The LMF User Primer

Order Number: preliminary

This document describes the DIGITAL Vendor-Specific Module (DVSM) Interface to the VMS™ License Management Facility (LMF). This document is divided into two parts: part one provides an overview of the LMF and part numbering; part two describes the subroutine calls necessary to support LMF in a layered product.

This document is intended for use with layered products only; no attempt is made to explain the licensing scheme for the VMS system-integrated products (SIPs).

Tabular information included herein is from the U.S. Price Book, 10-January-1989.

For further information on the LMF, consult the *License Management Utility Manual*, documentation part number AA-LA33A-TE. For internal-use information on the Loan-of-Products program, consult the *Software Loan of Products Guide*, part number EE-C0290-76.

"The Unauthorized LMF Documentation"
contact Stephen (VOX::) Hoffman with comments
April 3, 1990

Part I LMF License Routines Primer

1 2-5-2 Part Number Overview

1.1 Part Number Format

All new DIGITAL software part numbering is based on the same scheme. The part number is broken down into a two character prefix, a five character part number, and a two character suffix. The scheme is often known as the 2-5-2 part number format. The 2-5-2 format looks like:

Chapter 1 describes the part number format in detail. The License Management Facility, and the relationship of the LMF with part numbers, is covered in Chapter 2.

1.1.1 The Prefix

The first two characters, known in the vernacular as the *prefix*, describe the general classification of the part number. For example, the prefix is what differentiates the software binary kit from the software license. Table 1–1 shows the most common prefixes.

Table 1–1 Software-Related License Prefixes

Prefix	Description
QL	License
QT	SPS Services
QA	Media and Documentation kits
QB	Packages Software Components

1.1.2 The Body

The middle five positions of the part number are comprised of three characters for the Unique Product Identifier (UPI), one character for the language variant (see Table 1–2) and the final character for the processor type (see Table 1–3). A unique UPI is assigned to every software product that DIGITAL sells.

Table 1–2 License Language Types

Code	Description
A	American English
B	British English
9	Language Neutral

Table 1–3 License Processor Codes

Code	Processor Name/System Marketing Model
B	MicroVAX™ 3500, 3600
C	VAXstation™ 3200, 3500, 8000, 3100, VAXserver™ 3500, 36mm, 3300, 3400
D	VAX™ 6240, 8800, 8820, 8820N
E	VAX 8700, 8550, 8810
F	VAX 86VV
G	VAX 8530, VAXserver 6220
H	VAX 83SS, VAXserver 6210
J	VAX 82SS, MicroVAX 3300, 3400
K	VAX 11/78S
L	VAX 11/750
M	VAX 11/730
N	MicroVAX II
P	MicroVAX 2000
Q	VAXstation II
R	VAXstation 2000
U	VAX 8830
V	VAX 8840, 8974, 8842
W	PC Software
X	PDP™ Cross Product Generic
2	VAX 6210
3	VAX 6220
4	VAX 6230
5	VAX 8978
9	Processor Tier Neutral/VAX Cross Product Generic

1.1.3 The Suffix

The suffix, also known as the *variant*, contains information specific to the part number prefix. Each prefix has a valid set of variants. Consult the current price book for the possible variants.

For license (QL) part numbers, the general license type is in the first position of the suffix (see Table 1–4), and license-specific information in the second position (see the front of the software section in the current price book for details).

Table 1–4 License Types

Type	License Type Description
A	Traditional License
J	ClusterWide License
L	Loan-of-Products License
P	User/Activity License

The LMF design allows great flexibility on licensing. For all 2-5-2 parts with a QL prefix, the suffix calls out the type of license. The type of license determines what will appear on the PAK. Section 1.1.3.1, Section 1.1.3.2, Section 1.1.3.3 and Section 1.1.3.4 describe the current, major flavours of product licenses. Example 2–1 contains an example PAK.

1.1.3.1 Traditional Or Tiered Licenses

A traditional license provides the customer the right to execute a given software product on a single processor.

