KAGUYA (SELENE)

Product Format Description

- Plasma energy Angle and Composition Experiment(PACE)-

Version 1.0

November 1, 2009

Index

1. Introduction

1.1 Purpose

This document describes the format*2 used for the catalog and product files for the Plasma energy Angle and Composition Experiment *1(PACE) that was board KAGUYA (SELENE). These files provided by Japan Aerospace Exploration Agency (JAXA).

*1 : Refer to the following "Project Homepage of KAGUYA" and "Image Gallery of KAGUYA" used for the PACE mission.

- ✓ Project Homepage for KAGUYA
  http://www.kaguya.jaxa.jp/en/equipment/pace_e.htm
- ✓ Image Gallery for KAGUYA
  http://wms.selene.darts.isas.jaxa.jp/selene_viewer/en/observation_mission/pace/

*2 : The data format used for SELENE is based on the PDS (Planetary Data System) by NASA. However, the data format is not fully compliant with the PDS format.

1.2 The composition of this format description

Table 1-1 shows the composition of this format description.

Table 1-1 the composition of this format description

No.	INDEX	Title	Description content
1	Section 1.3	Table 1-2 PACE Products List	The name of the product, the object form, and the composition of the product are described as a product list illustrated by this description.
		Table 1-3 Product Description	Concerning each product shown in the No1 product list, the content included in data and the description of the observation method are illustrated.
2	Chapter X	" Product Name"	Concerning the product shown in the No1 product list, rules used for file naming, label format, data object format and catalog information file format are described.
3	Section X.1	Rules used for File naming	Concerning the product shown in No2, the rules of file nomenclature is described.
4	Section X.2	Label Format	Concerning the product shown in No2, the label format is described.
5	Section X.3	Data Object Format	Concerning the product shown in No2, the data format of the data object is described. (The extension of the data file is unique in each product. Therefore, refer to the file nomenclature in No3.)
6	Section X.4	Catalog Information File Format	Concerning the product shown in No2, the format of the catalog information file (extension: .ctg) of the product is described.
7	Chapter X+1		
		Same as above	

1.3 Data Set

The Data Set refers to a set consisting of: Product, Catalog Information, and Thumbnail Image (JPEG format), which are tar-archived. This set is referred to as the "L2 Data Set". The file extension is "SL2". However, the thumbnail image may be omitted at the by composer's judgment.

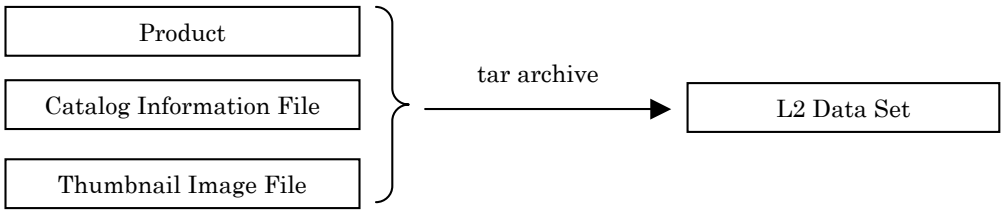

Figure 1-1 composition of the L2 Data Set

1.3.1 Product

For product composition, two possible options are available. Product Composition – Attached consists of label information and data information in a single data file. Product Composition – detached consists of separate files for the label file and data file.

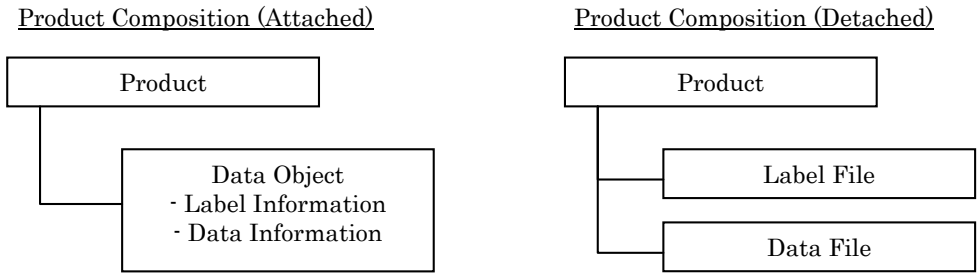

Figure 1-2 Product Composition : Attached and Detached

(1) Label File (Data Object (Label Information)

The Label File (Label Information) is storing as text format the information that identifies the Data File (Data Information).

(2) Data File (Data Object (Data Information))

The data File (Data Object (Data Information)) of the product are classified into the following four data types.

a) IMAGE : image data

An IMAGE is a two-dimensional array of values, all of the same type, each of which is referred to as a sample. IMAGE are normally processed with special display tools to produce a visual representation of the samples by assigning

brightness levels or display colors to the values. An IMAGE consists of a series of lines, each containing the same number of samples.
*Refer to the PDS Standard Reference V3.8 Appendix A.20 "IMAGE".

b) TABLE : tabular form data
TABLEs are a natural storage format for collections of data from many instruments. The TABLE is a uniform collection of rows containing ASCII or binary values stored in columns.
*Refer to the PDS Standard Reference V3.8 Appendix A.29 "TABLE".

c) SERIES : time series data
The SERIES is a sub-class of the TABLE. It is used for storing a sequence of measurements organized in a specific way. The sampling parameter keywords in the SERIES represent the variation between the ROWS of data.
*Refer to the PDS Standard Reference V3.8 Appendix A.24 "SERIES"

d) TEXT : text data
The TEXT describes a file which contains plain text.
*Refer to the PDS Standard Reference V3.8 Appendix A.30 "TEXT".

1.3.2 Catalog Information File

Catalog Information File is the information file attached to explain the general of the product and is used to search for the product from L2DB subsystem.

1.3.3 Thumbnail Image File

Thumbnail Image File is the reduced image of the data object, and is the JPEG format image. However, the thumbnail image may be omitted at the by composer's judgment.

1.4 PACE Products

The list of PACE products, which this document describes, is shown in Table 1-2. In addition, the description for each product is shown in Table 1-3.

Table 1-2 PACE Products List

Level *1	Product Name	Product ID	Data Type	Product Format*2
Higher Level	Magnetic anomaly map (Electron Reflectometer)	PACE_ERMA_MAP	IMAGE	A
Higher Level	Reflected Ion Map	PACE_SI_MAP	IMAGE	A
Standard	High Resoution Data of Electron/Ion Energy Spectrum (PBF1)	PACE_PBF_1	SERIES	D
Standard	High Resoution Data of Electron/Ion Energy Spectrum (CDF)	PACE_CDF	-*3	-*3
Standard	Summary Plot of Electron/Ion E-T Diagram	PACE_ET_summary	IMAGE	A

:Map products

*1 : Data obtained by equipments is not clear as it is, therefore various processings and correction treatment are necessary by the ground system. According to the difference in the process of processing and correction treatment, they can be classified to the standard processing and higher-level processing. The higher-level processing refers to the standard processing data to which various processing and correction treatment are conducted according to the research purpose et cetra

*2 Product Format : A - Attached, D - Detached

*3 : CDF Format. This completely conforms to CDF Ver3.1.

Table 1-3 Product Description

Product Name	Descriptions
Magnetic anomaly map (Electron Reflectometer)	two dimensional magnetic anomaly map obtained by using PACE-ESA and LMAG sensors as an electron reflectometer
Reflected Ion Map	two dimensional map of reflected ion from moon surface measuerd by PACE
High Resoution Data of Electron/Ion Energy Spectrum (PBF1)	high time resolution of electron & ion energy spectrum measured by PACE
High Resoution Data of Electron/Ion Energy Spectrum (CDF)	high time resolution of electron & ion energy spectrum measured by PACE
Summary Plot of Electron/Ion E-T Diagram	Summary plot of electron & ion energy spectrum measured by PACE

:Map products

(1) CDF

The PACE_CDF product is generated in CDF *1(Common Data Format) format. The catalog information is included in the CDF File.

Therefore, as a L2 product, the data product file and the catalog information file, in the CDF format, are included. The label is not defined. This completely conforms to CDF Ver3.1.

*1 : http://cdf.gsfc.nasa.gov/

2. Magnetic anomaly map (Electron Reflectometer) (Product ID:PACR_ERMA_MAP)

2.1 Rules used for File naming

The nomenclature used for Label, Data Object and Catalog Information File the product files are described below. In addition, the file names are case-independent.

PACE_AAAA_MAP_V###.dat

- AAAA : Product ID
  - ✓ ERMA : Magnetic anomaly map (Electron Reflectometer)
- ### : Version Number (three digits)
- dat : File Extension
  - ✓ img : Data File
  - ✓ ctg : Catalog Information File
  - ✓ sl2 : L2 Data Set (tar archive)

2.2 Label Format

The Label format for the IMAGE object used for the Magnetic anomaly map (Electron Reflectometer) product is shown in Table 2-1. The Label for the IMAGE object includes: Standard Item, Image Data Object Format Description Part and IMAGE_MAP_PROJECTION Object Description Part.

In Table 2-1 with the exception of the Values expressed as "STATIC", the numerical values and the character strings corresponding to the type of the product etc., are set.

Table 2-1 IMAGE Object Label

No	Items	Elements	Types	Values
Standard Item				
1	PDS version number	PDS_VERSION_ID = %s	char	PDS3 [STATIC]
2	Record format of the file	RECORD_TYPE = %s	char	UNDEFINED [STATIC]
3	File name	FILE_NAME = %s	char	See Section2.1 "Rules used for File naming"
4	Starting position of the image object	^IMAGE = %d <BYTES>	int	XXXX <BYTES>
5	Byte count of the file records	RECORD_BYTES = %d	int	XXXXXX
6	Name of the mission	MISSION_NAME = %s	char	SELENE [STATIC]
7	Name of the spacecraft	SPACECRAFT_NAME = %s	char	SELENE-M[STATIC]
8	Name of the instrument (Full name)	INSTRUMENT_NAME = %s	char	PACE [STATIC]
9	Product ID	PRODUCT_SET_ID = %s	char	See Table1-2 "Product_ID
10	Target name	TARGET_NAME = %s	char	MOON [STATIC]
11	Commen	COMMENT_TEXT = "%s"	char	
Image Data Object Format Description Part(/* IMAGE */)				
		OBJECT = IMAGE		
12	Band storage type	BAND_STORAGE_TYPE = %s	char	SAMPLE_INTERLEAVED [STATIC] *Refer to the PDS Standard Reference V3.5 Appendix A.19 "IMAGE"
13	Number of bands	BANDS = %d	smallint	X

14	Compression class and encoding type	ENCODING_TYPE = %s	char	N/A [STATIC]
15	Horizontal pixel count of image	LINE_SAMPLES = %d	int	XXX
16	Vertical pixel count of image	LINES = %d	int	XXX
17	Pixel bit length	SAMPLE_BITS = %d	int	X
18	Pixel type	SAMPLE_TYPE = %s	char	MSB_INTEGE R[STATIC] * Refer to the PDS Standard Reference V3.5 Appendix C.1 for further information about "MSB_INTEGER".
19	Stretched Flag	STRETCHED_FLAG = %s	char	FALSE[STATIC]
20	Alternative value outside assumption	INVALID_CONSTANT = %s	char	X
21	Offset	OFFSET = %f	float	X.X
22	Scaling factor	SCALING_FACTOR = %f	float	X.X
		END_OBJECT = IMAGE		
IMAGE_MAP_PROJECTION Object Description Part(/* IMAGE_MAP_PROJECTION */)				
		OBJECT = IMAGE_MAP_PROJECTION		
23	Resolution	MAP_RESOLUTION = %f<PIXEL/DEGREE>	float	X < PIXEL/DEGREE>
24	Semi-major axis of the ellipsoidal body	A_AXIS_RADIUS = %f<KM>	float	1737.400 <km> [STATIC]
25	medial axis of ellipsoidal body	B_AXIS_RADIUS = %f<KM>	float	1737.400 <km> [STATIC]
26	Semi-minor axis of ellipsoidal body	C_AXIS_RADIUS = %f<KM>	float	1737.400 <km> [STATIC]
27	Name of coordinate system	COORDINATE_SYSTEM_NAME = "%s"	char	"PLANETOCENTRIC "[STATIC]
28	Type of coordinate system	COORDINATE_SYSTEM_TYPE = "%s"	char	"BODY-FIXED ROTATING" [STATIC]
29	Maximum latitude	MAXIMUM_LATITUDE = %f	float	89.0 [STATIC]
30	Minimum latitude	MINIMUM_LATITUDE = %f	float	-89.0 [STATIC]
31	Westernmost longitude	WESTERNMOST_LONGITUDE = %f	float	0.0 [STATIC]
32	Easternmost longitude	EASTERNMOST_LONGITUDE = %f	float	359.0 [STATIC]
33	Type of map projection	MAP_PROJECTION_TYPE = "%s"	char	SIMPLE_CYLINDRICAL [STATIC]
34	Direction of positive longitude	POSITIVE_LONGITUDE_DIRECTION = "%s"	char	EAST [STATIC]
		END_OBJECT = IMAGE_MAP_PROJECTION		
END statement				
		END		

