

spi_tmscan

User Manual

Version 1.3.1
7 August 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

1	Introduction	1
2	Getting started	1
3	Parameter file	3
4	Interface definition	6
5	Algorithm	6
6	Error codes	7

1 Introduction

The executable `spi_tmscan` scans the INTEGRAL/SPI telemetry in the format of the CESR GSE acquisition system (`PktAll`) and launches the data processing queue that converts the data into ISDC compliant FITS files. In particular, `spi_tmscan` implements the cut of the data stream in so called science windows.

The algorithm that has been implemented follows closely the ISDC tech note ISDC/TEC018 Version 2.1 (8 March 2002) with some small deviations that are explained below.

`spi_tmscan` has been written in the ANSI C++ language, and requires the following software to be installed on the system:

<code>support-sw</code>	4.1	
<code>spi_toolslib</code>	1.7.0	
<code>spi_psdlib</code>	1.4.0	
<code>spi_tm2fits</code>	1.5.1	
<code>tools_lib</code>	2.2.4	(<code>sys-sw</code>)
<code>spi_merge_schk</code>	4.1	(<code>sys-sw</code>)
<code>dp_spi_cobt</code>	1.5	
<code>swg_create</code>	1.2	(<code>sys-sw</code>)
<code>swg_clean</code>	1.4	(<code>sys-sw</code>)
<code>spi_evts_obt_calc</code>	1.4.1	(<code>sys-sw</code>)
<code>dp_hkc</code>	1.8	(<code>sys-sw</code>)
<code>dp_spi_psd</code>	1.2.0	
<code>dp_status_gen</code>	2.5.1	(<code>sys-sw</code>)
<code>dp_aux_attr</code>	1.6.1	(<code>sys-sw</code>)
<code>spi_psd_adcgain</code>	1.3.0	
<code>spi_psd_efficiency</code>	1.3.0	
<code>spi_psd_performance</code>	1.4.0	
<code>spi_psd_si</code>	1.3.0	

Since the upwards compatibility of the software should be assured, higher software version numbers should also be possible. Software models flagged by (`sys-sw`) are part of the ISDC system software and can be obtained at the ISDC web site (<http://isdc.unige.ch>). The above mentioned versions are also available as bundle at the author's web site <http://www.cesr.fr/~jürgen/isdc>, where also all other software can be found.

2 Getting started

Before installing `spi_tmscan`, make sure that the ISDC support platform 4.1 or higher is installed on your system, and that the libraries `spi_psdlib` and `spi_toolslib` are also available (please consult the WWW site <http://www.cesr.fr/~jürgen/isdc> if these libraries are not available on your system).

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

```
$ gunzip spi_tmscan.tar.gz
$ tar xvf spi_tmscan.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
```