The traditional license, license type *A*, is a tiered license; that is, the cost of a license for the same product, when purchased for a faster VAX processor, is correspondingly higher. The LMF determines the relative processor performance by consulting the License Unit Requirement Table (LURT). The LURT is provided and maintained by VMS; the LURT is updated when new processors are released.

The first traditional license in a VAXcluster must be a standalone traditional license. (A part number with an *AA* suffix) Subsequent systems must use, if available, the *AB* cluster member license. If the cluster member license, the AB license, is not available, the AA license must be purchased. The AB license may remain in a VAXcluster only if accompanied by an AA licensed system.

In LMF-speak, the tiered, per-processor, licenses are AVAILIBILITY charged. See Section 2.1.3.3.

Example part numbers are:

1.1.3.2 ClusterWide Licenses

The ClusterWide License, license type *J*, provides for the execution of layered products on all processors within a cluster provides that the combined license rating of all processors within the cluster executing the software at a given time shall not exceed the combined license rating of the ClusterWide license(s) designated to the cluster for a given software product.

Rephrased somewhat, a ClusterWide license provides a pool of license units that are available for use by any system in the VAXcluster. When a ClusterWide license is loaded (activated) on a specific processor, a processor-specific number of units are deducted from the pool of available units. Conversely, when a ClusterWide license is unloaded (deactivated) the number of units deducted by the activation are returned to the VAXcluster pool of available units. The specified software product is licensed and is available for use only when the license units are loaded.

This license is purchased on a unit basis and is available for standalone and well as VAXclusters.

LMF supports a combination of the ClusterWide license with the User/Activity license mentioned below. In LMF-speak, the ClusterWide and tiered per-processor licenses are AVAILIBILITY charged. See Section 2.1.3.3.

Example part numbers are:

1.1.3.3 User/Activity Licenses

The layered product user/activity license, license type *P*, is a per-user or a per-activity license. This license restricts the number of simultaneous users, or activations, of the licensed product.

The User/Activity license is very similar to the ClusterWide license in capabilities. The salient difference is the blanket licensing granted a processor for the ClusterWide license. The User/Activity licenses provide a finer granularity of licensing.

The LMF supports the User/Activity licensing scheme combined with the ClusterWide licensing: a tiered per-processor charge, plus a charge of activations. In LMF-speak the tiered per-processor charge is the AVAILIBILITY charge. See Section 2.1.3.3. The per-user charge is the ACTIVITY charge. See Section 2.1.3.4.

An example part number is:

1.1.3.4 Loan-of-Products Licenses

The loan-of-products license, license type *L*, is a special-purpose license.

For information on the Loan-of-Products program, consult the *Software Loan of Products Guide*, part number EE-C0290-76.

2 LMF Overview

2.1 Product Authorization Key

Now for a brief overview of the VMS License Management Facility and the Product Authorization Key (PAK).

Temporary Service PAKs (TSPs), used to fix customer problems and for product demonstrations, are obtained from an LKG via the Customer Support Centers. (See Table 2–1)

Products released before VMS V5.0 are required to issue a Service Update PAK (SUP) with each release after VMS V5.0. All authorization number fields for SUPs commence with the three characters "AWS". PAKs intended for internal use are obtained from various sources depending on employee location and planned usage for the PAK: consult the notesfile VOX::VOICE_PRODUCTS note 329.* for more information.

Table 2–1 Customer Support Numbers

Number	Center
800-525-7100	Colorado Customer Support/Software
800-525-6570	Colorado Customer Support/Hardware
800-332-8000	Georgia Customer Support Center/HW & SW
800-272-2001	Massacusetts Customer Support/HW & SW

2.1.1 The License Key Generator

The PAKs for all products are obtained from a License Key Generator (LKG) system. Access to an LKG is typically subject to rather stringent restrictions. Software Shipping Business (SSB, formerly SDC) has access to an LKG for generation of PAKs for external sites and official internal distributions. Software Quality Management (SQM) has an LKG, currently located on MADRID::, used by product development to generate PAKs for both internal and field test purposes.

