

**spi\_psd\_lib2moc**

# User Manual

Version 1.4.0  
23 November 2002

Jürgen Knödseder  
Centre d'Etude Spatiale des Rayonnements  
knodseder@cesr.fr  
<http://www.cesr.fr/~jurgen/index.html>

#### Note to the user

This software has been written to analyse data of the SPI telescope onboard INTEGRAL. Particular care has been taken in making the software user friendly and well documented. If you appreciated this effort, and if this software and User Manual were useful for your scientific work, the author would appreciate a corresponding acknowledgment in your published work.

## **Contents**

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Getting started</b>	<b>1</b>
<b>3</b>	<b>Parameter file</b>	<b>2</b>
<b>4</b>	<b>Interface definition</b>	<b>5</b>
<b>5</b>	<b>Error codes</b>	<b>6</b>

## 1 Introduction

The executable `spi_psd_lib2moc`, written in the ANSI C++ language, is an executable that transforms library templates given in the ISDC FITS format into a MOC command file for library upload. It also creates an EEPROM memory in the OBSMS format for verification of the library upload.

Also, a library template file and an algorithm parameter file are generated that conform to the content of the MOC command file (since one could select only part of the templates of the input template file it is only guaranteed in that way that an exact image of the uploaded library is also available at ISDC).

`spi_psd_lib2moc` has been developed under ISDC support platform 4.1 and requires `spi_psdlib` version 1.6.0 or higher and `spi_toolslib` version 1.8.0 or higher.

## 2 Getting started

Before installing `spi_psd_lib2moc`, make sure that the ISDC support platform 4.1 or higher is installed on your system, and that you have installed the libraries `spi_psdlib` version 1.6.0 or higher and `spi_toolslib` version 1.8.0 or higher.

After downloading the `spi_psd_lib2moc.tar.gz` file, step into a directory that should hold the distribution, move the `spi_psd_lib2moc.tar.gz` file into this directory and type:

```
$ gunzip spi_psd_lib2moc.tar.gz
$ tar xvf spi_psd_lib2moc.tar
```

The first command uncompresses the distribution file, the second unpacks the files.

Before configuration, the distribution needs to be reset to a clean state. To do this, type

```
$ make distclean
```

Then, configure the distribution. It is assumed here that you have previously installed the ISDC support platform, thus you should type

```
$ ~/bin/ac_stuff/configure
```

Finally, build the distribution by typing

```
$ make global_install
```

To perform a unit test, type

```
$ make test
```

Make sure that the test data `spi_test_data-1.0.tar.gz` are available at your site (they should reside in a directory whose name is defined by the `ISDC_TEST_DATA_DIR` environment variable).