<Example of Label : Magnetic anomaly map (Electron Reflectometer)>

```
PDS_VERSION_ID                  = PDS3
RECORD_TYPE                     = UNDEFINED
FILE_NAME                       = PACE_SI_MAP_001.img
^IMAGE                          = 1082 <BYTES>
RECORD_BYTES                    = 579960
SPACECRAFT_NAME                 = SELENE-M
INSTRUMENT_NAME                 = PACE
PRODUCT_SET_ID                  = PACE_SI_MAP
TARGET_NAME                     = MOON
COMMENT_TEXT                    = "Magnetic field anomaly map.   9 bands."

/* IMAGE */
OBJECT                  = IMAGE
   BAND_STORAGE_TYPE            = SAMPLE_INTERLEAVED
   BANDS                        = 9
   ENCODING_TYPE                = N/A
   LINE_SAMPLES                 = 360
   LINES                        = 179
   SAMPLE_BITS                  = 8
```

```
   SAMPLE_TYPE                    = MSB_INTEGER
   STRETCHED_FLAG                 = FALSE
   INVALID_CONSTANT               = 0
   OFFSET                         = 0.0
   SCALING_FACTORE                = 0.5
END_OBJECT              = IMAGE

/* IMAGE_MAP_PROJECTION */
OBJECT                  = IMAGE_MAP_PROJECTION
   MAP_RESOLUTION                 = 1 < PIXEL / DEGREE>
   A_AXIS_RADIUS                  = 1737.400 <km>
   B_AXIS_RADIUS                  = 1737.400 <km>
   C_AXIS_RADIUS                  = 1737.400 <km>
   COORDINATE_SYSTEM_NAME         = "PLANETOCENTRIC"
   COORDINATE_SYSTEM_TYPE         = "BODY-FIXED ROTATING"
   MAXIMUM_LATITUDE               = 89.0
   MINIMUM_LATITUDE               = -89.0
   WESTERNMOST_LONGITUDE          = 0.0
   EASTERNMOST_LONGITUDE          = 359.0
   MAP_PROJECTION_TYPE            ="SIMPLE_CYLINDRICAL"
   POSITIVE_LONGITUDE_DIRECTION          =EAST
END_OBJECT              = IMAGE_MAP_PROJECTION
END
```

2.3 Data Object Format

T.B.D

2.4 Catalog Information File Format

The Catalog Information File Format is shown in Table 2-2.

Table 2-2 Catalog Information File Format

Item Name	Elements	Format of Value	Range of Value	Values
Name of the data file (*1)	DataFileName	AAAA….AAAA (Up to 31 digits)	alphanumeric characters	dependent on the product (See Section 2.1 "Rules used for File naming".)
Size of the data file	DataFileSize	NNNNNNNNNNNN (Up to 12 digits)	unit:<byte>	dependent on the product
File format of the data file	DataFileFormat	AAAA….AAAA (Up to 16 digits)	character strings	PDS[STATIC]
Name of the instrument	InstrumentName	AAAA….AAAA (Up to 16 digits)	character strings	PACE [STATIC]
Processing level	ProcessingLevel	AAAA….AAAA (Up to 16 digits)	character strings	dependent on the product (See Table 1-2 "Level")
Product ID	ProductID	AAAA….AAAA (Up to 30 digits)	character strings	dependent on the product (See Table 1-2 "Product_ID")
Version number of the product	ProductVersion	AAAA….AAAA (Up to 16 digits)	character strings	dependent on the product
Access level	AccessLevel	N	the value of 0-4	N/A

(*1) "DataFileName" is the stored file name of the product. For the detached format, this is the stored file name.

<Example of Catalog Information : Magnetic anomaly map (Electron Reflectometer)>

```
DataFileName = PACE_SI_MAP_001.img
DataFileSize = 581041
```

DataFileFormat = PDS
InstrumentName = PACE
ProcessingLevel = Higher Level
ProductID = PACE_SI_MAP
ProductVersion = 1.0
  AccessLevel = 4

3. Reflected Ion Map (Product ID:PACE_SI_MAP)

3.1 Rules used for File naming

The nomenclature used for Label, Data Object and Catalog Information File the PACE product files are described below. In addition, the file names are case-independent.

PACE_AAAA_MAP_V###.dat

- AAAA : Product ID
  - ✓ SI : Reflected Ion Map
- ### : Version Number (three digits)
- dat : File Extension
  - ✓ img : Data File
  - ✓ ctg : Catalog Information File
  - ✓ sl2 : L2 Data Set (tar archive)

3.2 Label Format

The Label format for the IMAGE object used for the Reflected Ion Map product is shown in Table 3-1. The Label for the IMAGE object includes: Standard Item, Image Data Object Format Description Part and IMAGE_MAP_PROJECTION Object Description Part.

In Table 3-1 with the exception of the Values expressed as “STATIC”, the numerical values and the character strings corresponding to the type of the product etc., are set.

Table 3-1 IMAGE Object Label

No	Items	Elements	Types	Values
Standard Item				
1	PDS version number	PDS_VERSION_ID = %s	char	PDS3 [STATIC]
2	Record format of the file	RECORD_TYPE = %s	char	UNDEFINED [STATIC]
3	File name	FILE_NAME = %s	char	See Section 3.1 “Rules used for File naming”
4	Starting position of the image object	^IMAGE = %d <BYTES>	int	XXXX <BYTES>
5	Byte count of the file records	RECORD_BYTES = %d	int	XXXXXX
6	Name of the mission	MISSION_NAME = %s	char	SELENE [STATIC]
7	Name of the spacecraft	SPACECRAFT_NAME = %s	char	SELENE-M[STATIC]
8	Name of the instrument (Full name)	INSTRUMENT_NAME = %s	char	PACE [STATIC]
9	Product ID	PRODUCT_SET_ID = %s	char	See Table1-2 “Product_ID”
10	Target name	TARGET_NAME = %s	char	MOON [STATIC]
11	Comment	COMMENT_TEXT = "%s"	char	
Image Data Object Format Description Part(/* IMAGE */)				
		OBJECT = IMAGE		
12	Band storage type	BAND_STORAGE_TYPE = %s	char	SAMPLE_INTERLEAVED [STATIC] *Refer to the PDS Standard Reference V3.5 Appendix A.19 “IMAGE”
13	Number of bands	BANDS = %d	smallint	X
14	Compression class and encoding type	ENCODING_TYPE = %s	char	N/A [STATIC]

15	Horizontal pixel count of image	LINE_SAMPLES = %d	int	XXX
16	Vertical pixel count of image	LINES = %d	int	XXX
17	Pixel bit length	SAMPLE_BITS = %d	int	X
18	Pixel type	SAMPLE_TYPE = %s	char	MSB_INTEGE R[STATIC] * Refer to the PDS Standard Reference V3.5 Appendix C.1 for further information about "MSB_INTEGER"
19	Stretched Flag	STRETCHED_FLAG = %s	char	FALSE[STATIC]
20	Alternative value outside assumption	INVALID_CONSTANT = %s	char	X
21	Offset	OFFSET = %f	float	X.X
22	Scaling factor	SCALING_FACTOR = %f	float	X.X
		END_OBJECT = IMAGE		
IMAGE_MAP_PROJECTION Object Description Part(/* IMAGE_MAP_PROJECTION */)				
		OBJECT = IMAGE_MAP_PROJECTION		
23	Resolution	MAP_RESOLUTION = %f<PIXEL/DEGREE>	float	X < PIXEL/DEGREE>
24	Semi-major axis of the ellipsoidal body	A_AXIS_RADIUS = %f<KM>	float	1737.400 <km> [STATIC]
25	medial axis of ellipsoidal body	B_AXIS_RADIUS = %f<KM>	float	1737.400 <km> [STATIC]
26	Semi-minor axis of ellipsoidal body	C_AXIS_RADIUS = %f<KM>	float	1737.400 <km> [STATIC]
27	Name of coordinate system	COORDINATE_SYSTEM_NAME = "%s"	char	"PLANETOCENTRIC "[STATIC]
28	Type of coordinate system	COORDINATE_SYSTEM_TYPE = "%s"	char	"BODY-FIXED ROTATING" [STATIC]
29	Maximum latitude	MAXIMUM_LATITUDE = %f	float	89.0 [STATIC]
30	Minimum latitude	MINIMUM_LATITUDE = %f	float	-89.0 [STATIC]
31	Westernmost longitude	WESTERNMOST_LONGITUDE = %f	float	0.0 [STATIC]
32	Easternmost longitude	EASTERNMOST_LONGITUDE = %f	float	359.0 [STATIC]
33	Type of map projection	MAP_PROJECTION_TYPE = "%s"	char	SIMPLE_CYLINDRICAL [STATIC]
34	Direction of positive longitude	POSITIVE_LONGITUDE_DIRECTION = "%s"	char	EAST [STATIC]
		END_OBJECT = IMAGE_MAP_PROJECTION		
END statement				
		END		

<Example of Label : Reflected Ion Map >

```
PDS_VERSION_ID                  = PDS3
RECORD_TYPE                     = UNDEFINED
FILE_NAME                       = PACE_SI_MAP_001.img
^IMAGE                          = 1082 <BYTES>
RECORD_BYTES                    = 579960
SPACECRAFT_NAME                 = SELENE-M
INSTRUMENT_NAME                 = PACE
PRODUCT_SET_ID                  = PACE_SI_MAP
TARGET_NAME                     = MOON
COMMENT_TEXT                    = "Magnetic field anomaly map.   9 bands."

/* IMAGE */
OBJECT                  = IMAGE
   BAND_STORAGE_TYPE            = SAMPLE_INTERLEAVED
   BANDS                        = 9
   ENCODING_TYPE                = N/A
   LINE_SAMPLES                 = 360
   LINES                        = 179
   SAMPLE_BITS                  = 8
   SAMPLE_TYPE                  = MSB_INTEGER
   STRETCHED_FLAG               = FALSE
```

```
   INVALID_CONSTANT                = 0
   OFFSET                          = 0.0
   SCALING_FACTORE                 = 0.5
END_OBJECT              = IMAGE

/* IMAGE_MAP_PROJECTION */
OBJECT                  = IMAGE_MAP_PROJECTION
   MAP_RESOLUTION                  = 1 < PIXEL / DEGREE>
   A_AXIS_RADIUS                   = 1737.400 <km>
   B_AXIS_RADIUS                   = 1737.400 <km>
   C_AXIS_RADIUS                   = 1737.400 <km>
   COORDINATE_SYSTEM_NAME          = "PLANETOCENTRIC"
   COORDINATE_SYSTEM_TYPE          = "BODY-FIXED ROTATING"
   MAXIMUM_LATITUDE                = 89.0
   MINIMUM_LATITUDE                = -89.0
   WESTERNMOST_LONGITUDE           = 0.0
   EASTERNMOST_LONGITUDE           = 359.0
   MAP_PROJECTION_TYPE             =”SIMPLE_CYLINDRICAL”
   POSITIVE_LONGITUDE_DIRECTION          =EAST
END_OBJECT              = IMAGE_MAP_PROJECTION
END
```

3.3 Data Object Format

T.B.D

3.4 Catalog Information File Format

The Catalog Information File Format is shown in Table 3-2.

Table 3-2 Catalog Information File Format

Item Name	Elements	Format of Value	Range of Value	Values
Name of the data file (*1)	DataFileName	AAAA….AAAA (Up to 31 digits)	alphanumeric characters	dependent on the product (See Section 3.1 “Rules used for File naming”.)
Size of the data file	DataFileSize	NNNNNNNNNNNN (Up to 12 digits)	unit:<byte>	dependent on the product
File format of the data file	DataFileFormat	AAAA….AAAA (Up to 16 digits)	character strings	PDS[STATIC]
Name of the instrument	InstrumentName	AAAA….AAAA (Up to 16 digits)	character strings	PACE [STATIC]
Processing level	ProcessingLevel	AAAA….AAAA (Up to 16 digits)	character strings	dependent on the product (See Table 1-2 “Level”)
Product ID	ProductID	AAAA….AAAA (Up to 30 digits)	character strings	dependent on the product (See Table 1-2 “Product_ID”)
Version number of the product	ProductVersion	AAAA….AAAA (Up to 16 digits)	character strings	dependent on the product
Access level	AccessLevel	N	the value of 0-4	N/A

(*1) “DataFileName” is the stored file name of the product. For the detached format, this is the stored file name.

<Example of Catalog Information : Reflected Ion Map >

```
DataFileName = PACE_SI_MAP_001.img
DataFileSize = 581041
DataFileFormat = PDS
InstrumentName = PACE
ProcessingLevel = Higher Level
```

ProductID = PACE_SI_MAP
ProductVersion = 1.0
  AccessLevel = 4

4. High Resoution Data of Electron/Ion Energy Spectrum (PBF1) (Product ID: PACE_PBF_1)

4.1 Rules used for File naming

The nomenclature used for Label, Data Object and Catalog Information File the product files are described below. In addition, the file names are case-independent.

xPACE_PBFn_yyyymmdd_BBB_V###.dat

- x : Binary Format
  - ✓ I : Intel Format
  - ✓ S : Non-Intel (Standard) Format
- n : Product ID
  - ✓ 1 : PBF1
- yyyymmdd : Observation Date
- BBB : Sensor used (ESA1, ESA2, IMA, IEA)
- ### : Version Number (three digits)
- dat : File Extension
  - ✓ dat.gz : Data File
  - ✓ lbl : Label File
  - ✓ ctg : Catalog Information File
  - ✓ sl2 : L2 Data Set (tar archive)

4.2 Label Format

The Label format for the SERIES object used for the High Resoution Data of Electron/Ion Energy Spectrum (PBF1) is shown in Table 4-1. The Label for the SERIES object includes: Standard Item and Object Format Description Part.