2.1.2 The License Database

Licenses are stored in the license database. During bootstrap, the VMS system loads all active, valid, unexpired licenses for the local node. Licenses can also be loaded by an explicit DCL "LICENSE LOAD" command. On node shutdown or crash, all licenses are released and all availability and activity charges are cleared.

The license database can be shared across nodes in a VAXcluster™; again, subject to the number of units available, the types of license(s) present, and the terms and conditions of the license(s).

The ClusterWide™ licenses and all layered products typically use the common license database file: SYS$SYSTEM:LMF$LICENSE.LDB. This database is shared among all nodes in a VAXcluster. Some system-specific licenses, such as VAX-VMS, reside in the system-specific database: SYS$SYSTEM:LMF$SYSTEM.LDB.

By default, the DCL LICENSE command references the LMF$LICENSE.LDB database. To alter the default database for a specific DCL LICENSE command, specify the /DATABASE qualifier.

For further information consult the *License Management Utility Manual*, documentation part number AA-LA33A-TE.

2.1.3 Breakdown of a License

The following sections describe the various component fields of a Product Authorization Key.

An example PAK is located in Example 2–1. Table 2–3 contains a synopsis of the various fields.

The major fields of a PAK are the *product name*, *units*, *availability*, and *activity* fields. Various other fields may also be specified.

2.1.3.1 Product Name

The product name is the name of the product that is licensed. Each product name is typically associated with a Unique Product Identifier (UPI, see Section 1.1.2). The product name may represent a single product, or it may be the name of a group of products sold together as a package.

2.1.3.2 Number Of Units

The *number of units* field on the PAK controls the maximum number of (completely arbitrary) units of license that can be used on the system. A zero indicates there is no limitation. If specified, the *key options* field value "MOD_UNITS" allows the System Manager to modify the number of units on a license, subject to the terms and conditions of the license.

2.1.3.3 Availability Table Code

The *availability table code* field determines the number of units to deduct when the license is loaded (made available) on the VMS system. The code can be a letter code, layered products typically use an *F* character, or it can be a constant value, such as "CONSTANT=0". Table 2–2 contains the complete list of valid availability codes.

The availibility codes used on many tiered products (see Section 1.1.3.1) determine the specific charges used for the current processor. The charge table for the current processor, broken down by the product availibility codes, can be displayed with the DCL command SHOW LICENSE/CHARGE_TABLE. The following charge table display is from a VAX 8800:

```
$  SHOW LICENSE/CHARGE_TABLE
VMS/LMF Charge Information for node VOX
This is a VAX 8800, hardware model type 18
Type: A, Units Required: 93     (VMS Capacity)
Type: B, * Not Permitted *      (VMS Server)
Type: C, * Not Permitted *      (VMS Concurrent User)
Type: D, * Not Permitted *      (VMS Workstation)
Type: E, Units Required: 400    (System Integrated Products)
Type: F, Units Required: 1200   (Layered Products)
```

An *availibility table code* of *F* indicates VMS is to determine the unit charge based on the standard processor tier unit charge from the License Unit Requirement Table (LURT). The *F* availibility code is the traditional tiered license, as implemented in the LMF environment. It is the most common availability table code for DIGITAL layered products.

The "CONSTANT" option in the availability table code field forces a constant number of units regardless of the processor. "CONSTANT=0" indicates a license charged solely on the activity.

Table 2–2 Availability Codes

Type	License Type Description
A	VMS Capacity/Timeshare
B	VMS Server
C	VMS Concurrent User
D	VMS Workstation
E	VMS System Integrated Products
F	Layered Product/LURT-based Charging
CONSTANT	User/Activity License

2.1.3.4 Activity Table Code

The *activity table code* field, which is typically non-blank only on "CONSTANT" availability licenses, controls the number of units deducted for each instance of active use of the licensed layered product. The LMF software dynamically determines the activity deduction for the *F* availability code based on the License Unit Requirement Table (LURT)—thus nothing is present in the activity table code field.

2.1.3.5 Miscellaneous Fields on the PAK

Table 2–3 describes the various fields found on a Product Authorization Key. The contents of these fields vary, depending on the PAK.