### 3 Parameter file

```
#####
#
#           Centre d'Etude Spatiale des Rayonnements           #
#           (in collaboration with ISDC)                       #
#
#           SPI PSD ANALYSIS                                   #
#
# -----#
#
# File:      spi_psd_lib2moc.par                               #
# Version:   1.4.0                                           #
# Component: PA                                              #
#
# Author:    Juergen Knoedlseder                             #
#            knodlseder@cesr.fr                              #
#            CESR                                             #
#
# Purpose:   Parameter file of the SPI PSD library to MOC format #
#            conversion executable                             #
#
# History:   1.4.0 23-Nov-2002 First ISDC delivery (Rev. 4)  #
#
#####
#
# The DOLs of the new templates/algorithm parameters
#=====
inlibDOL,s,ql,"psdFM280701.fits[SPI.-LIB.-PSD]",,, "PSD template library DOL"
inalgDOL,s,ql,"psdFM280701.fits[SPI.-ALGO-PSD]",,, "PSD algorithm parameters DOL"
set,      i,ql, 0,0,1, "Load in template set number (0/1)?"
#
# The output DOLs
#=====
outlibDOL, s,ql,"psd.fits",,, "PSD result template library DOL or filename"
outalgDOL, s,ql,"psd.fits",,, "PSD result algorithm parameters DOL or filename"
#
# The previous uplinked DOLs
#=====
prevlibDOL, s,ql,"psdFM280701.fits[SPI.-LIB.-PSD]",,, "PSD previous template library DOL"
prevalgDOL, s,ql,"psdFM280701.fits[SPI.-ALGO-PSD]",,, "PSD previous algorithm parameters DOL"
#
# Result file parameters
#=====
devphase,  s,ql,"X",X|A|B|V,, "Developement phase (X,A,B,V)"
rel_level, i,ql,0,0,999,      "Release level (0-999)"
sub_level, i,ql,0,0,999,      "Sub-release level (0-999)"
int_level, i,ql,0,0,9,        "Internal release level (0-9)"
database,  s,ql,"27/02/01",,, "Database version"
#
# Task parameters
#=====
createlib, b,ql,yes,,, "Create library command blocks (and DOL) ?"
```

```

createpar, b,ql,yes,,, "Create algorithm parameter command blocks (and DOL) ?"
creategse, b,ql,yes,,, "Create GSE library binaries ?"
tarstk,    b,ql,yes,,, "Tar STK files ?"
zipstk,    b,ql,yes,,, "Compress STK files ?"
targse,    b,ql,yes,,, "Tar GSE files ?"
#
# Template usage
#=====
tpls0, s,hl,"6-28",,, "Detector 0 template TTPs"
tpls1, s,hl,"6-28",,, "Detector 1 template TTPs"
tpls2, s,hl,"6-28",,, "Detector 2 template TTPs"
tpls3, s,hl,"6-28",,, "Detector 3 template TTPs"
tpls4, s,hl,"6-28",,, "Detector 4 template TTPs"
tpls5, s,hl,"6-28",,, "Detector 5 template TTPs"
tpls6, s,hl,"6-28",,, "Detector 6 template TTPs"
tpls7, s,hl,"6-28",,, "Detector 7 template TTPs"
tpls8, s,hl,"6-28",,, "Detector 8 template TTPs"
tpls9, s,hl,"6-28",,, "Detector 9 template TTPs"
tpls10,s,hl,"6-28",,, "Detector 10 template TTPs"
tpls11,s,hl,"6-28",,, "Detector 11 template TTPs"
tpls12,s,hl,"6-28",,, "Detector 12 template TTPs"
tpls13,s,hl,"6-28",,, "Detector 13 template TTPs"
tpls14,s,hl,"6-28",,, "Detector 14 template TTPs"
tpls15,s,hl,"6-28",,, "Detector 15 template TTPs"
tpls16,s,hl,"6-28",,, "Detector 16 template TTPs"
tpls17,s,hl,"6-28",,, "Detector 17 template TTPs"
tpls18,s,hl,"6-28",,, "Detector 18 template TTPs"
#
# Standard parameters
#=====
clobber, b,h, no,    , , "Clobber Flag"
mode,    s,h, "ql",  , , "Mode"

```

The parameters have the following meaning:

- **inlibDOL** specifies the PSD library from which the template set specified by **set** will be extracted for library uploading.
- **inalgDOL** specifies the PSD algorithm parameters from which the set specified by **set** will be extracted for parameter uploading.
- **set** specifies which template set of the data structures **inlibDOL** and **inalgDOL** should replace the existing PSD library templates. Since the PSD holds two template sets, and since a coherent image of the PSD library status is needed for SPI data analysis, the complementary set (i.e. the set that should not be replaced, given by **1 - set**) is taken from the data structures specified by **prevlibDOL** and **prevalgDOL**, and stored together with the new templates and algorithm parameters in the result data structures specified by **outlibDOL** and **outalgDOL**.
- **outlibDOL** specifies the DOL or filename of the resulting PSD library that presents a coherent image of the PSD library status after the MOC command files produced by **spi\_psd\_lib2moc** have been executed correctly. The HDU is not required by this parameter, but its specification does no harm to the executable.
- **outalgDOL** specifies the DOL or filename of the resulting PSD algorithm parameters that present a coherent image of the PSD algorithm parameter status after the MOC command files produced

by `spi_psd_lib2moc` have been executed correctly. The HDU is not required by this parameter, but its specification does no harm to the executable. **It is recommended to store the algorithm parameters in the same file as the library templates to be sure that associated templates and algorithm parameters are always indeed associated.** Thus as default, the same filename as given by `outlibDOL` should be specified.

