



```

SSSSSSSS  TTTTTTTTTT  AAAAAA  RRRRRRRR  LL  EEEEEEEEE  TTTTTTTTTT
SSSSSSSS  TTTTTTTTTT  AAAAAA  RRRRRRRR  LL  EEEEEEEEE  TTTTTTTTTT
SS        TT        AA        AA  RR        RR  LL        EE        TT
SS        TT        AA        AA  RR        RR  LL        EE        TT
SS        TT        AA        AA  RR        RR  LL        EE        TT
SS        TT        AA        AA  RR        RR  LL        EE        TT
SSSSSSS   TT        AA        AA  RRRRRRRR  LL        EEEEEEEE  TT
SSSSSSS   TT        AA        AA  RRRRRRRR  LL        EEEEEEEE  TT
SS        TT        AA        AA  RRRRRRRR  LL        EE        TT
SS        TT        AAAAAAAAAA  RR  RR        LL        EE        TT
SS        TT        AAAAAAAAAA  RR  RR        LL        EE        TT
SS        TT        AA        AA  RR        RR  LL        EE        TT
SSSSSSSS  TT        AA        AA  RR        RR  LLLLLLLLLL  EEEEEEEEE  TT
SSSSSSSS  TT        AA        AA  RR        RR  LLLLLLLLLL  EEEEEEEEE  TT

```

```

SSSSSSSS  DDDDDDDD  LL
SSSSSSSS  DDDDDDDD  LL
SS        DD        DD  LL
SS        DD        DD  LL
SS        DD        DD  LL
SS        DD        DD  LL
SSSSSSS   DD        DD  LL
SSSSSSS   DD        DD  LL
SS        DD        DD  LL
SS        DD        DD  LL
SS        DD        DD  LL
SS        DD        DD  LL
SSSSSSSS  DDDDDDDD  LLLLLLLLLL
SSSSSSSS  DDDDDDDD  LLLLLLLLLL

```

EI

E

```

{ .TITLE STARLET - VMS SYSTEM SERVICE DEFINITIONS
{ .IDENT 'V04-000' /* Please read the comment about the TYPE key-
{ /* word before adding services to this file!
{ /* The comment follows the modification history.

```

```

{*****
{*
{* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
{* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
{* ALL RIGHTS RESEKVED.
{*
{* 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: System Service Definitions
{
{ ABSTRACT:
{
{ This module contains the SDL entry point definitions for system
{ services. These definitions allow the user to call the system
{ services from any language supported by SDL.
{
{ ENVIRONMENT:
{
{ AUTHOR: H. M. Levy , CREATION DATE: 1-May-1977
{
{ MODIFIED BY:
{
{ V03-041 MMD0326 Meg Dumont, 27-Aug-1984 10:15
{ Change the defaults on $MTACCESS. Also fix comments.
{
{ V03-040 ACG0399 Andrew C. Goldstein, 10-Apr-1984 13:35
{ Change media name arg of $ALLOC to flags
{
{ V03-039 LY0470 Larry Yetto 22-MAR-1984 14:17
{ Add support for new parameters to $MOD_IDENT
{
{ V03-038 TMH0028 Tim Halvorsen 10-Mar-1984
{ Fix V03-037 to change parameter 'LABEL' to 'LBLNAM'.
{ LABEL is a reserved word in BLISS.
{

```

EI

EI

EI

E

{ V03-037 MMD0256 Meg Dumont, 5-Mar-1984 17:32  
{ Add support for \$MTACCESS  
{  
{ V03-036 SSA0007 Stan Amway 3-Feb-1984  
{ In \$SETPFM, changed parameter type of ASTADR.  
{  
{ V03-035 RSH0099 R. Scott Hanna 03-Feb-1984  
{ Make the attrib parameter optional in \$ADD\_HOLDER and  
{ \$ADD\_IDENT. Make the id parameter mandatory in \$IDTOASC.  
{  
{ V03-034 LMP0186 L. Mark Pilant, 1-Feb-1984 10:41  
{ Minor tweeking of the \$CHANGE\_ACL interface.  
{  
{ V03-033 LMP0184 L. Mark Pilant, 12-Jan-1984 14:50  
{ Make the ACCNAM argument added in LMP0183 optional. Also  
{ make the optional args for \$FORMAT\_ACL really optional.  
{  
{ V03-032 LMP0183 L. Mark Pilant, 11-Jan-1984 15:23  
{ Add the ACCNAM field to the \$FORMAT\_ACL and \$PARSE\_ACL  
{ definitions.  
{  
{ V03-031 SSA0004 Stan Amway 28-Dec-1983  
{ Added new parameters to \$SETPFM.  
{  
{ V03-030 LMP0177 L. Mark Pilant, 7-Dec-1983 13:06  
{ Add a new service, \$CHANGE\_ACL, for twiddling ACLs on random  
{ objects.  
{  
{ V03-029 CWH3029 CW Hobbs 6-Dec-1983  
{ For \$IMGACT the DFLNAM parameter should be optional, fix it.  
{  
{ V03-028 KBT0579 Keith B. Thompson 8-Aug-1983  
{ Add new parameter to \$FILESCAN  
{  
{ V03-027 JLV0277 Jake VanNoy 27-JUL-1983  
{ Correct default value for CARCON in \$BRKTHRU  
{  
{ V03-026 MLJ0114 Martin L. Jack 22-Jun-1983  
{ Add \$GETCUI.  
{  
{ V03-025 RAS0160 Ron Schaefer 16-Jun-1983  
{ Change arguments to \$CRELNT, \$CRELNM, \$DELLNM and \$TRNLNM  
{ to be by-reference.  
{  
{ V03-024 RAS0158 Ron Schaefer 27-May-1983  
{ Change the PROT argument to PROMSK in \$CRELNT.  
{  
{ V03-023 RSH0028 R. Scott Hanna 24-May-1983  
{ Change the CONTEXT argument to CONXT in the \$FIND\_HELD,  
{ \$FIND\_HOLDER, \$IDTOASC, \$FINISH\_RDB system services.  
{  
{ V03-022 KFH0001 Ken Henderson 18 May 1983  
{ Changed the call interface to \$GETSYI to  
{ make the two null arguments become  
{ CSIDADR and NODENAME.

V03-021 CWH0006 CW Hobbs 28-Apr-1983  
 Change the comment delimiter on the TYPE comments because of an SDL bug. Include the \$BRKTHRU service at Jake VanNoy's request. Remove \$INPUT and \$OUTPUT from STARLET, since these cannot easily be extended to multiple languages. \$INPUT and \$OUTPUT will continue to be supported for MACRO only (as has always been the case), but they will not be documented in the system services manual.

V03-020 RAS0147 Ron Schaefer 28-Apr-1983  
 Add support for the RMS service SYSSFILESCAN.

V03-019 CWH0006 CW Hobbs 24-Apr-1983  
 Add the TYPE keyword to every parameter so that automatic documentation generators will have enough information to produce useful results. Also change names of TABNAM, TABLEN and LOGNAM at Todd Katz' request.

V03-018 LMP0098 L. Mark Pilant, 8-Apr-1983 14:50  
 Add the \$FORMAT\_ACL and \$PARSE\_ACL system services.

V03-017 ACG0319 Andrew C. Goldstein, 25-Mar-1983 23:24  
 Add \$CHKPRO, \$GRANTID, and \$REVOKID services

V03-016 TMK0001 Todd M. Katz 25-Mar-1983  
 Two of the parameters to SYSSCRELNT are in the wrong order. Reverse the ordering of ACMODE and PARTAB.

V03-015 CWH0005 CW Hobbs 25-Mar-1983  
 Clean up a couple of minor mistakes pointed out by the documentation people.

V03-014 KTA3045 Kerbey T. Altmann 23-Mar-1983  
 Add new parameter to \$ALLOC - MEDIATYP

V03-013 RSH0001 R. Scott Hanna 9-Feb-1983  
 Add the rights database system services  
 \$ADD HOLDER \$ADD\_IDENT \$ASCTOID  
 \$CREATE\_RDB \$FIND\_HOLD \$FIND HOLDER  
 \$FINISH\_RDB \$IDTOASC \$MOD HOLDER  
 \$MOD\_IDENT \$REM HOLDER \$REM\_IDENT

V03-012 RNG0012 Rod N. Gamache 8-Feb-1983  
 Add \$GETLKI.

V03-011 SBL3011 Steve Lionel 31-Jan-1983  
 Remove OPTIONAL keyword from ERR and SUC parameters to \$RENAME only.

V03-010 BLS0203 Benn Schreiber 18-Jan-1983  
 Add OPTIONAL keyword to RMS service calls ERR and SUC parameters

V03-009 CWH0004 CW Hobbs 18-Jan-1983  
 Fix default for nullargs for \$GETSYI, need to be

value, not address

V03-008 DMW4022 DMWalp 7-Jan-1983  
Added \$CRELNT, \$CRELNM, \$DELNM and \$TRNLNM

V03-007 WMC0001 Wayne Cardoza 04-Oct-1982  
Add optional item list parameter to CREPRC.

V03-006 STJ3029 Steven Jeffreys 26-Sep-1982  
Add \$ERAPAT.

V03-005 CWH0003 CW Hobbs 25-Sep-1982  
Complete rework - replace STARLET.MAR definitions  
with this file. Remaining audit trails are for  
historical purposes only - this is a new file.

V03-004 CWH0002 CW Hobbs 13-Sep-1982  
Add \$\$SNDJBCW, fix defaults for \$\$SNDJBC

V03-003 CWH0001 CW Hobbs 28-Aug-1982  
Add the \$\$SNDJBC service (for Marty Jack)

V03-002 PHL0101 Peter H. Lipman 19-Jun-1982  
Add \$\$SYNCH system service for synchronizing EFN and IOSB  
Add \$GETDVIW, \$GETJPIW, \$GETSYIW, \$UPSECW services.

V03-001 JLV0210 Jake VanNoy 12-APR-1982  
Change default for CARCON in \$BRDCST.

Each of the parameter declarations must have a TYPE entity which  
is used to supply additional information for the documentation group.  
A special SDL backend takes this information and automatically  
produces data type information for the system services manual.

Please try to find the entity in this table which most accurately  
describes the datatype of a parameter for a service. If you have  
questions about these datatypes, please direct them to CW Hobbs  
(DELPHI::HOBBBS) or Mike Fallet (GALAXY::FALLET).

SDL currently does not support the TYPE keyword, so these entities  
are added as local comments. Note that each parameter line is  
followed by exactly:

```
<tab>{ /*<space>TYPE(xxx)?
```

(where ? is a comma, semicolon, or null depending on how the  
original line was terminated) so that the comments can be quickly  
converted to actual lines when the TYPE keyword is added to SDL.  
This format is also required by some temporary tools used by the  
documentation group (temporary until TYPE is a formal keyword).

Please try to use EXACTLY this format for your TYPE lines.

ACMODE - Access mode  
Hardware access mode, as in User, Supervisor, Executive and Kernel

ADDRESS - Memory address  
Address of a location in memory, of either data or code. Not the address of an entry mask for code.

ARGLIST - Procedure argument list  
Structure is a counted argument list, as for the VAX CALL instructions.

ASTADR - Address of AST routine  
Address of the entry mask of routine which will be called at AST level, which includes RMS Error and Success routines.

BOOLEAN - Boolean truth value flag  
(Some parameters which only allow 0 and 1 are not really boolean - see if saying that parm=true makes sense. If not, call it an NUMBER. For an example, look at the REGION parameter on \$EXPREG. REGION=TRUE sounds quite silly even though 0 and 1 are the only possibilities.)

CHANNEL - I/O channel  
Address of an I/O channel

CHARDESC - Character string  
Character string descriptor. Note that several common character strings have their own types, e.g. DEVNAME, LOGNAME, SECTNAME.

CNTRLBLK - Control block  
A structure which is interpreted by the service. The elements of the structure are heterogeneous. Note that several common structures have their own types, e.g. FAB, RAB, EXHELOCK. (Contrast with LIST and VECTOR).

CONDVALU - Condition value  
A return status or system condition code, as is returned by a procedure in R0.

CONTEXT - Context  
A piece of information used by the service to maintain position over an iterative sequence of calls. Probably initialized by the user to start the sequence, but thereafter manipulated by the service.

DEVNAME - Device name  
A character string describing a device name. This can be a logical name, but it must translate to a name which is valid for a device name.

EFCLUSTER - Event flag cluster name  
A character string describing an event flag cluster name. This can be a logical name, but it must translate to a name which is valid for an event flag cluster.

EFNUM - Event flag number  
An integer representing the number of an event flag.

ENTRYADR - Procedure entry address  
The address of a procedure entry mask. This procedure will not be called at AST level (use ASTADR type for that).

EXHBLOCK - Exit handler control block  
A control block describing an exit handler

FAB - File access block  
An RMS File Access Block

FILEPROT - File protection mask  
A 16-bit mask describing the file protection for system, owner, group and world

FUNCCODE - Function code  
A function code, as for a QIO service. This is a combination of a NUMBER and a MASK. If a function code is a simple enumeration of values, it should be type NUMBER.

HOLDER - Access rights holder  
The holder entity for the access rights services.

IOSB - I/O status block  
A 64-bit structure which describes the results of an I/O or similar (i.e. \$GETxxx) operation

ITELIST - Item list  
An item list, consisting of groups of 3 longword items which is terminated by a longword 0

LIST - List  
An array of elements, terminated by an element (or partial element) with a particular value. The quota list for \$CREPRC is an example.

LOCKID - Lock identifier  
A number identifying a particular lock, assigned by the system when the lock was granted

LU \* TAT - Lock status block  
Receives status of a lock request, contains the LOCKID and an optional LOCKVALU block

LOCKVALU - Lock value block  
A 16-byte block to contain a lock value

LOGNAM - Logical name string  
A 1 to 63 character string for a logical or equivalence name. Some types are passed by logical names, but ultimately must translate to a string more restricted than a logical name, for example a DEVNAME. If this is the case, use the restricted name.

MASK - Mask  
A group of flags or bitmasks, interpreted by the individual service

NULLARG - Null argument  
A place holding argument

NUMBER - An integer count  
A signed or unsigned quantity which is a count of something, such as a number of pages or the length of a character string. Also used for an indicator with a number of unique values, such as the TBLFLG argument to \$CRELOG, where 0->system, 1->group, and 2-> process name table.

PAGEPROT - Hardware page protection  
The 4-bit memory management page protection recognized by the VAX hardware

PRIVMASK - Privilege mask  
A 64-bit mask of process privileges

PROCID - Process ID  
A longword number identifying a process, assigned by the system when the process is created

PROCNAME - Process name  
A character string describing the name of a process

QUADID - Quadword identifier  
An access rights entity consisting of an id (USERID) and the attributes mask associated with the id.

QUADTIME - Quadword system time  
A time value in the 64-bit system time format

RAB - Record access block  
An RMS Record Access Block

RETID - Returned id  
An identifier which is created by the system and assigned to an object, and used on subsequent references to that object.

SECTID - Section version and validation  
A quadword specifying the version for a global section and specifying the criteria for matching that version

SECTNAME - Section name  
A character string describing a process or global section name. This can be a logical name, but it must translate to a name which is valid for a section.

SYSTEMID - System access id  
An identifier which is used to define a access rights system.

TIMDESC - Time descriptor

```

{
  A character string describing a time value in the standard
  system format.
{
  USERID - User identifier
  A user identification code, e.g. UIC
{
  USERPARM - User interpreted argument
  A longword quantity interpreted at the discretion of the user,
  for example the ASTPRM parameter for AST services or the
  REQIDT associated with timer services.
{
  VARANGE - Virtual address range
  A pair of longwords specifying the beginning and ending virtual
  address of a range of memory, as used by memory management
  services
{
  VARIES - Varying parameter
  A parameter which can take multiple types depending on other
  arguments in the call. Examples are $FAO parameters and the
  P1-P6 for QIO calls.
{
  VECTOR - Homogeneous array
  An array of identical items, length either implied or described
  by one of the parameters. (A vector terminated by an item with
  a special value is described as a LIST.)
{
  ---

```

MODULE STARLET;

```

/*
/* SYSTEM SERVICE ENTRY POINT DESCRIPTIONS
/*

```

```

/*
/* $ADD_HOLDER
/*
/* Add Holder Record To The Rights Database
/*
/* $ADD_HOLDER id, holder, [attrib]
/*
/* id = identifier longword to associate the
/* holder record with
/* holder = address of the holder identifier quadword
/* attrib = attributes longword to grant to the holder
/*
ENTRY SYSSADD HOLDER ALIAS $ADD HOLDER PARAMETER (
LONGWORD UNSIGNED VALUE NAMED ID IN, /* TYPE(USERID),
QUADWORD UNSIGNED NAMED HOLDER IN, /* TYPE(HOLDER),
LONGWORD UNSIGNED VALUE NAMED ATTRIB IN DEFAULT 0 /* TYPE(MASK)
) RETURNS LONGWORD; /* TYPE(CONDVALU);

/*
/* $ADD_IDENT
/*