Table 2–3 PAK Fields

Field	Description
Issuer	This field indicates the organization that issued the PAK. For DIGITAL products, this field will typically be set to DEC.
Authorization	This is a number used to track the PAK. A PAK with the first three characers "AWS" is a Service Update PAK (SUP).
Product Name	This is the LMF name of the licensed product, or group of products.
Producer	This is the number of the product producer.
Units	This is the number of units available for the product. For tiered licenses, zero indicates no limit.
Version	This is the numeric version number, in the form mm.nn where mm is the major identifier and nn is the minor identifer, of the highest product release that this license can be used with. Cannot be combined with product release date.
Release Date	This is the date of the final product release with which the PAK can successfully be used on. The product release date is advanced only when a new release of the product is installed. This field has a use similar to, but cannot be combined with, the product version field. Not to be confused with termination date.
Termination Date	This is the last calendar date on which the PAK can be used on. The PAK is said to have expired when the termination date has passed. Not to be confused with release date.
Availability	The number of units deducted when the product is made available. Explained in Section 1.1.3.3.
Activity	The number of units deducted when the product is activated. Explained in Section 1.1.3.3.
Options	This field has various uses, most common are the NO_SHARE and the MOD_UNITS settings. NO_SHARE ties the PAK to a system. MOD_UNITS allows modification of the units field.
Product Token	This field is typically not used.
Hardware-ID	This field is typically not used.
Checksum	This field contains a checksum of all the other fields on the PAK. The first character of the checksum field is a numeric, the rest of the checksum contains alphabetic characters only.

2.2 Part Numbers and the LMF

Now for the tie-in with the layered product part numbers: an LMF availability code of *F* has a part number with a prefix of "QL-", the product UPI, language and processor, and a suffix. (See Table 2–4) User/Activity licenses have a suffix from Table 2–5).

The License Key Generator (LKG) system used by SBB (formerly SDC) parses the part number for content and validity, and in the case of a user/activity license, multiples the number of users (indicated by the final character in the "-P*" suffix) times the value of the activity constant to determine the number of units to generate on the customer PAK.

Table 2–4 Tiered License Suffixes

Suffix	Description
AA	Traditional License
AB	Traditional VAXcluster License
A*	Other variants—see the price book
L*	Loan-of-Products License
J*	First ClusterWide License
U*	Subsequent ClusterWide Licenses

Table 2–5 Non-Tiered License Suffixes

Suffix	Description
L*	Loan-of-Products License
P*	User/Activity License

An example Product Authorization Key is located in Example 2–1.

Example 2–1 PAK Example

```
 - - - - - - -
| | | | | | | |   LICENSE SOFTWARE PRODUCT     | DOCUMENT ISSUE DATE |
|d|i|g|i|t|a|l|   PRODUCT AUTHORIZATION KEY    |    14-APR-1989      |
| | | | | | | |                                |_____________________|
 - - - - - - -
Digital Equipment Corporation
Maynard, MA.

 _____________________________________
| LICENSE ADMINISTRATION LOCATION:    |  ORDERED BY: Enterprise Eng.
|                                     |              Dept. NCC-1701D
| Digital Equipment Corporation       |              San Francisco, CA.
| Software Quality Management/        |
|    Systems Quality Group            |
| 110 Spit Brook Road                 |
| Nashua, N.H. 03062                  |
 -------------------------------------
**********************************************************************
 PAK ID:
                    Issuer: DEC
      Authorization Number: SQM006887

 PRODUCT ID:
               Product Name: VOX
                   Producer: DEC

 NUMBER OF UNITS:
            Number of units: 0

 KEY LEVEL:
                    Version: 1.1
      Product Release Date:

 KEY TERMINATION DATE:
      Key Termination Date: 31-JUL-1989

 RATING:
   Availability Table Code: F
       Activity Table Code:

 MISCELLANEOUS:
               Key Options: MOD_UNITS
             Product Token:
               Hardware-Id:
                  Checksum: 1-BOGU-SCHE-CKSU-MMIS
**********************************************************************
```

Part II LMF License Routines

SYS$LOOKUP_LICENSE

SYS$LOOKUP_LICENSE is intended to be the primary interface for the VAX/VMS License Management Facility for DIGITAL layered products. The routine will grant the caller permission to continue execution if a valid license exists for the specified product. SYS$LOOKUP_LICENSE is part of the DIGITAL Vendor-Specific Module (DVSM).