- `prevlibDOL` specifies the DOL of the previous PSD library that has been uplinked to the PSD. The complementary template set is extracted from this DOL, and will be stored in the result library, specified by `outlibDOL`. Note, that the specification of the previous library is optional. If no DOL is specified, no complementary set will be added to the result library.
- `prevalgDOL` specifies the DOL of the previous PSD algorithm parameters that have been uplinked to the PSD. The complementary algorithm parameters are extracted from this DOL, and will be stored in the result algorithm parameters, specified by `outalgDOL`. Note, that the specification of the previous algorithm parameters is optional. If no DOL is specified, no complementary set will be added to the result algorithm parameters.
- `devphase` specifies for which development phase the MOC command files should be produced.
- `rel_level` specifies the release level of the MOC command files.
- `sub_level` specifies the sub-release level of the MOC command files.
- `int_level` specifies the internal release level of the MOC command files.
- `database` specifies the database version for the MOC command files.
- `createlib` specifies if MOC commands for library template uplink should be added to the MOC command file. The default option is **yes**. If, however, only the algorithm parameters should be modified, one may specify **no** to suppress library uplink commands.
- `createpar` specifies if MOC commands for algorithm parameter uplink should be added to the MOC command file. The default option is **yes**. If, however, only the library templates should be changed, one may specify **no** to suppress algorithm parameter uplink commands.
- `creategse` specifies if PSD GSE library uplink binaries should be generated. These binaries are needed to verify library uplink either on the PSD Laboratory Model (LM) or the Electrical Model (EM).
- `tarstk` specifies if the MOC command files (STK files) should be tared as specified in the MOC ICD.
- `zipstk` specifies if the MOC command files (STK files) should be compressed as specified in the MOC ICD.
- `targse` specifies if the GSE binary files should be tared (otherwise `spi_psd_lib2moc` may produce several 100 files!)
- `tplsn` specifies for each of the 19 PSD channels ( $n = 0 - 18$ ) which library templates should be used for uplink. Only these templates will be extracted from the input library and stored in the result library (see below for a deeper discussion of the field syntax).
- `clobber` ISDC standard parameter.
- `mode` ISDC standard parameter.

The detector template fields specify the templates for each detector that should be converted from the input library DOL into a MOC command file. There is one such field per detector. The template list may be a list of comma-separated template identifiers, or a range of templates. For example, `tpls0,s,h1,"0-18"` adds the templates 0 to 18 from the input library DOL to the MOC command file for detector 0. The same is achieved by `tpls0,s,h1,"0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18"`. If a range of templates is specified, the start and/or the end template identifier may be omitted. If the start template identifier is omitted, it is assumed that the range starts from template 0. If the end template identifier is omitted, the range is assumed to end with 79. If both are omitted, the entire template range is selected (0-79). Hence, the four commands `0-79`, `-79`, `0-`, `-` are identical and specify all 80 templates from the library input DOL (note that there are not necessarily 80 templates in this file, but 80 is just the maximum number one could imagine).

Comma separated template identifiers have the special functionality of toggling between templates. For example, `3,3` means that no template is selected. The first `3` signals to use template 3, while the second `3` deletes template 3 from the list of templates that should be used. The above example is, of course, not very meaningful. However, `-,3` makes much more sense, since now all templates except of template 3 will be used (the `-` selects all templates while the `3` deletes template 3 from the list).

If the template field is left empty, nothing is done for the corresponding detector (i.e. no templates will be added to the MOC command file). Blanks may be added deliberately to the template field.

## 4 Interface definition

On input, `spi_psd_lib2moc` requires a library template DOL of type `SPI.-LIB.-PSD` and a PSD algorithm parameter DOL of type `SPI.-ALGO-PSD`. No specific keyword is required in these data structures.

On output, `spi_psd_lib2moc` produces

- an ASCII MOC command file (for library upload)
- an ASCII MOC memory dump file (for EEPROM dump comparison)
- a library template DOL containing all templates that have been selected for the MOC command file (for keeping consistent information at ISDC and MOC)
- a PSD algorithm parameter DOL containing the same information that has been added to the MOC command file (again for keeping consistent information at ISDC and MOC)

Note that adding library templates and PSD algorithm parameter information to the MOC command file is optionally. If `createlib` is set to `no`, no library templates are added to the MOC command file. However, input library templates are processed as usual and the resulting PSD algorithm parameter information is still valid. Hence one can use `spi_psd_lib2moc` to only produce the PSD algorithm parameter information. In this case no library template output DOL will be created.

If `createpar` is set to `no`, no PSD algorithm parameter information will be added to the MOC command file. This does not affect the processing of the input library templates.

## 5 Error codes

The following error codes are defined:

SPI_PSD_LIB2MOC_ERROR_BASE	-230700 // Error base
SPI_PSD_LIB2MOC_ERROR_MEM_ALLOC	-230700 // Memory allocation failure
SPI_PSD_LIB2MOC_ERROR_FILE_OPEN	-230701 // Result file opening error
SPI_PSD_LIB2MOC_ERROR_TAR	-230702 // UNIX "tar" error
SPI_PSD_LIB2MOC_ERROR_RM	-230703 // UNIX "rm" error
SPI_PSD_LIB2MOC_ERROR_COMPRESS	-230704 // UNIX "compress" error
SPI_PSD_LIB2MOC_ERROR_MEM_OVERFLOW	-230705 // Internal memory overflow