```

EI

```
/* Add Identifier To The Rights Database
/*
/* $ADD_IDENT name, [id], [attrib], [resid]
/*
/* name = address of the identifier name character
/* string descriptor
/* id = identifier longword to associate with 'name'
/* attrib = attributes longword to grant to the
/* identifier
/* resid = address of a longword to return the assigned
/* identifier
/*
ENTRY SYSSADD IDENT ALIAS $ADD_IDENT PARAMETER (
CHARACTER DESCRIPTOR NAMED NAME IN, /* TYPE(CHARDESC),
LONGWORD UNSIGNED VALUE NAMED ID IN DEFAULT 0, /* TYPE(USERID),
LONGWORD UNSIGNED VALUE NAMED ATTRIB IN DEFAULT 0, /* TYPE(MASK),
LONGWORD UNSIGNED NAMED RESID OUT DEFAULT 0 /* TYPE(USERID)
) RETURNS LONGWORD; /* TYPE(CONDVALU);

/*
/* $ADJSTK
/*
/* Adjust Outer Mode Stack Pointer
/*
/* $ADJSTK [acmode] ,[adjust] ,newadr
/*
/* acmode = access mode for which to adjust stack pointer
/* adjust = 16-bit signed adjustment value
/* newadr = address of longword to store updated value
/*
ENTRY SYSSADJSTK ALIAS $ADJSTK PARAMETER (
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0, /* TYPE(ACMODE),
WORD VALUE NAMED ADJUST DEFAULT 0, /* TYPE(NUMBER),
LONGWORD UNSIGNED NAMED NEWADR IN OUT /* TYPE(NUMBER)
) RETURNS LONGWORD; /* TYPE(CONDVALU);

/*
/* $ADJWSL
/*
/* Adjust Working Set Limit
/*
/* $ADJWSL [pagcnt] ,[wsetlm]
/*
/* pagcnt = number of pages to add to working set (if positive),
/* Number of pages to subtract from working set (if
/* negative).
/* wsetlm = address of longword to receive new working set limit,
/* or current working set limit if pagcnt not specified.
/*
ENTRY SYSSADJWSL ALIAS $ADJWSL PARAMETER (
LONGWORD VALUE NAMED PAGCNT DEFAULT 0, /* TYPE(NUMBER),
LONGWORD UNSIGNED NAMED WSETLM OUT DEFAULT 0 /* TYPE(NUMBER)
) RETURNS LONGWORD; /* TYPE(CONDVALU);
```

```

/*
/* $ALLOC
/*
/* Allocate Device
/*
/* $ALLOC devnam ,[phylen] ,[phybuf] ,[acmode] ,[flags]
/*
/* devnam = address of device name or logical name string
/* descriptor
/* phylen = address of word to receive length of physical name
/* phybuf = address of physical name buffer descriptor
/* acmode = access mode associated with allocated device
/* flags = options flags longword
/*
ENTRY SYSSALLOC ALIAS $ALLOC PARAMETER (
CHARACTER DESCRIPTOR NAMED DEVNAM IN,  (/* TYPE(DEVNAME),
WORD UNSIGNED NAMED PHYLEN OUT DEFAULT 0,  (/* TYPE(NUMBER),
CHARACTER DESCRIPTOR NAMED PHYBUF OUT DEFAULT 0,  (/* TYPE(DEVNAME),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0, (/* TYPE(ACMODE),
LONGWORD UNSIGNED VALUE NAMED FLAGS DEFAULT 0 (/* TYPE(MASK)
) RETURNS LONGWORD;  (/* TYPE(CONDVALU);

/*
/* $ASCEFC
/*
/* Associate Common Event Flag Cluster
/*
/* $ASCEFC efn ,name ,[prot] ,[perm]
/*
/* efn = number of any event flag in the cluster with which to
/* associate
/* name = address of the text name string descriptor
/* prot = protection indicator for the cluster
/* 0 -> default, any process in group
/* 1 -> only owner's UIC
/* perm = permanent indicator
/* 0 -> temporary cluster
/* 1 -> permanent cluster
/*
ENTRY SYSSASCEFC ALIAS $ASCEFC PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN,  (/* TYPE(EFNUM),
CHARACTER DESCRIPTOR NAMED NAME IN,  (/* TYPE(EFCLUSTR),
BOOLEAN VALUE NAMED PROT DEFAULT 0,  (/* TYPE(BOOLEAN),
BOOLEAN VALUE NAMED PERM DEFAULT 0  (/* TYPE(BOOLEAN)
) RETURNS LONGWORD;  (/* TYPE(CONDVALU);

/*
/* $ASCTIM
/*
/* Convert Binary Time to ASCII String
/*
/* $ASCTIM [timlen] ,timbuf ,[timadr] ,[cvtflg]
/*

```

```

/*      timlen = address of a word to receive the number of characters
/*      inserted into the output buffer.
/*      timbuf = address of a character string descriptor describing
/*      the buffer to receive the converted time.
/*      timadr = address of the quadword containing the 64-bit time to
/*      be converted to ASCII. If 0, use current time.
/*      cvtflg = conversion indicator
/*      0 -> return full date and time
/*      1 -> return converted time only
/*
ENTRY SYSSASCTIM ALIAS SASCTIM PARAMETER (
WORD UNSIGNED NAMED TIMLEN OUT DEFAULT 0,      (/* TYPE(NUMBER),
CHARACTER DESCRIPTOR NAMED TIMBUF OUT,      (/* TYPE(TIMEDESC),
QUADWORD UNSIGNED NAMED TIMADR IN DEFAULT 0,      (/* TYPE(QUADTIME),
BOOLEAN VALUE NAMED CVTFLG DEFAULT 0      (/* TYPE(NUMBER)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*
/* $ASCTOID
/*
/* Ascii To Identifier Conversion
/*
/* $ASCTOID name, [id], [attrib]
/*
/* name      = address of the identifier name character
/*            string descriptor to be converted
/* id        = address to return the identifier longword
/* attrib    = address to return the attributes longword
/*
ENTRY SYSSASCTOID ALIAS SASCTOID PARAMETER (
CHARACTER DESCRIPTOR NAMED NAME IN,      (/* TYPE(CHARDESC),
LONGWORD UNSIGNED NAMED ID OUT DEFAULT 0,      (/* TYPE(USERID),
LONGWORD UNSIGNED NAMED ATTRIB OUT DEFAULT 0      (/* TYPE(MASK)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*
/* $ASSIGN
/*
/* Assign I/O Channel
/*
/* $ASSIGN devnam ,chan ,[acmode] ,[mbxnam]
/*
/* devnam = address of device name or logical name string
/*         descriptor
/* chan   = address of word to receive channel number assigned
/* acmode = access mode associated with channel
/* mbxnam = address of mailbox logical name string descriptor, if
/*         mailbox associated with device
/*
ENTRY SYSSASSIGN ALIAS SASSIGN PARAMETER (
CHARACTER DESCRIPTOR NAMED DEVNAM IN,      (/* TYPE(DEVNAME),
WORD UNSIGNED NAMED CHAN OUT,      (/* TYPE(CHANNEL),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0, (/* TYPE(ACMODE),
CHARACTER DESCRIPTOR NAMED MBXNAM IN DEFAULT 0 (/* TYPE(DEVNAME)

```

```

) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*
/* SBINTIM
/*
/* Convert ASCII String to Binary Time
/*
/* SBINTIM timbuf ,timadr
/*
/* timbuf = address of string descriptor for ASCII time string
/* timadr = address of quadword to receive 64-bit binary time
/* value
/*
/* Absolute time strings are specified in the format:
/* dd-mmm-yyyy hh:mm:ss.cc
/* Delta time strings are specified in the format:
/* dddd hh:mm:ss.cc
/*
ENTRY SYSSBINTIM ALIAS SBINTIM PARAMETER (
CHARACTER DESCRIPTOR NAMED TIMBUF IN,      (/* TYPE(TIMEDESC),
QUADWORD UNSIGNED NAMED TIMADR OUT        (/* TYPE(QUADTIME)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*
/* $BRDCST
/*
/* Broadcast
/*
/* $BRDCST msgbuf ,[devnam]
/*
/* msgbuf = address of message buffer string descriptor
/* devnam = terminal device name string descriptor. If 0, send
/* message to all terminals. If first word in
/* descriptor is 0, send message to all allocated
/* terminals.
/*
ENTRY SYSSBRDCST ALIAS $BRDCST PARAMETER (
CHARACTER DESCRIPTOR NAMED MSGBUF IN,      (/* TYPE(CHARDESC),
CHARACTER DESCRIPTOR NAMED DEVNAM IN DEFAULT 0, (/* TYPE(DEVNAME),
LONGWORD UNSIGNED VALUE NAMED FLAGS DEFAULT 0, (/* TYPE(MASK),
LONGWORD UNSIGNED VALUE NAMED CARCON DEFAULT $2 (/* TYPE(NUMBER)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/* $BRKTHRU
/*
/* Write to terminal breakthru
/*
/* $BRKTHRU [efn] ,msgbuf, [sendto], [sndtyp], [iosb],
/* ($BRKTHRUW) [carcon], [flags], [reqid], [timeout],
/* [astadr], [astprm]
/*
/* efn = event flag to be set at completion

```

```

/*
/*   msgbuf = address of message buffer descriptor
/*
/*   sendto = address of send address descriptor
/*
/*   sndtyp = value to describe sendto
/*
/*   iosb   = address of a quadword I/O status block
/*
/*   carcon = carriage control
/*
/*   flags = flags to modify broadcast
/*
/*   reqid  = broadcast class requestor id
/*
/*   timeout = address of timeout value
/*
/*   astadr = address of entry mask of AST routine
/*
/*   astprm = value to be passed to AST routine as an argument
/*
/*

```

```

ENTRY SYS$BRKTHRU ALIAS $BRKTHRU PARAMETER (
LONGWORD UNSIGNED VALUE      NAMED EFN      IN  DEFAULT 0,      { /* TYPE(EFNUM),
CHARACTER DESCRIPTOR         NAMED MSGBUF   IN  DEFAULT 0,      { /* TYPE(CHARDESC),
CHARACTER DESCRIPTOR         NAMED SENDTO   IN  DEFAULT 0,      { /* TYPE(CHARDESC),
LONGWORD UNSIGNED VALUE      NAMED SNDTYP   IN  DEFAULT 0,      { /* TYPE(NUMBER),
QUADWORD UNSIGNED           NAMED IOSB     OUT  DEFAULT 0,      { /* TYPE(IOSB),
LONGWORD UNSIGNED VALUE      NAMED CARCON  IN  DEFAULT 32,     { /* TYPE(NUMBER),
LONGWORD UNSIGNED VALUE      NAMED FLAGS    IN  DEFAULT 0,      { /* TYPE(MASK),
LONGWORD UNSIGNED VALUE      NAMED REQID    IN  DEFAULT 0,      { /* TYPE(NUMBER),
LONGWORD UNSIGNED VALUE      NAMED TIMEOUT  IN  DEFAULT 0,      { /* TYPE(NUMBER),
ADDRESS(ENTRY)               NAMED ASTADR   DEFAULT 0,      { /* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE      NAMED ASTPRM   DEFAULT 0,      { /* TYPE(USERPARM)
) RETURNS LONGWORD;          { /* TYPE(CONDVALU);

```

```

ENTRY SYS$BRKTHRUW ALIAS $BRKTHRUW PARAMETER (
LONGWORD UNSIGNED VALUE      NAMED EFN      IN  DEFAULT 0,      { /* TYPE(EFNUM),
CHARACTER DESCRIPTOR         NAMED MSGBUF   IN  DEFAULT 0,      { /* TYPE(CHARDESC),
CHARACTER DESCRIPTOR         NAMED SENDTO   IN  DEFAULT 0,      { /* TYPE(CHARDESC),
LONGWORD UNSIGNED VALUE      NAMED SNDTYP   IN  DEFAULT 0,      { /* TYPE(NUMBER),
QUADWORD UNSIGNED           NAMED IOSB     OUT  DEFAULT 0,      { /* TYPE(IOSB),
LONGWORD UNSIGNED VALUE      NAMED CARCON  IN  DEFAULT 32,     { /* TYPE(NUMBER),
LONGWORD UNSIGNED VALUE      NAMED FLAGS    IN  DEFAULT 0,      { /* TYPE(MASK),
LONGWORD UNSIGNED VALUE      NAMED REQID    IN  DEFAULT 0,      { /* TYPE(NUMBER),
LONGWORD UNSIGNED VALUE      NAMED TIMEOUT  IN  DEFAULT 0,      { /* TYPE(NUMBER),
ADDRESS(ENTRY)               NAMED ASTADR   DEFAULT 0,      { /* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE      NAMED ASTPRM   DEFAULT 0,      { /* TYPE(USERPARM)
) RETURNS LONGWORD;          { /* TYPE(CONDVALU);

```

```

/*
/* $CANCEL
/*
/*   Cancel I/O on Channel

```

```
/*
/*  $CANCEL chan
/*
/*  chan = number of the channel on which I/O is to be canceled
/*
ENTRY SYSSCANCEL ALIAS $CANCEL PARAMETER (
WORD UNSIGNED VALUE NAMED CHAN { /* TYPE(CHANNEL)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/*
/*  $SCANEXH
/*
/*  Cancel Exit Handler
/*
/*  $SCANEXH [desblk]
/*
/*  desblk = address of exit control block describing exit handler
/*           to be deleted. If 0, delete all.
/*
ENTRY SYSSCANEXH ALIAS $SCANEXH PARAMETER (
ANY NAMED DESBLK IN DEFAULT 0 { /* TYPE(EXHBLOCK)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/*
/*  $SCANTIM
/*
/*  Cancel Timer Request
/*
/*  $SCANTIM [reqidt] ,[acmode]
/*
/*  reqidt = request identification for request to be canceled.
/*           If 0, all requests canceled.
/*  acmode = access mode of requests to be canceled
/*
ENTRY SYSSCANTIM ALIAS $SCANTIM PARAMETER (
LONGWORD UNSIGNED VALUE NAMED REQIDT DEFAULT 0, { /* TYPE(USERPARM),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0 { /* TYPE(ACMODE)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/*
/*  $SCANWAK
/*
/*  Cancel Wakeup
/*
/*  $SCANWAK [pidadr] ,[prcnam]
/*
/*  pidadr = address of process identification of process for
/*           which wakeups are to be canceled
/*  prcnam = address of process name string descriptor
/*
ENTRY SYSSCANWAK ALIAS $SCANWAK PARAMETER (
LONGWORD UNSIGNED NAMED PIDADR IN OUT DEFAULT 0, { /* TYPE(PROCID),
CHARACTER DESCRIPTOR NAMED PRCNAM IN DEFAULT 0 { /* TYPE(PROCNAME)
```

```
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);
```

```
/* $CHANGE_ACL
```

```
/*
```

```
/* Set or modify an object's ACL
```

```
/*
```

```
/* $CHANGE_ACL [chan], objtyp, [objnam],  
/* itmlst, [acmode], [iosb], [contxt]
```

```
/*
```

```
/* chan = number of a channel assigned to the object or  
/* 0 if object is specified by the objnam parameter
```

```
/*
```

```
/* objtyp = address of an object type code
```

```
/*
```

```
/* objnam = address of object name
```

```
/*
```

```
/* itmlst = address of a list of item descriptors
```

```
/*
```

```
/* acmode = address of a byte containing the access mode in  
/* which the arguments will be validated
```

```
/*
```

```
/* iosb = address of a quadword I/O status block
```

```
/*
```

```
/* contxt = address of a context long word (used for iterative  
/* calls or a multi-entry item list)
```

```
/*
```

```
ENTRY SYSSCHANGE_ACL ALIAS $CHANGE_ACL PARAMETER (  
WORD UNSIGNED VALUE NAMED CHAN DEFAULT 0,      (/* TYPE(CHANNEL),  
LONGWORD UNSIGNED NAMED OBJTYP IN,            (/* TYPE(NUMBER),  
CHARACTER DESCRIPTOR NAMED OBJNAM IN DEFAULT 0, (/* TYPE(CHARDESC),  
ANY NAMED ITMLST IN,                          (/* TYPE(ITEMLIST),  
LONGWORD UNSIGNED NAMED ACMODE IN DEFAULT 0,   (/* TYPE (ALMODE)  
QUADWORD UNSIGNED NAMED IOSB OUT DEFAULT 0,    (/* TYPE(IOSB),  
LONGWORD UNSIGNED NAMED CONTXT IN OUT DEFAULT 0 (/* TYPE(CONTEXT),  
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);
```

```
/*
```

```
/* $CHKPRO
```

```
/*
```

```
/* Check Access Protection
```

```
/*
```

```
/* $CHKPRO itmlst
```

```
/*
```

```
/* itmlst = address of a list of item descriptors
```

```
/*
```

```
ENTRY SYSSCHKPRO ALIAS $CHKPRO PARAMETER (  
ANY NAMED ITMLST IN (/* TYPE(ITEMLIST)  
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);
```

```
/*
```

```
/* $CLREF
```

```
/*
```

```
/* Clear Event Flag
```

```
/*
```

EM

EM

```
/*
/*  $CLREF efn
/*
/*  efn    = number of event flag to be cleared
/*
ENTRY SYSSCLREF ALIAS $CLREF PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN    (/* TYPE(EFNUM)
) RETURNS LONGWORD;    (/* TYPE(CONDVALU);

/*
/*  $CLOSE
/*
/*  Close File
/*
/*  $CLOSE fab, [err], [suc]
/*
/*  fab     = address of fab
/*
/*  err     = address of user error completion routine
/*
/*  suc     = address of user success completion routine
/*
ENTRY SYSSCLOSE ALIAS $CLOSE LINKAGE $RMSCALL PARAMETER (
ANY NAMED FAB IN OUT,    (/* TYPE(FAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL,    (/* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL    (/* TYPE(ASTADR)
) RETURNS LONGWORD;    (/* TYPE(CONDVALU);

/*
/*  $CMEXEC
/*
/*  Change to Executive Mode
/*
/*  $CMEXEC  routin ,[arglst]
/*
/*  routin = address of the routine to be executed in executive
/*          mode
/*  arglst = address of argument list to be supplied to the
/*          routine
/*
ENTRY SYSSCMEXEC ALIAS $CMEXEC PARAMETER (
ADDRESS(ENTRY) NAMED ROUTIN,    (/* TYPE(ENTRYADR),
ANY NAMED ARGST IN DEFAULT 0    (/* TYPE(ARGLIST)
) RETURNS LONGWORD;    (/* TYPE(CONDVALU);

/*
/*  $CMKRNL
/*
/*  Change to Kernel Mode
/*
/*  $CMKRNL  routin ,[arglst]
/*
/*  routin = address of routine to be executed in kernel mode
/*
```

```
/*      arglst = address of argument list to be supplied to routine
/*
ENTRY SYSSCMKRNL ALIAS $CMKRNL PARAMETER (
ADDRESS(ENTRY) NAMED ROUTIN,      (/* TYPE(ENTRYADR),
ANY NAMED ARGVLIST IN DEFAULT 0   (/* TYPE(ARGVLIST)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*
/* $CONNECT
/*
/* Connect File
/*
/* $CONNECT rab, [err], [suc]
/*
/* rab      = address of rab
/*
/* err      = address of user error completion routine
/*
/* suc      = address of user success completion routine
/*
ENTRY SYSSCONNECT ALIAS $CONNECT LINKAGE $RMSCALL PARAMETER (
ANY NAMED RAB IN OUT,      (/* TYPE(RAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL,      (/* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL      (/* TYPE(ASTADR)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/* $CNTREG
/*
/* Contract Program/Control Region
/*
/* $CNTREG pagcnt ,[retadr] ,[acmode] ,[region]
/*
/* pagcnt = number of pages to be deleted from end of region
/*
/* retadr = address of 2-longword array to receive virtual
/*          addresses of starting and ending page of deleted area
/*
/* acmode = access mode for which service is performed
/*
/* region = region indicator
/*          0 -> program (P0) region  1 -> control (P1) region
/*
ENTRY SYSSCNTREG ALIAS $CNTREG PARAMETER (
LONGWORD UNSIGNED VALUE NAMED PAGCNT,      (/* TYPE(NUMBER),
LONGWORD UNSIGNED DIMENSION 2 NAMED RETADR OUT DEFAULT 0,      (/* TYPE(VARANGE),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0, (/* TYPE(ACMODE),
BOOLEAN VALUE NAMED REGION DEFAULT 0      (/* TYPE(NUMBER)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*
/* $CREATE
/*
/* Create File
```

```

/*
/* $CREATE fab, [err], [suc]
/*
/* fab = address of fab
/*
/* err = address of user error completion routine
/*
/* suc = address of user success completion routine
/*
ENTRY SYSS$CREATE ALIAS $CREATE LINKAGE $RMSCALL PARAMETER (
ANY NAMED FAB IN OUT, { /* TYPE(FAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL, { /* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL { /* TYPE(ASTADR)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/*
/* $CREATE_RDB
/*
/* Create The Rights Database
/*
/* $CREATE_RDB [sysid]
/*
/* sysid = address of the quadword system identifier
/* to store in the maintenance record
/*
ENTRY SYSS$CREATE RDB ALIAS $CREATE RDB PARAMETER (
QUADWORD UNSIGNED NAMED SYSID IN DEFAULT 0 { /* TYPE(SYSTEMID)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/* $CRELNM
/*
/* Create Logical Name
/*
/* $CRELNM [attr], tabnam, lognam, [acmode], [itemlist]
/*
/* attr = address of logical name attributes
/*
/* Attribute Meaning
/* LNMSM_CONFINE Logical name not to be copied into sub-process
/* LNMSM_NO_ALIAS Logical name can not be aliased
/* LNMSM_CRELOG Logical name created using $CRELOG
/*
/* tabnam = address of logical name table string descriptor
/*
/* lognam = address of logical name string descriptor
/*
/* acmode = address of access mode for logical name
/*
/* itm1st = address of a list of item descriptors
/*
ENTRY SYSS$CRELNM ALIAS $CRELNM PARAMETER (
LONGWORD UNSIGNED NAMED ATTR DEFAULT 0, { /* TYPE(MASK),
CHARACTER DESCRIPTOR NAMED TABNAM IN, { /* TYPE(LOGNAME),
CHARACTER DESCRIPTOR NAMED LOGNAM IN, { /* TYPE(LOGNAME),