3 Parameter file

```
#####
#
#           Centre d'Etude Spatiale des Rayonnements           #
#           (in collaboration with ISDC)                       #
#
#           PRE PROCESSING                                     #
#
#
#
# File:      spi_tmscan.par                                   #
# Version:   1.3.0                                           #
# Component: pp                                              #
#
# Authors:   Juergen Knoedlseder                             #
#            knodlseder@cesr.fr                              #
#            CESR                                             #
#
# Purpose:   Parameter file of the SPI TM scanner           #
#
# Revision History:                                         #
#
# Version 1.3.0: 18-Jul-2002 JK First ISDC version (rev. 3) #
#
#####
#
# Path name definition
#=====
tmpath,  s, h,           "/x02/tm",,, "TM Base Path"
base,    s, h,           "/villa-1/jurgen/arc/tst_tm",,, "Repository Base Path"
cfgbase, s, h,           "/users/jurgen/isdc/templates",,, "Template Base Path"
#
# TM input definition
#=====
nfirst,  i, ql,          419,, "First TM file number"
nlast,   i, ql,          419,, "Last TM file number"
pktmin,  i, ql,          26048,1,10000000, "First packet to scan in first TM file"
pktmax,  i, ql,          61927,1,10000000, "Last packet to scan in last TM file"
header,  i, h,           0,, "Header size (in Bytes)"
timechk, i, ql,          60,, "TM scanning cycle (sec)"
timeout, i, ql,          300,, "TM scanning timeout (sec)"
#
# First SCW definition
#=====
previd,  s, hl,          "000000000000",,, "Previous SCW ID"
revno,   i, hl,          9001,, "Revolution Number"
pointid, i, hl,          0000,, "Pointing ID (minimum is 0000)"
subid,   i, hl,          001,, "Subdivision ID (minimum is 001)"
scwtype, i, hl,          2,0|1|2,, "Science Window Type (0=pointing, 1=slew, 2=other)"
version, i, hl,          1,, "Version number"
bcpid,   s, hl,          "00000000",,, "Broadcast packet ID"
```

```

#
# SCW cut definition
#=====
cutOPER,    b, hl, yes,,,    "New SCW if OPER mode change"
cutCALB,    b, hl, yes,,,    "New SCW if CALIB mode change"
cutDIAG,    b, hl, yes,,,    "New SCW if DIAG mode change"
cutEMER,    b, hl, yes,,,    "New SCW if EMER mode change"
cutSTB1,    b, hl, no,,,     "New SCW if STANDBY1 mode change"
cutSTB2,    b, hl, no,,,     "New SCW if STANDBY2 mode change"
cutPoint,   b, hl, yes,,,    "New SCW if pointing change"
cutOTF,     b, hl, no,,,     "Consider OTF for pointing change"
cutPerigee, b, hl, yes,,,    "New SCW at perigee passage"
cutTime,    b, hl, yes,,,    "New SCW if time exceeded"
timescw,    i, hl, 3600,0,1000000, "Maximum SCW length (sec)"
maxgap,     i, hl, 1800,0,32000, "Maximum time gap (sec)"
notf,       i, hl, 3,0,10000, "Requested number of stable OTF flags"
#
# Automatic processing parameters
#=====
autolaunch, b, hl, no,,,     "Automatically launch processing"
shell,      s, hl, "/bin/csh",,, "Processing shell"
script,     s, hl, "spi_tmscan_process.C",,, "Processing script"
#
#
# Standard parameters
#=====
clobber, b, h, no,,, "Clobber Flag"
mode,    s, h, "ql",,, ""

```

The parameters have the following meanings:

- **tmpath** specified the directory which holds the **PktAll** telemetry files in the CESR GSE format (in this format, telemetry frames are stored as successive blocks of 440 Bytes in a single binary file).
- **scwbase** specifies the Science window base path into which the science window data structures should be written.
- **cfgbase** specifies the absolute directory in which the science window group configuration files will be found. Normally, the directory where the data structure templates are found should be specified.
- **nfirst** specifies the file number of the first **PktAll** telemetry file that should be processed.
- **nlast** specifies the file number of the last **PktAll** telemetry file that should be processed.
- **pktmin** specifies the first packet in the first telemetry binary file (specified by the parameter **nfirst**) that should be processed (the first packet has the packet number 1). A packet is defined as a 440 Bytes block in the telemetry binary file. If **pktmin** = 1, pre-processing starts from the beginning of the telemetry file.
- **pktmax** specifies the last packet in the last telemetry binary file (specified by the parameter **nlast**) that should be processed (packets start from number 1). A packet is defined as a 440 Bytes block in the telemetry binary file. If **pktmax** is larger than the number of available packets, the telemetry file is analysed until the end of the file is reached.
- **header** specifies the number of Bytes in the telemetry binary file header. Earlier versions of the **PktAll** files had a header attached. The actual version do not have a header anymore, hence as default, 0 should be specified.