FORMAT

SYS$LOOKUP_LICENSE *PRDNAM, ITMLST, PRODUCER [,FLAGS , CONTXT, LICPROD*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

PRDNAM

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

The product name corresponding to the name on the product authorization key license.

ITMLST

VMS Usage: **item_list_3**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

This is the address of a contiguous array of standard VMS three-longword item list structures. The item list must contain at least the items LMF$_PROD_DATE and LMF$_PROD_VERSION. The item list is terminated with a zero-filled entry.

3 LMF$_PROD_DATE, the quadword binary date when the software is to be released. This date must always be prior to the SBB shipment date. PAKs issued internally are currently coded with a release date ONE YEAR after the date the internal-use PAK is issued. A product will work with any PAK with a release date in the future relative to the release date coded into the program. (this argument establishes the release date that is coded into the product.)

4 LMF$_PROD_VERSION, the product version is encoded into a longword field. The upper word is the product major version, the lower longword is the minor version. If the version encoded into the product LMF call is less than, or equal to, the product version on the PAK, this part of the check with succeed. (This argument establishes the version coded into the product.)

PRODUCER

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

The product producer. This argument must be defaulted to zero or must be specified as the address of a descriptor referencing the string "DEC".

FLAGS

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

A flags longword containing various control bits.

0 LMF$V_RETURN_FAILURES, when set, indicates the procedure ~~with~~ will return both success and failure status to the calling routine. If this flag is clear or false, the procedure will signal all failure conditions.

1 LMF$V_BROADCAST_FAILURES, when set, OR if the logical name LMF$DISPLAY_OPCOM_MESSAGE is defined, an OPCOM message will be broadcast to all LICENSE operators on any license failures.

2 LMF$V_OUTPUT_FAILURES, if set and LMF$V_SIGNAL_FAILURES is false, license failures will result in one or more messages being output to the user via SYS$PUTMSG.

CONTXT

VMS Usage: **context**
type: **octaword (unsigned)**
access: **write only**
mechanism: **by reference**

This octaword contains the license authorization context required for license release by SYS$RELEASE_LICENSE. This argument is required only for user/activity licenses—licenses that should be released prior to image run-down.

LICPROD

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**

Will be loaded with the product name of the actual product or group licensed. This can be different than the PRDNAM field if the product is part of a group license. The field is 24 bytes and is fixed length.

DESCRIPTION

SYS$LOOKUP_LICENSE implements a DIGITAL-specific interface to the SYS$GRANT_LICENSE system service. The routine is designed specifically for use by DIGITAL layered products and includes the notion of a product group.

The procedure is not intended for use by customers or by non-DIGITAL software as it incorporates a number of DIGITAL-specific business rules.

SYS$LOOKUP_LICENSE calls the system service SYS$GRANT_LICENSE. All DIGITAL layered products should call SYS$LOOKUP_LICENSE, rather than calling SYS$GRANT_LICENSE, for consistency across all licensing implementations.

The LICENSE$_xxx return status codes are defined in, among other places, SYS$SHARE:STARLET.MLB, in the module $LICENSEDEF.

CONDITION VALUES RETURNED

SS$_NORMAL	Routine successfully completed.
LICENSE$_BADPARAM	Product name length too large or return product length buffer is too small.
LICENSE$_ILLPRODUCER	Producer argument isn't "DEC".
LICENSE$_WRONUMARG	Too few or too many arguments were specified.
LICENSE$_NOAUTH	All attempts to grant access to the caller failed.
LICENSE$_SYSMGR	Information signaled with LICENSE$_NOAUTH.
LICENSE$_NOT_STARTED	LMF has not yet been started.
STR$_xxx	Invalid string parameter specified.