```

E

E

E

```

BYTE UNSIGNED NAMED ACMODE DEFAULT 0,  { /* TYPE(ACMODE),
ANY NAMED ITMLST IN DEFAULT 0  { /* TYPE(ITEMLIST)
) RETURNS LONGWORD;  { /* TYPE(CONDVALU);

```

```

/* $CRELNT

```

```

/* Create Logical Name Table
/*
/* $CRELNT [attr], [resnam], [reslen], [quota],
/* [promsk], [tabnam], [acmode], [partab]
/*
/* attr = address of logical name table attributes
/*
/* resnam = address of descriptor of a buffer to receive the
/* created table's name
/*
/* reslen = address of word to receive length of created table's
/* name
/*
/* quota = address of quota associated with logical name table
/*
/* promsk = address of protection mask
/*
/* tabnam = address of descriptor of logical name table name to be created
/*
/* partab = address of name of table parent string descriptor
/*
/* acmode = address of access mode for logical name
/*

```

```

ENTRY SYSSCRELNT ALIAS $CRELNT PARAMETER (
LONGWORD UNSIGNED NAMED ATTR DEFAULT 0, { /* TYPE(MASK),
CHARACTER DESCRIPTOR NAMED RESNAM OUT DEFAULT 0, { /* TYPE(LOGNAME),
WORD UNSIGNED NAMED RESELEN OUT DEFAULT 0, { /* TYPE(NUMBER),
LONGWORD UNSIGNED NAMED QUOTA DEFAULT 0, { /* TYPE(NUMBER),
WORD UNSIGNED NAMED PROMSK DEFAULT 0, { /* TYPE(FILEPROT),
CHARACTER DESCRIPTOR NAMED TABNAM IN DEFAULT 0, { /* TYPE(LOGNAME),
CHARACTER DESCRIPTOR NAMED PARTAB IN DEFAULT 0, { /* TYPE(CHARDESC),
BYTE UNSIGNED NAMED ACMODE DEFAULT 0 { /* TYPE(ACMODE)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

```

```

/* $CRELOG

```

```

/* Create Logical Name
/*
/* $CRELOG [tblflg] ,lognam ,eqlnam ,[acmode]
/*
/* tblflg = logical name table number
/* 0 -> system (default) 1 -> group 2 -> process
/*
/* lognam = address of logical name string descriptor
/*
/* eqlnam = address of equivalence name string descriptor
/*
/* acmode = access mode for logical name (process table only)
/*

```

```
/*
ENTRY SYSSCRELOG ALIAS SCRELOG PARAMETER (
LONGWORD UNSIGNED VALUE NAMED TBLFLG DEFAULT 0, { /* TYPE(NUMBER),
CHARACTER DESCRIPTOR NAMED LOGNAM IN, { /* TYPE(LOGNAME),
CHARACTER DESCRIPTOR NAMED EQLNAM IN, { /* TYPE(LOGNAME),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0 { /* TYPE(ACMODE)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);
```

```
/* SCREMBX
```

```
/*
```

```
/* Create Mailbox and Assign Channel
```

```
/*
```

```
/* SCREMBX [prmflg] ,chan ,[maxmsg] ,[bufquo] ,[promsk] ,[acmode] ,[lognam]
```

```
/*
```

```
/* prmflg = permanent flag
```

```
/* 0 -> temporary (default) 1 -> permanent
```

```
/*
```

```
/* chan = address of word to receive channel
```

```
/*
```

```
/* maxmsg = maximum message size that may be received
```

```
/*
```

```
/* bufquo = number of bytes that can be used to buffer messages
```

```
/*
```

```
/* promsk = protection mask
```

```
/*
```

```
/* acmode = access mode of created mailbox
```

```
/*
```

```
/* lognam = address of logical name string descriptor for mailbox
```

```
/*
```

```
ENTRY SYSSCREMBX ALIAS SCREMBX PARAMETER (
BOOLEAN VALUE NAMED PRMFLG DEFAULT 0, { /* TYPE(BOOLEAN),
WORD UNSIGNED NAMED CHAN OUT, { /* TYPE(CHANNEL),
LONGWORD UNSIGNED VALUE NAMED MAXMSG DEFAULT 0, { /* TYPE(NUMBER),
LONGWORD UNSIGNED VALUE NAMED BUFQUO DEFAULT 0, { /* TYPE(NUMBER),
LONGWORD UNSIGNED VALUE NAMED PROMSK DEFAULT 0, { /* TYPE(FILEPROT),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0, { /* TYPE(ACMODE),
CHARACTER DESCRIPTOR NAMED LOGNAM IN DEFAULT 0 { /* TYPE(LOGNAME)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);
```

```
/* SCREPRC
```

```
/*
```

```
/* Create Process
```

```
/*
```

```
/* SCREPRC [pidadr] ,[image] ,[input] ,[output] ,[error] ,[prvadr]
/* ,[quota] ,[prcnam] ,[baspri] ,[uic] ,[mbxunt] ,[stsflg] ,[itmlst]
```

```
/*
```

```
/* pidadr = address of longword to return id of created process
```

```
/*
```

```
/* image = address of string descriptor for image name
```

```
/*
```

```
/* input = address of string descriptor for SYSS$INPUT
```

```
/*
```

```
/* output = address of string descriptor for SYSS$OUTPUT
```

```
/*
```

E

E

E

```

/* error = address of string descriptor for SYS$ERROR
/*
/* prvadr = address of quadword privilege list
/*
/* quota = address of quota list
/*
/* prcnam = address of string descriptor for process name
/*
/* baspri = base priority (0-31)
/*
/* uic = user identification code. If 0, create a subprocess
/*
/* mbxunt = mailbox unit for termination message
/*
/* stsflg = status and mode flag bits
/*
/*      Bit  Meaning
/*
/*      0  disable resource wait mode
/*      1  enable system service failure exception mode
/*      2  inhibit process swapping
/*      3  disable accounting messages
/*      4  batch process
/*      5  cause created process to hibernate
/*      6  allow login without authorization file check
/*      7  process is a network connect object
/*
/*      itmlst = address of a list of item descriptors
/*
ENTRY SYSSCREPRC ALIAS $CREPRC PARAMETER (
LONGWORD UNSIGNED NAMED PIDADR OUT DEFAULT 0,  { /* TYPE(PROCID),
CHARACTER DESCRIPTOR NAMED IMAGE IN DEFAULT 0,  { /* TYPE(LOGNAME),
CHARACTER DESCRIPTOR NAMED INPUT IN DEFAULT 0,  { /* TYPE(LOGNAME),
CHARACTER DESCRIPTOR NAMED OUTPUT IN DEFAULT 0, { /* TYPE(LOGNAME),
CHARACTER DESCRIPTOR NAMED ERROR IN DEFAULT 0,  { /* TYPE(LOGNAME),
QUADWORD UNSIGNED NAMED PRVADR IN DEFAULT 0,    { /* TYPE(PRIVMASK),
ANY NAMED QUOTA IN DEFAULT 0,  { /* TYPE(LIST),
CHARACTER DESCRIPTOR NAMED PRCNAM IN DEFAULT 0, { /* TYPE(PROCNAME),
LONGWORD UNSIGNED VALUE NAMED BASPRI DEFAULT 2, { /* TYPE(NUMBER),
LONGWORD UNSIGNED VALUE NAMED UIC DEFAULT 0,    { /* TYPE(USERID),
WORD UNSIGNED VALUE NAMED MBXUNT DEFAULT 0,     { /* TYPE(NUMBER),
LONGWORD UNSIGNED VALUE NAMED STSFLG DEFAULT 0, { /* TYPE(MASK),
ANY NAMED ITMLST IN DEFAULT 0  { /* TYPE(ITEMLIST)
) RETURNS LONGWORD;  { /* TYPE(CONDVALU);

/* $CRETVA
/*
/* Create Virtual Address Space
/*
/* $CRETVA inadr ,[retadr] ,[acmode]
/*
/* inadr = address of 2-longword array containing starting and
/* ending virtual address of pages to be created
/*
/* retadr = address of a 2-longword array to receive starting and

```

```

/*      ending virtual address of pages actually created
/*
/*      acmode = access mode for the new pages (protection is
/*      read/write for acmode and more privileged modes)
/*
ENTRY SY$CRETVA ALIAS $CRETVA PARAMETER (
LONGWORD UNSIGNED DIMENSION 2 NAMED INADR IN,      { /* TYPE(VARANGE),
LONGWORD UNSIGNED DIMENSION 2 NAMED RETADR OUT DEFAULT 0, { /* TYPE(VARANGE),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0 { /* TYPE(ACMODE)
) RETURNS LONGWORD;      { /* TYPE(CONDVALU);

/* $CRMPSC
/*
/* Create and Map Section
/*
/* $CRMPSC [inadr] ,[retadr] ,[acmode] ,[flags] ,[gsdnam] ,[ident]
/*      ,[relpag] ,[chan] ,[pagcnt] ,[vbn] ,[prot] ,[pfc]
/*
/* inadr = address of 2-longword array containing starting and
/*      ending virtual addresses of space to map section
/*
/* retadr = address of 2-longword array to receive addresses
/*      actually mapped
/*
/* acmode = access mode of owner of pages
/*
/* flags = section characteristics
/*
/*      Flag      Meaning
/*
/*      SECSM_GBL   Global section
/*      SECSM_CRF   Copy-on-reference pages
/*      SECSM_DZRO  Demand zero pages
/*      SECSM_EXPREG Find first available space
/*      SECSM_PERM  Permanent section
/*      SECSM_PFNMAP Physical page frame section
/*      SECSM_SYSGBL System global section
/*      SECSM_WRT   Read/write section
/*
/* gsdnam = address of global section name string descriptor
/*
/* ident = address of quadword containing version id and match control
/*
/* relpag = relative page number within section
/*
/* chan = number of channel on which file is accessed
/*
/* pagcnt = number of pages in section
/*
/* vbn = virtual block number of beginning of section or
/*      physical page frame number of beginning of section
/*
/* prot = protection mask
/*
/* pfc = page fault cluster size

```

```
/*
ENTRY SYSSCRMPSC ALIAS SCRMPSC PARAMETER (
LONGWORD UNSIGNED DIMENSION 2 NAMED INADR IN DEFAULT 0, { /* TYPE(VARANGE),
LONGWORD UNSIGNED DIMENSION 2 NAMED RETADR OUT DEFAULT 0, { /* TYPE(VARANGE),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0, { /* TYPE(ACMODE),
LONGWORD UNSIGNED VALUE NAMED FLAGS DEFAULT 0, { /* TYPE(MASK),
CHARACTER DESCRIPTOR NAMED GSDNAM IN DEFAULT 0, { /* TYPE(SECTNAME),
QUADWORD UNSIGNED NAMED 'IDENT' IN DEFAULT 0, { /* TYPE(SECTID),
LONGWORD UNSIGNED VALUE NAMED RELPAG DEFAULT 0, { /* TYPE(NUMBER),
WORD UNSIGNED VALUE NAMED CHAN DEFAULT 0, { /* TYPE(CHANNEL),
LONGWORD UNSIGNED VALUE NAMED PAGCNT DEFAULT 0, { /* TYPE(NUMBER),
LONGWORD UNSIGNED VALUE NAMED VBN DEFAULT 0, { /* TYPE(NUMBER),
LONGWORD UNSIGNED VALUE NAMED PROT DEFAULT 0, { /* TYPE(FILEPROT),
LONGWORD UNSIGNED VALUE NAMED PFC DEFAULT 0 { /* TYPE(NUMBER)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);
```

```
/* $DACEFC
/*
/* Disassociate Common Event Flag Cluster
/*
/* $DACEFC efn
/*
/* efn = number of any event flag in the cluster
/*
```

```
ENTRY SYSSDACEFC ALIAS $DACEFC PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN { /* TYPE(EFNUM)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);
```

```
/* $DALLOC
/*
/* Deallocate Device
/*
/* $DALLOC [devnam] [,acmode]
/*
/* devnam = address of device name descriptor. If 0, deallocate all
/*
/* acmode = access mode associated with device
/*
```

```
ENTRY SYSSDALLOC ALIAS $DALLOC PARAMETER (
CHARACTER DESCRIPTOR NAMED DEVNAM IN DEFAULT 0, { /* TYPE(DEVNAME),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0 { /* TYPE(ACMODE)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);
```

```
/* $DASSGN
/*
/* Deassign I/O Channel
/*
/* $DASSGN chan
/*
/* chan = number of channel to be deassigned
/*
```

```
ENTRY SYSSDASSGN ALIAS $DASSGN PARAMETER (
WORD UNSIGNED VALUE NAMED CHAN { /* TYPE(CHANNEL)
```

E

E

E

```

) RETURNS LONGWORD;      { /* TYPE(CONDVALU);

/* $DCLAST
/*
/* Declare AST
/*
/* $DCLAST  astadr ,[astprm] ,[acmode]
/*
/* astadr = address of entry mask of AST routine
/*
/* astprm = value to be passed to AST routine
/*
/* acmode = access mode for which the AST is to be declared
/*
ENTRY SYSSDCLAST ALIAS $DCLAST PARAMETER (
ADDRESS(ENTRY) NAMED ASTADR,      { /* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0, { /* TYPE(USERPARM),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0 { /* TYPE(ACMODE)
) RETURNS LONGWORD;      { /* TYPE(CONDVALU);

/* $DCLCMH
/*
/* Declare Change Mode or Compatibility Mode Handler
/*
/* $DCLCMH  address ,[prvhnd] ,[type]
/*
/* address = address of handler
/*
/* prvhnd = address of longword to receive previous handler address
/*
/* type    = handler type indicator
/*          0 -> change mode (current mode)  1 -> compatibility mode
/*
ENTRY SYSSDCLCMH ALIAS $DCLCMH PARAMETER (
ADDRESS(ENTRY) NAMED ADDRESS,      { /* TYPE(ENTRYADR),
LONGWORD UNSIGNED NAMED PRVHND OUT DEFAULT 0, { /* TYPE(NUMBER),
BOOLEAN VALUE NAMED "TYPE" DEFAULT 0 { /* TYPE(NUMBER)
) RETURNS LONGWORD;      { /* TYPE(CONDVALU);

/* $DCLEXH
/*
/* Declare Exit Handler
/*
/* $DCLEXH  desblk
/*
/* desblk = address of exit control block containing:
/*
/* -----
/*          | forward link |
/*          |-----|
/*          | exit handler address |
/*          |-----|
/*          | argument count |
/*          |-----|

```

E

E

```

/*      +-----+
/*      | address to store reason |
/*      +-----+
/*      | additional arguments   |
/*      | for exit handler,     |
/*      | if any                 |
/*      +-----+
/*
ENTRY SYSSDCLEXH ALIAS $DCLEXH PARAMETER (
ANY NAMED DESBLK IN  (/* TYPE(EXHBLOCK)
) RETURNS LONGWORD;  (/* TYPE(CONDVALU);

/*
/* $DELETE
/*
/* Delete Record
/*
/* $DELETE rab, [err], [suc]
/*
/* rab = address of rab
/*
/* err = address of user error completion routine
/*
/* suc = address of user success completion routine
/*
ENTRY SYSSDELETE ALIAS $DELETE LINKAGE $RMSCALL PARAMETER (
ANY NAMED RAB IN OUT, (/* TYPE(RAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL, (/* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL (/* TYPE(ASTADR)
) RETURNS LONGWORD; (/* TYPE(CONDVALU);

/* $DELLNM
/*
/* Delete Logical Name and/or Table
/*
/* $DELLNM [tabnam], [lognam], [acmode]
/*
/* tabnam = address of descriptor of logical name table name string
/*
/* lognam = address of descriptor of logical name string
/*
/* acmode = address of access mode for logical name
/*
ENTRY SYSSDELLNM ALIAS $DELLNM PARAMETER (
CHARACTER DESCRIPTOR NAMED TABNAM IN DEFAULT 0, (/* TYPE(LOGNAME),
CHARACTER DESCRIPTOR NAMED LOGNAM IN DEFAULT 0, (/* TYPE(LOGNAME),
BYTE UNSIGNED NAMED ACMODE DEFAULT 0 (/* TYPE(ACMODE)
) RETURNS LONGWORD; (/* TYPE(CONDVALU);

/* $DELLOG
/*
/* Delete Logical Name
/*

