SMOS R/W Library Software User Manual

Authors: R. Onrubia, R. Oliva
Zenithal Blue technologies
Date: 18/04/23

DOCUMENT SIGNATURE PAGE

	Name & Function	Date	Signature
Prepared by:	R. Onrubia *Project Engineer*	22/06/2023	
Checked by:	R. Oliva *Project Manager*	22/06/2023	

TABLE OF CONTENTS

1 Introduction

This document is the software user manual for the Matlab SMOS universal Read and Write Library (SMOSRWLib) implemented by Zenithal Blue Technologies (ZBT) for all DBL, HDR and EEF products types and versions. To do so, this software uses product schemas in order to avoid updating the reader and writer code every time a new product is released or updated.

This document first explains under which license it is distributed, and the system requirements. Next follows the installation procedure and the schema update procedure, which include the authenticity validation for both procedures. After these points, the user can find the commands to read and write any kind of DBL, HDR and EEF products. The document explains then how the product data is given to the user. Last, a list of typical reading times is shown, so the user can know beforehand if the products typically have long or short reading times.

2 Reference Documents

[RD1] J. Barbosa, G. Lopes, M. Zapata, J. Ortega, M. Rodriguez, “SMOS Level 1 and Auxiliary Data Products Specifications”, INDRA Sistemas S.A., Madrid, Spain, SO-TN-IDR-GS-0005, 09/12/2021. [Online]. Available: https://earth.esa.int/eogateway/documents/20142/37627/SMOS-L1-Aux-Data-Product-Specification.pdf [Accessed: 25/05/2023]

[RD1] B. Bengoa, M. Zapata, J. Ortega, M. Rodriguez, “SMOS Level 2 and Auxiliary Data Products Specifications”, INDRA Sistemas S.A., Madrid, Spain, SO-TN-IDR-GS-0006, 09/12/2021. [Online]. Available: https://earth.esa.int/eogateway/documents/20142/0/SMOS-L2-Aux-Data-Product-Specification.pdf [Accessed: 25/05/2023]

3 License considerations

smosRWLib is distributed under GNU Public License version 3. The license file can be found at the root folder (LICENSE.TXT).

4 System Requirements

smosRWLib requires:

Software	Version
Operative System	Tested with Windows 10, Ubuntu 18.04
Matlab	Tested with:

	2016b (Windows 10 and Ubuntu 18.04), 2020b (Windows 10) 2022b (Windows 10)
Matlab toolboxes	(none)

5 Installation

5.1 Unzip the software to a desired location

The software is delivered in the file smosRWLib_[*VERSION*].zip. To certificate its authenticity, the smosRWLib_[*VERSION*].zip.md5 128-bits MD5 hash signature file can be downloaded. This file contains the MD5 hash number and the number of bytes, and the filename of the software release smosRWLib_[*VERSION*].zip. *VERSION* stands for the software release version.

In order to validate the software authenticity, the user must compute the MD5 of the downloaded file smosRWLib_[*VERSION*].zip. To do so, the user must run the following command in the Windows command prompt:

certutil -hashfile smosRWLib_[*VERSION*].zip MD5

or the following command in a Linux terminal:

md5sum smosRWLib_[*VERSION*].zip

Then, the user must compare the obtained MD5 hash number with the one in smosRWLib_[*VERSION*].zip.md5. If the MD5 signature matches, the authenticity is validated.

Unzip the provided smosRWLib_[*VERSION*].zip file to a desired location. The SMOSRWLib main folder should contain:

Folder	Content
examples	Folder containing a EEF, HDR and DBL product examples to read.
functions	Folder containing the functions used to read and write any product.
preprocessedSchemas	Folder containing all the preprocessed schemas in MAT format.
File	**Content**
Example.m	Code example to read and write the EEF, HDR or DBL files from the Examples folder.
LICENSE.TXT	License file

README.md	Readme file.
smosReader.m/smosReader.p	Function to read any SMOS product
smosWriter.m/smosWriter.p	Function to write any SMOS product

5.2 Add smosRWLib root folder to the matlab path

The smosRWLib root folder must be added to the matlab path. To do so, in Matlab execute:

addpath(*smosRWLibPath*)

Where *smosRWLibPath* is the path to the smosRWLib root folder. The code itself adds the required subfolders to the path. If an error occurs and some functions are not found by smosRWLib, the user can solve it by executing:

addpath(genpath(*smosRWLibPath*))

5.3 Schema update

The code is delivered with a set of pre-processed schemas in the "preprocessedSchemas" folder. If newer product versions are released, they will be released in the file smosRWLib_schemas_[*VERSION*].zip, and the 128-bit MD5 hash signature will be in smosRWLib_schemas_[*VERSION*].zip.md5.

To update the schemas, first validate its authenticity following the procedure in 5.1. Then unzip the file smosRWLib_schemas_[*VERSION*].zip file in the "preprocessedSchemas" folder and overwrite all files.

6 smosRWLib usage

This section explains how to use smosRWLib. All commands here must be executed in the matlab command line.

6.1 Product reader

In order to read any version of any DBL, EEF or HDR product, the user must execute:

outputData = smosReader(filename);

where filename is the path to the desired DBL, EEF, or HDR file with the filename and extension in it, and outputData is a structure containing all the data in the DBL file as explained in Section 0. Products must have been unzipped prior to be read. The reader will automatically detect the product type and version. For this purpose, DBL files will require the corresponding HDR file.

If the product will be modified later and written again to a file, the user must execute:

[outputData, dataStructure] = smosReader(filename);

where dataStructure has information on file is structured internally (see Section 7.2 for an example).