SYS$RELEASE_LICENSE

SYS$RELEASE_LICENSE revokes a process' authorization to execute a licensed software product. The authorization must have been previously granted by a call to SYS$GRANT_LICENSE or to the DIGITAL-specific routine SYS$LOOKUP_LICENSE.

FORMAT

SYS$RELEASE_LICENSE *CONTXT*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

CONTXT

VMS Usage: **context**
type: **octaword (unsigned)**
access: **write only**
mechanism: **by reference**

License authorization context from a call to SYS$LOOKUP_LICENSE or SYS$GRANT_LICENSE.

DESCRIPTION

SYS$RELEASE_LICENSE revokes a process' authorization to execute a licensed product. This subroutine is generally useful for user/activity licenses only.

SYS$RELEASE_LICENSE is available to release licenses prior to image run-down. This routine allows the implemwentation of user or activity based licenses.

The LICENSE$_xxx return status codes are defined in, among other places, SYS$SHARE:STARLET.MLB, in module $LICENSEDEF.

CONDITION VALUES RETURNED

SS$_NORMAL	Routine successfully completed.
SS$_xxx	System service error.

SYS$GETLUI

SYS$GETLUI returns information about the in-memory license database. SYS$GETLUI is not yet implemented.

FORMAT

SYS$GETLUI *PRDNAM, PRODUCER, ITMLST*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

PRDNAM

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

The product name corresponding to the name on the product authorization key license.

PRODUCER

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Address of a descriptor containing the product producer. The argument must be defaulted to zero or specified as a string descriptor for the ASCII string "DEC".

ITMLST

VMS Usage: **item_list_3**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Address of a contiguous array of standard VMS three-longword item list entries. The list is terminated with an itemlist entry containing zeroes.

DESCRIPTION

The SYS$GETUAI call is not currently implemented.

The LICENSE$_xxx return status codes are defined in, among other places, SYS$SHARE:STARLET.MLB, in module $LICENSEDEF.

CONDITION VALUES RETURNED

SS$_NORMAL	Routine successfully completed.
SS$_EXENQLM	Process has exceeded its ENQUEUE authorization quota.
SS$_INSFVM	Sufficient system dynamic memory does not exist to allocate the lock or resource block.
SS$_ACCVIO	Product or producer strings cannot be read, or the itemlist buffers cannot be read or written by the calling access mode.
SS$_BADPARAM	Invalid iten code specified.
LICENSE$_NOLICENSE	No license exists for this product in the in-memory database.
LICENSE$_NOT_STARTED	LMF has not yet been started.
STR$_xxx	Invalid string parameter specified.

A Program Example

A VAX C example program that makes calls to the routines of the DIGITAL vendor-specific module (DVSM) is shown in Example A–1.

Example A–1 DVSM LMF Program Example

```
#module vox$$checks "T1.1-001"
/*
/* vox$$lmf_check()
/* vox$$lmf_cashin()
/*
/*
/* vox$$lmf_check.c
/*  this subroutine is used to check for a hardwired product
/*  license. If the license is not installed the subroutine
/*  signals the error.
/*
/*  vox$$lmf_check( [product, flags, contxt ] [, itmlst] )
/*
/*  product addr (or a zero) of a descr of the product to
/*     check for (which, for our purposes, defaults to
/*     a descriptor to the product "VOX")
/*
/*  flags   addr of LMF flags longword to be passed into LMF.
/*
/*  contxt  addr (or zero) of a LONGWORD used to hold the
/*     address of a dynamically allocated block that
/*     holds the LMF context OCTAWORD for releasing
/*     license.  (needed for user/activity lice only.)
/*
/*  itmlst  is the address of an itemlist_3 structure to be
/*     passed in.  Not currently used.  This hook allows
/*     (even weirder) variant calls and argument lists.
/*
/*  modified:
/*  10-Mar-1989 Hoffman
/* account for VOX FT1 V1.1 dates; allow a couple of
/* [optional] parameters so that we do not necessary
/* go down in flames if the license is not registered.
/* (we can now return a polite NOLICENSE status...)
/*  19-Apr-1989 Hoffman
/* Shuffled the argument list around.  Added a context
/* argument so that user/activity licenses can be
/* released.  And added the vox$$lmf_cashin() routine
/* to actually perform the releasing.
*/
/*
/*  The following four definitions are the defaults.  Change
/*  these and you can check any product (by default):
*/
#define PRODUCT  "VOX"
#define PRODUCER "DEC"
#define PROD_VERSION 0x010001
#define PROD_DATE "1-MAR-1989"
```