```

```
/* $DELLOG [tblflg] ,[lognam] ,[acmode]
/*
/* tblflg = logical name table number
/*          0 -> system  1 -> group  2 -> process
/*
/* lognam = address of logical name string descriptor.  If 0,
/*          delete all names
/*
/* acmode = access mode of logical name (process table only)
/*
ENTRY SYSSDELLOG ALIAS $DELLOG PARAMETER (
LONGWORD UNSIGNED VALUE NAMED TBLFLG DEFAULT 0, { /* TYPE(NUMBER)
CHARACTER DESCRIPTOR NAMED LOGNAM IN DEFAULT 0, { /* TYPE(LOGNAME)
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0 { /* TYPE(ACMODE)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/* $DELMBX
/*
/* Delete Mailbox
/*
/* $DELMBX chan
/*
/* chan = channel number assigned to the mailbox
/*
ENTRY SYSSDELMBX ALIAS $DELMBX PARAMETER (
WORD UNSIGNED VALUE NAMED CHAN { /* TYPE(CHANNEL)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/* $DELPRC
/*
/* Delete Process
/*
/* $DELPRC [pidadr] ,[prcnam]
/*
/* pidadr = address of longword containing id of process to be deleted
/*
/* prcnam = address of string descriptor for process name to be deleted
/*
ENTRY SYSSDELPRC ALIAS $DELPRC PARAMETER (
LONGWORD UNSIGNED NAMED PIDADR IN OUT DEFAULT 0, { /* TYPE(PROCID),
CHARACTER DESCRIPTOR NAMED PRCNAM IN DEFAULT 0 { /* TYPE(PROCNAME)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/* $DELTVA
/*
/* Delete Virtual Address Space
/*
/* $DELTVA inadr ,[retadr] ,[acmode]
/*
/* inadr = address of 2-longword array containing starting and
/*          ending virtual addresses of pages to delete
/*
/* retadr = address of 2-longword array to receive starting and
```

```
/*          ending addresses of pages actually deleted
/*
/*          acmode = access mode for which service is performed
/*
ENTRY SYSSDELTA ALIAS $DELTA PARAMETER (
LONGWORD UNSIGNED DIMENSION 2 NAMED INADR IN,  { /* TYPE(VARANGE),
LONGWORD UNSIGNED DIMENSION 2 NAMED RETADR OUT DEFAULT 0, { /* TYPE(VARANGE),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0 { /* TYPE(ACMODE)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/* $DEQ
/*
/* Dequeue Lock
/*
/* $DEQ [lkid] ,[valblk] ,[acmode] ,[flags]
/*
/* lkid = lock ID of the lock to be dequeued
/*
/* valblk = address of the lock value block
/*
/* acmode = access mode of the locks to be dequeued
/*
/* flags = optional flags.
/*
/*          LCKSM_DEQALL
/*
ENTRY SYSSDEQ ALIAS $DEQ PARAMETER (
LONGWORD UNSIGNED VALUE NAMED LKID DEFAULT 0, { /* TYPE(LOCKID),
ANY NAMED VALBLK IN OUT DEFAULT 0, { /* TYPE(LOCKVALU),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0, { /* TYPE(ACMODE),
LONGWORD UNSIGNED VALUE NAMED FLAGS DEFAULT 0 { /* TYPE(MASK)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/* $DGBLSC
/*
/* Delete Global Section
/*
/* $DGBLSC [flags] ,gsdnam ,[ident]
/*
/* flags = type of section
/*          0 -> group section  SEC$M_SYSGBL -> system section
/*
/* gsdnam = address of global section name string descriptor
/*
/* ident = address of quadword containing version id and match control
/*
ENTRY SYSSDGBLSC ALIAS $DGBLSC PARAMETER (
LONGWORD UNSIGNED VALUE NAMED FLAGS DEFAULT 0, { /* TYPE(MASK),
CHARACTER DESCRIPTOR NAMED GSDNAM IN, { /* TYPE(SECTNAME),
QUADWORD UNSIGNED NAMED 'IDENT' IN DEFAULT 0 { /* TYPE(SECTID)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/*
```

```

/* $DISCONNECT
/*
/* Disconnect Record Stream
/*
/* $DISCONNECT rab, [err], [suc]
/*
/* rab = address of rab
/*
/* err = address of user error completion routine
/*
/* suc = address of user success completion routine
/*
ENTRY SYSSDISCONNECT ALIAS $DISCONNECT LINKAGE $RMSCALL PARAMETER (
ANY NAMED RAB IN OUT,      (/* TYPE(RAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL,    (/* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL     (/* TYPE(ASTADR)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/* $DISMOU
/*
/* Dismount Volume
/*
/* $DISMOU devnam ,[flags]
/*
/* devnam = address of device name string descriptor
/*
/* flags = 32-bit status mask selecting options for the dismount
/* The symbols for the flags are defined by the $DMTDEF
/* macro.
/*
/* Flag      Meaning
/*
/* DMTSM_NOUNLOAD  Do not unload the volume.
/*
/* DMTSM_UNIT      Dismount the specified device, rather
/* than the entire volume set.
/*
ENTRY SYSSDISMOU ALIAS $DISMOU PARAMETER (
CHARACTER DESCRIPTOR NAMED DEVNAM IN,  (/* TYPE(DEVNAME),
LONGWORD UNSIGNED VALUE NAMED FLAGS DEFAULT 0 (/* TYPE(MASK)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*
/* $DISPLAY
/*
/* Display File
/*
/* $DISPLAY fab, [err], [suc]
/*
/* fab = address of fab
/*
/* err = address of user error completion routine
/*
/* suc = address of user success completion routine
/*

```

EI

E

```

ENTRY SYSSDISPLAY ALIAS $DISPLAY LINKAGE $RMSCALL PARAMETER (
  ANY NAMED FAB IN OUT,      (/* TYPE(FAB),
  ADDRESS(ENTRY) NAMED ERR OPTIONAL,    (/* TYPE(ASTADR),
  ADDRESS(ENTRY) NAMED SUC OPTIONAL      (/* TYPE(ASTADR)
) RETURNS LONGWORD:      (/* TYPE(CONDVALU);

```

```

/* $DLCEFC

```

```

/* Delete Common Event Flag Cluster

```

```

/* $DLCEFC name

```

```

/* name = address of cluster name string descriptor

```

```

ENTRY SYSSDLCEFC ALIAS $DLCEFC PARAMETER (
  CHARACTER DESCRIPTOR NAMED NAME IN      (/* TYPE(EFCLUSTER)
) RETURNS LONGWORD:      (/* TYPE(CONDVALU);

```

```

/* $ENQ

```

```

/* Enqueue Lock Request

```

```

/* $ENQ [efn] ,lkmode ,lksb ,[flags] ,[resnam] ,[parid]
/* ($ENQW) ,[astadr] ,[astprm] ,[blkast] ,[acmode] ,[prot]

```

```

/* efn = event flag to be set at completion

```

```

/* lkmode = type of lock mode requested. Modes are:

```

```

/* LCK$K_NLMODE null lock
/* LCK$K_CRMODE concurrent read
/* LCK$K_CWMODE concurrent write
/* LCK$K_PRMODE protected read
/* LCK$K_PWMODE protected write
/* LCK$K_EXMODE exclusive lock

```

```

/* lksb = address of the lock status block

```

```

/* flags = flags defining the characteristics of the lock. These are:

```

```

/* LCK$M_NOQUEUE
/* LCK$M_SYNCSTS
/* LCK$M_SYSTEM
/* LCK$M_VALBLK
/* LCK$M_CONVERT

```

```

/* resnam = address of string descriptor of the resource name

```

```

/* parid = lock ID of the parent lock

```

```

/* astadr = address of entry mask of AST routine

```

```

/* astprm = value to be passed to AST routine

```

```

/* blkast = address of entry mask of blocking AST routine

```

```

/*
/*  acmode = Access mode to be associated with the lock
/*
/*  prot  = optional additional arg for extension
/*
ENTRY SY$SENQ ALIAS $ENQ PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0,      (/* TYPE(EFNUM),
LONGWORD UNSIGNED VALUE NAMED LKMODE,           (/* TYPE(NUMBER),
ANY NAMED LKSB OUT,                             (/* TYPE(LOCKSTAT),
LONGWORD UNSIGNED VALUE NAMED FLAGS DEFAULT 0,   (/* TYPE(MASK),
CHARACTER DESCRIPTOR NAMED RESNAM IN DEFAULT 0,  (/* TYPE(CHARDÉSC),
LONGWORD UNSIGNED VALUE NAMED PARID DEFAULT 0,   (/* TYPE(LOCKID),
ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0,          (/* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0,  (/* TYPE(USERPARM),
ADDRESS(ENTRY) NAMED BLKAST DEFAULT 0,          (/* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0,  (/* TYPE(ACMODE),
LONGWORD UNSIGNED VALUE NAMED PROT DEFAULT 0    (/* TYPE(NULLARG)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

ENTRY SY$SENQW ALIAS $ENQW PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0,      (/* TYPE(EFNUM),
LONGWORD UNSIGNED VALUE NAMED LKMODE,           (/* TYPE(NUMBER),
ANY NAMED LKSB OUT,                             (/* TYPE(LOCKSTAT),
LONGWORD UNSIGNED VALUE NAMED FLAGS DEFAULT 0,   (/* TYPE(MASK),
CHARACTER DESCRIPTOR NAMED RESNAM IN DEFAULT 0,  (/* TYPE(CHARDÉSC),
LONGWORD UNSIGNED VALUE NAMED PARID DEFAULT 0,   (/* TYPE(LOCKID),
ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0,          (/* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0,  (/* TYPE(USERPARM),
ADDRESS(ENTRY) NAMED BLKAST DEFAULT 0,          (/* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0,  (/* TYPE(ACMODE),
LONGWORD UNSIGNED VALUE NAMED PROT DEFAULT 0    (/* TYPE(NULLARG)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*
/* $ENTER
/*
/*  Enter file
/*
/*  $ENTER fab, [err], [suc]
/*
/*  fab      = address of fab
/*
/*  err      = address of user error completion routine
/*
/*  suc      = address of user success completion routine
/*
ENTRY SY$SE$ENTER ALIAS $ENTER LINKAGE $RMSCALL PARAMETER (
ANY NAMED FAB IN OUT,      (/* TYPE(FAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL,      (/* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL,      (/* TYPE(ASTADR)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*
/* $ERAPAT
/*
/*  Generate a security erase pattern.

```

```
/*
/*  $SERPAT type, [count], patadr
/*
/*  type    = type of security erase
/*
/*  count   = iteration count (seed for erase pattern)
/*
/*  patadr  = address of longword to receive erase pattern
/*
ENTRY SYSSERPAT ALIAS $SERPAT PARAMETER (
LONGWORD VALUE NAMED TYPE DEFAULT 0,    { /* TYPE(NUMBER),
LONGWORD UNSIGNED VALUE NAMED COUNT DEFAULT 1, { /* TYPE(NUMBER),
LONGWORD UNSIGNED NAMED PATADR OUT DEFAULT 0 { /* TYPE(NUMBER)
) RETURNS LONGWORD;    { /* TYPE(CONDVALU);

/*  $ERASE
/*
/*  Erase File
/*
/*  $EPASE fab, [err], [suc]
/*
/*  fab      = address of fab
/*
/*  err      = address of user error completion routine
/*
/*  suc      = address of user success completion routine
/*
ENTRY SYSSERASE ALIAS $ERASE LINKAGE $RMSCALL PARAMETER (
ANY NAMED FAB IN OUT,    { /* TYPE(FAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL,    { /* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL    { /* TYPE(ASTADR)
) RETURNS LONGWORD;    { /* TYPE(CONDVALU);

/*  $EXIT
/*
/*  Exit image
/*
/*  $EXIT [code]
/*
/*  code    = longword completion status
/*
ENTRY SYSEXIT ALIAS $EXIT PARAMETER (
LONGWORD UNSIGNED VALUE NAMED CODE DEFAULT 1    { /* TYPE(CONDVALU)
) RETURNS LONGWORD;    { /* TYPE(CONDVALU);

/*  $EXPREG
/*
/*  Expand Program/Control Region
/*
/*  $EXPREG pagcnt ,[retadr] ,[acmode] ,[region]
/*
/*  pagcnt = number of pages to add to end of specified region
/*
```

```

/*      retadr = address of 2-longword array to receive virtual
/*      addresses of starting and ending pages
/*
/*      acmode = access mode of the new pages
/*
/*      region = region indicator
/*      0 -> program (P0) region  1 -> control (P1) region
/*
ENTRY SYS$EXPREG ALIAS $EXPREG PARAMETER (
LONGWORD UNSIGNED VALUE NAMED PAGCNT,  { /* TYPE(NUMBER),
LONGWORD UNSIGNED DIMENSION 2 NAMED RETADR OUT DEFAULT 0, { /* TYPE(VARANGE),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0, { /* TYPE(ACMODE),
BOOLEAN VALUE NAMED REGION DEFAULT 0 { /* TYPE(NUMBER)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/*
/* $EXTEND
/*
/* Extend File
/*
/* $EXTEND fab, [err], [suc]
/*
/* fab      = address of fab
/*
/* err      = address of user error completion routine
/*
/* suc      = address of user success completion routine
/*
ENTRY SYS$EXTEND ALIAS $EXTEND LINKAGE $RMSCALL PARAMETER (
ANY NAMED FAB IN OUT, { /* TYPE(FAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL, { /* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL { /* TYPE(ASTADR)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/* $FAO
/*
/* Formatted ASCII Output
/*
/* $FAO ctrstr ,[outlen] ,outbuf ,[p1] ,[p2]...[pn]
/*
/* ctrstr = address of string descriptor for control string
/*
/* outlen = address of word in which to store output length
/*
/* outbuf = address of output buffer string descriptor
/*
/* p1...  = variable number of arguments to FAO
/*
ENTRY SYSS$FAO ALIAS $FAG PARAMETER (
CHARACTER DESCRIPTOR NAMED CTRSTR IN, { /* TYPE(CHARDESC),
WORD UNSIGNED NAMED OUTLEN OUT DEFAULT 0, { /* TYPE(NUMBER),
CHARACTER DESCRIPTOR NAMED OUTBUF OUT, { /* TYPE(CHARDESC),
LONGWORD VALUE NAMED P1 { /* TYPE(VARIABLES)
) VARIABLE RETURNS LONGWORD; { /* TYPE(CONDVALU);

```

```
/* $FAOL
/*
/* Formatted ASCII Output With List Parameter
/*
/* $FAOL  ctrstr ,[outlen] ,outbuf ,prmlst
/*
/* ctrstr = address of string descriptor for control string
/*
/* outlen = address of word to receive output string length
/*
/* outbuf = address of output buffer string descriptor
/*
/* prmlst = address of a list of longword parameters
/*
ENTRY SYSS$FAOL ALIAS $FAOL PARAMETER (
CHARACTER DESCRIPTOR NAMED CTRSTR IN,  { /* TYPE(CHARDESC),
WORD UNSIGNED NAMED OUTLEN OUT DEFAULT 0,  { /* TYPE(NUMBER),
CHARACTER DESCRIPTOR NAMED OUTBUF OUT,  { /* TYPE(CHARDESC),
ANY NAMED PRMLST IN  { /* TYPE(VECTOR)
) RETURNS LONGWORD;  { /* TYPE(CONDVALU);

/* $FILESCAN
/*
/* Scan a string and identify a file specification
/*
/* $FILESCAN  srcstr, [valuelst], [fldflags]
/*
/* srcstr = address of string descriptor for source string
/*
/* valuelst = address of a list of item descriptors
/*
/* fldflags = address of a longword to receive flags
/*
ENTRY SYSS$FILESCAN ALIAS $FILESCAN PARAMETER (
CHARACTER DESCRIPTOR NAMED SRCSTR IN,  { /* TYPE(CHARDESC),
ANY NAMED VALUELST IN OUT DEFAULT 0,  { /* TYPE(LIST),
LONGWORD UNSIGNED NAMED FLDFLAGS OUT DEFAULT 0  { /* TYPE(MASK)
) RETURNS LONGWORD;  { /* TYPE(CONDVALU);

/*
/* $FIND
/*
/* Find Record in File
/*
/* $FIND rab, [err], [suc]
/*
/* rab      = address of rab
/*
/* err      = address of user error completion routine
/*
/* suc      = address of user success completion routine
/*
ENTRY SYSS$FIND ALIAS $FIND LINKAGE $RMSCALL PARAMETER (
ANY NAMED RAB IN OUT,  { /* TYPE(RAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL,  { /* TYPE(ASTADR),
```

E

E

```
ADDRESS(ENTRY) NAMED SUC OPTIONAL      (/* TYPE(ASTADR)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);
```

```
/*
/* $FIND_HELD
/*
/* Find Identifiers Held By Holder
/*
/* $FIND_HELD holder, [id], [attrib], [contxt]
/*
/* holder = address of a quadword which specifies the holder
/*          id of the records to find
/* id      = address to return the identifier longword
/* attrib  = address to return the holder attributes longword
/* contxt  = address of a longword containing the record stream
/*          context. initially should be zero, value output
/*          on first call, value input on subsequent calls.
/*
```

```
ENTRY SYSS$FIND_HELD ALIAS $FIND_HELD PARAMETER (
QUADWORD UNSIGNED NAMED HOLDER IN,      (/* TYPE(HOLDER),
LONGWORD UNSIGNED NAMED ID OUT DEFAULT 0, (/* TYPE(USERID),
LONGWORD UNSIGNED NAMED ATTRIB OUT DEFAULT 0, (/* TYPE(MASK),
LONGWORD UNSIGNED NAMED CONTXT IN OUT DEFAULT 0 (/* TYPE(CONTEXT)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);
```

```
/*
/* $FIND_HOLDER
/*
/* Find Holder Of Identifier
/*
/* $FIND_HOLDER id, [holder], [attrib], [contxt]
/*
/* id      = identifier longword whose holder records
/*          are to be found
/* holder  = address to return the holder id quadword
/* attrib  = address to return the attributes longword
/* contxt  = address of a longword containing the record stream
/*          context. initially should be zero, value output
/*          on first call, value input on subsequent calls.
/*
```

```
ENTRY SYSS$FIND_HOLDER ALIAS $FIND_HOLDER PARAMETER (
LONGWORD UNSIGNED VALUE NAMED ID IN,      (/* TYPE(USERID),
QUADWORD UNSIGNED NAMED HOLDER OUT DEFAULT 0, (/* TYPE(HOLDER),
LONGWORD UNSIGNED NAMED ATTRIB OUT DEFAULT 0, (/* TYPE(MASK),
LONGWORD UNSIGNED NAMED CONTXT IN OUT DEFAULT 0 (/* TYPE(CONTEXT)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);
```

```
/*
/* $FINISH_RDB
/*
/* Clean up RMS Stream
/*
/* $FINISH_RDB contxt
```

```

/*
/*      contxt = address of a longword containing the record stream
/*      context.
/*
ENTRY SYSS$FINISH RDB ALIAS $FINISH RDB PARAMETER (
LONGWORD UNSIGNED NAMED CONTEXT IN OUT  { /* TYPE(CONTEXT)
) RETURNS LONGWORD;      { /* TYPE(CONDVALU);

/* $FORCEX
/*
/*      Force Exit
/*
/*      $FORCEX [p:adr] ,[prcnam] ,[code]
/*
/*      pidadr = address of process id of process to be forced to exit
/*
/*      prcnam = address of process name descriptor for forced process
/*
/*      code   = longword completion status for exit service
/*
ENTRY SYSS$FORCEX ALIAS $FORCEX PARAMETER (
LONGWORD UNSIGNED NAMED PIDADR IN OUT DEFAULT 0,      { /* TYPE(PROCID),
CHARACTER DESCRIPTOR NAMED PRCNAM IN DEFAULT 0, { /* TYPE(PROCNAME),
LONGWORD UNSIGNED VALUE NAMED CODE DEFAULT 0      { /* TYPE(CONDVALU)
) RETURNS LONGWORD;      { /* TYPE(CONDVALU);

/*
/* $FORMAT_ACL
/*
/*      Format an Access Control List Entry
/*
/*      $FORMAT_ACL acl-entry, [acl-length], acl-string,
/*      [line-width], [term-desc], [line-indent],
/*      [bit-names]
/*
/*      acl-entry      = address of a descriptor of a buffer which
/*                      contains the ACE
/*
/*      acl-length     = address of a word to receive the length of
/*                      the output string
/*
/*      acl-string     = address of a descriptor of a buffer into
/*                      which the output string is to be stored
/*
/*      line-width     = address of the maximum line width
/*                      (0 = infinite)
/*
/*      term_desc      = address of a character descriptor containing
/*                      a character string to be inserted whenever
/*                      the line segment length exceeds the line-width
/*
/*      line-indent    = address of the number of columns to indent
/*                      the output line segment
/*

