal
     * digits in the external representation. If
     * zero, the number of digits can be computed
     * based on DSC$W_LENGTH.
    /* Array flag bits.

/*/
/*/
/*/
/* end
agg
```

```

    /* i.e., DSC$A_A0, DSC$L_Mi, DSC$L_Li, and
    /* DSC$L_Ui may be changed. The redimensioned
    /* array cannot exceed the size allocated to
    /* the array (i.e. DSC$L_ARSIZE).
    /* If set, the elements of the array are
    /* stored by columns (FORTRAN). Otherwise
    /* the elements are stored by rows.

FL_COLUMN bitfield;
    /* If set, the multiplicative coefficients in
    /* Block 2 are present.

FL_COEFF bitfield;
    /* If set, the bounds information in Block 3
    /* is present.

FL_BOUNDS bitfield;
end AFLAGS_BITS;

end AFLAGS_OVERLAY;
DIMCT byte unsigned;
    /* Number of dimensions

constant SD_BLN equals . prefix DSC$ tag K;
constant SD_BLN equals . prefix DSC$ tag C;
    /* Block length in bytes for SD class descr.
    /* Block length in bytes for SD class descr.

ARSIZE longword unsigned;
    /* Total size of array (in bytes unless DTTYPE is
    /* EQUAL DSC$K_DTTYPE_V or DSC$K_DTTYPE_P).

A0 address;
    /* Address of element A(0,0,...,0). This
    /* need not be within the actual array/* it
    /* is the same as DSC$A_POINTER for 0-origin
    /* arrays.

end DSCDEF5;

aggregate DSCDEF6 structure prefix DSC$;
    /* The following two fields have meaning only if the array is of
    /* class DSC$K_CLASS_NCA (Non-contiguous array).
    /* Signed bit offset of element A(0,0,...,0) with
    /* respect to BASE. (Class UBA)

FILL_7 byte dimension 16 fill prefix DSCDEF tag $$;
V0 longword unsigned;
    /* Stride of the first dimension, i.e. the
    /* difference between the addresses of
    /* successive elements of the first dimension.

S1 longword unsigned;
    /* Stride of the second dimension.

S2 longword unsigned;

end DSCDEF6;

aggregate DSCDEF7 structure prefix DSC$;
    /* Addressing coefficient M1 = U1-L1+1
    /* Addressing coefficient M2 = U2-L2+1
    /* The following two fields have meaning only if the array is of
    /* class DSC$K_CLASS_NCA (Non-contiguous array).
    /* Signed bit offset of element A(0,0,...,0) with
    /* respect to BASE. (Class UBA)

FILL_8 byte dimension 20 fill prefix DSCDEF tag $$;
M1 longword unsigned;
M2 longword unsigned;

```

```
/*+
/* Procedure Incarnation descriptor (DSC$K_CLASS_PI) and
/* Label Incarnation descriptor (DSC$K_CLASS_JI).
*/
end DSCDEF7;

aggregate DSCDEF8 structure prefix DSC$;
    FILL 9 byte dimension 8 fill prefix DSCDEF tag $$;
    FRAME address;                                /* Address of frame

constant PI_BLN equals . prefix DSC$ tag K;      /* Block length in bytes for PI class descr.
constant PI_BLN equals . prefix DSC$ tag C;      /* Block length in bytes for PI class descr.
constant JI_BLN equals . prefix DSC$ tag K;      /* block length in bytes for JI class descr.
constant JI_BLN equals . prefix DSC$ tag C;      /* block length in bytes for JI class descr.

*/
/* String with Bounds descriptor (DSC$K_CLASS_SB).
*/
end DSCDEF8;

aggregate DSCDEF9 structure prefix DSC$;
    FILL 10 byte dimension 8 fill prefix DSCDEF tag $$;
    SB_LT longword;                            /* Signed lower bound of first dimension
    SB_U1 longword;                            /* Signed upper bound of first dimension

*/
/* Unaligned Bit String with Bounds descriptor (DSC$K_CLASS_UBSB).
*/
end DSCDEF9;

aggregate DSCDEF10 structure prefix DSC$;
    FILL 11 byte dimension 12 fill prefix DSCDEF tag $$;
    UBSB_L1 longword;                          /* Signed lower bound of first dimension
    UBSB_U1 longword;                          /* Signed upper bound of first dimension

end DSCDEF10;
end_module $DSCDEF;
module $SRMDEF;

*/
/* Define SRM Hardware symbols
*/
```

STA
mod
/*+
/*
/*
/*
/*-
agg

```

constant INT_OVF_T      equals 1  prefix SRM tag $K;    /* Integer overflow trap code
constant INT_DIV_T     equals 2  prefix SRM tag $K;    /* Integer divide by zero trap code
constant FLT_OVF_T     equals 3  prefix SRM tag $K;    /* Floating overflow trap code
constant FLT_DIV_T     equals 4  prefix SRM tag $K;    /* Floating/decimal Divide by zero trap code
constant FLT_UND_T     equals 5  prefix SRM tag $K;    /* Floating Underflow trap code
constant DEC_OVF_T     equals 6  prefix SRM tag $K;    /* Decimal string overflow trap code
constant SUB_RNG_T     equals 7  prefix SRM tag $K;    /* Subscript range trap
constant FLT_OVF_F     equals 8  prefix SRM tag $K;    /* Floating Overflow fault code
constant FLT_DIV_F     equals 9  prefix SRM tag $K;    /* Floating Divide by zero fault code
constant FLT_UND_F     equals 10 prefix SRM tag $K;   /* Floating Underflow fault code

```

```

end_module $$RMDEF;
module $PSWDEF;

```

```

/**+
/* Define PSW bits (STARDEF.MDL has PSL bits)
/**-

```

```

aggregate PSWDEF union prefix PSW$;
PSWDEF BITS structure fill;
C bitfield mask;                      /* carry
V bitfield mask;                      /* overflow
Z bitfield mask;                      /* zero
N bitfield mask;                      /* negative
TBIT bitfield mask;                   /* trace trap enable
IV bitfield mask;                     /* integer overflow enable
FU bitfield mask;                     /* floating underflow enable
DV bitfield mask;                     /* decimal overflow enable
end PSWDEF_BITS;
end PSWDEF;
end_module $$PSWDEF;
module $$FDEF;

```

```

/**+
/* Define stack frame offsets as a separate structure SF$
/**-

```

```

aggregate SFDEF structure prefix SF$;

```

```
HANDLER address;
SAVE_PSW_OVERLAY union fill;
  SAVE_PSW word unsigned;
  SAVE_PSW_BITS structure fill;
    C bitfield mask;
    V bitfield mask;
    Z bitfield mask;
    N bitfield mask;
    TBIT bitfield mask;
    IV bitfield mask;
    FU bitfield mask;
    DV bitfield mask;
  end SAVE_PSW_BITS;
end SAVE_PSW_OVERLAY;
SAVE_MASK_OVERLAY union fill;
  SAVE_MASK word unsigned;
  SAVE_MASK_BITS structure fill;
    SAVE_MASK bitfield length 12;
    FILL_1 bitfield fill prefix SFDEF tag $$;
    CALLS bitfield;
    STACKOFFS bitfield length 2;
  end SAVE_MASK_BITS;
end SAVE_MASK_OVERLAY;
SAVE_AP longword unsigned;
SAVE_FP longword unsigned;
SAVE_PC longword unsigned;
SAVE_REGS longword unsigned;

end SFDEF;
end_module $$FDEF;
```

```
{+
{ A
{-
mod
agg
```

0433 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

STARDEFL
SDL