Example A–1 Cont'd on next page

Example A–1 (Cont.) DVSM LMF Program Example

```
/*
/*  Definitions required to get off the ground:  (note that
/*  the LMF stuff does not yet have a C header file... VAX
/*  C V3.0 cures this by adding the LMFDEF.H file.)
*/
#include "sys$share:stsdef.h"
#include "sys$share:ssdef.h"
#include "sys$share:syidef.h"
#include "sys$share:descrip.h"
#include "sys$share:varargs.h"

#define LMF$M_RETURN_FAILURES     0x01
#define LMF$M_BROADCAST_FAILURES    0x02
#define LMF$_PROD_VERSION     3
#define LMF$_PROD_DATE      4

#ifdef TEST_MAIN
/*
/*  Test main program -- for debugging purposes only.
/*  Use the command "CC/DEFINE=TEST_MAIN file.c" to
/*  enable this stub main program.
*/
main()
MAIN_PROGRAM
    {
    unsigned long int retstat;
    unsigned long int ctx = 0;
    $DESCRIPTOR( lmfcrecog, "VOX-CRECOG" );
    $DESCRIPTOR( lmfvox, "VOX" );

    /*
    /* See if default LMF "VOX" check works.
    */
    printf("checking LMF -- fully defaulted / no arguments\n");
    retstat = vox$$lmf_check();
    if (!$VMS_STATUS_SUCCESS( retstat )) LIB$SIGNAL( retstat );

    /*
    /* See if non-default LMF "VOX" check works.
    */
    printf("checking LMF -- 3 args / traditional license\n");
    retstat = vox$$lmf_check( &lmfvox,
 &LMF$M_RETURN_FAILURES, 0 );
    if (!$VMS_STATUS_SUCCESS( retstat )) LIB$SIGNAL( retstat );

    /*
    /* Check a user/activity license -- retaining context for
    /* the eventual release.
    */
    printf("checking LMF -- 3 args / user-activity license\n");
    retstat = vox$$lmf_check( &lmfcrecog,
 &LMF$M_RETURN_FAILURES, &ctx );
    if (!$VMS_STATUS_SUCCESS( retstat )) LIB$SIGNAL( retstat );

    /*
    /* And return what we have used in the above check
    */
    printf("checking LMF -- releasing user-activity license\n");
    retstat = vox$$lmf_cashin( &ctx );
    if (!$VMS_STATUS_SUCCESS( retstat )) LIB$SIGNAL( retstat );

    return( SS$_NORMAL );
    }
#endif
```