```

```
/*      bit-names      = address of a access bit name table (32 entries)
/*
ENTRY SYSS$FORMAT ACL ALIAS $FORMAT ACL PARAMETER (
  CHARACTER DESCRIPTOR NAMED ACLENT IN,      (/* TYPE(CHARDESC),
  WORD UNSIGNED NAMED ACLEN OUT DEFAULT 0,   (/* TYPE(NUMBER),
  CHARACTER DESCRIPTOR NAMED ACLSTR OUT,    (/* TYPE(CHARDESC),
  WORD UNSIGNED NAMED WIDTH IN DEFAULT 0,   (/* TYPE(NUMBER),
  CHARACTER DESCRIPTOR NAMED TRMDSC IN DEFAULT 0, (/* TYPE(CHARDESC),
  WORD UNSIGNED NAMED INDENT IN DEFAULT 0,   (/* TYPE(NUMBER),
  ANY NAMED ACCNAM IN DEFAULT 0 (/* TYPE(VECTOR)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*
/* $FREE
/*
/* Free Record
/*
/* $FREE rab, [err], [suc]
/*
/* rab      = address of fab
/*
/* err      = address of user error completion routine
/*
/* suc      = address of user success completion routine
/*
ENTRY SYSS$FREE ALIAS $FREE LINKAGE $RMSCALL PARAMETER (
  ANY NAMED RAB IN OUT,      (/* TYPE(RAB),
  ADDRESS(ENTRY) NAMED ERR OPTIONAL,      (/* TYPE(ASTADR),
  ADDRESS(ENTRY) NAMED SUC OPTIONAL      (/* TYPE(ASTADR)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*
/* $FLUSH
/*
/* Flush Record
/*
/* $FLUSH rab, [err], [suc]
/*
/* rab      = address of rab
/*
/* err      = address of user error completion routine
/*
/* suc      = address of user success completion routine
/*
ENTRY SYSS$FLUSH ALIAS $FLUSH LINKAGE $RMSCALL PARAMETER (
  ANY NAMED RAB IN OUT,      (/* TYPE(RAB),
  ADDRESS(ENTRY) NAMED ERR OPTIONAL,      (/* TYPE(ASTADR),
  ADDRESS(ENTRY) NAMED SUC OPTIONAL      (/* TYPE(ASTADR)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*
/* $GET
```

```
/*
/*  Get Record from File
/*
/*  $GET rab, [err], [suc]
/*
/*  rab      = address of rab
/*
/*  err      = address of user error completion routine
/*
/*  suc      = address of user success completion routine
/*
ENTRY SYSSGET ALIAS $GET LINKAGE $RMSCALL PARAMETER (
ANY NAMED RAB IN OUT,      (/* TYPE(RAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL,      (/* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL      (/* TYPE(ASTADR)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*  $GETCHN
/*
/*  Get I/O Channel Information
/*
/*  $GETCHN chan ,[prilen] ,[pribuf] ,[scdlen] ,[scdbuf]
/*
/*  chan     = number of a channel assigned to the device
/*
/*  prilen  = address of word to receive length of primary buffer
/*
/*  pribuf  = address of primary buffer descriptor
/*
/*  scdlen  = address of word to receive length of secondary buffer
/*
/*  scdbuf  = address of secondary buffer descriptor
/*
ENTRY SYSSGETCHN ALIAS $GETCHN PARAMETER (
WORD UNSIGNED VALUE NAMED CHAN, (/* TYPE(CHANNEL),
WORD UNSIGNED NAMED PRILEN OUT DEFAULT 0,      (/* TYPE(NUMBER),
CHARACTER DESCRIPTOR NAMED PRIBUF OUT DEFAULT 0,      (/* TYPE(CHARDESC),
WORD UNSIGNED NAMED SCMLEN OUT DEFAULT 0,      (/* TYPE(NUMBER),
CHARACTER DESCRIPTOR NAMED SCDBUF OUT DEFAULT 0 (/* TYPE(CHARDESC)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*  $GETDEV
/*
/*  Get I/O Device Information
/*
/*  $GETDEV devnam ,[prilen] ,[pribuf] ,[scdlen] ,[scdbuf]
/*
/*  devnam   = address of device name or logical name descriptor
/*
/*  prilen  = address of word to receive length of primary buffer
/*
/*  pribuf  = address of primary buffer descriptor
/*
/*  scdlen  = address of word to receive length of secondary buffer
/*
```

```

/*      scdbuf = address of secondary buffer descriptor
/*
ENTRY SYSS$GETDEV ALIAS $GETDEV PARAMETER (
CHARACTER DESCRIPTOR NAMED DEVNAM IN,      (/* TYPE(DEVNAME),
WORD UNSIGNED NAMED PRILEN OUT DEFAULT 0,  (/* TYPE(NUMBER),
CHARACTER DESCRIPTOR NAMED PRIBUF OUT DEFAULT 0, (/* TYPE(CHARDESC),
WORD UNSIGNED NAMED SCDLEN OUT DEFAULT 0, (/* TYPE(NUMBER),
CHARACTER DESCRIPTOR NAMED SCDBUF OUT DEFAULT 0 (/* TYPE(CHARDESC),
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/* $GETDVI
/*
/*      Get Device/Volume Information
/*
/*      $GETDVI [efn] ,[chan] ,[devnam] ,itmlst ,[iosb] ,[astadr]
/*      ($GETDVIW)      ,[astprm] ,[nullarg]
/*
/*      efn      = event flag to be set at completion
/*
/*      chan     = number of a channel assigned to the device or
/*                0 if device is specified by the devnam parameter
/*
/*      devnam   = address of device name or logical name descriptor
/*
/*      itmlst  = address of a list of item descriptors
/*
/*      iosb    = address of a quadword I/O status block
/*
/*      astadr  = address of entry mask of AST routine
/*
/*      astprm  = value to be passed to AST routine
/*
/*      nullarg = reserved argument
/*
/*
F TRY SYSS$GETDVI ALIAS $GETDVI PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0,      (/* TYPE(EFNUM),
WORD UNSIGNED VALUE NAMED CHAN DEFAULT 0,        (/* TYPE(CHANNEL),
CHARACTER DESCRIPTOR NAMED DEVNAM IN DEFAULT 0, (/* TYPE(DEVNAME),
ANY NAMED ITMLST IN,      (/* TYPE(ITEMLIST),
QUADWORD UNSIGNED NAMED IOSB OUT DEFAULT 0,      (/* TYPE(IOSB),
ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0, (/* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0, (/* TYPE(USERPARM),
QUADWORD UNSIGNED NAMED NULLARG DEFAULT 0      (/* TYPE(NULLARG)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

ENTRY SYSS$GETDVIW ALIAS $GETDVIW PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0,      (/* TYPE(EFNUM),
WORD UNSIGNED VALUE NAMED CHAN DEFAULT 0,        (/* TYPE(CHANNEL),
CHARACTER DESCRIPTOR NAMED DEVNAM IN DEFAULT 0, (/* TYPE(DEVNAME),
ANY NAMED ITMLST IN,      (/* TYPE(ITEMLIST),
QUADWORD UNSIGNED NAMED IOSB OUT DEFAULT 0,      (/* TYPE(IOSB),
ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0, (/* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0, (/* TYPE(USERPARM),
QUADWORD UNSIGNED NAMED NULLARG DEFAULT 0      (/* TYPE(NULLARG)

```

```
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);
```

```
/* $GETJPI
/*
/* Get Job/Process Information
/*
/* $GETJPI [efn] ,[pidadr] ,[prcnam] ,itmlst ,[iosb], [astadr],
/* ($GETJPIW) [astprm]
/*
/* efn = event flag to be set at completion
/*
/* pidadr = address of process identification
/*
/* prcnam = address of process name string descriptor
/*
/* itmlst = address of a list of item descriptors
/*
/* iosb = address of a quadword I/O status block
/*
/* astadr = address of entry mask of AST routine
/*
/* astprm = value to be passed to AST routine as an argument
/*
```

```
ENTRY SYSS$GETJPI ALIAS $GETJPI PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0,      (/* TYPE(EFNUM),
LONGWORD UNSIGNED NAMED PIDADR IN OUT DEFAULT 0,  (/* TYPE(PROCID),
CHARACTER DESCRIPTOR NAMED PRCNAM IN DEFAULT 0, (/* TYPE(PROCNAME),
ANY NAMED ITMLST IN, (/* TYPE(ITEMLIST),
QUADWORD UNSIGNED NAMED IOSB OUT DEFAULT 0, (/* TYPE(IOSB),
ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0, (/* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0 (/* TYPE(USERPARM)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);
```

```
ENTRY SYSS$GETJPIW ALIAS $GETJPIW PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0,      (/* TYPE(EFNUM),
LONGWORD UNSIGNED NAMED PIDADR IN OUT DEFAULT 0,  (/* TYPE(PROCID),
CHARACTER DESCRIPTOR NAMED PRCNAM IN DEFAULT 0, (/* TYPE(PROCNAME),
ANY NAMED ITMLST IN, (/* TYPE(ITEMLIST),
QUADWORD UNSIGNED NAMED IOSB OUT DEFAULT 0, (/* TYPE(IOSB),
ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0, (/* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0 (/* TYPE(USERPARM)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);
```

```
/* $GETLKI
/*
/* Get Lock Information
/*
/* $GETLKI [efn] ,lkidadr ,itmlst ,[iosb], [astadr], [astprm],
/* ($GETLKIW) [reserved]
/*
/* efn = event flag to be set at completion
/*
/* lkidadr = address of lock identification
```

E

E

E

```

/*
/* itmlst = address of a list of item descriptors
/*
/* iosb = address of a quadword I/O status block
/*
/* astadr = address of entry mask of AST routine
/*
/* astprm = value to be passed to AST routine as an argument
/*
/* reserved = reserved parameter
/*
ENTRY SYSSGETLKI ALIAS $GETLKI PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0,      (/* TYPE(EFNUM),
LONGWORD UNSIGNED NAMED LKIDADR IN OUT DEFAULT 0,  (/* TYPE(LOCKID),
ANY NAMED ITMLST IN,      (/* TYPE(ITEMLIST),
QUADWORD UNSIGNED NAMED IOSB OUT DEFAULT 0,      (/* TYPE(IOSB),
ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0, (/* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0, (/* TYPE(USERPARM),
LONGWORD UNSIGNED VALUE NAMED RESERVED DEFAULT 0 (/* TYPE(NULLARG)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

ENTRY SYSSGETLKIW ALIAS $GETLKIW PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0,      (/* TYPE(EFNUM),
LONGWORD UNSIGNED NAMED LKIDADR IN OUT DEFAULT 0,  (/* TYPE(LOCKID),
ANY NAMED ITMLST IN,      (/* TYPE(ITEMLIST),
QUADWORD UNSIGNED NAMED IOSB OUT DEFAULT 0,      (/* TYPE(IOSB),
ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0, (/* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0, (/* TYPE(USERPARM),
LONGWORD UNSIGNED VALUE NAMED RESERVED DEFAULT 0 (/* TYPE(NULLARG)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/* $GETMSG
/*
/* Get Message
/*
/* $GETMSG msgid ,msglen ,bufadr ,[flags] ,[outadr]
/*
/* msgid = identification of message to be retrieved
/*
/* msglen = address of a word to receive length of string
/* returned
/*
/* bufadr = address of buffer descriptor of buffer to receive
/* string
/*
/* flags = flag bits for message content (macro default = 15)
/*
/* Bit Value/Meaning
/*
/* 0 1 Include text
/* 0 Do not include text
/* 1 1 Include identifier
/* 0 Do not include identifier
/* 2 1 Include severity
/*

```

```

/*          3  0 Do not include severity
/*          3  1 Include component
/*          0  0 Do not include component
/*
/*      outadr = address of 4-byte array to receive the following values:
/*
/*      Byte      Contents
/*
/*      0  Reserved
/*      1  Count of FAO arguments
/*      2  User value
/*      3  Reserved
/*
ENTRY SYSS$GETMSG ALIAS $GETMSG PARAMETER (
LONGWORD UNSIGNED VALUE NAMED MSGID,      (/* TYPE(CONDVALU),
WORD UNSIGNED NAMED MSGLEN OUT, (/* TYPE(NUMBER),
CHARACTER DESCRIPTOR NAMED BUFADR OUT, (/* TYPE(CHARDESC),
LONGWORD UNSIGNED VALUE NAMED FLAGS DEFAULT 15, (/* TYPE(MASK),
BYTE UNSIGNED DIMENSION 4 NAMED OUTADR OUT DEFAULT 0 (/* TYPE(VECTOR)
) RETURNS LONGWORD; (/* TYPE(CONDVALU);

/* $GETPTI
/*
/*      Get Page Table Information
/*
/*      $GETPTI [inadr],[retadr],[acmode],[mask],[pagcnt]
/*
/*      inadr      = address of two longwords containing starting
/*                  and ending virtual address to operate on
/*
/*      retadr     = address of two longwords into which starting
/*                  and ending address of pages operated on is returned
/*
/*      acmode     = access mode against which ownership is checked
/*
/*      mask       = mask of page table information control bits
/*
/*      pagcnt     = minimum page count
/*
ENTRY SYSS$GETPTI ALIAS $GETPTI PARAMETER (
LONGWORD UNSIGNED DIMENSION 2 NAMED INADR IN, (/* TYPE(VARANGE),
LONGWORD UNSIGNED DIMENSION 2 NAMED RETADR OUT DEFAULT 0, (/* TYPE(VARANGE),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0, (/* TYPE(ACMODE),
LONGWORD UNSIGNED VALUE NAMED MASK DEFAULT 0, (/* TYPE(MASK),
LONGWORD UNSIGNED VALUE NAMED PAGCNT DEFAULT 0 (/* TYPE(NUMBER)
) RETURNS LONGWORD; (/* TYPE(CONDVALU);

/* $GETQUI
/*
/*      Get Queue Information
/*
/*      $GETQUI [efn], func, [nullarg], [itmlst],
/*      ($GETQUIW) [iosb], [astadr], [astprm]

```

```

/*
/*  efn      = event flag to be set when request completes
/*  func     = code specifying function to be performed
/*  nullarg  = reserved argument for similarity with $getxxx services
/*  itmlst   = address of a list of item descriptors for the operation
/*  iosb     = address of a quadword status block to receive the final status
/*  astadr   = address of an ast routine to be called when request completes
/*  astprm   = 32-bit ast parameter
/*
ENTRY SYSSGETQUI ALIAS $GETQUI PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0,      (/* TYPE(EFNUM),
WORD UNSIGNED VALUE NAMED FUNC, (/* TYPE(FUNCCODE),
LONGWORD UNSIGNED VALUE NAMED NULLARG DEFAULT 0,   (/* TYPE(NULLARG),
ANY NAMED ITMLST IN DEFAULT 0, (/* TYPE(ITEMLIST),
QUADWORD UNSIGNED NAMED IOSB OUT DEFAULT 0,       (/* TYPE(IOSB),
ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0, (/* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0 (/* TYPE(USERPARM)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

ENTRY SYSSGETQUIW ALIAS $GETQUIW PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0,      (/* TYPE(EFNUM),
WORD UNSIGNED VALUE NAMED FUNC, (/* TYPE(FUNCCODE),
LONGWORD UNSIGNED VALUE NAMED NULLARG DEFAULT 0,   (/* TYPE(NULLARG),
ANY NAMED ITMLST IN DEFAULT 0, (/* TYPE(ITEMLIST),
QUADWORD UNSIGNED NAMED IOSB OUT DEFAULT 0,       (/* TYPE(IOSB),
ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0, (/* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0 (/* TYPE(USERPARM)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/* $GETSYI
/*
/*  Get System-Wide Information
/*
/*  $GETSYI [efn] ,[csidadr],[nodename],itmlst ,[iosb] ,[astadr],
/*  ($GETSYIW) [astprm]
/*
/*  efn      = event flag to be set at completion
/*
/*  csidadr  = address of cluster system identification
/*
/*  nodename = address of node name string descriptor
/*
/*  itmlst   = address of a list of item descriptors
/*
/*  iosb     = address of a quadword I/O status block
/*
/*  astadr   = address of entry mask of AST routine
/*
/*  astprm   = value to be passed to AST routine
/*
/*
/*  The second and third arguments in the $GETSYI argument list are
/*  not used; they are reserved for future use.
/*
ENTRY SYSSGETSYI ALIAS $GETSYI PARAMETER (

```

```

LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0,      { /* TYPE(EFNUM)
LONGWORD UNSIGNED NAMED CSIDADR IN OUT DEFAULT 0,  { /* TYPE(PROCID)
CHARACTER DESCRIPTOR NAMED NODENAME IN DEFAULT 0,  { /* TYPE(PROCNAME)
ANY NAMED ITMLST IN,      { /* TYPE(ITEMLIST)
QUADWORD UNSIGNED NAMED IOSB OUT DEFAULT 0,      { /* TYPE(IOSB)
ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0, { /* TYPE(ASTADR)
LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0 { /* TYPE(USERPARM)
) RETURNS LONGWORD;      { /* TYPE(CONDVALU);

```

```

ENTRY SYSSGETSYIW ALIAS $GETSYIW PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0,      { /* TYPE(EFNUM)
LONGWORD UNSIGNED NAMED CSIDADR IN OUT DEFAULT 0,  { /* TYPE(PROCID)
CHARACTER DESCRIPTOR NAMED NODENAME IN DEFAULT 0,  { /* TYPE(PROCNAME)
ANY NAMED ITMLST IN,      { /* TYPE(ITEMLIST)
QUADWORD UNSIGNED NAMED IOSB OUT DEFAULT 0,      { /* TYPE(IOSB)
ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0, { /* TYPE(ASTADR)
LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0 { /* TYPE(USERPARM)
) RETURNS LONGWORD;      { /* TYPE(CONDVALU);

```

```

/* $GETTIM
/*
/* Get Time
/*
/* $GETTIM timadr
/*
/* timadr = address of a quadword to receive 64-bit current time value
/*

```

```

ENTRY SYSSGETTIM ALIAS $GETTIM PARAMETER (
QUADWORD UNSIGNED NAMED TIMADR OUT      { /* TYPE(QUADTIME)
) RETURNS LONGWORD;      { /* TYPE(CONDVALU);

```

```

/*
/* $GRANTID
/*
/* Grant Identifier to Process
/*
/* $GRANTID [pidadr], [prcnam], [id], [name], [prvatr]
/*
/* pidadr = address of PID of affected process
/* prcnam = address of string descriptor of process name
/* id = address of quadword identifier and attributes
/* name = address of string descriptor of identifier name
/* prvatr = address to store attributes of superseded id
/*

```

```

ENTRY SYSSGRANTID ALIAS $GRANTID PARAMETER (
LONGWORD UNSIGNED NAMED PIDADR IN OUT DEFAULT 0,      { /* TYPE(PROCID)
CHARACTER DESCRIPTOR NAMED PRCNAM IN DEFAULT 0, { /* TYPE(PROCNAME)
QUADWORD UNSIGNED NAMED ID IN OUT DEFAULT 0,      { /* TYPE(QUADID)
CHARACTER DESCRIPTOR NAMED NAME IN DEFAULT 0, { /* TYPE(CHARDESC)
LONGWORD UNSIGNED NAMED PRVATR OUT DEFAULT 0 { /* TYPE(MASK)
) RETURNS LONGWORD;      { /* TYPE(CONDVALU);

