Multimission Software Interface Specification (SIS)

SPICE
Meta-Kernel
(a.k.a. FURNSH Kernel)

MK

NAIF Document No. 417
Version 1.0

Prepared by: C. Acton

Navigation and Ancillary Information Facility (NAIF)
Jet Propulsion Laboratory
National Aeronautics and Space Administration

PURPOSE: This SIS describes the format and content of a SPICE Meta-Kernel (MK) file. The MK file is a convenience feature that makes it easier for a SPICE user to manage which kernels are to be loaded into a program.

CHANGE LOG

Version	Date	Page Nos.	Reason
1.0	20 July 2011	All	New multimission version.

List of Acronyms

ASCII	American Standard Code for Information Interchange
JPL	Caltech/Jet Propulsion Laboratory
MK	Meta-kernel
NAIF	Navigation and Ancillary Information Facility
SIS	Software Interface Specification
SPICE	S-, P-, I-, C- and E-kernels; the principal logical data components of a particular NASA ancillary information system

Section 1
General Description

1.1 Purpose of Document

This Software Interface Specification (SIS) describes the contents and structure of a SPICE Meta-Kernel (MK) file.

1.2 Scope

This is a multimission SIS, applicable for all flight projects.

1.3 Reference Documents

No.	**NAIF Document ID**	**Title**
1.	318	Kernel Required Reading

This reference documents is included in each delivery of the SPICE Toolkit.

Also available is a SPICE tutorial, named intro_to_kernels, available from the NAIF server: ftp://naif.jpl.nasa.gov/pub/naif/toolkit_docs/Tutorials/pdf/individual_docs/

1.4 Functional Description

A meta-kernel contains the names and path locations of one or more SPICE kernels that are to be loaded into an application program.

Because it is implemented using the SPICE text kernel standards, it could also contain additional information needed by an application program when that information may be provided in the text kernel's KEYWORD = VALUE(s) style.

1.4.1 Data Source, Destination and Transfer Method

Unlike for standard SPICE kernels, a meta-kernel is most often made by a SPICE end user and placed wherever is appropriate for that use.

On occasion a SPICE operations entity, such as the NAIF Group at JPL, will have need to produce and delivery a meta-kernel to support a SPICE-based application that is being provided to the project, such as the CHRONOS time conversion application. In such a case the MK will be placed in the appropriate place on a project server.

1.4.2 Labeling and Identification

There is no generic requirement for labeling (file naming) for an MK. However it is recommended that an MK use ".tm" as its file name extesnion.

An MK is a text kernel, and as such can be viewed by any available text display tool. If one displays or prints an MK, one will see the following identifier at the very top of the file: `KPL/MK.` These two sets of characters stand for: "Kernel Pool" and "Meta-kernel."

1.4.3 Assumptions and Constraints

Contents of an MK file must adhere to SPICE text kernel specifications, and special meta-kernel specifications, all as described in the SPICE technical reference document Kernel Required Reading (Reference 1).

Section 2
Data Object Definition

2.1 Structure and Organization

A meta-kernel is a simple ASCII file containing data sections and descriptive text sections. The contents and format follow the SPICE text kernel specifications described in the NAIF document Kernel Required Reading (Reference 1).

Text sections of an MK are used to describe the data. They are preceded by the token:

```
\begintext
```

If it appears first in the file, before any data, the first text section does not need this delimiter—it is interpreted as a text section by default.

All data sections start with the begin data delimiter,

```
\begindata
```

Data are provided using a "keyword = value" syntax. The data sections are parsed by SPICE kernel file readers and so must adhere to the format specified in the NAIF document Kernel Required Reading (Reference 1).

2.2 Data Format and Definition

2.2.1 Text Description

A text description about the MK, often called "comments," explaining its purpose and perhaps providing creator and creation date information, is provided in a text block initiated with the \begintext delimiter. There is no restriction on the number of lines of such descriptive text, nor on the number of these text blocks. However, each line of text within a text block should be limited to 80 characters or less.

2.2.2 Data Description

The data sections of an MK must conform to the text kernel specifications, including special, extra restrctions on meta-kernels, found in the SPICE document Kernel Required Reading (Reference 1).

2.2.2.1 Example of a Simple Meta-kernel

In its simplest form the data section of a meta-kernel contains a single assignment, consisting of the keyword KERNELS_TO_LOAD, with one or more kernel file names as the value(s). An example is shown below.

```
  \begindata

       KERNELS_TO_LOAD = ( 'leapseconds.ker',
                           'mgs.tsc',
                           'generic.bsp',
                           'mgs.bc',
                           'earth.bpc',
                           'mgs.bes'           )
```

A meta-kernel may contain multiple KERNELS_TO_LOAD assignments.