In Table 4-1, the numerical Values and character strings that correspond to the type of product, etc. is set, with exception of the Value shown as “STATIC”.

Table 4-1 SERIES Object Label

No	Items	Elements	Types	Values
Standard Item				
1	PDS version number	PDS_VERSION_ID = %s	char	PDS3 [STATIC]
2	Record format of the file	RECORD_TYPE = %s	char	UNDEFINED [STATIC]
3	File name	FILE_NAME = %s	char	See Section 4.1 “Rules used for File naming”.
4	Name of the mission	MISSION_NAME = %s	char	SELENE [STATIC]
5	Name of the spacecraft	SPACECRAFT_NAME = %s	char	SELENE-M[STATIC]
6	Name of the instrument (Full name)	INSTRUMENT_NAME = %s	char	PACE [STATIC]
7	Product ID	PRODUCT_SET_ID = %s	char	See Table 1-2 “Product_ID
8	Product version number	PRODUCT_VERSION_ID = %s	char	Ver.X.X
9	Target name	TARGET_NAME = %s	char	MOON [STATIC]
10	Comment	COMMENT_TEXT = "%s"	char	
11	Starting position of the object	^TIME_SERIES = %d <BYTES>	int	0 <BYTES> [STATIC]
Object Format Description Part				
		OBJECT = TIME_SERIES		

12	Type of data	INTERCHANGE_FORMAT = "%s"	char	ASCI I[STATIC]
13	Number of lines of data	ROWS = %d	int	XXXXX
14	Number of columns of data	COLUMNS = %d	int	XX
15	Byte count of data line	ROW_BYTES = %d	int	XX
16	Parameter name	SAMPLING_PARAMETER_NAME = "%s"	char	TIME [STATIC]
17	Unit of parameter	SAMPLING_PARAMETER_UNIT = "%s"	char	SECOND [STATIC]
18	Parameter interval	SAMPLING_PARAMETER_INTERVAL = "%f"	float	X.X
19	Start time of data	START_TIME = "%s"	char	yyyy-mm-ddThh:mm:ss
20	Stop time of data	STOP_TIME = "%s"	char	yyyy-mm-ddThh:mm:ss
		END_OBJECT = TIME_SERIES		
END Statements				
		END		

<Example of Label : High Resoution Data of Electron/Ion Energy Spectrum >

```
PDS_VERSION_ID              = PDS3
RECORD_TYPE                 = UNDEFINED
FILE_NAME                   = PACE_ET_20080610.dat
MISSION_NAME                = SELENE
SPACECRAFT_NAME             = SELENE-M
INSTRUMENT_NAME             = PACE
PRODUCT_SET_ID              = PACE_ET
PRODUCT_VERSION_ID          = Ver.1.0
TARGET_NAME                 = MOON
COMMENT_TEXT                = "Energy-time diagram"
^TIME_SERIES                = 0 <BYTES>

/* TIME SERIES */
OBJECT                      = TIME_SERIES
   INTERCHANGE_FORMAT       = ASCII
   ROWS                     = 86400
   COLUMNS                  = 33
   ROW_BYTES                = 64
   SAMPLING_PARAMETER_NAME        = TIME
   SAMPLING_PARAMETER_UNIT        = SECOND
   SAMPLING_PARAMETER_INTERVAL  = 1.0
   START_TIME               = 2008-06-10T00:00:00
   STOP_TIME                = 2008-06-10T23:59:56
END_OBJECT                  = TIME_SERIES
END
```

4.3 Data Object Format

The content of data corresponding to each sensor's (ESA1, ESA2, IMA, and IEA) observation mode of High Resolution Data of Electron / Ion Energy Spectrum is described in the following thesis.

Saito et al. 2008("Low-energy charged particle measurement by MAP-PACE onboard SELENE") (Appendix-1)

In addition, all specific formats are in the header file "paceql_outputdata_070904.h" (Appendix-2). The data can be read by the Sun program "dump_pbf.c" (Appendix-3) and that uses the above-mentioned header file.

4.4 Catalog Information File Format

The Catalog Information File Format is shown in Table 4-2.

Table 4-2 Catalog Information File Format

Item Name	Elements	Format of Value	Range of Value	Values
Name of the data file (*1)	DataFileName	AAAA….AAAA (Up to 31 digits)	alphanumeric characters	dependent on the product (See Section 4.1 "Rules used for File naming".)
Size of the data file	DataFileSize	NNNNNNNNNNNN (Up to 12 digits)	unit:<byte>	dependent on the product
File format of the data file	DataFileFormat	AAAA….AAAA (Up to 16 digits)	character strings	PDS[STATIC]
Name of the instrument	InstrumentName	AAAA….AAAA (Up to 16 digits)	character strings	PACE [STATIC]
Processing level	ProcessingLevel	AAAA….AAAA (Up to 16 digits)	character strings	dependent on the product (See Table 1-2 "Level")
Product ID	ProductID	AAAA….AAAA (Up to 30 digits)	character strings	dependent on the product (See Table 1-2 "Product_ID")
Version number of the product	ProductVersion	AAAA….AAAA (Up to 16 digits)	character strings	dependent on the product
Access level	AccessLevel	N	values of 0-4	N/A
Start time	StartDateTime	yyyy- mmddT hh: mm: ssZ	DATE & TIME	dependent on the product
Stop time	EndDateTime	yyyy- mmddT hh: mm: ssZ	DATE & TIME	dependent on the product

(*1) "DataFileName" is the stored file name of the product. For the detached format, this is the stored file name.

<Example of Catalog Information : High Resoution Data of Electron/Ion Energy Spectrum >

DataFileName =PACE_ET_20080610.dat
DataFileSize =10875290
DataFileFormat =24*60*60
InstrumentName =PACE
ProcessingLevel =Standard
ProductID =PACE_ET
ProductVersion =1.0
AccessLevel =3
StartDateTime = 2008-06-10T00:00:00Z
EndDateTime = 2008-06-10T23:59:59Z

5. High Resoution Data of Electron/Ion Energy Spectrum (CDF) (Product ID:PACE_CDF)

5.1 Rules used for File naming

The nomenclature used for Label, Data Object and Catalog Information File the product files are described below. In addition, the file names are case-independent.

PACE_CDFn_yyyymmdd_BBB_V###.dat

- n : Product ID
  - ✓ 0 : before treatment
  - ✓ 1 : after treatment
- yyyymmdd : Observation Date
- BBB : Sensor used(ESA,IMA,IEA)
- ### : Version Number (three digits)
- dat : File Extension
  - ✓ cdf : Data Fole
  - ✓ ctg : Catalog Information File
  - ✓ sl2 : L2 Data Set (tar archive)

5.2 Label Format

The High Resoution Data of Electron/Ion Energy Spectrum (CDF) is generated in CDF (Common Data Format) format. Therefore, as a L2 product, the data product file and the catalog information file, in the CDF format, are included. The label is not defined.

5.3 Data Object Format

The High Resoution Data of Electron/Ion Energy Spectrum (CDF) is generated in CDF*1 (Common Data Format) format. This completely conforms to CDF Ver3.1.

*1 : http://cdf.gsfc.nasa.gov/

5.4 Catalog Information File Format

T.B.D

6. Summary Plot of Electron/Ion E-T Diagram (Product ID:PACE_ET_sammary)

6.1 Rules used for File naming

The nomenclature used for Label, Data Object and Catalog Information File the product files are described below. In addition, the file names are case-independent.

PACE_ETn_yyyymmdd _V###.dat

- n : Product ID
  - ✓ 0 : before treatment
  - ✓ 1 : after treatment
- yyyymmdd : Observation Date
- ### : Version Number (three digits)
- dat : File Extension
  - ✓ png : Data File
  - ✓ ctg : Catalog Information File
  - ✓ sl2 : L2 Data Set (tar archive)

6.2 Label Format

The Label format for the IMAGE object used for the Summary Plot of Electron/Ion E-T Diagram product is shown in Table 6-1. The Label for the IMAGE object includes: Standard Item and Image Data Object Format Description Part.

In Table 6-1 with the exception of the Values expressed as "STATIC", the numerical values and the character strings corresponding to the type of the product etc., are set.

Table 6-1 IMAGE Object Label

No	Items	Elements	Types	Values
Standard Item				
1	PDS version number	PDS_VERSION_ID = %s	char	PDS3 [STATIC]
2	Record format of the file	RECORD_TYPE = %s	char	UNDEFINED [STATIC]
3	File name	FILE_NAME = %s	char	See Section 6.1 "Rules used for File naming".
4	Starting position of the image object	^IMAGE = %d <BYTES>	int	XXXX <BYTES>
5	Byte count of the file records	RECORD_BYTES = %d	int	XXXXXX
6	Name of the mission	MISSION_NAME = %s	char	SELENE [STATIC]
7	Name of the spacecraft	SPACECRAFT_NAME = %s	char	SELENE-M[STATIC]
8	Name of the instrument (Full name)	INSTRUMENT_NAME = %s	char	PACE [STATIC]
9	Product ID	PRODUCT_SET_ID = %s	char	See Table 1-2 "Product_ID"
10	Version number of the product	PRODUCT_VERSION_ID = %s	char	Ver.X.X
11	Target name	TARGET_NAME = %s	char	MOON [STATIC]
12	Comment	COMMENT_TEXT = "%s"	char	
Image Data Object Format Description Part(/* IMAGE */)				
		OBJECT = IMAGE		
13	Band storage type	BAND_STORAGE_TYPE = %s	char	SAMPLE_INTERLEAVED [STATIC] *Refer to the PDS Standard Reference V3.5 Appendix

				A.19 “IMAGE”
14	Number of bands	BANDS = %d	smallint	X
15	Compression class and encoding type	ENCODING_TYPE = %s	char	N/A [STATIC]
16	Horizontal pixel count of image	LINE_SAMPLES = %d	int	XXX
17	Vertical pixel count of image	LINES = %d	int	XXX
18	Pixel bit length	SAMPLE_BITS = %d	int	X
19	Pixel type	SAMPLE_TYPE = %s	char	MSB_INTEGE R[STATIC] * Refer to the PDS Standard Reference V3.5 Appendix C.1 for further information about “MSB_INTEGER”.
20	Stretched Flag	STRETCHED_FLAG = %s	char	FALS E[STATIC]
21	Alternative value outside assumption	INVALID_CONSTANT = %s	char	X
22	Offset	OFFSET = %f	float	X.X
23	Scaling factor	SCALING_FACTOR = %f	float	X.X
24	Start time	START_TIME = %s	char	YYYY-MM-DDThh:mm:ss
25	Stop time	STOP_TIME = %s	char	YYYY-MM-DDThh:mm:ss
		END_OBJECT = IMAGE		
END statement				
		END		

<Example of Label : Summary Plot of Electron/Ion E-T Diagram >

```
PDS_VERSION_ID          = PDS3
RECORD_TYPE             = UNDEFINED
FILE_NAME               = PACE_ET1_20080601_V001.png
^IMAGE                  = 0 <BYTES>
RECORD_BYTES            = 64475
MISSION_NAME            = SELENE
SPACECRAFT_NAME         = SELENE-M
INSTRUMENT_NAME         = PACE
PRODUCT_SET_ID          = PACE_ET_summary
PRODUCT_VERSION_ID      = Ver.1.0
TARGET_NAME             = MOON
COMMENT_TEXT            = "Energy-time diagram, Magnetic field and S/C orbit"


/* IMAGE */
   OBJECT = IMAGE
   BAND_STORAGE_TYPE = SAMPLE_INTERLEAVED
   BANDS                = 1
   ENCODING_TYPE        = N/A
   LINE_SAMPLES         = 865
   LINES                = 690
   SAMPLE_BITS          = 24
   SAMPLE_TYPE          = MSB_INTEGER
   STRETCHED_FLAG       = FALSE
   INVALID_CONSTANT     = 0
   OFFSET               = 0.0
   SCALING_FACTOR       = 0.0
   START_TIME           = 2008-06-01T00:00:00
   STOP_TIME            = 2008-06-01T23:59:59
END_OBJECT = IMAGE
END
```

6.3 Data Object Format

The Summary Plot of Electron/Ion E-T Diagram product is image data and is specified in PNG format.

6.4 Catalog Information File Format

The Catalog Information File Format is shown in Table 6-2.