```

```

/* $HIBER

```

```

/*
/*  Hibernate
/*
/*  $HIBER_S
/*
ENTRY SYSSHIBER ALIAS $HIBER
RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*
/*  $IDTOASC
/*
/*  Identifier To Ascii Conversion
/*
/*  $IDTOASC id, [namlen], [nambuf], [resid], [attrib], [contxt]
/*
/*  id      = identifier longword to convert, or zero to
/*           find all identifiers (wildcarding)
/*  namlen  = address of a word to store the length of the
/*           identifier name
/*  nambuf  = address of a character string descriptor that
/*           describes the buffer to return the identifier name
/*  resid   = address of a longword to return the id found
/*           while wildcarding
/*  attrib  = address to return the attributes longword
/*  contxt  = address of a longword containing the record stream
/*           context. initially should be zero, value output
/*           on first call, value input on subsequent calls.
/*
ENTRY SY$IDTOASC ALIAS $IDTOASC PARAMETER (
LONGWORD UNSIGNED VALUE NAMED ID IN,      (/* TYPE(USERID),
WORD UNSIGNED NAMED NAMLEN OUT DEFAULT 0, (/* TYPE(NUMBER),
CHARACTER DESCRIPTOR NAMED NAMBUF OUT DEFAULT 0, (/* TYPE(CHARDESC),
LONGWORD UNSIGNED NAMED RESID OUT DEFAULT 0, (/* TYPE(USERID),
LONGWORD UNSIGNED NAMED ATTRIB OUT DEFAULT 0, (/* TYPE(MASK),
LONGWORD UNSIGNED NAMED CONTXT IN OUT DEFAULT 0 (/* TYPE(CONTEXT)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*  $IMGACT
/*
/*  Image Activation
/*
/*  $IMGACT name,[dflnam],hdrbuf,[imgctl],[inadr],
/*           [retadr], [ident], [acmode]
/*
/*  name    = address of descriptor for file name string
/*  dflnam  = descriptor for file name string
/*  hdrbuf  = address of 512 byte buffer to write in
/*  imgctl  = image activation control flags
/*  inadr   = address of quadword specifying virtual address

```

```

/*          range to be mapped
/*
/*  retadr  = address of quadword specifying virtual address
/*          range actually mapped
/*
/*  ident   = address of quadword holding image section match
/*          control and identifier
/*
/*  acmode  = access mode to be the owner of the created pages
/*
ENTRY SYSSIMGACT ALIAS $IMGACT PARAMETER (
CHARACTER DESCRIPTOR NAMED NAME IN,      (/* TYPE(CHARDESC),
CHARACTER DESCRIPTOR NAMED DFLNAM IN DEFAULT 0, (/* TYPE(CHARDESC),
ADDRESS NAMED HDRBUF OUT,                (/* TYPE(CNTRLBLK),
LONGWORD UNSIGNED VALUE NAMED IMGCTL DEFAULT 0, (/* TYPE(MASK),
LONGWORD UNSIGNED DIMENSION 2 NAMED INADR IN DEFAULT 0, (/* TYPE(VARANGE),
LONGWORD UNSIGNED DIMENSION 2 NAMED RETADR OUT DEFAULT 0, (/* TYPE(VARANGE),
QUADWORD UNSIGNED NAMED "IDENT" IN DEFAULT 0, (/* TYPE(SECTID),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0 (/* TYPE(ACMODE)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*  $IMGFIX
/*
/*  Image Address Fixup Service
/*
/*  $IMGFIX
/*
ENTRY SYSSIMGFIX ALIAS $IMGFIX
RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*  $LCKPAG
/*
/*  Lock Pages in Memory
/*
/*  $LCKPAG  inadr ,[retadr] ,[acmode]
/*
/*  inadr   = address of 2-longword array containing starting and
/*          ending addresses of pages to be locked
/*
/*  retadr  = address of 2-longword array to receive addresses of
/*          pages actually locked
/*
/*  acmode  = access mode to check against the owner of the pages
/*
ENTRY SYSSLCKPAG ALIAS $LCKPAG PARAMETER (
LONGWORD UNSIGNED DIMENSION 2 NAMED INADR IN,      (/* TYPE(VARANGE),
LONGWORD UNSIGNED DIMENSION 2 NAMED RETADR OUT DEFAULT 0, (/* TYPE(VARANGE),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0 (/* TYPE(ACMODE)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*  $LKWSET
/*
/*  Lock Pages in Working Set

```

```

/*
/*  $LKWSET  inadr ,[retadr] ,[acmode]
/*
/*  inadr  = address of 2-longword array containing starting and
/*           ending virtual addresses of pages to be locked
/*
/*  retadr = address of a 2-longword array to receive starting and
/*           ending virtual addresses of pages actually locked
/*
/*  acmode = access mode to be checked against the page owner
/*
ENTRY SYSSLKWSET ALIAS $LKWSET PARAMETER (
LONGWORD UNSIGNED DIMENSION 2 NAMED INADR IN,  (/* TYPE(VARANGE),
LONGWORD UNSIGNED DIMENSION 2 NAMED RETADR OUT DEFAULT 0,  (/* TYPE(VARANGE),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0 (/* TYPE(ACMODE)
) RETURNS LONGWORD;  (/* TYPE(CONDVALU);

/*  $MGBLSC
/*
/*  Map Global Section
/*
/*  $MGBLSC  inadr ,[retadr] ,[acmode] ,[flags] ,gsdnam ,[ident] ,[relpag]
/*
/*  inadr  = address of 2-longword array containing starting and
/*           ending addresses of pages to be mapped
/*
/*  retadr = address of 2-longword array to receive virtual
/*           addresses of pages mapped
/*
/*  acmode = access mode of owner of mapped pages
/*
/*  flags  = flags overriding default section characteristics
/*
/*          Flag          Meaning
/*
/*          SECSM_WRT      Read/write section
/*          SECSM_SYSGBL   System global section
/*          SECSM_EXPREG   Find first available space
/*
/*  gsdnam = address of global section name descriptor
/*
/*  ident  = address of quadword containing version id and match control
/*
/*  relpag = relative page number within global section
/*
ENTRY SYSMGBLSC ALIAS $MGBLSC PARAMETER (
LONGWORD UNSIGNED DIMENSION 2 NAMED INADR IN,  (/* TYPE(VARANGE),
LONGWORD UNSIGNED DIMENSION 2 NAMED RETADR OUT DEFAULT 0,  (/* TYPE(VARANGE),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0, (/* TYPE(ACMODE),
LONGWORD UNSIGNED VALUE NAMED FLAGS DEFAULT 0, (/* TYPE(MASK),
CHARACTER DESCRIPTOR NAMED GSDNAM IN,  (/* TYPE(SECTNAME),
QUADWORD UNSIGNED NAMED "IDENT" IN DEFAULT 0,  (/* TYPE(SECTID),
LONGWORD UNSIGNED VALUE NAMED RELPAG DEFAULT 0 (/* TYPE(NUMBER)
) RETURNS LONGWORD;  (/* TYPE(CONDVALU);

```

```
/*
/* $MODIFY
/*
/* Modify File
/*
/* $MODIFY fab, [err], [suc]
/*
/* fab = address of fab
/*
/* err = address of user error completion routine
/*
/* suc = address of user success completion routine
/*
ENTRY SYSSMODIFY ALIAS $MODIFY LINKAGE $RMSCALL PARAMETER (
ANY NAMED FAB IN OUT,      (/* TYPE(FAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL,    (/* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL    (/* TYPE(ASTADR)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*
/* $MOD_HOLDER
/*
/* Modify Holder Record In Rights Database
/*
/* $MOD_HOLDER id, holder, [set_attrib], [clr_attrib]
/*
/* id = identifier longword
/* holder = address of the holder identifier quadword
/* set_attrib = longword containing the attributes to set
/* into the holder record
/* clr_attrib = longword containing the attributes to clear
/* in the holder record
/*
ENTRY SYSSMOD_HOLDER ALIAS $MOD_HOLDER PARAMETER (
LONGWORD UNSIGNED VALUE NAMED ID IN,    (/* TYPE(USERID),
QUADWORD UNSIGNED NAMED HOLDER IN,      (/* TYPE(HOLDER),
LONGWORD UNSIGNED VALUE NAMED SET_ATTRIB IN DEFAULT 0, (/* TYPE(MASK),
LONGWORD UNSIGNED VALUE NAMED CLR_ATTRIB IN DEFAULT 0 (/* TYPE(MASK)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*
/* $MOD_IDENT
/*
/* Modify Identifier Record In Rights Database
/*
/* $MOD_IDENT id, [set_attrib], [clr_attrib], [new_name], [new_value]
/*
/* id = identifier longword
/* set_attrib = longword containing the attributes
/* to set into the identifier record
/* clr_attrib = longword containing the attributes
/* to clear in the identifier record
/* new_name = address of the new identifier name character
```

```

/*      string descriptor
/*      new_value = new identifier value longword
/*
ENTRY SYSSMOD IDENT ALIAS $MOD IDENT PARAMETER (
LONGWORD UNSIGNED VALUE NAMED ID IN      (/* TYPE(USERID)),
LONGWORD UNSIGNED VALUE NAMED SET_ATTRIB IN DEFAULT 0, (/* TYPE(MASK),
LONGWORD UNSIGNED VALUE NAMED CLR_ATTRIB IN DEFAULT 0, (/* TYPE(MASK)
CHARACTER DESCRIPTOR NAMED NEW_NAME IN DEFAULT 0,      (/* TYPE(CHARDESC),
LONGWORD UNSIGNED VALUE NAMED NEW_VALUE IN DEFAULT 0   (/* TYPE(USERID),
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

```

```

/* $MOUNT
/*
/* Mount Volume
/*
/* $MOUNT itmlst
/*
/* itmlst = Address of a list of item identifiers
/*
ENTRY SYSSMOUNT ALIAS $MOUNT PARAMETER (
ANY NAMED ITMLST IN      (/* TYPE(ITEMLIST)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

```

```

/* SMTACCESS
/*
/* Installation specific accessibility routine
/*
/* SMTACCESS [lblnam, [uic], [std_version], [access_char],
/* [access_spec],type
/*
/* lblnam = On input this field is the address of the
/* VOL1 or HDR1 label read off the magnetic tape. On
/* output of labels this field is zero. The type
/* of label is determined by the TYPE field.
/* uic = This field contains the longword volume UIC passed
/* by value.
/* std_version = This field contains a byte value of the
/* decimal equivalent of the version number
/* gotten from the VOL1 label, passed by value.
/* access_char = This field contains the byte value of the
/* accessibility character specified by the user,
/* passed by value. For output of labels only.
/* access_spec = This field determines if the character passed
/* in ACCESS_CHAR was specified by user, passed
/* by value. It must contain one of the following
/* values:
/* MTASK_CHARVALID = YES
/* MTASK_NOCHAR = NO
/* For output of labels only.
/* type = This field contains the type of accessibility to process,
/* passed by value. It must contain one of the following
/* values:
/* MTASK_INVOL1 = Input a VOL1 label

```

```
/*          MTASK_INHDR1 = Input a HDR1 label
/*          MTASK_OUTVOL1 = Output a VOL1 label
/*          MTASK_OUTHDR1 = Output a HDR1 label
/*
ENTRY SYSSMTACCESS ALIAS SMTACCESS PARAMETER (
LONGWORD UNSIGNED NAMED LBLNAM, { /* TYPE(VECTOR),
LONGWORD UNSIGNED VALUE NAMED UIC DEFAULT 0, { /* TYPE(USERID),
LONGWORD UNSIGNED VALUE NAMED STD VERSION DEFAULT 3, { /* TYPE(NUMBER),
LONGWORD UNSIGNED VALUE NAMED ACCESS_CHAR DEFAULT 32, { /* TYPE(NUMBER),
LONGWORD UNSIGNED VALUE NAMED ACCESS_SPEC DEFAULT 0, { /* TYPE(NUMBER),
LONGWORD UNSIGNED VALUE NAMED TYPE { /* TYPE(NUMBER)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/* $NUMTIM
/*
/* Convert Binary Time to Numeric Time
/*
/* $NUMTIM timbuf ,[timadr]
/*
/* timbuf = address of a 7-word buffer to receive numeric time
/*          information
/*
/* timadr = address of a quadword containing the 64-bit time. If
/*          0, use current time
/*
ENTRY SYSSNUMTIM ALIAS $NUMTIM PARAMETER (
WORD UNSIGNED DIMENSION 7 NAMED TIMBUF OUT, { /* TYPE(VECTOR),
QUADWORD UNSIGNED NAMED TIMADR IN DEFAULT 0 { /* TYPE(QUADTIME)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/*
/* $NXTVOL
/*
/* Go to Next Volume
/*
/* $NXTVOL rab, [err], [suc]
/*
/* rab = address of rab
/*
/* err = address of user error completion routine
/*
/* suc = address of user success completion routine
/*
ENTRY SYSSNXTVOL ALIAS $NXTVOL LINKAGE $RMSCALL PARAMETER (
ANY NAMED RAB IN OUT, { /* TYPE(RAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL, { /* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL { /* TYPE(ASTADR)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/*
/* $OPEN
/*
/* Open File
/*
```

```
/* $OPEN fab, [err], [suc]
/*
/* fab = address of fab
/*
/* err = address of user error completion routine
/*
/* suc = address of user success completion routine
/*
ENTRY SYSSOPEN ALIAS $OPEN LINKAGE $RMSCALL PARAMETER (
ANY NAMED FAB IN OUT,      (/* TYPE(FAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL,      (/* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL      (/* TYPE(ASTADR)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*
/* $PARSE
/*
/* Parse File Name
/*
/* $PARSE fab, [err], [suc]
/*
/* fab = address of fab
/*
/* err = address of user error completion routine
/*
/* suc = address of user success completion routine
/*
ENTRY SYSSPARSE ALIAS $PARSE LINKAGE $RMSCALL PARAMETER (
ANY NAMED FAB IN OUT,      (/* TYPE(FAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL,      (/* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL      (/* TYPE(ASTADR)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/*
/* $PARSE_ACL
/*
/* Parse an Access Control List Entry
/*
/* $PARSE_ACL acl-string ,acl-entry ,[error-position],
/* [bit-names]
/*
/* acl-string = address of a descriptor of a buffer which
/* contains the text to be parsed
/*
/* acl-entry = address of a descriptor of a buffer into
/* which the converted ACE is to be written
/*
/* error-position = address of a word to receive the number of
/* characters actually processed by the service.
/* If the service fails, this count points to
/* the failing point in the input string.
/*
/* bit-names = address of a access bit name table (32 entries)
/*
```

```
ENTRY SYSSPARSE_ACL ALIAS $PARSE_ACL PARAMETER (
  CHARACTER DESCRIPTOR NAMED ACLSTR IN,  (/* TYPE(CHARDESC),
  CHARACTER DESCRIPTOR NAMED ACLENT OUT, (/* TYPE(CHARDESC),
  WORD UNSIGNED NAMED ERRPOS OUT DEFAULT 0, (/* TYPE(NUMBER),
  ANY NAMED ACCNAM IN DEFAULT 0 (/* TYPE(VECTOR)
) RETURNS LONGWORD; (/* TYPE(CONDVALU);

/* $PURGWS
/*
/* Purge Working Set
/*
/* $PURGWS inadr
/*
/* inadr = address of 2-longword array containing starting and
/* ending addresses of pages to be removed
/*
ENTRY SYSSPURGWS ALIAS $PURGWS PARAMETER (
  LONGWORD UNSIGNED DIMENSION 2 NAMED INADR IN (/* TYPE(VARANGE)
) RETURNS LONGWORD; (/* TYPE(CONDVALU);

/*
/* $PUT
/*
/* Put Record to File
/*
/* $PUT rab, [err], [suc]
/*
/* rab = address of rab
/*
/* err = address of user error completion routine
/*
/* suc = address of user success completion routine
/*
ENTRY SYSSPUT ALIAS $PUT LINKAGE $RMSCALL PARAMETER (
  ANY NAMED RAB IN OUT, (/* TYPE(RAB),
  ADDRESS(ENTRY) NAMED ERR OPTIONAL, (/* TYPE(ASTADR),
  ADDRESS(ENTRY) NAMED SUC OPTIONAL (/* TYPE(ASTADR)
) RETURNS LONGWORD; (/* TYPE(CONDVALU);

/* $PUTMSG
/*
/* Put Message
/*
/* $PUTMSG msgvec ,[actrtn] ,[facnam], [actprm]
/*
/* msgvec = address of message argument vector
/*
/* actrtn = address of entry mask of action routine
/*
/* facnam = address of facility name string descriptor
/*
/* actprm = parameter to pass to action routine
/*
```

```

ENTRY SYSSPUTMSG ALIAS $PUTMSG PARAMETER (
  ANY NAMED MSGVEC IN,      (/* TYPE(CNTRLBLK),
  ADDRESS(ENTRY) NAMED ACTRTN DEFAULT 0, (/* TYPE(ENTRYADR),
  CHARACTER DESCRIPTOR NAMED FACNAM IN DEFAULT 0, (/* TYPE(CHARDESC),
  LONGWORD UNSIGNED VALUE NAMED ACTPRM DEFAULT 0 (/* TYPE(USERPARM)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/* $QIO
/*
/* Queue I/O Request
/*
/* $QIO [efn] ,chan ,func ,[iosb] ,[astadr] ,[astprm]
/* ($QIOW) ,[p1] ,[p2] ,[p3] ,[p4] ,[p5] ,[p6]
/*
/* efn = number of event flag to set on completion
/*
/* chan = number of channel on which I/O is directed
/*
/* func = function code specifying action to be performed
/*
/* iosb = address of quadword I/O status block to receive final
/* completion status
/*
/* astadr = address of entry mask of AST routine
/*
/* astprm = value to be passed to AST routine as argument
/*
/* p1... = optional device- and function-specific parameters
/*
ENTRY SYSSQIO ALIAS $QIO PARAMETER (
  LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0, (/* TYPE(EFNUM),
  WORD UNSIGNED VALUE NAMED CHAN, (/* TYPE(CHANNEL),
  WORD UNSIGNED VALUE NAMED FUNC, (/* TYPE(FUNCCODE),
  QUADWORD UNSIGNED NAMED IOSB OUT DEFAULT 0, (/* TYPE(IOSB),
  ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0, (/* TYPE(ASTADR),
  LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0, (/* TYPE(USERPARM),
  ANY NAMED P1 IN DEFAULT 0, (/* TYPE(VARIES),
  LONGWORD VALUE NAMED P2 DEFAULT 0, (/* TYPE(VARIES),
  LONGWORD VALUE NAMED P3 DEFAULT 0, (/* TYPE(VARIES),
  LONGWORD VALUE NAMED P4 DEFAULT 0, (/* TYPE(VARIES),
  LONGWORD VALUE NAMED P5 DEFAULT 0, (/* TYPE(VARIES),
  LONGWORD VALUE NAMED P6 DEFAULT 0 (/* TYPE(VARIES)
) RETURNS LONGWORD; (/* TYPE(CONDVALU);

ENTRY SYSSQIOW ALIAS $QIOW PARAMETER (
  LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0, (/* TYPE(EFNUM),
  WORD UNSIGNED VALUE NAMED CHAN, (/* TYPE(CHANNEL),
  WORD UNSIGNED VALUE NAMED FUNC, (/* TYPE(FUNCCODE),
  QUADWORD UNSIGNED NAMED IOSB OUT DEFAULT 0, (/* TYPE(IOSB),
  ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0, (/* TYPE(ASTADR),
  LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0, (/* TYPE(USERPARM),
  ANY NAMED P1 IN DEFAULT 0, (/* TYPE(VARIES),
  LONGWORD VALUE NAMED P2 DEFAULT 0, (/* TYPE(VARIES),
  LONGWORD VALUE NAMED P3 DEFAULT 0, (/* TYPE(VARIES),
  LONGWORD VALUE NAMED P4 DEFAULT 0, (/* TYPE(VARIES),