2.2.2.2 Use of PATH NAMES and PATH VALUES

It is sometimes necessary to qualify kernel names with their path names. To reduce both typing and the need to continue kernel names over multiple lines, meta-kernels allow users to define symbols for paths. This is done using two kernel variables:

```
       PATH_VALUES
       PATH_SYMBOLS
```

To create symbols for path names, one assigns an array of path names to the variable PATH_VALUES. Next, one assigns an array of corresponding symbol names to the variable PATH_SYMBOLS. The nth symbol in the second array represents the nth path name in the first array. Then you can prefix with path symbols the kernel names specified in the KERNELS_TO_LOAD variable. Each symbol is prefixed with a dollar sign to indicate that it is in fact a symbol.

Suppose in our example above the MGS kernels reside in the path:

/flight_projects/mgs/SPICE_kernels

and the other kernels reside in the path

/generic/SPICE_kernels

Then we can add paths to our meta-kernel as follows:

```
       \begindata
```

```
      PATH_VALUES  = ( '/flight_projects/mgs/SPICE_kernels',
                       '/generic/SPICE_kernels'              )

      PATH_SYMBOLS = ( 'MGS',
                       'GEN' )


      KERNELS_TO_LOAD = ( '$GEN/leapseconds.ker',
                          '$MGS/mgs.tsc',
                          '$GEN/generic.bsp',
                          '$MGS/mgs.bc',
                          '$GEN/earth.bpc',
                          '$MGS/mgs.bes'           )
```

It is not required that paths be abbreviated using path symbols; it's simply a convenience available to you.

Caution: the symbols defined using PATH_SYMBOLS are not related to the symbols supported by a host shell or any other operating system interface.

2.2.2.3 Example of an MK with Additional Kernel Information

The example meta-kernel shown below was prepared for the MSL project, in support of the NAIF-provided CHRONOS time conversion utility. This example begins with a brief textual description of the MK ("comments"), followed by a data block, another text block, and a final data block. The first data block provides the KERNELS_TO_LOAD functionality that is the prime purpose of an MK. But just after the KERNELS_TO_LOAD assignment one also sees several standard, text kernel assignments not related to loading kernels. This is a convenient way to provide additional, non-kernel-loading information to the kernel pool where it may be used by the user's application program. (Such information could, instead, have been provided in a separate text kernel made just for that purpose. In this example the extra assignments are used by NAIF's *chronos* time conversion program, configured for a surface vehicle.)

```
KPM/MK

CHRONOS setup file for MSL at ``Eberswalde Crater''

For landing time 2012-08-06T17:02:23 and SOLs indexed from 0.

By Boris Semenov, November 30, 2010.

\begindata

    PATH_VALUES       = (
                        '/proj/msl/ops/ods/cedl/strategic/naif/kernels'
                        )
    PATH_SYMBOLS      = (
                        'KERNELS'
                        )
    KERNELS_TO_LOAD   = (
                        '$KERNELS/lsk/naif0009.tls'
                        '$KERNELS/sclk/msl.tsc'
                        '$KERNELS/sclk/msl_lmst_ec120806_v1.tsc'
                        '$KERNELS/pck/pck00008.tpc'
                        '$KERNELS/spk/de418.bsp'
                        '$KERNELS/spk/msl_ls_ec120806_iau2000_v1.bsp'
                        '$KERNELS/spk/msl_atls_ec120806_v1.bsp'
                        '$KERNELS/fk/msl_tp_ec120806_iau2000_v1.tf'
                        )

    SPACECRAFT_ID    = -76
    CENTER_ID        = 499
    LANDING_TIME     = '2012-08-06T17:02:23'
    LANDING_SOL_INDEX = 0

\begintext

Sun GM from DE410.

\begindata

BODY10_GM  =   132712440035.0199

\begintext
```

Section 3
Using a Meta-Kernel

3.1 Loading Kernels Into a Program

All of the kernels identified in a meta-kernel KERNELS_TO_LOAD assignment may be loaded into a user's program with a single API call, shown below using Fortran, C, IDL and MATLAB syntax.

```
     CALL FURNSH ('<mk file name>')
     furnsh_c ('<mk file name>')
     cspice_furnsh, '<mk file name>'
     cspice_furnsh ('<mk file name>')
```