Table 6-2 Catalog Information File Format

Item Name	Elements	Format of Value	Range of Value	Values
Name of the data file (*1)	DataFileName	AAAA….AAAA (Up to 31 digits)	alphanumeric characters	dependent on the product (See Section 6.1 "Rules used for File naming".)
Size of the data file	DataFileSize	NNNNNNNNNNN (Up to 12 digits)	unit:<byte>	dependent on the product
File format of the data file	DataFileFormat	AAAA….AAAA (Up to 16 digits)	character strings	PDS[STATIC]
Name of the instrument	InstrumentName	AAAA….AAAA (Up to 16 digits)	character strings	PACE [STATIC]
Processing level	ProcessingLevel	AAAA….AAAA (Up to 16 digits)	character strings	dependent on the product (See Table 1-2 "Level")
Product ID	ProductID	AAAA….AAAA (Up to 30 digits)	character strings	dependent on the product (See Table 1-2 "Product_ID")
Version number of the product	ProductVersion	AAAA….AAAA (Up to 16 digits)	character strings	dependent on the product
Access level	AccessLevel	N	the value of 0-4	N/A

(*1) "DataFileName" is the stored file name of the product. For the detached format, this is the stored file name.

<Example of Catalog Information : Summary Plot of Electron/Ion E-T Diagram >

DataFileName = PACE_SI_MAP_001.img
DataFileSize = 581041
DataFileFormat = PDS
InstrumentName = PACE
ProcessingLevel = Higher Level
ProductID = PACE_SI_MAP
ProductVersion = 1.0
AccessLevel = 4

Appendix - 1
Saito et al. 2008("Low-energy charged particle measurement by MAP-PACE onboard SELENE")