```

E

E

```
LONGWORD VALUE NAMED P5 DEFAULT 0,      { /* TYPE(VARIES),
LONGWORD VALUE NAMED P6 DEFAULT 0,      { /* TYPE(VARIES)
) RETURNS LONGWORD;      { /* TYPE(CONDVALU);
```

```
/*
/* $READ
/*
/* Read Block from File
/*
/* $READ rab, [err], [suc]
/*
/* rab = address of rab
/*
/* err = address of user error completion routine
/*
/* suc = address of user success completion routine
/*
```

```
ENTRY SYSS$READ ALIAS $READ LINKAGE $RMSCALL PARAMETER (
ANY NAMED RAB IN OUT, { /* TYPE(RAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL, { /* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL { /* TYPE(ASTADR)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);
```

```
/* $READEF
/*
/* Read Event Flag
/*
/* $READEF efn ,state
/*
/* efn = event flag number of any flag in the cluster
/*
/* state = address of a longword to receive current state of all
/* flags in the cluster
/*
```

```
ENTRY SYSS$READEF ALIAS $READEF PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN, { /* TYPE(EFNUM),
LONGWORD UNSIGNED NAMED STATE OUT { /* TYPE(MASK)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);
```

```
/*
/* $RELEASE
/*
/* Release Record
/*
/* $RELEASE rab, [err], [suc]
/*
/* rab = address of rab
/*
/* err = address of user error completion routine
/*
/* suc = address of user success completion routine
/*
```

```
ENTRY SYSS$RELEASE ALIAS $RELEASE LINKAGE $RMSCALL PARAMETER (
ANY NAMED RAB IN OUT, { /* TYPE(RAB),
```

```
ADDRESS(ENTRY) NAMED ERR OPTIONAL,      { /* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL        { /* TYPE(ASTADR)
) RETURNS LONGWORD;      { /* TYPE(CONDVALU);
```

```
/*
/* $REMOVE
/*
/* Remove File
/*
/* $REMOVE fab, [err], [suc]
/*
/* fab = address of fab
/*
/* err = address of user error completion routine
/*
/* suc = address of user success completion routine
/*
```

```
ENTRY SYS$REMOVE ALIAS $REMOVE LINKAGE $RMSCALL PARAMETER (
ANY NAMED FAB IN OUT,      { /* TYPE(FAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL,      { /* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL        { /* TYPE(ASTADR)
) RETURNS LONGWORD;      { /* TYPE(CONDVALU);
```

```
/*
/* $REM_HOLDER
/*
/* Remove Holder Record From Rights Database
/*
/* $REM_HOLDER id, holder
/*
/* id = identifier longword
/* holder = address of the holder identifier quadword
/*
```

```
ENTRY SYS$REM_HOLDER ALIAS $REM_HOLDER PARAMETER (
LONGWORD UNSIGNED VALUE NAMED ID IN,      { /* TYPE(USERID),
QUADWORD UNSIGNED NAMED HOLDER IN          { /* TYPE(HOLDER)
) RETURNS LONGWORD;      { /* TYPE(CONDVALU);
```

```
/*
/* $REM_IDENT
/*
/* Remove Identifier From Rights Database
/*
/* $REM_IDENT id
/*
/* id = identifier longword
/*
```

```
ENTRY SYS$REM_IDENT ALIAS $REM_IDENT PARAMETER (
LONGWORD UNSIGNED VALUE NAMED ID IN      { /* TYPE(USERID)
) RETURNS LONGWORD;      { /* TYPE(CONDVALU);
```

```
/*
```

```

/* $RENAME
/*
/* Rename File
/*
/* $RENAME oldfab, [err], [suc], newfab
/*
/* oldfab = address of oldfab
/*
/* err = address of user error completion routine
/*
/* suc = address of user success completion routine
/*
/* newfab = address of new fab
ENTRY SYSS$RENAME ALIAS $RENAME LINKAGE $RMSCALLTWO PARAMETER (
ANY NAMED OLDFAB IN OUT,          { /* TYPE(FAB) },
ADDRESS(ENTRY) NAMED ERR DEFAULT 0, { /* TYPE(ASTADR) },
ADDRESS(ENTRY) NAMED SUC DEFAULT 0, { /* TYPE(ASTADR) },
ANY NAMED NEWFAB IN OUT { /* TYPE(FAB) }
) RETURNS LONGWORD;          { /* TYPE(CONDVALU) };

/* $RESUME
/*
/* Resume Suspended Process
/*
/* $RESUME [pidadr] ,[prcnam]
/*
/* pidadr = address of process id of process to be resumed
/*
/* prcnam = address of process name string descriptor
/*
ENTRY SYSS$RESUME ALIAS $RESUME PARAMETER (
LONGWORD UNSIGNED NAMED PIDADR IN OUT DEFAULT 0,          { /* TYPE(PROCID) },
CHARACTER DESCRIPTOR NAMED PRCNAM IN DEFAULT 0 { /* TYPE(PROCNAME) }
) RETURNS LONGWORD;          { /* TYPE(CONDVALU) };

/*
/* $REVOKID
/*
/* Revoke Identifier from Process
/*
/* $REVOKID [pidadr], [prcnam], [id], [name], [prvatr]
/*
/* pidadr = address of PID of affected process
/*
/* prcnam = address of string descriptor of process name
/*
/* id = address of quadword identifier and attributes
/*
/* name = address of string descriptor of identifier name
/*
/* prvatr = address to store attributes of superseded id
/*
ENTRY SYSS$REVOKID ALIAS $REVOKID PARAMETER (
LONGWORD UNSIGNED NAMED PIDADR IN OUT DEFAULT 0,          { /* TYPE(PROCID) },
CHARACTER DESCRIPTOR NAMED PRCNAM IN DEFAULT 0, { /* TYPE(PROCNAME) },
QUADWORD UNSIGNED NAMED ID IN OUT DEFAULT 0,          { /* TYPE(QUADID) },
CHARACTER DESCRIPTOR NAMED NAME IN DEFAULT 0,          { /* TYPE(CHARDESC) },
LONGWORD UNSIGNED NAMED PRVATR OUT DEFAULT 0          { /* TYPE(MASK) }

```

```
) RETURNS LONGWORD;      { /* TYPE(CONDVALU);

/*
/* $REWIND
/*
/*   Rewind File
/*
/*   $REWIND rab, [err], [suc]
/*
/*   rab   = address of rab
/*
/*   err   = address of user error completion routine
/*
/*   suc   = address of user success completion routine
/*
ENTRY SYSS$REWIND ALIAS $REWIND LINKAGE $RMSCALL PARAMETER (
ANY NAMED RAB IN OUT,      { /* TYPE(RAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL,      { /* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL      { /* TYPE(ASTADR)
) RETURNS LONGWORD;      { /* TYPE(CONDVALU);

/*
/* $RUNDWN
/*
/*   Rundown Process
/*
/*   $RUNDWN [acmode]
/*
/*   acmode = access mode to rundown; this and all less
/*           privileged access modes are rundown
/*
ENTRY SYSS$RUNDWN ALIAS $RUNDWN PARAMETER (
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0 { /* TYPE(ACMODE)
) RETURNS LONGWORD;      { /* TYPE(CONDVALU);

/*
/* $SCHDWK
/*
/*   Schedule Wakeup
/*
/*   $SCHDWK [pidadr] ,[prcnam] ,daytim ,[reptim]
/*
/*   pidadr = address of process id of process to be awakened
/*
/*   prcnam = address of process name string descriptor
/*
/*   daytim = address of quadword containing time to wake
/*
/*   reptim = address of quadword containing repeat time interval
/*
ENTRY SYSS$SCHDWK ALIAS $SCHDWK PARAMETER (
LONGWORD UNSIGNED NAMED PIDADR IN OUT DEFAULT 0,      { /* TYPE(PROCID),
CHARACTER DESCRIPTOR NAMED PRCNAM IN DEFAULT 0, { /* TYPE(PROCNAME),
QUADWORD UNSIGNED NAMED DAYTIM IN,      { /* TYPE(QUADTIME),
QUADWORD UNSIGNED NAMED REPTIM IN DEFAULT 0      { /* TYPE(QUADTIME)
```

E

E

E

```
) RETURNS LONGWORD;    { /* TYPE(CONDVALU);

/*
/* $SEARCH
/*
/* Search for File Name
/*
/* $SEARCH fab, [err], [suc]
/*
/* fab      = address of fab
/*
/* err      = address of user error completion routine
/*
/* suc      = address of user success completion routine
/*
ENTRY SYS$SEARCH ALIAS $SEARCH LINKAGE $RMSCALL PARAMETER (
  ANY NAMED FAB IN OUT,    { /* TYPE(FAB),
  ADDRESS(ENTRY) NAMED ERR OPTIONAL,    { /* TYPE(ASTADR),
  ADDRESS(ENTRY) NAMED SUC OPTIONAL    { /* TYPE(ASTADR)
) RETURNS LONGWORD;    { /* TYPE(CONDVALU);

/*
/* $SETAST
/*
/* Set AST Enable
/*
/* $SETAST enbflg
/*
/* enbflg = AST enable indicator for current access mode
/*          0 -> disable    1 -> enable
/*
ENTRY SYS$SETAST ALIAS $SETAST PARAMETER (
  BOOLEAN VALUE NAMED ENBFLG    { /* TYPE(BOOLEAN)
) RETURNS LONGWORD;    { /* TYPE(CONDVALU);

/*
/* $SETEF
/*
/* Set Event Flag
/*
/* $SETEF efn
/*
/* efn     = event flag number of flag to set
/*
ENTRY SYS$SETEF ALIAS $SETEF PARAMETER (
  LONGWORD UNSIGNED VALUE NAMED EFN    { /* TYPE(EFNUM)
) RETURNS LONGWORD;    { /* TYPE(CONDVALU);

/*
/* $SETEXV
/*
/* Set Exception Vector
/*
/* $SETEXV [vector] ,[addres] ,[acmode] ,[prvhnd]
/*
```

```

/*      vector = vector number
/*          0 -> primary vector  1 -> secondary  2 -> last chance
/*
/*      adres = exception handler address (0 indicates deassign vector)
/*
/*      acmode = access mode for which vector is set
/*
/*      prvhnd = address of longword to receive previous handler address
/*
ENTRY SYS$SETEXV ALIAS $SETEXV PARAMETER (
LONGWORD UNSIGNED VALUE NAMED VECTOR DEFAULT 0, { /* TYPE(NUMBER),
ADDRESS(ENTRY) NAMED ADRES DEFAULT 0, { /* TYPE(ENTRYADR),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0, { /* TYPE(ACMODE),
LONGWORD UNSIGNED NAMED PRVHND OUT DEFAULT 0 { /* TYPE(ENTRYADR)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

```

```

/* $SETIME
/*
/*      Set System Time
/*
/*      $SETIME [timadr]
/*
/*      timadr = address of quadword containing new system time in
/*              64-bit format.  If 0, recalibrate system time using
/*              hardware time-of-year clock.
/*
ENTRY SYS$SETIME ALIAS $SETIME PARAMETER (
QUADWORD UNSIGNED NAMED TIMADR IN DEFAULT 0 { /* TYPE(QUADTIME)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

```

```

/* $SETIMR
/*
/*      Set Timer
/*
/*      $SETIMR [efn] ,daytim ,[astadr] ,[reqidt]
/*
/*      efn      = event flag to set when timer expires
/*
/*      daytim   = address of quadword containing 64-bit time value
/*
/*      astadr   = address of entry mask of AST routine
/*
/*      reqidt   = request identification of this timer request
/*
ENTRY SYS$SETIMR ALIAS $SETIMR PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0, { /* TYPE(EFNUM),
QUADWORD UNSIGNED NAMED DAYTIM IN, { /* TYPE(QUADTIME),
ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0, { /* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE NAMED REQIDT DEFAULT 0 { /* TYPE(USERPARM)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

```

```

/* $SETPFM
/*

```

```

/* Set Page Fault Monitoring
/*
/* $SETPFM [pfmflg] ,[astadr] ,[astprm] ,[acmode]
/*
/* pfmflg = function/subfunction bits
/*
/* astadr = address of entry mask of AST routine
/*
/* astprm = value to be passed to AST routine
/*
/* acmode = access mode for which the AST is to be declared
/*
ENTRY SYS$SETPFM ALIAS $SETPFM PARAMETER (
LONGWORD UNSIGNED VALUE NAMED PFMFLG IN DEFAULT 0,      { /* TYPE(FUNCCODE)
ADDRESS(ENTRY)          NAMED ASTADR IN DEFAULT 0,      { /* TYPE(ASTADR)
LONGWORD UNSIGNED VALUE NAMED ASTPRM IN DEFAULT 0,      { /* TYPE(USERPARM)
LONGWORD UNSIGNED VALUE NAMED ACMODE IN DEFAULT 0       { /* TYPE(ACMODE)
) RETURNS LONGWORD;   { /* TYPE(CONDVALU);

/* $SETPRA
/*
/* Set Power Recovery AST
/*
/* $SETPRA astadr ,[acmode]
/*
/* astadr = address of power recovery AST routine
/*
/* acmode = access mode of AST
/*
ENTRY SYS$SETPRA ALIAS $SETPRA PARAMETER (
ADDRESS(ENTRY) NAMED ASTADR,      { /* TYPE(ASTADR)
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0 { /* TYPE(ACMODE)
) RETURNS LONGWORD;   { /* TYPE(CONDVALU);

/* $SETPRI
/*
/* Set Priority
/*
/* $SETPRI [pidadr] ,[prcnam] ,pri ,[prvpri]
/*
/* pidadr = address of process id of process to be set
/*
/* prcnam = address of process name string descriptor
/*
/* pri    = new base priority for the process
/*
/* prvpri = address of longword to receive previous base priority
/*
ENTRY SYS$SETPRI ALIAS $SETPRI PARAMETER (
LONGWORD UNSIGNED NAMED PIDADR IN OUT DEFAULT 0,      { /* TYPE(PROCID),
CHARACTER DESCRIPTOR NAMED PRCNAM IN DEFAULT 0,      { /* TYPE(PROCNAME),
LONGWORD UNSIGNED VALUE NAMED PRI,                   { /* TYPE(NUMBER),
LONGWORD UNSIGNED NAMED PRVPRI OUT DEFAULT 0         { /* TYPE(NUMBER)
) RETURNS LONGWORD;   { /* TYPE(CONDVALU);

```

```

/* $SETPRN
/*
/* Set Process Name
/*
/* $SETPRN [prcnam]
/*
/* prcnam = address of the process name string descriptor
/*
ENTRY SYS$SETPRN ALIAS $SETPRN PARAMETER (
CHARACTER DESCRIPTOR NAMED PRCNAM IN DEFAULT 0 { /* TYPE(PROCNAME)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/* $SETPRT
/*
/* Set Protection on Pages
/*
/* $SETPRT inadr ,[retadr] ,[acmode] ,prot ,[prvprt]
/*
/* inadr = address of 2-longword array containing starting and
/* ending virtual addresses of pages to change
/*
/* retadr = address of 2-longword array containing starting and
/* ending addresses of pages which were changed
/*
/* acmode = access mode of request
/*
/* prot = new protection
/*
/* prvprt = address of byte to receive previous protection of
/* last page changed
/*
ENTRY SYS$SETPRT ALIAS $SETPRT PARAMETER (
LONGWORD UNSIGNED DIMENSION 2 NAMED INADR IN, { /* TYPE(VARANGE),
LONGWORD UNSIGNED DIMENSION 2 NAMED RETADR OUT DEFAULT 0, { /* TYPE(VARANGE),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0, { /* TYPE(ACMODE),
LONGWORD UNSIGNED VALUE NAMED PROT, { /* TYPE(PAGEPROT),
BYTE UNSIGNED NAMED PRVPRT OUT DEFAULT 0 { /* TYPE(PAGEPROT)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/* $SETPRV
/*
/* Set privileges.
/*
/* $SETPRV [enbflg] ,[prvadr] ,[prmflg] ,[prvprv]
/*
/* enbflg = enable indicator
/* 0 -> disable 1 -> enable
/*
/* prvadr = 64-bit mask defining the privileges to be enabled or
/* disabled
/*
/* prmflg = permanent indicator
/* 0 -> temporary (for this image) 1 -> permanent