- **timechk** specifies the TM scanning cycle in seconds. If the end of the telemetry file has been reached, **spi_tmscan** is put into sleep for the specified period in order to liberate processor capacities. After the period, **spi_tmscan** wakes up and checks if new TM has been received. If yes, the new TM is scanned. Otherwise, **spi_tmscan** is put into sleep for another cycle.
- **timeout** specifies the TM scanning timeout in seconds after which **spi_tmscan** stops execution if no new TM has been received.
- **previd** specifies the science window identifier of the last science window that has been processed before starting **spi_tmscan**. The information is used to establish a continuous link of telemetry. If no previous science window exists, this parameter may be set to **000000000000**.
- **revno** specifies the revolution number of the first science window that will be created by **spi_tmscan**.
- **pointid** specifies the pointing identifier of the first science window that will be created by **spi_tmscan**. The minimum pointing identifier is **0000**.
- **subid** specifies the subdivision identifier of the first science window that will be created by **spi_tmscan**. The minimum subdivision identifier is **001**.
- **scwtype** specifies the type of the first science window that will be created by **spi_tmscan**. The following types are allowed: **0** is a pointing, **1** is a slew, and **2** is any other science window type.
- **version** specifies the science window version number. The version number will not be changed by **spi_tmscan**.
- **bcpid** specifies the Broadcast packet identifier for the telemetry, to be stored in the headers of the RAW data structures.
- **cutOPER** specifies if a new science window should be created if a transition from or to the **OPER** mode occurs.
- **cutCALB** specifies if a new science window should be created if a transition from or to the **CALIB** mode occurs.
- **cutDIAG** specifies if a new science window should be created if a transition from or to the **DIAG** mode occurs.
- **cutEMER** specifies if a new science window should be created if a transition from or to the **EMER** mode occurs.
- **cutSTB1** specifies if a new science window should be created if a transition from or to the **STANDBY1** mode occurs.
- **cutSTB2** specifies if a new science window should be created if a transition from or to the **STANDBY2** mode occurs.
- **cutPoint** specifies if a new science window should be created if the pointing identifier changes.
- **cutOTF** specifies if a new science window should be created if the On Target Flag (OTF) makes a verifiable change. A verifiable change means that the OTF must be stable for at least **notf** consecutive readouts of the OTF value from the CDMU packet. Note that **cutOTF** is only considered if **cutPoint** is **yes**.
- **cutPerigee** specifies if a new science window should be created if the revolution number changes.
- **cutTime** specifies if a new science window should be created if the time period specified by **timescw** has been exceeded. This allows to cut fixed pointing observations into smaller pieces.

- **timescw** specifies the time interval for science window cutting using the timeout mechanism (**cutTime = yes**).
- **notf** specifies the required number of stable OTF flag readings for a science window cut to occur. If the minimum value of 0 is specified, a single OTF change will lead to a science window cut.
- **autolaunch** specifies if **spi_tmscan** should automatically launch the processing of the telemetry using the processing script specified by the parameter **script**.
- **shell** specifies the UNIX shell that should be used for the processing script. The user has to make sure that the shell initialisation script sets the ISDC environment variables correctly.
- **script** specifies the ISDCROOT script that should be used for telemetry processing. A standard script comes with the software distribution.
- **clobber** ISDC standard parameter (see Common User Manual).
- **mode** ISDC standard parameter (see Common User Manual).

4 Interface definition

The task **spi_tmscan** does not produce any data structures. The telemetry file in the CESR GSE format (**PktA11**) are scanned and some relevant information (such as On-Event Messages and telemetry gaps) are recorded into a LOG file. If requested, **spi_tmscan** starts a processing pipeline that converts the raw telemetry data into ISDC compliant FITS data structures.

5 Algorithm

The science window ID has always the following form:

RRRRPPPPSSSF

where

- **RRRR** is the revolution number of the spacecraft as defined from the perigee to perigee passage,
- **PPPP** is the pointing number within the revolution and is always reset to 0000 when the revolution number increments,
- **SSS** is the subdivision number for a science window with a given **RRRR** and **PPPP** value. The minimum value of **SSS** is 001 and always resets with a change of value of **PPPP**,
- **F** is the type identifier with the allowed values of 0 for a pointing, 1 for a slew, and 2 for other science window types.

The following table summarises the science window ID changes that occurs after a given boundary has been reached. Science window boundaries that are implemented in **spi_tmscan** are

- SPI mode transitions
- SPI reset
- timeout
- slew

- pointing
- perigee passage

Boundary	RRRR	PPPP	SSS	F	Comments
Mode transition	-	+1	1	-	
SPI reset	-	+1	1	-	may occur if data loss
Timeout	-	-	+1	-	
Slew	-	-	+1	1	
Pointing	-	+1*	1	0	
Perigee	+1	0	1	2	

*if the current revolution number in the CDMU packet corresponds to RRRR, the pointing identifier PPPP is extracted from the CDMU packet if the CDMU packet PID is larger than the actual PPPP (see ISDC/TECH018).

6 Error codes

The following error codes may be returned by `spi_tmscan`:

```

SPI_TM2FITS_ERROR_BASE           -10000 // Error base
SPI_TM2FITS_ERROR_MEM_ALLOC      -10000 // Memory allocation error
SPI_TM2FITS_ERROR_EOF           -10001 // End of file reached
SPI_TM2FITS_ERROR_FILE_ERROR     -10002 // TM file error
SPI_TM2FITS_ERROR_INVALID_BUFFER -10003 // Invalid buffer

```