Example A–1 Cont'd on next page

Example A–1 (Cont.) DVSM LMF Program Example

```
vox$$lmf_check( va_alist )
va_dcl
    {
    va_list ap;
    int retstat;
    int SYS$LOOKUP_LICENSE();
    $DESCRIPTOR( product, PRODUCT );
    $DESCRIPTOR( producer, PRODUCER );
    $DESCRIPTOR( prod_date, PROD_DATE );
    char retprodnambuf[25];
    struct dsc$descriptor retprodnam =
{ 24, DSC$K_DTYPE_T, DSC$K_CLASS_S, retprodnambuf };
    int prod_version = PROD_VERSION;
    int argcnt;
    unsigned long int *usrflags;
    unsigned long int *usrcontxt;
    unsigned long int *contxt;
    struct dsc$descriptor *usrproduct;
    char bintim[8];
    struct itmlst_3
{
short buflen;
short itmcod;
int bufadr;
int bufrla;
} itmlst[3], *usritmlst;

    va_start( ap );
    va_count( argcnt );
    switch ( argcnt )
{
case 0:
     usrproduct = 0;
     usrflags = 0;
     usrcontxt = 0;
     usritmlst = 0;
     break;
case 3:
     usrproduct = va_arg( ap, struct dsc$descriptor * );
     usrflags = va_arg( ap, unsigned long int *);
     usrcontxt = va_arg( ap, unsigned long int * );
     usritmlst = 0;
     break;
case 4:
     usrproduct = va_arg( ap, struct dsc$descriptor * );
     usrflags = va_arg( ap, unsigned long int *);
     usrcontxt = va_arg( ap, unsigned long int * );
     usritmlst = va_arg( ap, struct itmlst_3 *);
     break;
default:
     retstat = LIB$SIGNAL( SS$_BADPARAM );
     return( SS$_BADPARAM );
     break;
}

    /*
    /* Fill in the item list required by the call.  Includes
    /* the required final, zero-filled, block.  This call will
    /* pass the version and release date checks if the currently
    /* loaded PAK is of a more recent product version or of a
    /* more recent product release date.  (And the termination
    /* date on the PAK has not passed, and various other checks
    /* succeed.)
    */
    itmlst[0].buflen = 8;
    itmlst[0].itmcod = LMF$_PROD_DATE;
    itmlst[0].bufadr = bintim;
    itmlst[0].bufrla = 0;
```

Example A–1 Cont'd on next page

Example A–1 (Cont.) DVSM LMF Program Example

```
    itmlst[1].buflen = 4;
    itmlst[1].itmcod = LMF$_PROD_VERSION;
    itmlst[1].bufadr = &prod_version;
    itmlst[1].bufrla = 0;

    itmlst[2].buflen = itmlst[2].itmcod = 0;
    itmlst[2].bufadr = itmlst[2].bufrla = 0;

    /*
    /*  convert the time from human-readable ASCII to the
    /* internal binary quadword format.
    */
    retstat = SYS$BINTIM( &prod_date, bintim );
    if ( retstat != SS$_NORMAL )
 LIB$SIGNAL( retstat );

    /*
    /* Deal with the usrcontxt argument.  If it's present,
    /* it must contain a zero.  (Signal an error if it
    /* doesn't -- programming error)  If it's not present,
    /* then assume this is not a user/activity license.
    */
    if ( (int)usrcontxt )
{
if ( (*(int*) usrcontxt ) )
    LIB$SIGNAL( SS$_BADPARAM );
else contxt = *usrcontxt = malloc( 32 );
}
    else
{
contxt = 0;
}

    /*
    /* The LMF signals on errors in the zero-argument
    /* version.  In the two argument version it does
    /* what you ask it...
    */
    if ( !((int)usrproduct )) usrproduct = &product;
    retstat = SYS$LOOKUP_LICENSE( usrproduct,
&itmlst, &producer, usrflags, contxt, 0 );

    return( retstat );
    }

/*
/*  vox$$lmf_cashin( usrcontxt )
/*  Cash in any license units we've used -- releasing
/*  them back to the available pool.  (Only needed if
/*  releasing them on process run-down doesn't cut it.
/*  This is typically only required for user/activity
/*  licensing.)
/*
/*  usrcontxt is the address of a longword.  This
/*  longword references the octaword dynamically
/*  allocated.  (This means the caller (hypothetically)
/*  has one less thing to aim at the proverbial foot
/*  -- there is no second longword to trash a variable.)
*/
vox$$lmf_cashin( usrcontxt )
unsigned long int *usrcontxt;
    {
    int retstat;
    int SYS$RELEASE_LICENSE();
    unsigned long int contxt = *usrcontxt;

    *usrcontxt = 0;

    retstat = SYS$RELEASE_LICENSE( contxt );

    free( contxt );
```

Example A–1 Cont'd on next page

Example A–1 (Cont.) DVSM LMF Program Example

```
    return( retstat );
    }
```