```

```
/*
/*   prvprv = address of quadword buffer to receive previous privileges
/*
ENTRY SYS$SETPRV ALIAS $SETPRV PARAMETER (
  BOOLEAN VALUE NAMED ENBFLG DEFAULT 0,  /* TYPE(BOOLEAN),
  QUADWORD UNSIGNED NAMED PRVADR IN DEFAULT 0,  /* TYPE(PRIVMASK),
  BOOLEAN VALUE NAMED PRMFLG DEFAULT 0,  /* TYPE(BOOLEAN),
  QUADWORD UNSIGNED NAMED PRVPRV OUT DEFAULT 0  /* TYPE(PRIVMASK)
) RETURNS LONGWORD;  /* TYPE(CONDVALU);

/* $SETRWM
/*
/*   Set Resource Wait Mode
/*
/*   $SETRWM [watflg]
/*
/*   watflg = wait indicator
/*           0 -> wait for resources  1 -> return failure immediately
/*           (type is NUMBER rather than BOOLEAN since WATFLG=TRUE
/*           means don't wait, which is the opposite of what it seems)
/*
ENTRY SYS$SETRWM ALIAS $SETRWM PARAMETER (
  BOOLEAN VALUE NAMED WATFLG DEFAULT 0  /* TYPE(NUMBER)
) RETURNS LONGWORD;  /* TYPE(CONDVALU);

/* $SETSFM
/*
/*   Set System Service Failure Mode
/*
/*   $SETSFM [enbflg]
/*
/*   enbflg = enable indicator
/*           0 -> disable generation of exceptions on service failures
/*           1 -> generate exceptions for system service failures
/*
ENTRY SYS$SETSFM ALIAS $SETSFM PARAMETER (
  BOOLEAN VALUE NAMED ENBFLG DEFAULT 0  /* TYPE(BOOLEAN)
) RETURNS LONGWORD;  /* TYPE(CONDVALU);

/* $SETSSF
/*
/*   Set System Service Filter
/*
/*   $SETSSF [mask]
/*
/*   mask = flags for services to inhibit
/*
ENTRY SYS$SETSSF ALIAS $SETSSF PARAMETER (
  LONGWORD UNSIGNED VALUE NAMED "MASK" DEFAULT 0  /* TYPE(MASK)
) RETURNS LONGWORD;  /* TYPE(CONDVALU);

/* $SETSTK
```

```
/*
/* SET VIRTUAL STACK LIMITS
/*
/* $SETSTK inadr ,[retadr] ,[acmode]
/*
/* inadr = address of 2-longword array containing starting and
/* ending virtual address of stack limits to set
/*
/* retadr = address of a 2-longword array to receive starting and
/* ending virtual address of stack limits to set
/*
/* acmode = access mode for the stack to change
/*
ENTRY SYS$SETSTK ALIAS $SETSTK PARAMETER (
LONGWORD UNSIGNED DIMENSION 2 NAMED INADR IN,  { /* TYPE(VARANGE),
LONGWORD UNSIGNED DIMENSION 2 NAMED RETADR OUT DEFAULT 0, { /* TYPE(VARANGE),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0 { /* TYPE(ACMODE)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/* $SETSWM
/*
/* Set Process Swap Mode
/*
/* $SETSWM [swpflg]
/*
/* swpflg = swap indicator
/* 0 -> enable swapping 1 -> disable swapping
/*
ENTRY SYS$SETSWM ALIAS $SETSWM PARAMETER (
BOOLEAN VALUE NAMED SWPFLG DEFAULT 0 { /* TYPE(NUMBER)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/* $SENDACC
/*
/* Send Message to Accounting Manager
/*
/* $SENDACC msgbuf ,[chan]
/*
/* msgbuf = address of message buffer string descriptor
/*
/* chan = number of channel assigned to mailbox to receive reply
/*
ENTRY SYS$SENDACC ALIAS $SENDACC PARAMETER (
CHARACTER DESCRIPTOR NAMED MSGBUF IN, { /* TYPE(CHARDESC),
WORD UNSIGNED VALUE NAMED CHAN DEFAULT 0 { /* TYPE(CHANNEL)
) RETURNS LONGWORD; { /* TYPE(CONDVALU);

/* $SENDERR
/*
/* Send Message to Error Logger
/*
/* $SENDERR msgbuf
/*
```

```

/*      msgbuf = address of message buffer string descriptor
/*
ENTRY SYS$SENDERR ALIAS $SENDERR PARAMETER (
CHARACTER DESCRIPTOR NAMED MSGBUF IN      (/* TYPE(CHARDESC)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/* $SENDOPR
/*
/*      Send Message to Operator
/*
/*      $SENDOPR msgbuf ,[chan]
/*
/*      msgbuf = address of message buffer string descriptor
/*
/*      chan   = number of channel assigned to mailbox to receive reply
/*
ENTRY SYS$SENDOPR ALIAS $SENDOPR PARAMETER (
CHARACTER DESCRIPTOR NAMED MSGBUF IN,      (/* TYPE(CHARDESC),
WORD UNSIGNED VALUE NAMED CHAN DEFAULT 0      (/* TYPE(CHANNEL)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/* $SENDMSMB
/*
/*      Send Message to Symbiont Manager
/*
/*      $SENDMSMB msgbuf ,[chan]
/*
/*      msgbuf = address of message buffer string descriptor
/*
/*      chan   = number of channel assigned to mailbox to receive reply
/*
ENTRY SYS$SENDMSMB ALIAS $SENDMSMB PARAMETER (
CHARACTER DESCRIPTOR NAMED MSGBUF IN,      (/* TYPE(CHARDESC),
WORD UNSIGNED VALUE NAMED CHAN DEFAULT 0      (/* TYPE(CHANNEL)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/* $SENDJBC
/*
/*      Send Message to Job Controller
/*
/*      $SENDJBC [efn], func, [nullarg], [itmlst],
/*      ($SENDJBCW) [iosb], [astadr], [astprm]
/*
/*      efn      = event flag to be set when request completes
/*      func     = code specifying function to be performed
/*      nullarg  = reserved argument for similarity with $getxxx services
/*      itmlst   = address of a list of item descriptors for the operation
/*      iosb    = address of a quadword status block to receive the final status
/*      astadr  = address of an ast routine to be called when request completes
/*      astprm  = 32-bit ast parameter
/*
ENTRY SYS$SENDJBC ALIAS $SENDJBC PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0,      (/* TYPE(EFNUM),

```

```

WORD UNSIGNED VALUE NAMED FUNC, (/* TYPE(FUNCCODE),
LONGWORD UNSIGNED VALUE NAMED NULLARG DEFAULT 0, (/* TYPE(NULLARG),
ANY NAMED ITMLST IN DEFAULT 0, (/* TYPE(ITEMLIST),
QUADWORD UNSIGNED NAMED IOSB OUT DEFAULT 0, (/* TYPE(IOSB),
ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0, (/* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0 (/* TYPE(USERPARM)
) RETURNS LONGWORD; (/* TYPE(CONDVALU);

```

```

ENTRY SYSSNDJBCW ALIAS $SNDJBCW PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0, (/* TYPE(EFNUM),
WORD UNSIGNED VALUE NAMED FUNC, (/* TYPE(FUNCCODE),
LONGWORD UNSIGNED VALUE NAMED NULLARG DEFAULT 0, (/* TYPE(NULLARG),
ANY NAMED ITMLST IN DEFAULT 0, (/* TYPE(ITEMLIST),
QUADWORD UNSIGNED NAMED IOSB OUT DEFAULT 0, (/* TYPE(IOSB),
ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0, (/* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0 (/* TYPE(USERPARM)
) RETURNS LONGWORD; (/* TYPE(CONDVALU);

```

```

/*
/* $SPACE
/*
/* Space to Record in File
/*
/* $SPACE rab, [err], [suc]
/*
/* rab = address of rab
/*
/* err = address of user error completion routine
/*
/* suc = address of user success completion routine
/*

```

```

ENTRY SYSSPACE ALIAS $SPACE LINKAGE $RMSCALL PARAMETER (
ANY NAMED RAB IN OUT, (/* TYPE(RAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL, (/* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL (/* TYPE(ASTADR)
) RETURNS LONGWORD; (/* TYPE(CONDVALU);

```

```

/* $SUSPND
/*
/* Suspend Process
/*
/* $SUSPND [pidadr] ,[prcnam]
/*
/* pidadr = address of process id of process to be suspended
/*
/* prcnam = address of name string descriptor of process
/*

```

```

ENTRY SYSSUSPND ALIAS $SUSPND PARAMETER (
LONGWORD UNSIGNED NAMED PIDADR IN OUT DEFAULT 0, (/* TYPE(PROCID),
CHARACTER DESCRIPTOR NAMED PRCNAM IN DEFAULT 0 (/* TYPE(PROCNAME)
) RETURNS LONGWORD; (/* TYPE(CONDVALU);

```

```

/* $SYNCH
/*

```

```

/* Synchronize Event Completion
/*
/*  $$SYNCH [iosb] ,[efn]
/*
/*  efn    = event flag to be set at completion
/*
/*  iosb   = address of a quadword I/O status block
/*
ENTRY SYSSYNCH ALIAS $$SYNCH PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0,    { /* TYPE(EFNUM),
QUADWORD UNSIGNED NAMED IOSB OUT DEFAULT 0      { /* TYPE(IOSB)
) RETURNS LONGWORD;    { /* TYPE(CONDVALU);

/* $STRNLNM
/*
/* Translate Logical Name
/*
/* $STRNLNM [attr], tabnam, [lognam], [acmode], [itemlist]
/*
/* attr = address of logical name translation attributes
/*
/* Attribute      Meaning
/* LNMSM_CASE_BLIND Search of string is case insensitive
/*
/* tabnam = address of logical name table name string descriptor
/*
/* lognam = address of logical name string descriptor
/*
/* acmode = address of access mode for logical name
/*
/* itmlst = address of a list of item descriptors
/*
ENTRY SYSSTRNLNM ALIAS $STRNLNM PARAMETER (
LONGWORD UNSIGNED NAMED ATTR DEFAULT 0, { /* TYPE(MASK),
CHARACTER DESCRIPTOR NAMED TABNAM IN,   { /* TYPE(LOGNAME),
CHARACTER DESCRIPTOR NAMED LOGNAM IN DEFAULT 0, { /* TYPE(LOGNAME),
BYTE UNSIGNED NAMED ACMODE DEFAULT 0,    { /* TYPE(ACMODE),
ANY NAMED ITMLST IN DEFAULT 0 { /* TYPE(ITEMLIST)
) RETURNS LONGWORD;    { /* TYPE(CONDVALU);

/* $STRNLOG
/*
/* Translate Logical Name
/*
/* $STRNLOG lognam ,[rslLEN] ,rslbuf ,[table] ,[acmode] ,[dsbmsk]
/*
/* lognam = address of logical name string descriptor
/*
/* rslLEN = address of word to receive length of resultant name
/*
/* rslbuf = address of descriptor for buffer to receive resultant
/*          string
/*
/* table = address of byte to receive logical name table number

```

```

/*
/*  acmode = address of byte to receive access mode of entry
/*          (process table only)
/*
/*  dsbmsk = table search disable mask
/*
/*      Bit Set   Meaning
/*
/*      0         Do not search system table
/*      1         Do not search group table
/*      2         Do not search process table
/*

```

```

ENTRY SYS$STRNLOG ALIAS $STRNLOG PARAMETER (
CHARACTER DESCRIPTOR NAMED LOGNAM IN,  (/* TYPE(LOGNAME),
WORD UNSIGNED NAMED RSLLEN OUT DEFAULT 0, (/* TYPE(NUMBER),
CHARACTER DESCRIPTOR NAMED RSLBUF OUT, (/* TYPE(CHARDESC),
BYTE UNSIGNED NAMED TABLE OUT DEFAULT 0, (/* TYPE(NUMBER),
BYTE UNSIGNED NAMED ACMODE OUT DEFAULT 0, (/* TYPE(ACMODE),
LONGWORD UNSIGNED VALUE NAMED DSBMSK DEFAULT 0 (/* TYPE(MASK)
) RETURNS LONGWORD; (/* TYPE(CONDVALU);

```

```

/*
/* $STRUNCATE
/*
/* Truncate Record
/*
/* $STRUNCATE rab, [err], [suc]
/*
/* rab = address of rab
/*
/* err = address of user error completion routine
/*
/* suc = address of user success completion routine
/*

```

```

ENTRY SYS$STRUNCATE ALIAS $STRUNCATE LINKAGE $RMSCALL PARAMETER (
ANY NAMED RAB IN OUT, (/* TYPE(RAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL, (/* TYPE(ADDR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL (/* TYPE(ADDR)
) RETURNS LONGWORD; (/* TYPE(CONDVALU);

```

```

/*
/* $SULKPAG
/*
/* Unlock Pages From Memory
/*
/* $SULKPAG inadr [,retadr] [,acmode]
/*
/* inadr = address of 2-longword array containing starting and
/*          ending virtual addresses of pages to be unlocked
/*
/* retadr = address of a 2-longword array to receive starting and
/*          ending virtual addresses of pages actually unlocked
/*
/* acmode = access mode to check against the owner of the pages
/*

```

```
ENTRY SYSSULKPAG ALIAS SULKPAG PARAMETER (
LONGWORD UNSIGNED DIMENSION 2 NAMED INADR IN,  { /* TYPE(VARANGE),
LONGWORD UNSIGNED DIMENSION 2 NAMED RETADR OUT DEFAULT 0,  { /* TYPE(VARANGE),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0 { /* TYPE(ACMODE)
) RETURNS LONGWORD;  { /* TYPE(CONDVALU);
```

```
/* $ULWSET
/*
/* Unlock Pages From Working Set
/*
/* $ULWSET inadr ,[retadr] ,[acmode]
/*
/* inadr = address of 2-longword array containing starting and
/* ending virtual addresses of pages to be unlocked
/*
/* retadr = address of a 2-longword array to receive starting and
/* ending virtual addresses of pages actually unlocked
/*
/* acmode = access mode to check against the owner of the pages
/*
```

```
ENTRY SYSSULWSET ALIAS SULWSET PARAMETER (
LONGWORD UNSIGNED DIMENSION 2 NAMED INADR IN,  { /* TYPE(VARANGE),
LONGWORD UNSIGNED DIMENSION 2 NAMED RETADR OUT DEFAULT 0,  { /* TYPE(VARANGE),
LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0 { /* TYPE(ACMODE)
) RETURNS LONGWORD;  { /* TYPE(CONDVALU);
```

```
/* $UNWIND
/*
/* Unwind Call Stack
/*
/* $UNWIND [depadr] ,[newpc]
/*
/* depadr = address of longword containing number of logical frames
/* (depth) to unwind call stack
/*
/* newpc = address to be given control when the unwind is complete
/*
```

```
ENTRY SYSSUNWIND ALIAS $UNWIND PARAMETER (
LONGWORD UNSIGNED NAMED DEPADR IN DEFAULT 0,  { /* TYPE(NUMBER),
LONGWORD UNSIGNED NAMED NEWPC IN DEFAULT 0  { /* TYPE(ADDRESS)
) RETURNS LONGWORD;  { /* TYPE(CONDVALU);
```

```
/* $UPDATE
/*
/* Update Record
/*
/* $UPDATE fab, [err], [suc]
/*
/* rab = address of rab
/*
/* err = address of user error completion routine
/*
```

```

/*      suc      = address of user success completion routine
/*
ENTRY SYSSUPDATE ALIAS $UPDATE LINKAGE $RMSCALL PARAMETER (
  ANY NAMED RAB IN OUT,      (/* TYPE(RAB),
  ADDRESS(ENTRY) NAMED ERR OPTIONAL,      (/* TYPE(ASTADR),
  ADDRESS(ENTRY) NAMED SUC OPTIONAL,      (/* TYPE(ASTADR)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

/* $UPDSEC
/*
/*      Update Section File on Disk
/*
/*      $UPDSEC inadr ,[retadr] ,[acmode] ,[updflg] ,[efn] ,[iosb]
/*      ($UPDSECW) ,[astadr] ,[astprm]
/*
/*      inadr = address of 2-longword array containing starting and
/*              ending addresses of the pages to be potentially
/*              written
/*
/*      retadr = address of 2-longword array to receive addresses of
/*              the first and last page queued in the first I/O
/*              request
/*
/*      acmode = access mode on behalf of which the service is
/*              performed
/*
/*      updflg = update indicator for read/write global sections
/*              0 -> write all read/write pages in the section
/*              1 -> write all pages modified by the caller
/*
/*      efn    = number of event flag to set when the section file is
/*              updated
/*
/*      iosb   = address of quadword I/O status block
/*
/*      astadr = address of entry mask of an AST service routine
/*
/*      astprm = AST parameter to be passed to the AST service routine
/*
ENTRY SYSSUPDSEC ALIAS $UPDSEC PARAMETER (
  LONGWORD UNSIGNED DIMENSION 2 NAMED INADR IN,      (/* TYPE(VARANGE),
  LONGWORD UNSIGNED DIMENSION 2 NAMED RETADR OUT DEFAULT 0,      (/* TYPE(VARANGE),
  LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0,      (/* TYPE(ACMODE),
  BOOLEAN VALUE NAMED UPDFLG DEFAULT 0,      (/* TYPE(NUMBER),
  LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0,      (/* TYPE(EFNUM),
  QUADWORD UNSIGNED NAMED IOSB OUT DEFAULT 0,      (/* TYPE(IOSB),
  ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0,      (/* TYPE(ASTADR),
  LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0      (/* TYPE(USERPARM)
) RETURNS LONGWORD;      (/* TYPE(CONDVALU);

ENTRY SYSSUPDSECW ALIAS $UPDSECW PARAMETER (
  LONGWORD UNSIGNED DIMENSION 2 NAMED INADR IN,      (/* TYPE(VARANGE),
  LONGWORD UNSIGNED DIMENSION 2 NAMED RETADR OUT DEFAULT 0,      (/* TYPE(VARANGE),
  LONGWORD UNSIGNED VALUE NAMED ACMODE DEFAULT 0,      (/* TYPE(ACMODE),
  BOOLEAN VALUE NAMED UPDFLG DEFAULT 0,      (/* TYPE(NUMBER),

```

```
LONGWORD UNSIGNED VALUE NAMED EFN DEFAULT 0,  (/* TYPE(EFNUM),
QUADWORD UNSIGNED NAMED IOSB OUT DEFAULT 0,  (/* TYPE(IOSB),
ADDRESS(ENTRY) NAMED ASTADR DEFAULT 0,  (/* TYPE(ASTADR),
LONGWORD UNSIGNED VALUE NAMED ASTPRM DEFAULT 0 (/* TYPE(USERPARM)
) RETURNS LONGWORD;  (/* TYPE(CONDVALU);
```

```
/*
/* $WAIT
/*
/* Wait on File
/*
/* $WAIT rab
/*
/* rab = address of rab
/*
/*
```

```
ENTRY SYSSWAIT ALIAS $WAIT LINKAGE $RMSCALL PARAMETER (
ANY NAMED RAB IN OUT (/* TYPE(RAB)
) RETURNS LONGWORD; (/* TYPE(CONDVALU);
```

```
/* $WAITFR
/*
/* Wait for Single Event Flag
/*
/* $WAITFR efn
/*
/* efn = event flag number to wait for
/*
```

```
ENTRY SYSSWAITFR ALIAS $WAITFR PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN (/* TYPE(EFNUM)
) RETURNS LONGWORD; (/* TYPE(CONDVALU);
```

```
/* $WAKE
/*
/* Wake
/*
/* $WAKE [pidadr] ,[prcnam]
/*
/* pidadr = address of process id of process to be awakened
/*
/* prcnam = address of name string descriptor of process to be
/* awakened
/*
```

```
ENTRY SYSSWAKE ALIAS $WAKE PARAMETER (
LONGWORD UNSIGNED NAMED PIDADR IN OUT DEFAULT 0, (/* TYPE(PROCID),
CHARACTER DESCRIPTOR NAMED PRCNAM IN DEFAULT 0 (/* TYPE(PROCNAME)
) RETURNS LONGWORD; (/* TYPE(CONDVALU);
```

```
/* $WFLAND
/*
/* Wait for Logical AND of Event Flags
/*
```

```
/* $WFLAND efn ,mask
/*
/* efn = event flag number of any flag within the cluster
/* mask = 32-bit mask of flags that must be set
/*
ENTRY SYSSWFLAND ALIAS $WFLAND PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN,      (/* TYPE(EFNUM),
LONGWORD UNSIGNED VALUE NAMED 'MASK'    (/* TYPE(MASK)
) RETURNS LONGWORD:      (/* TYPE(CONDVALU);

/* $WFLOR
/*
/* Wait for Logical OR of Event Flags
/*
/* $WFLOR efn ,mask
/*
/* efn = event flag number of any flag within the cluster
/*
/* mask = 32-bit mask of flags, any of which must be set
/*
ENTRY SYSSWFLOR ALIAS $WFLOR PARAMETER (
LONGWORD UNSIGNED VALUE NAMED EFN,      (/* TYPE(EFNUM),
LONGWORD UNSIGNED VALUE NAMED 'MASK'    (/* TYPE(MASK)
) RETURNS LONGWORD:      (/* TYPE(CONDVALU);

/* $WRITE
/*
/* Write Block to File
/*
/* $WRITE rab, [err], [suc]
/*
/* rab = address of rab
/*
/* err = address of user error completion routine
/*
/* suc = address of user success completion routine
/*
ENTRY SYSSWRITE ALIAS $WRITE LINKAGE $RMSCALL PARAMETER (
ANY NAMED RAB IN OUT,      (/* TYPE(RAB),
ADDRESS(ENTRY) NAMED ERR OPTIONAL,      (/* TYPE(ASTADR),
ADDRESS(ENTRY) NAMED SUC OPTIONAL      (/* TYPE(ASTADR)
) RETURNS LONGWORD:      (/* TYPE(CONDVALU);
```

```
END_MODULE;
```