Appendix - 2
paceql_outputdata_070904.h

```
/*
        SELENE PACE level2B data header file
                version     0.4         15 February 2007
                            mode 01,02,11,12,13,14,24,15,25,16,26,17,18,19,29,1A,2A,1B, (+80:internal count)
                            mode 72->(17),73->(11),74->(17), 71, 78, 70
                version     1.0         05 March 2007
                version     1.1         01 May 2007
                version 1.2 14 June 2007
                version 1.3 18 June 2007
                version 1.4 19 June 2007
                version 1.5 20 June 2007
                version 1.6 30 June 2007
                version 2.0 08 August 2007
                version 2.1 10 August 2007
                version 3.0 23 August 2007
                version 4.0 04 September 2007

*/

#define             UCHAR           unsigned char
#define             USHORT          unsigned short
#define             ULONG           unsigned long



/*************************************************************************/
/*************************************************************************/
/* HEADER                                                                */
/*************************************************************************/
/*************************************************************************/


typedef struct Header_PACE {
        ULONG   sensor; /* 0 ESA-S1   1 ESA-S2   2 IMA   3 IEA 4 ALL */
        ULONG   mode; /* data mode = data mode command   */
        ULONG   mode2; /* sub-data mode */
        ULONG   type; /* data type */
        ULONG   size; /* data size */
        ULONG   time_resolution; /* time resolution (msec) */
        ULONG   sampl_time; /* sampling time (16000/*** msec)*/
        ULONG   ver; /* data version */
        ULONG   tbl_ver; /* onboard table version */
        ULONG   obs_ver; /* onboard software version */
        ULONG   timeH; /* 1pps TI High Word */
        ULONG   timeM; /* 1pps TI Medium Word */
        ULONG   timeL; /* 1pps TI Low Word */
        ULONG   bc; /* base clock */
        ULONG   ic; /* increment counter */
        ULONG   sc; /* base counter */
        ULONG   sc_step0; /* sc @ energy sweep 0 */
        ULONG   t_date; /* total date */
        ULONG   time_ms; /* msec of day @ energy sweep 0 */
        ULONG   yyyymmdd;
        ULONG   hhmmss;
        ULONG   tof_tbl; /* IMA */
        ULONG   pd_pha;
        ULONG   svg_tbl; /* IMA IEA */
        ULONG   sva_tbl;
        ULONG   svs_tbl;
        ULONG   obs_tbl;
        ULONG   obs_ctr;
        ULONG   nv_high;
        ULONG   nv_low;
        ULONG   data_quality; /* data quality */
        ULONG   pol_step; /* polar angle step number */
        ULONG   az_step; /* azimuthal angle step number */
        ULONG   ene_step; /* energy step number */
        ULONG   mass_step; /* mass step number */
        ULONG   pitch_step; /* pitch angle step number */
```

```
		ULONG	tof_step; /* tof step number */
		ULONG	solwnd_step; /* solar wind    number */
		ULONG	exb_step; /* ExB    number */
		ULONG	event_step; /* event counter number */
		ULONG	trash_step; /* trash counter number */
		ULONG	tof_disc_start; /* TOF DISCRI SCAN h'73 MODE IMA IEA ONLY */
		ULONG	tof_disc_stop; /* TOF DISCRI SCAN h'73 MODE IMA IEA ONLY */
		ULONG	hv_scan_level; /* 1Byte HV SCAN h'72 MODE ONLY */
		ULONG	spare[20]; /* total header 256bytes = 64 long */
} H_P_t;


/*====================== ESA DATA TYPE ======================*/
/* ------- TYPE 00 ------- */
struct D_ESA_TYPE00 {
		ULONG	event[16];
		USHORT	cnt[32][16][64];
		USHORT	trash[32][16][2];
};
/* ------- TYPE 01 ------- */
struct D_ESA_TYPE01 {
		ULONG	event[16];
		USHORT	cnt[32][4][16];
		USHORT	trash[32][4][2];
};
/* ------- TYPE 02 ------- */
struct D_ESA_TYPE02 {
		ULONG	event[16];
		USHORT	cnt[32][32];
};
/* ------- TYPE 03 ------- */
struct D_ESA_TYPE03 {
		ULONG	event[16];
		USHORT	cnt[32][8][64];
		USHORT	trash[32][8][2];
};

/*====================== IMA DATA TYPE ======================*/
/* ------- TYPE 40 ------- */
struct D_IMA_TYPE40 {
		ULONG event[4][16];
		USHORT cnt[4][32][1024];
};
/* ------- TYPE 41 ------- */
struct D_IMA_TYPE41 {
		ULONG event[4][16];
		USHORT cnt[32][16][64];
		USHORT trash[32][16][2];
};
/* ------- TYPE 42 ------- */
struct D_IMA_TYPE42 {
		ULONG event[4][16];
		USHORT cnt[32][4][16];
		USHORT trash[32][4][2];
};
/* ------- TYPE 43 ------- */
struct D_IMA_TYPE43 {
		ULONG event[4][16];
		USHORT cnt[8][32][4][16];
		USHORT trash[8][32][4][2];
};
/* ------- TYPE 44 ------- */
struct D_IMA_TYPE44 {
		ULONG event[4][16];
		USHORT s_cnt[16][32][64];
		USHORT cnt[16][32][16][64];
};
/* ------- TYPE 45 ------- */
struct D_IMA_TYPE45 {
		ULONG event[4][16];
```

```
                USHORT cnt[16][32][4][16];
                USHORT trash[16][32][4][2];
};

/*====================== IEA DATA TYPE ====================*/
/* ------- TYPE 80 ------- */
struct D_IEA_TYPE80 {
                ULONG event[16];
                USHORT cnt[32][4][16];
                USHORT trash[32][4][2];
};
/* ------- TYPE 81 ------- */
struct D_IEA_TYPE81 {
                ULONG event[16];
                USHORT cnt[32][16][64];
                USHORT trash[32][16][2];
};
/* ------- TYPE 82 ------- */
struct D_IEA_TYPE82 {
                ULONG event[16];
                USHORT s_cnt[32][128];
                USHORT cnt[32][16][64];
};

/*====================== SPECIAL DATA TYPE ====================*/
/* ------- TYPE SV_MONITOR ------- */
struct D_TYPEF0_ESA_SV_MONITOR {
                USHORT sv_unit;
                USHORT sp[1024];
                USHORT ang1[1024];
                USHORT ang2[1024];
};
/* ------- TYPE SV_MONITOR ------- */
struct D_TYPEF1_IMEA_SV_MONITOR {
                USHORT sv_unit;
                USHORT sp[1024];
                USHORT ang1[1024];
                USHORT ang2[1024];
                USHORT g[1024];
};

/* ------- TYPE LMAG_MONITOR ------- */
struct D_TYPEF2_LMAG_MONITOR {
                ULONG time;
                ULONG B[16];
                ULONG x[16];
                ULONG y[16];
                ULONG z[16];
};

/* ------- TYPE INTERNAL_INFORMATION ------- */
struct D_TYPEF3_INTERNAL_INFORMATION {
            ULONG irl1;
            USHORT esa_s1_irl1_addr;
            USHORT esa_s2_irl1_addr;
            USHORT ima_irl1_addr;
            USHORT iea_irl1_addr;
            ULONG esa_s1_intr;
            ULONG esa_s2_intr;
            ULONG ima_intr;
            ULONG iea_intr;
            USHORT tof_disc_start_org;
            USHORT tof_disc_flg;
            ULONG hv_scan_start;
            ULONG hv_scan_gap;
            ULONG esa_s1_sva1ovsvs_p;
            ULONG esa_s2_sva1ovsvs_p;
            ULONG ima_sva1ovsvs_p;
            ULONG iea_sva1ovsvs_p;
            ULONG esa_s1_sva2ovsvs_p;
```

```
        ULONG esa_s2_sva2ovsvs_p;
        ULONG ima_sva2ovsvs_p;
        ULONG iea_sva2ovsvs_p;
        ULONG esa_s1_sva1ovsvs_h;
        ULONG esa_s2_sva1ovsvs_h;
        ULONG ima_sva1ovsvs_h;
        ULONG iea_sva1ovsvs_h;
        ULONG esa_s1_sva2ovsvs_h;
        ULONG esa_s2_sva2ovsvs_h;
        ULONG ima_sva2ovsvs_h;
        ULONG iea_sva2ovsvs_h;
        USHORT tof_disc_start;
        USHORT tof_disc_start_mon;
        USHORT tof_disc_stop;
        USHORT tof_disc_stop_mon;
        USHORT tof_disc_start_scan1;
        USHORT tof_disc_start_scan2;
        USHORT tof_disc_start_scan3;
        USHORT tof_disc_start_scan4;
        USHORT tof_disc_stop_scan1;
        USHORT tof_disc_stop_scan2;
        USHORT tof_disc_stop_scan3;
        USHORT tof_disc_stop_scan4;
        ULONG nv_l;
        ULONG nv_l_cmd;
        ULONG nv_h;
        ULONG nv_h_cmd;
        ULONG errbufclr_ctr;
        USHORT esa_s1_obs_mode;
        USHORT esa_s2_obs_mode;
        USHORT ima_obs_mode;
        USHORT iea_obs_mode;
        USHORT esa_s1_ena_dis;
        USHORT esa_s2_ena_dis;
        USHORT ima_ena_dis;
        USHORT iea_ena_dis;
        USHORT esa_s1_clk_status;
        USHORT esa_s2_clk_status;
        USHORT ima_clk_status;
        USHORT iea_clk_status;
        USHORT tof_disc_start2;
        USHORT tof_disc_stop2;
};

/*============================================================*/

/*********************************************************************/
/* ELECTRON MODE EC-N                                                */
/*     Electron Check Mode 0x00                                      */
/*                 16X64(mode1) 16s RAM67    NB    TYPE00            */
/*********************************************************************/
typedef struct D_ESA_TYPE00 D_ESAS1_TYPE00_EC_N_t;          /* ESAS1 */
typedef struct D_ESA_TYPE00     D_ESAS2_TYPE00_EC_N_t;      /* ESAS2 */

/*********************************************************************/
/* ELECTRON MODE EM_N                                                */
/*     Electron Magnetosphere Noraml Mode 0x01                       */
/*                 4X16(mode3) 2s RAM67    NB    TYPE01              */
/*********************************************************************/
typedef struct D_ESA_TYPE01 D_ESAS1_TYPE01_EM_N_t;/* ESAS1 */
typedef struct D_ESA_TYPE01 D_ESAS2_TYPE01_EM_N_t;/* ESAS2 */

/*********************************************************************/
/* ELECTRON MODE EM_H                                                */
/*     Electron Magnetosphere High Mode 0x08                         */
/*                 4X16(mode3) 1s RAM67    NB    TYPE01              */
```

```
/*****************************************************************/
typedef struct          D_ESA_TYPE01 D_ESAS1_TYPE01_EM_H_t;/* ESAS1 */
typedef struct          D_ESA_TYPE01 D_ESAS2_TYPE01_EM_H_t;/* ESAS2 */


/*****************************************************************/
/* ELECTRON MODE EM_R                                            */
/*     Electron Magnetosphere No Compression Mode 0x02           */
/*              4X16(mode3) 4s RAM67   NB   TYPE01               */
/*****************************************************************/

typedef struct          D_ESA_TYPE01 D_ESAS1_TYPE01_EM_R_t;/* ESAS1 */
typedef struct          D_ESA_TYPE01 D_ESAS2_TYPE01_EM_R_t;/* ESAS2 */


/*****************************************************************/
/* ELECTRON MODE ER_N                                            */
/*     Electron Electron Reflectometer Normal Mode 0x03          */
/*              16X64(mode1) 16s(2s) RAM0123   NB   TYPE00       */
/*****************************************************************/
typedef struct          D_ESA_TYPE00 D_ESAS1_TYPE00_ER_N_t;/* ESAS1 */
typedef struct          D_ESA_TYPE00 D_ESAS2_TYPE00_ER_N_t;/* ESAS2 */


/*****************************************************************/
/* ELECTRON MODE ER_H                                            */
/*     Electron Electron Reflectometer High Mode 0x08            */
/*              16X64(mode1) 8s(1s) RAM0123   B   TYPE02         */
/*****************************************************************/
typedef struct          D_ESA_TYPE02 D_ESAS1_TYPE02_ER_H_t;/* ESAS1 */
typedef struct          D_ESA_TYPE02 D_ESAS2_TYPE02_ER_H_t;/* ESAS2 */


/*****************************************************************/
/* ELECTRON MODE ER_W                                            */
/*     Electron Electron Reflectometer Wake Mode 0x04            */
/*              16X64(mode1) 8s(1s) RAM0123   NB POL8 TYPE03     */
/*****************************************************************/
typedef struct          D_ESA_TYPE03 D_ESAS1_TYPE03_ER_W_t;/* ESAS1 */
typedef struct          D_ESA_TYPE03 D_ESAS2_TYPE03_ER_W_t;/* ESAS2 */


/*****************************************************************/
/* ELECTRON MODE ER_B                                            */
/*     Electron Electron Reflectometer Backup Mode 0x05          */
/*              16X64(mode1) 16s(4s) RAM45   NB POL16 TYPE00     */
/*****************************************************************/
typedef struct          D_ESA_TYPE00 D_ESAS1_TYPE00_ER_B_t;/* ESAS1 */
typedef struct          D_ESA_TYPE00 D_ESAS2_TYPE00_ER_B_t;/* ESAS2 */


/*****************************************************************/
/* ELECTRON MODE ER_R                                            */
/*     Electron Electron Reflectometer No Compression Mode 0x07  */
/*              16X64(mode1) 8s(1s) RAM0123   B TYPE02           */
/*****************************************************************/
typedef struct          D_ESA_TYPE02 D_ESAS1_TYPE02_ER_R_t;/* ESAS1 */
typedef struct          D_ESA_TYPE02 D_ESAS2_TYPE02_ER_R_t;/* ESAS2 */


/*****************************************************************/
/* ELECTRON MODE PM-1                                            */
/*     Electron PM NORMAL 1 Mode 0x21                            */
/*              4X16(mode2) 16s RAM01234567   NB TYPE01          */
/*****************************************************************/
typedef struct          D_ESA_TYPE01 D_ESAS1_TYPE01_PM_1_t;/* ESAS1 */
typedef struct          D_ESA_TYPE01 D_ESAS2_TYPE01_PM_1_t;/* ESAS2 */


/*****************************************************************/
```

```
/* ELECTRON MODE PM-2                                                    */
/*     Electron PM NORMAL 2 Mode 0x22                                    */
/*                 16X64(mode1) 32s RAM01234567    NB TYPE00             */
/*************************************************************************/
typedef struct          D_ESA_TYPE00 D_ESAS1_TYPE00_PM_2_t;/* ESAS1 */
typedef struct          D_ESA_TYPE00 D_ESAS2_TYPE00_PM_2_t;/* ESAS2 */


/*-----------------------------------------------------------------------*/

/*************************************************************************/
/* ION MODE IC_T                                                         */
/*     Ion Time Check Mode 0x10                                          */
/*          IMA 4X1X1024(direct1) 16s RAM0123 MASS1024 TYPE40            */
/*          IEA 4X16(mode3) 16s RAM0123                TYPE80            */
/*************************************************************************/
typedef struct                  D_IMA_TYPE40 D_IMA_TYPE40_IC_T_t;/*   IMA   */
typedef struct                  D_IEA_TYPE80 D_IEA_TYPE80_IC_T_t;/*   IEA   */


/*************************************************************************/
/* ION MODE IC_P                                                         */
/*     Ion Position Check Mode 0x11                                      */
/*          IMA 16X64X16(mode1) 16s RAM0123 MASS1    TYPE41              */
/*          IEA 16X64(mode1) 16s RAM0123             TYPE81              */
/*************************************************************************/
typedef struct                  D_IMA_TYPE41 D_IMA_TYPE41_IC_P_t;/*   IMA   */
typedef struct                  D_IEA_TYPE81 D_IEA_TYPE81_IC_P_t;/*   IEA   */


/*************************************************************************/
/* ION MODE IM_N                                                         */
/*     Ion Magnetosphere Normal Mode 0x12                                */
/*          IMA 4X16X16(mode3) 1s RAM0123 MASS1 TYPE42                   */
/*          IEA 4X16(mode3) 1s RAM0123          TYPE80                   */
/*************************************************************************/
typedef struct                  D_IMA_TYPE42 D_IMA_TYPE42_IM_N_t;/*   IMA   */
typedef struct                  D_IEA_TYPE80 D_IEA_TYPE80_IM_N_t;/*   IEA   */


/*************************************************************************/
/* ION MODE IM_L                                                         */
/*     Ion Magnetosphere Low Mode 0x19                                   */
/*          IMA 4X16X16(mode3) 2s RAM0123 MASS1    TYPE42                */
/*          IEA 4X16(mode3) 2s RAM0123             TYPE80                */
/*************************************************************************/
typedef struct                  D_IMA_TYPE42 D_IMA_TYPE42_IM_L_t;/*   IMA   */
typedef struct                  D_IEA_TYPE80 D_IEA_TYPE80_IM_L_t;/*   IEA   */


/*************************************************************************/
/* ION MODE IL_S                                                         */
/*     Ion Lunar Solar Wind Mode 0x13                                    */
/*          IMA 4X16X16(mode3) 8s RAM0    MASS8    TYPE43                */
/*          IEA 16X64(mode1) 2s RAM0 128PT         TYPE82                */
/*************************************************************************/
typedef struct                  D_IMA_TYPE43 D_IMA_TYPE43_IL_S_t;/*   IMA   */
typedef struct                  D_IEA_TYPE82 D_IEA_TYPE82_IL_S_t;/*   IEA   */


/*************************************************************************/
/* ION MODE IL_V                                                         */
/*     Ion Lunar WAKE Mode 0x14                                          */
/*          IMA 4X16X16(mode3) 8s RAM1 MASS8    TYPE43                   */
/*          IEA 16X64(mode1) 8s RAM1               TYPE81                */
/*************************************************************************/
typedef struct                  D_IMA_TYPE43 D_IMA_TYPE43_IL_V_t;/*   IMA   */
typedef struct                  D_IEA_TYPE81 D_IEA_TYPE81_IL_V_t;/*   IEA   */
```

```
/*********************************************************************/
/* ION MODE IL_M                                                     */
/*     Ion Lunar Ion Mass Survey Mode 0x15                           */
/*          IMA 4X16X64(drct3) 16s RAM0123 Time Profile   TYPE45     */
/*          IEA 4X16(mode3) 16s RAM0123              TYPE80          */
/*********************************************************************/
typedef struct                  D_IMA_TYPE45 D_IMA_TYPE45_IL_M_t;/*   IMA   */
typedef struct                  D_IEA_TYPE80 D_IEA_TYPE80_IL_M_t;/*   IEA   */


/*********************************************************************/
/* ION MODE IL_R                                                     */
/*     Ion Lunar Ion Wake No Compression Mode 0x16                   */
/*          IMA 4X16X16(mode3) 16s RAM1 MASS8   TYPE43               */
/*          IEA 4X16(mode3) 16s RAM1            TYPE80               */
/*********************************************************************/
typedef struct                  D_IMA_TYPE43 D_IMA_TYPE43_IL_R_t;/*   IMA   */
typedef struct                  D_IEA_TYPE80 D_IEA_TYPE80_IL_R_t;/*   IEA   */


/*********************************************************************/
/* ION MODE IS_A                                                     */
/*     Ion Lunar Ion Sputtering Mode 0x17                            */
/*          IMA 16X64X16(mode1) 16s(4s) RAM2 8ENERGY MASS16   TYPE44 */
/*          IEA 16X64(mode1) 2s RAM0 128PT        TYPE82             */
/*********************************************************************/
typedef struct                  D_IMA_TYPE44 D_IMA_TYPE44_IS_A_t;/*   IMA   */
typedef struct                  D_IEA_TYPE82 D_IEA_TYPE82_IS_A_t;/*   IEA   */


/*********************************************************************/
/* ION MODE PM_1                                                     */
/*     Ion PM NORMAL1 Mode 0x21                                      */
/*          IMA 16X4X16(mode2) 16s RAM0123 MASS16 TYPE45             */
/*          IEA 4X16(mode2) 16s RAM0123           TYPE80             */
/*********************************************************************/
typedef struct          D_IMA_TYPE45 D_IMA_TYPE45_PM_1_t;/*   IMA   */
typedef struct          D_IEA_TYPE80 D_IEA_TYPE80_PM_1_t;/*   IEA   */


/*********************************************************************/
/* ION MODE PM_2                                                     */
/*     Ion PM NORMAL2 Mode 0x22                                      */
/*          IMA 16X16X64(mode1) 32s RAM0123 MASS1   TYPE41           */
/*          IEA 16X64(mode1) 32s RAM0123            TYPE81           */
/*********************************************************************/
typedef struct          D_IMA_TYPE41 D_IMA_TYPE41_PM_2_t;/*   IMA   */
typedef struct          D_IEA_TYPE81 D_IEA_TYPE81_PM_2_t;/*   IEA   */


/*********************************************************************/
/* SPECIAL MODE SV_MONITOR                                           */
/*     SV MONITOR MODE   Command Mode 0x0071                         */
/*          ESA-S1 sp 1024 , ang1 1024 , ang2 1024                   */
/*          ESA-S2 sp 1024 , ang1 1024 , ang2 1024                   */
/*          IMA      sp 1024 , ang1 1024 , ang2 1024 , g 1024        */
/*          IEA      sp 1024 , ang1 1024 , ang2 1024 , g 1024        */
/*********************************************************************/
typedef struct                  D_TYPEF0_ESA_SV_MONITOR         D_TYPEF0_ESAS1_SV_MONITOR_t;/*
ESAS1 */
typedef struct                  D_TYPEF0_ESA_SV_MONITOR         D_TYPEF0_ESAS2_SV_MONITOR_t;/*
ESAS2 */
typedef struct                  D_TYPEF1_IMEA_SV_MONITOR        D_TYPEF1_IMA_SV_MONITOR_t;/*
IMA */
typedef struct                  D_TYPEF1_IMEA_SV_MONITOR        D_TYPEF1_IEA_SV_MONITOR_t;/*
IEA */


/*********************************************************************/
/* SPECIAL MODE LMAG_MONITOR                                         */
```

```
/*     LMAG MONITOR MODE    Command Mode 0x0078                                    */
/*************************************************************************/
typedef struct                          D_TYPEF2_LMAG_MONITOR     D_TYPEF2_LMAG_MONITOR_t;


/*************************************************************************/
/* SPECIAL MODE INTERNAL_INFORMATION                                      */
/*     INTERNAL INFORMATION MODE    Command Mode 0x0070                    */
/*************************************************************************/
typedef struct                          D_TYPEF3_INTERNAL_INFORMATION
        D_TYPEF3_INTERNAL_INFORMATION_t;




/*************************************************************************/
/* data type definition ESA                                               */
/*************************************************************************/
typedef struct D_ESA_TYPE00 D_ESAS1_TYPE00_t;     /* ESAS1 */
typedef struct D_ESA_TYPE00 D_ESAS2_TYPE00_t;     /* ESAS2 */
typedef struct D_ESA_TYPE01 D_ESAS1_TYPE01_t;     /* ESAS1 */
typedef struct D_ESA_TYPE01 D_ESAS2_TYPE01_t;     /* ESAS2 */
typedef struct D_ESA_TYPE02 D_ESAS1_TYPE02_t;     /* ESAS1 */
typedef struct D_ESA_TYPE02 D_ESAS2_TYPE02_t;     /* ESAS2 */
typedef struct D_ESA_TYPE03 D_ESAS1_TYPE03_t;     /* ESAS1 */
typedef struct D_ESA_TYPE03 D_ESAS2_TYPE03_t;     /* ESAS2 */
/*************************************************************************/
/* data type definition IMA                                               */
/*************************************************************************/
typedef struct D_IMA_TYPE40 D_IMA_TYPE40_t;       /* IMA */
typedef struct D_IMA_TYPE41 D_IMA_TYPE41_t;       /* IMA */
typedef struct D_IMA_TYPE42 D_IMA_TYPE42_t;       /* IMA */
typedef struct D_IMA_TYPE43 D_IMA_TYPE43_t;       /* IMA */
typedef struct D_IMA_TYPE44 D_IMA_TYPE44_t;       /* IMA */
typedef struct D_IMA_TYPE45 D_IMA_TYPE45_t;       /* IMA */
/*************************************************************************/
/* data type definition IEA                                               */
/*************************************************************************/
typedef struct D_IEA_TYPE80 D_IEA_TYPE80_t;       /* IEA */
typedef struct D_IEA_TYPE81 D_IEA_TYPE81_t;       /* IEA */
typedef struct D_IEA_TYPE82 D_IEA_TYPE82_t;       /* IEA */




/*************************************************************************/
/*************************************************************************/
/* MODE 01 PM NORMAL1                                                     */
/*************************************************************************/
/*************************************************************************/
typedef struct S_ESAS1_MODE_01 {
        H_P_t       header;
        D_ESAS1_TYPE01_PM_1_t       data;
} S_ESAS1_MODE_01_t;

typedef struct S_ESAS2_MODE_01 {
        H_P_t       header;
        D_ESAS2_TYPE01_PM_1_t       data;
} S_ESAS2_MODE_01_t;

typedef struct S_IMA_MODE_01 {
        H_P_t       header;
        D_IMA_TYPE45_PM_1_t         data;
} S_IMA_MODE_01_t;
```

```
typedef struct S_IEA_MODE_01 {
        H_P_t       header;
        D_IEA_TYPE80_PM_1_t         data;
} S_IEA_MODE_01_t;


/*************************************************************************/
/*************************************************************************/
/* MODE 02 PM NORMAL2                                                    */
/*************************************************************************/
/*************************************************************************/
typedef struct S_ESAS1_MODE_02 {
        H_P_t       header;
        D_ESAS1_TYPE00_PM_2_t       data;
} S_ESAS1_MODE_02_t;

typedef struct S_ESAS2_MODE_02 {
        H_P_t       header;
        D_ESAS2_TYPE00_PM_2_t       data;
} S_ESAS2_MODE_02_t;

typedef struct S_IMA_MODE_02 {
        H_P_t       header;
        D_IMA_TYPE41_PM_2_t         data;
} S_IMA_MODE_02_t;

typedef struct S_IEA_MODE_02 {
        H_P_t       header;
        D_IEA_TYPE81_PM_2_t         data;
} S_IEA_MODE_02_t;


/*************************************************************************/
/*************************************************************************/
/* MODE 11 TOFCAL                                                        */
/*************************************************************************/
/*************************************************************************/
typedef struct S_ESAS1_MODE_11 {
        H_P_t       header;
        D_ESAS1_TYPE00_EC_N_t       data;
} S_ESAS1_MODE_11_t;

typedef struct S_ESAS2_MODE_11 {
        H_P_t       header;
        D_ESAS2_TYPE00_EC_N_t       data;
} S_ESAS2_MODE_11_t;

typedef struct S_IMA_MODE_11 {
        H_P_t       header;
        D_IMA_TYPE40_IC_T_t         data;
} S_IMA_MODE_11_t;

typedef struct S_IEA_MODE_11 {
        H_P_t       header;
        D_IEA_TYPE80_IC_T_t         data;
} S_IEA_MODE_11_t;


/*************************************************************************/
/*************************************************************************/
/* MODE 12 POSCAL                                                        */
/*************************************************************************/
/*************************************************************************/
typedef struct S_ESAS1_MODE_12 {
        H_P_t       header;
        D_ESAS1_TYPE00_EC_N_t       data;
} S_ESAS1_MODE_12_t;

typedef struct S_ESAS2_MODE_12 {
```

```
        H_P_t       header;
        D_ESAS2_TYPE00_EC_N_t       data;
} S_ESAS2_MODE_12_t;

typedef struct S_IMA_MODE_12 {
        H_P_t       header;
        D_IMA_TYPE41_IC_P_t         data;
} S_IMA_MODE_12_t;

typedef struct S_IEA_MODE_12 {
        H_P_t       header;
        D_IEA_TYPE81_IC_P_t         data;
} S_IEA_MODE_12_t;


/*****************************************************************************/
/*****************************************************************************/
/* MODE 13 E2sI1s                                                            */
/*****************************************************************************/
/*****************************************************************************/
typedef struct S_ESAS1_MODE_13 {
        H_P_t       header;
        D_ESAS1_TYPE01_EM_N_t       data;
} S_ESAS1_MODE_13_t;

typedef struct S_ESAS2_MODE_13 {
        H_P_t       header;
        D_ESAS2_TYPE01_EM_N_t       data;
} S_ESAS2_MODE_13_t;

typedef struct S_IMA_MODE_13 {
        H_P_t       header;
        D_IMA_TYPE42_IM_N_t         data;
} S_IMA_MODE_13_t;

typedef struct S_IEA_MODE_13 {
        H_P_t       header;
        D_IEA_TYPE80_IM_N_t         data;
} S_IEA_MODE_13_t;


/*****************************************************************************/
/*****************************************************************************/
/* MODE 14 3D MASS SOLAR WIND                                                */
/*****************************************************************************/
/*****************************************************************************/
typedef struct S_ESAS1_MODE_14 {
        H_P_t       header;
        D_ESAS1_TYPE01_EM_N_t       data;
} S_ESAS1_MODE_14_t;

typedef struct S_ESAS2_MODE_14 {
        H_P_t       header;
        D_ESAS2_TYPE01_EM_N_t       data;
} S_ESAS2_MODE_14_t;

typedef struct S_IMA_MODE_14 {
        H_P_t       header;
        D_IMA_TYPE43_IL_S_t         data;
} S_IMA_MODE_14_t;

typedef struct S_IEA_MODE_14 {
        H_P_t       header;
        D_IEA_TYPE82_IL_S_t         data;
} S_IEA_MODE_14_t;


/*****************************************************************************/
/*****************************************************************************/
/* MODE 24 3D MASS WAKE                                                      */
```

```
/*****************************************************************************/
/*****************************************************************************/
typedef struct S_ESAS1_MODE_24 {
        H_P_t       header;
        D_ESAS1_TYPE01_EM_N_t       data;
} S_ESAS1_MODE_24_t;

typedef struct S_ESAS2_MODE_24 {
        H_P_t       header;
        D_ESAS2_TYPE01_EM_N_t       data;
} S_ESAS2_MODE_24_t;

typedef struct S_IMA_MODE_24 {
        H_P_t       header;
        D_IMA_TYPE43_IL_V_t         data;
} S_IMA_MODE_24_t;

typedef struct S_IEA_MODE_24 {
        H_P_t       header;
        D_IEA_TYPE81_IL_V_t         data;
} S_IEA_MODE_24_t;


/*****************************************************************************/
/*****************************************************************************/
/* MODE 15 ER LUNAR ION SOLAR WIND                                           */
/*****************************************************************************/
/*****************************************************************************/
typedef struct S_ESAS1_MODE_15E {
        H_P_t       header;
        D_ESAS1_TYPE02_ER_H_t       data;
} S_ESAS1_MODE_15E_t;
typedef struct S_ESAS1_MODE_15T {
        H_P_t       header;
        D_ESAS1_TYPE01_EM_N_t       data;
} S_ESAS1_MODE_15T_t;

typedef struct S_ESAS2_MODE_15E {
        H_P_t       header;
        D_ESAS2_TYPE02_ER_H_t       data;
} S_ESAS2_MODE_15E_t;
typedef struct S_ESAS2_MODE_15T {
        H_P_t       header;
        D_ESAS2_TYPE01_EM_N_t       data;
} S_ESAS2_MODE_15T_t;

typedef struct S_IMA_MODE_15 {
        H_P_t       header;
        D_IMA_TYPE43_IL_S_t         data;
} S_IMA_MODE_15_t;

typedef struct S_IEA_MODE_15 {
        H_P_t       header;
        D_IEA_TYPE82_IL_S_t         data;
} S_IEA_MODE_15_t;


/*****************************************************************************/
/*****************************************************************************/
/* MODE 25 ER LUNAR ION WAKE                                                 */
/*****************************************************************************/
/*****************************************************************************/
typedef struct S_ESAS1_MODE_25E {
        H_P_t       header;
        D_ESAS1_TYPE03_ER_W_t       data;
} S_ESAS1_MODE_25E_t;
typedef struct S_ESAS1_MODE_25T {
        H_P_t       header;
        D_ESAS1_TYPE01_EM_N_t       data;
} S_ESAS1_MODE_25T_t;
```

```
typedef struct S_ESAS2_MODE_25E {
        H_P_t       header;
        D_ESAS2_TYPE03_ER_W_t       data;
} S_ESAS2_MODE_25E_t;
typedef struct S_ESAS2_MODE_25T {
        H_P_t       header;
        D_ESAS2_TYPE01_EM_N_t       data;
} S_ESAS2_MODE_25T_t;

typedef struct S_IMA_MODE_25 {
        H_P_t       header;
        D_IMA_TYPE43_IL_V_t         data;
} S_IMA_MODE_25_t;

typedef struct S_IEA_MODE_25 {
        H_P_t       header;
        D_IEA_TYPE81_IL_V_t         data;
} S_IEA_MODE_25_t;


/*************************************************************************/
/*************************************************************************/
/* MODE 16 ER SPUTTERING ION SOLAR WIND                                  */
/*************************************************************************/
/*************************************************************************/
typedef struct S_ESAS1_MODE_16E {
        H_P_t       header;
        D_ESAS1_TYPE02_ER_H_t       data;
} S_ESAS1_MODE_16E_t;
typedef struct S_ESAS1_MODE_16T {
        H_P_t       header;
        D_ESAS1_TYPE01_EM_N_t       data;
} S_ESAS1_MODE_16T_t;

typedef struct S_ESAS2_MODE_16E {
        H_P_t       header;
        D_ESAS2_TYPE02_ER_H_t       data;
} S_ESAS2_MODE_16E_t;
typedef struct S_ESAS2_MODE_16T {
        H_P_t       header;
        D_ESAS2_TYPE01_EM_N_t       data;
} S_ESAS2_MODE_16T_t;

typedef struct S_IMA_MODE_16 {
        H_P_t       header;
        D_IMA_TYPE44_IS_A_t         data;
} S_IMA_MODE_16_t;

typedef struct S_IEA_MODE_16 {
        H_P_t       header;
        D_IEA_TYPE82_IS_A_t         data;
} S_IEA_MODE_16_t;


/*************************************************************************/
/*************************************************************************/
/* MODE 26 ER SPUTTERING ION WAKE                                        */
/*************************************************************************/
/*************************************************************************/
typedef struct S_ESAS1_MODE_26E {
        H_P_t       header;
        D_ESAS1_TYPE03_ER_W_t       data;
} S_ESAS1_MODE_26E_t;
typedef struct S_ESAS1_MODE_26T {
        H_P_t       header;
        D_ESAS1_TYPE01_EM_N_t       data;
} S_ESAS1_MODE_26T_t;

typedef struct S_ESAS2_MODE_26E {
```

```
        H_P_t       header;
        D_ESAS2_TYPE03_ER_W_t       data;
} S_ESAS2_MODE_26E_t;
typedef struct S_ESAS2_MODE_26T {
        H_P_t       header;
        D_ESAS2_TYPE01_EM_N_t       data;
} S_ESAS2_MODE_26T_t;

typedef struct S_IMA_MODE_26 {
        H_P_t       header;
        D_IMA_TYPE44_IS_A_t         data;
} S_IMA_MODE_26_t;

typedef struct S_IEA_MODE_26 {
        H_P_t       header;
        D_IEA_TYPE82_IS_A_t         data;
} S_IEA_MODE_26_t;


/*************************************************************************/
/*************************************************************************/
/* MODE 17 LUNAR ION WAKE NO COMPRESSION                                 */
/*************************************************************************/
/*************************************************************************/
typedef struct S_ESAS1_MODE_17 {
        H_P_t       header;
        D_ESAS1_TYPE01_EM_R_t       data;
} S_ESAS1_MODE_17_t;

typedef struct S_ESAS2_MODE_17 {
        H_P_t       header;
        D_ESAS2_TYPE01_EM_R_t       data;
} S_ESAS2_MODE_17_t;

typedef struct S_IMA_MODE_17 {
        H_P_t       header;
        D_IMA_TYPE43_IL_R_t         data;
} S_IMA_MODE_17_t;

typedef struct S_IEA_MODE_17 {
        H_P_t       header;
        D_IEA_TYPE80_IL_R_t         data;
} S_IEA_MODE_17_t;


/*************************************************************************/
/*************************************************************************/
/* MODE 18 ER LUNAR ION NO COMPRESSION                                   */
/*************************************************************************/
/*************************************************************************/
typedef struct S_ESAS1_MODE_18E {
        H_P_t       header;
        D_ESAS1_TYPE02_ER_R_t       data;
} S_ESAS1_MODE_18E_t;
typedef struct S_ESAS1_MODE_18T {
        H_P_t       header;
        D_ESAS1_TYPE01_EM_R_t       data;
} S_ESAS1_MODE_18T_t;

typedef struct S_ESAS2_MODE_18E {
        H_P_t       header;
        D_ESAS2_TYPE02_ER_R_t       data;
} S_ESAS2_MODE_18E_t;
typedef struct S_ESAS2_MODE_18T {
        H_P_t       header;
        D_ESAS2_TYPE01_EM_R_t       data;
} S_ESAS2_MODE_18T_t;

typedef struct S_IMA_MODE_18 {
        H_P_t       header;
```

```
        D_IMA_TYPE43_IL_R_t         data;
} S_IMA_MODE_18_t;

typedef struct S_IEA_MODE_18 {
        H_P_t       header;
        D_IEA_TYPE80_IL_R_t         data;
} S_IEA_MODE_18_t;


/*****************************************************************************/
/*****************************************************************************/
/* MODE 19 ER LUNAR ION SOLAR WIND BACKUP                                    */
/*****************************************************************************/
/*****************************************************************************/
typedef struct S_ESAS1_MODE_19E {
        H_P_t       header;
        D_ESAS1_TYPE02_ER_H_t       data;
} S_ESAS1_MODE_19E_t;
typedef struct S_ESAS1_MODE_19T {
        H_P_t       header;
        D_ESAS1_TYPE01_EM_N_t       data;
} S_ESAS1_MODE_19T_t;

typedef struct S_ESAS2_MODE_19E {
        H_P_t       header;
        D_ESAS2_TYPE02_ER_H_t       data;
} S_ESAS2_MODE_19E_t;
typedef struct S_ESAS2_MODE_19T {
        H_P_t       header;
        D_ESAS2_TYPE01_EM_N_t       data;
} S_ESAS2_MODE_19T_t;

typedef struct S_IMA_MODE_19 {
        H_P_t       header;
        D_IMA_TYPE43_IL_S_t         data;
} S_IMA_MODE_19_t;

typedef struct S_IEA_MODE_19 {
        H_P_t       header;
        D_IEA_TYPE82_IL_S_t         data;
} S_IEA_MODE_19_t;


/*****************************************************************************/
/*****************************************************************************/
/* MODE 29 ER LUNAR ION WAKE BACKUP                                          */
/*****************************************************************************/
/*****************************************************************************/
typedef struct S_ESAS1_MODE_29E {
        H_P_t       header;
        D_ESAS1_TYPE00_ER_N_t       data;
} S_ESAS1_MODE_29E_t;
typedef struct S_ESAS1_MODE_29T {
        H_P_t       header;
        D_ESAS1_TYPE01_EM_N_t       data;
} S_ESAS1_MODE_29T_t;

typedef struct S_ESAS2_MODE_29E {
        H_P_t       header;
        D_ESAS2_TYPE00_ER_N_t       data;
} S_ESAS2_MODE_29E_t;
typedef struct S_ESAS2_MODE_29T {
        H_P_t       header;
        D_ESAS2_TYPE01_EM_N_t       data;
} S_ESAS2_MODE_29T_t;

typedef struct S_IMA_MODE_29 {
        H_P_t       header;
        D_IMA_TYPE43_IL_V_t         data;
} S_IMA_MODE_29_t;
```

```
typedef struct S_IEA_MODE_29 {
        H_P_t       header;
        D_IEA_TYPE81_IL_V_t         data;
} S_IEA_MODE_29_t;


/*****************************************************************************/
/*****************************************************************************/
/* MODE 1A ER SPUTTERING ION SOLAR WIND BACKUP                               */
/*****************************************************************************/
/*****************************************************************************/
typedef struct S_ESAS1_MODE_1AE {
        H_P_t       header;
        D_ESAS1_TYPE02_ER_H_t       data;
} S_ESAS1_MODE_1AE_t;
typedef struct S_ESAS1_MODE_1AT {
        H_P_t       header;
        D_ESAS1_TYPE01_EM_N_t       data;
} S_ESAS1_MODE_1AT_t;

typedef struct S_ESAS2_MODE_1AE {
        H_P_t       header;
        D_ESAS2_TYPE02_ER_H_t       data;
} S_ESAS2_MODE_1AE_t;
typedef struct S_ESAS2_MODE_1AT {
        H_P_t       header;
        D_ESAS2_TYPE01_EM_N_t       data;
} S_ESAS2_MODE_1AT_t;

typedef struct S_IMA_MODE_1A {
        H_P_t       header;
        D_IMA_TYPE44_IS_A_t         data;
} S_IMA_MODE_1A_t;

typedef struct S_IEA_MODE_1A {
        H_P_t       header;
        D_IEA_TYPE82_IS_A_t         data;
} S_IEA_MODE_1A_t;


/*****************************************************************************/
/*****************************************************************************/
/* MODE 2A ER SPUTTERING ION WAKE BACKUP                                     */
/*****************************************************************************/
/*****************************************************************************/
typedef struct S_ESAS1_MODE_2AE {
        H_P_t       header;
        D_ESAS1_TYPE00_ER_N_t       data;
} S_ESAS1_MODE_2AE_t;
typedef struct S_ESAS1_MODE_2AT {
        H_P_t       header;
        D_ESAS1_TYPE01_EM_N_t       data;
} S_ESAS1_MODE_2AT_t;

typedef struct S_ESAS2_MODE_2AE {
        H_P_t       header;
        D_ESAS2_TYPE00_ER_N_t       data;
} S_ESAS2_MODE_2AE_t;
typedef struct S_ESAS2_MODE_2AT {
        H_P_t       header;
        D_ESAS2_TYPE01_EM_N_t       data;
} S_ESAS2_MODE_2AT_t;

typedef struct S_IMA_MODE_2A {
        H_P_t       header;
        D_IMA_TYPE44_IS_A_t         data;
} S_IMA_MODE_2A_t;

typedef struct S_IEA_MODE_2A {
```

```
        H_P_t       header;
        D_IEA_TYPE82_IS_A_t         data;
} S_IEA_MODE_2A_t;


/*****************************************************************************/
/*****************************************************************************/
/* MODE 1B E1sI2s                                                            */
/*****************************************************************************/
/*****************************************************************************/
typedef struct S_ESAS1_MODE_1B {
        H_P_t       header;
        D_ESAS1_TYPE01_EM_H_t       data;
} S_ESAS1_MODE_1B_t;

typedef struct S_ESAS2_MODE_1B {
        H_P_t       header;
        D_ESAS2_TYPE01_EM_H_t       data;
} S_ESAS2_MODE_1B_t;

typedef struct S_IMA_MODE_1B {
        H_P_t       header;
        D_IMA_TYPE42_IM_L_t         data;
} S_IMA_MODE_1B_t;

typedef struct S_IEA_MODE_1B {
        H_P_t       header;
        D_IEA_TYPE80_IM_L_t         data;
} S_IEA_MODE_1B_t;



/*****************************************************************************/
/*****************************************************************************/
/* MODE 71 SV MONITOR                                                        */
/*****************************************************************************/
/*****************************************************************************/
typedef struct S_SVMONITOR_MODE_71 {
        H_P_t       header;
        D_TYPEF0_ESAS1_SV_MONITOR_t esas1_mon;
        D_TYPEF0_ESAS2_SV_MONITOR_t esas2_mon;
        D_TYPEF1_IMA_SV_MONITOR_t ima_mon;
        D_TYPEF1_IEA_SV_MONITOR_t iea_mon;
} S_SVMONITOR_MODE_71_t;


/*****************************************************************************/
/*****************************************************************************/
/* MODE 78 LMAG MONITOR                                                      */
/*****************************************************************************/
/*****************************************************************************/
typedef struct S_LMAGMONITOR_MODE_78 {
        H_P_t       header;
        D_TYPEF2_LMAG_MONITOR_t lmag_mon;
} S_LMAGMONITOR_MODE_78_t;

/*****************************************************************************/
/*****************************************************************************/
/* MODE 70 INTERNAL INFORMATION                                              */
/*****************************************************************************/
/*****************************************************************************/
typedef struct S_INTINF_MODE_70 {
        H_P_t       header;
        D_TYPEF3_INTERNAL_INFORMATION_t int_inf;
} S_INTINF_MODE_70_t;
```

Appendix - 3
dump_pbf.c

```
/*
		SELENE MAP PACE PBF Format Data
		Sample read program
	Created by Yoshifumi Saito on 2007/06/30 Ver. 0.0

*/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include <math.h>

#include "paceql_outputdata_070904.h"



/* DATA HEADER */
		H_P_t s_pace_header;
		size_t size_H_P_t=(sizeof(s_pace_header));
/* ESA DATA TYPE */
		D_ESAS1_TYPE00_t s_esas1_type00;
		size_t size_s_esas1_type00_t=(sizeof(s_esas1_type00));
		D_ESAS2_TYPE00_t s_esas2_type00;
		size_t size_s_esas2_type00_t=(sizeof(s_esas2_type00));
		D_ESAS1_TYPE01_t s_esas1_type01;
		size_t size_s_esas1_type01_t=(sizeof(s_esas1_type01));
		D_ESAS2_TYPE01_t s_esas2_type01;
		size_t size_s_esas2_type01_t=(sizeof(s_esas2_type01));
		D_ESAS1_TYPE02_t s_esas1_type02;
		size_t size_s_esas1_type02_t=(sizeof(s_esas1_type02));
		D_ESAS2_TYPE02_t s_esas2_type02;
		size_t size_s_esas2_type02_t=(sizeof(s_esas2_type02));
		D_ESAS1_TYPE03_t s_esas1_type03;
		size_t size_s_esas1_type03_t=(sizeof(s_esas1_type03));
		D_ESAS2_TYPE03_t s_esas2_type03;
		size_t size_s_esas2_type03_t=(sizeof(s_esas2_type03));
/* IMA DATA TYPE */
		D_IMA_TYPE40_t s_ima_type40;
		size_t size_s_ima_type40_t=(sizeof(s_ima_type40));
		D_IMA_TYPE41_t s_ima_type41;
		size_t size_s_ima_type41_t=(sizeof(s_ima_type41));
		D_IMA_TYPE42_t s_ima_type42;
		size_t size_s_ima_type42_t=(sizeof(s_ima_type42));
		D_IMA_TYPE43_t s_ima_type43;
		size_t size_s_ima_type43_t=(sizeof(s_ima_type43));
		D_IMA_TYPE44_t s_ima_type44;
		size_t size_s_ima_type44_t=(sizeof(s_ima_type44));
		D_IMA_TYPE45_t s_ima_type45;
		size_t size_s_ima_type45_t=(sizeof(s_ima_type45));
/* IEA DATA TYPE */
		D_IEA_TYPE80_t s_iea_type80;
		size_t size_s_iea_type80_t=(sizeof(s_iea_type80));
		D_IEA_TYPE81_t s_iea_type81;
		size_t size_s_iea_type81_t=(sizeof(s_iea_type81));
		D_IEA_TYPE82_t s_iea_type82;
		size_t size_s_iea_type82_t=(sizeof(s_iea_type82));

		FILE *fp_in;
		unsigned char c_in_head[1024];

int main(int argc, char **argv)
{
		char	fil_nme[500];

/* ************************** */
		FILE *fp_d;
		char dfname[100];
		int i,j,k,l;
		long today, time0, time1;
```

```
        long tmp[12][32][16][64];
/* ***************************** */

        if(argc != 5){
                fprintf(stderr, "¥n Usage: read_pbf input_file yyyymmdd, hhmmss1, hhmmss2, ¥n");
                fputs("¥n", stderr);
                exit(1);
        }

        today = atol(argv[2]);
        time0 = atol(argv[3]);
        time1 = atol(argv[4]);

        for (l = 0; l < 12; ++l)
                for (k = 0; k < 32; ++k)
                        for (j = 0; j < 16; ++j)
                                for (i = 0; i < 64; ++i)
                                        tmp[l][k][j][i] = 0;

        strcpy(fil_nme,argv[1]);
        //sprintf(dfname, "%s_%s_%s.dat", argv[2], argv[3], argv[4]);

/* INPUT DATA FILE OPEN */
         if (NULL == (fp_in=fopen(fil_nme,"rb")))
             {
             exit(-1);
             }

/* READ FILE HEADER DESCRIPTION */
         fread(c_in_head, 1024 , 1,fp_in);

/* FILE BINARY TYPE CHECK */
#ifdef PC
    if (c_in_head[1023] != 0xEE){
        printf("FILE_TYPE ERROR ¥n");
        exit(1);}
#else
    if (c_in_head[1023] != 0xDD){
        printf("FILE_TYPE ERROR ¥n");
        exit(1);}
#endif

/* PRINT FILE HEADER INFORMATION */
            printf("%s", c_in_head);



        while(1){
/* READ DATA HEADER */
                if (0 == fread(&s_pace_header, size_H_P_t, 1,fp_in))
                        break;

/* DATA MODE */
                switch (s_pace_header.type){
                        case 0x00:
                                printf("type:%x¥n", s_pace_header.type);

                                if(s_pace_header.sensor == 0){
                                        if (0 == fread(&s_esas1_type00, size_s_esas1_type00_t, 1,fp_in)) {
                                                printf ("FILE ERROR: No Data ¥n");
                                                break;
                                        }
                                        printf ("ESA-S1 DATA MODE %X DATA TYPE 00
time=%X%X%X event[0]=%X total_day=%X msec_day=%X ¥n",

        s_pace_header.mode,s_pace_header.timeH,s_pace_header.timeM,

        s_pace_header.timeL,s_esas1_type00.event[0],s_pace_header.t_date,s_pace_header.time_ms);

                                        }
```

```
                                        else if(s_pace_header.sensor == 1) {
                                                if (0 == fread(&s_esas2_type00, size_s_esas2_type00_t, 1,fp_in)) {
                                                        printf ("FILE ERROR: No Data ¥n");
                                                        break;
                                                }
                                                printf ("ESA-S2 DATA MODE %X DATA TYPE 00
time=%X%X%X event[0]=%X total_day=%X msec_day=%X ¥n",

        s_pace_header.mode,s_pace_header.timeH,s_pace_header.timeM,

        s_pace_header.timeL,s_esas2_type00.event[0],s_pace_header.t_date,s_pace_header.time_ms);
                                        }
                                        break;

                                case 0x01:
                                        if(s_pace_header.sensor == 0) {
                                                if (0 == fread(&s_esas1_type01, size_s_esas1_type01_t, 1,fp_in)) {
                                                        printf ("FILE ERROR: No Data ¥n");
                                                        break;
                                                }
                                                printf ("ESA-S1 DATA MODE %X DATA TYPE 01
time=%X%X%X event[0]=%X total_day=%X msec_day=%X ¥n",

        s_pace_header.mode,s_pace_header.timeH,s_pace_header.timeM,

        s_pace_header.timeL,s_esas1_type01.event[0],s_pace_header.t_date,s_pace_header.time_ms);
                                        }
                                        else if(s_pace_header.sensor == 1){
                                                if (0 == fread(&s_esas2_type01, size_s_esas2_type01_t, 1,fp_in)){
                                                        printf ("FILE ERROR: No Data ¥n");
                                                        break;
                                                }
                                                printf ("ESA-S2 DATA MODE %X DATA TYPE 01
time=%X%X%X event[0]=%X total_day=%X msec_day=%X ¥n",

        s_pace_header.mode,s_pace_header.timeH,s_pace_header.timeM,

        s_pace_header.timeL,s_esas2_type01.event[0],s_pace_header.t_date,s_pace_header.time_ms);
                                        }
                                        break;

                                case 0x02:
                                        if(s_pace_header.sensor == 0){
                                                if (0 == fread(&s_esas1_type02, size_s_esas1_type02_t, 1,fp_in)){
                                                        printf ("FILE ERROR: No Data ¥n");
                                                        break;
                                                }
                                                printf ("ESA-S1 DATA MODE %X DATA TYPE 02
time=%X%X%X event[0]=%X total_day=%X msec_day=%X ¥n",

        s_pace_header.mode,s_pace_header.timeH,s_pace_header.timeM,

        s_pace_header.timeL,s_esas1_type02.event[0],s_pace_header.t_date,s_pace_header.time_ms);
                                        }
                                        else if(s_pace_header.sensor == 1){
                                                if (0 == fread(&s_esas2_type02, size_s_esas2_type02_t, 1,fp_in)){
                                                        printf ("FILE ERROR: No Data ¥n");
                                                        break;
                                                }
                                                printf ("ESA-S2 DATA MODE %X DATA TYPE 02
time=%X%X%X event[0]=%X total_day=%X msec_day=%X ¥n",

        s_pace_header.mode,s_pace_header.timeH,s_pace_header.timeM,

        s_pace_header.timeL,s_esas2_type02.event[0],s_pace_header.t_date,s_pace_header.time_ms);
                                        }
                                        break;

                                case 0x03:
                                        if(s_pace_header.sensor == 0){
```

```
                                                if (0 == fread(&s_esas1_type03, size_s_esas1_type03_t, 1,fp_in)) {
                                                        printf ("FILE ERROR: No Data ¥n");
                                                        break;
                                                }
                                                printf ("ESA-S1 DATA MODE %X DATA TYPE 03 time=%X%X%X event[0]=%X total_day=%X msec_day=%X ¥n",

        s_pace_header.mode,s_pace_header.timeH,s_pace_header.timeM,

        s_pace_header.timeL,s_esas1_type03.event[0],s_pace_header.t_date,s_pace_header.time_ms);
                                        }
                                        else if(s_pace_header.sensor == 1){
                                                if (0 == fread(&s_esas2_type03, size_s_esas2_type03_t, 1,fp_in)){
                                                        printf ("FILE ERROR: No Data ¥n");
                                                        break;
                                                }
                                                printf ("ESA-S2 DATA MODE %X DATA TYPE 03 time=%X%X%X event[0]=%X total_day=%X msec_day=%X ¥n",

        s_pace_header.mode,s_pace_header.timeH,s_pace_header.timeM,

        s_pace_header.timeL,s_esas2_type03.event[0],s_pace_header.t_date,s_pace_header.time_ms);
                                        }
                                break;
                                case 0x40:
                                        if (0 == fread(&s_ima_type40, size_s_ima_type40_t, 1,fp_in)){
                                                printf ("FILE ERROR: No Data ¥n");
                                                break;
                                        }
                                        printf ("IMA DATA MODE %X DATA TYPE 40 time=%X%X%X event[0][0]=%X total_day=%X msec_day=%X ¥n",
                                                s_pace_header.mode,s_pace_header.timeH,s_pace_header.timeM,

        s_pace_header.timeL,s_ima_type40.event[0][0],s_pace_header.t_date,s_pace_header.time_ms);
                                        break;
                                case 0x41:
                                        if (0 == fread(&s_ima_type41, size_s_ima_type41_t, 1,fp_in)){
                                                printf ("FILE ERROR: No Data ¥n");
                                                break;
                                        }
                                        printf ("IMA DATA MODE %X DATA TYPE 41 time=%X%X%X event[0][0]=%X total_day=%X msec_day=%X ¥n",
                                                s_pace_header.mode,s_pace_header.timeH,s_pace_header.timeM,

        s_pace_header.timeL,s_ima_type41.event[0][0],s_pace_header.t_date,s_pace_header.time_ms);
                                        break;

                                case 0x42:
                                        if (0 == fread(&s_ima_type42, size_s_ima_type42_t, 1,fp_in)){
                                                printf ("FILE ERROR: No Data ¥n");
                                                break;
                                        }
                                        printf ("IMA DATA MODE %X DATA TYPE 42 time=%X%X%X event[0][0]=%X total_day=%X msec_day=%X ¥n",
                                                s_pace_header.mode,s_pace_header.timeH,s_pace_header.timeM,

        s_pace_header.timeL,s_ima_type42.event[0][0],s_pace_header.t_date,s_pace_header.time_ms);
                                        break;

                                case 0x43:
                                        if (0 == fread(&s_ima_type43, size_s_ima_type43_t, 1,fp_in)){
                                                printf ("FILE ERROR: No Data ¥n");
                                                break;
                                        }
                                        printf ("IMA DATA MODE %X DATA TYPE 43 time=%X%X%X event[0][0]=%X total_day=%X msec_day=%X ¥n",
                                                s_pace_header.mode,s_pace_header.timeH,s_pace_header.timeM,

        s_pace_header.timeL,s_ima_type43.event[0][0],s_pace_header.t_date,s_pace_header.time_ms);
                                        break;
```

```
                case 0x44:
                        if (0 == fread(&s_ima_type44, size_s_ima_type44_t, 1,fp_in)){
                                printf ("FILE ERROR: No Data ¥n");
                                break;
                        }
                        printf ("IMA DATA MODE %X DATA TYPE 44 time=%X%X%X
event[0][0]=%X total_day=%X msec_day=%X ¥n",
                                s_pace_header.mode,s_pace_header.timeH,s_pace_header.timeM,
        s_pace_header.timeL,s_ima_type44.event[0][0],s_pace_header.t_date,s_pace_header.time_ms);
                        break;

                case 0x45:
                        if (0 == fread(&s_ima_type45, size_s_ima_type45_t, 1,fp_in)){
                                printf ("FILE ERROR: No Data ¥n");
                                break;
                        }
                        printf ("IMA DATA MODE %X DATA TYPE 45 time=%X%X%X
event[0][0]=%X total_day=%X msec_day=%X ¥n",
                                s_pace_header.mode,s_pace_header.timeH,s_pace_header.timeM,
        s_pace_header.timeL,s_ima_type45.event[0][0],s_pace_header.t_date,s_pace_header.time_ms);
                        break;

                case 0x80:
                        if (0 == fread(&s_iea_type80, size_s_iea_type80_t, 1,fp_in)){
                                printf ("FILE ERROR: No Data ¥n");
                                break;
                        }
                        printf ("IEA DATA MODE %X DATA TYPE 80 time=%X%X%X event[0]=%X
total_day=%X msec_day=%X ¥n",
                                s_pace_header.mode,s_pace_header.timeH,s_pace_header.timeM,
        s_pace_header.timeL,s_iea_type80.event[0],s_pace_header.t_date,s_pace_header.time_ms);
                        break;

                case 0x81:
                        if (0 == fread(&s_iea_type81, size_s_iea_type81_t, 1,fp_in)) {
                                printf ("FILE ERROR: No Data ¥n");
                                break;
                        }

                        //printf ("IEA DATA MODE %X DATA TYPE 81 time=%X%X%X event[0]=%X
total_day=%X msec_day=%X ¥n",
                        //              s_pace_header.mode,s_pace_header.timeH,s_pace_header.timeM,
                        //
        s_pace_header.timeL,s_iea_type81.event[0],s_pace_header.t_date,s_pace_header.time_ms);

                        /* dump*/

                        //if (NULL == (fp_d = fopen(dfname, "a"))) {
                        //              printf ("    --- OUTPUT FILE cannot be open.¥n");
                        //              exit(1);
                        //}
                        //if (s_pace_header.yyyymmdd == today &&
                        //              s_pace_header.hhmmss >= 180000      &&
                        //              s_pace_header.hhmmss <= 181000) {
                                //fprintf(fp_d, "yyyymmdd:%ld hhmmss: %ld¥n",
                                //              s_pace_header.yyyymmdd, s_pace_header.hhmmss);

                                //for (k = 0; k < 16; ++k)
                                //              fprintf(fp_d, "%d ", s_iea_type81.event[k]);
                                //fprintf(fp_d, "¥n");
                                //fprintf(fp_d, "%ld ", s_pace_header.hhmmss);
                        //              for (k = 0; k < 32; ++k)
                        //                      for (j = 0; j < 16; ++j)
                        //                              for (i = 0; i < 64; ++i) {
                        //                                      //fprintf(fp_d,         "%d         ",
s_iea_type81.cnt[k][j][i]);
```

```
                                        //                                      tmp[0][k][j][i]                 +=
s_iea_type81.cnt[k][j][i];
                                        //                              }

                                                //fprintf(fp_d, "¥n");

                                        //}
                                        //fclose(fp_d);
                                        for (l = 0; l < 6; ++l)
                                                if (s_pace_header.yyyymmdd == today &&
                                                        s_pace_header.hhmmss >= 180000 +l*1000     &&
                                                        s_pace_header.hhmmss <= 180000 +(l+1)*1000)
                                                        for (k = 0; k < 32; ++k)
                                                                for (j = 0; j < 16; ++j)
                                                                        for  (i  =  0;  i  <  64;
++i)

        tmp[l][k][j][i] += s_iea_type81.cnt[k][j][i];

                                        for (l = 0; l < 6; ++l)
                                                if (s_pace_header.yyyymmdd == today &&
                                                        s_pace_header.hhmmss >= 190000 +l*1000     &&
                                                        s_pace_header.hhmmss <= 190000 +(l+1)*1000)
                                                        for (k = 0; k < 32; ++k)
                                                                for (j = 0; j < 16; ++j)
                                                                        for  (i  =  0;  i  <  64;
++i)

        tmp[l+6][k][j][i] += s_iea_type81.cnt[k][j][i];
                                        break;

                                case 0x82:
                                        if (0 == fread(&s_iea_type82, size_s_iea_type82_t, 1,fp_in)) {
                                                printf ("FILE ERROR: No Data ¥n");
                                                break;
                                        }
                                        printf ("IEA DATA MODE %X DATA TYPE 82 time=%X%X%X event[0]=%X
total_day=%X msec_day=%X ¥n",
                                                s_pace_header.mode,s_pace_header.timeH,s_pace_header.timeM,

        s_pace_header.timeL,s_iea_type82.event[0],s_pace_header.t_date,s_pace_header.time_ms);
                                        break;

                                default:
                                        break;
                        } /* switch type */



        } /* while(1) */

        for (l = 0; l < 12; ++l) {
                sprintf(dfname, "%02d_disp.dat", l);
                if (NULL == (fp_d = fopen(dfname, "w"))) {
                        printf ("    --- OUTPUT FILE cannot be open.¥n");
                        exit(1);
                }
                for (k = 0; k < 32; ++k) {
                        for (j = 0; j < 16; ++j)
                                for (i = 0; i < 64; ++i)
                                        fprintf(fp_d, "%d ", tmp[l][k][j][i]);

                        fprintf(fp_d, "¥n");
                }
                fclose(fp_d);
        }


    fclose(fp_in);
    exit(0);
```

}