The read data is returned in the original data type (not converted to double), and no scaling factors are applied. For instance, the user must first convert to double the incidence angle in L1C products and then divide it by 2^16 and multiply it by 90; this is not conducted by smosRWLib, since it is not included in the schemas but in the product specifications [RD1, RD2], hence must be done by the user.

6.2 Product writer

Once a product has been read and modified, to save it again to disk the user must use:

smosWriter(modifiedOutputData, filename, dataStructure, headerText);

where modifiedOutputData is the variable that contains the product data, dataStructure is the output variable obtained from Section 6.1, and filename is the name of the file to be written (if exists, it will overwrite it). The headerText variable is optional, it only applies to HDR and EEF files, and it is the XML optional preamble. It often is

headerText = ‘<?xml version=""1.0"" encoding=""UTF-8""standalone=""yes""?>’;

If not specified, no preamble will be added to the file.

The product writer does not check the consistency of the modifications. It is strongly recommended to first read an operational product, modify it, and write it to a file instead of creating a product from scratch; that is, creating each field and subfield manually.

The product writer does not de-scale the parameters and these are expected to be in the format specified in the product specifications [RD1, RD2], but the data type will be converted. Keeping with the example in 6.1, the incidence angle in L1c products must be integer numbers in the range from 0 to (2^16)-1.

This integer value can be stored in a double datatype variable, since it will be converted to a 16 bit unsigned integer in the writer.

6.3 Code example

The software includes an example script named Example.m that allows the user to choose between reading a DBL, HDR or EEF files that are allocated in the Examples folder, and then write it again in the “output” folder.

7 Software considerations

This section describes how data is kept in Matlab variables.

7.1 Matlab name conventions and data types

Matlab variable names do not follow the same rules than C/C++, where most SMOS products are generated. Therefore, some variable names had to be modified. This is carried out automatically inside smosRWLib and always follows the following rules:

- “-“ (hyphen) is not valid in Matlab names. They are replaced by “_” (underscore).
- “.” (dot) is not valid in Matlab names. They are replaced by “_” (underscore).
- Matlab names cannot start with a number. In this case, a “v_” (letter v followed by underscore) is added to the beginning of the variable name. In order to speed up the reading and writing of some products, some data structures have been bypassed, and this affects the variable naming. Please, check Section 7.4.

Note that this also applies to all DBL, HDR and EEF files. When Writing back HDR or EEF data to a file, the original names will be kept.

7.2 Data structure

This software will generate data structs containing all the product information. Fields and subfields follow the structure specified in the schemas, except the cases specified in Section 7.4 to speed up the reading process.

7.2.1 DBL Products

The struct only contains fields whose content are the data defined in the schemas. Figure 7-1 shows an example of a MIR SCLF1C DBL product. The returned struct has two fields, “Swath_Snapshot_List”, and “Temp_Swath_Full”. “Swath_Snapshot_List” contains multiple fields, two of which are structures: “Snapshot_Time” and “Quality_Information”. “Temp_Swath_Full” also contains multiple fields from which one is a struct: “BT_Data”. All structs are shown in the figure.

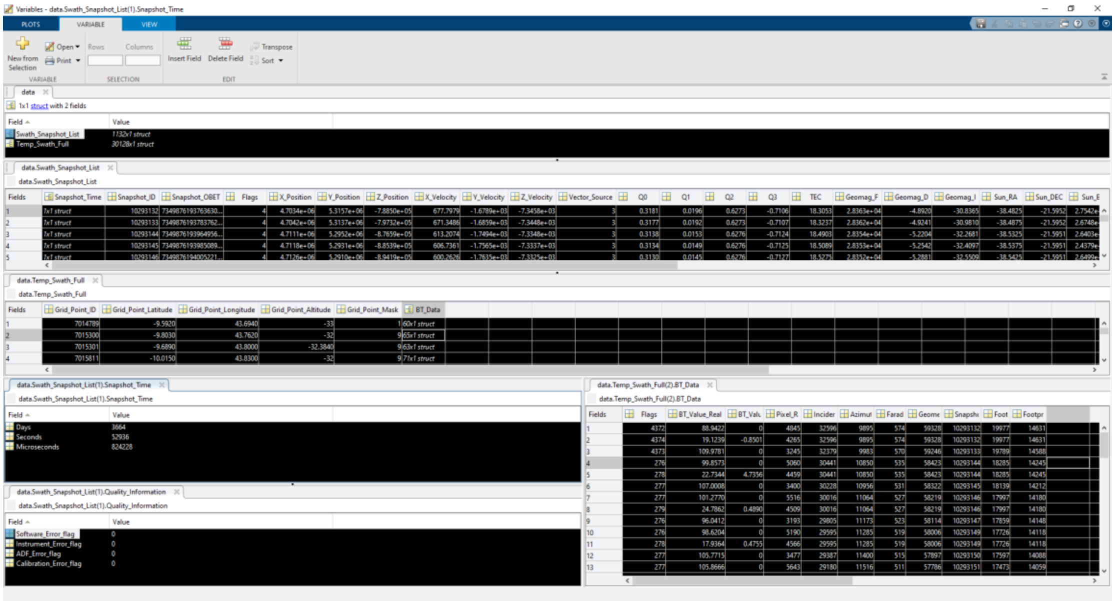

Figure 7-1 Example of a MIR SCLF1C DBL product read.

7.2.2 EEF and HDR Products

HDR data is structured in a similar way than DBL data, but also contains certain attributes, which are saved in variables with the same name but followed by “__attributes” (note the two underscores at the beginning). For example, the “Earth_Explorer_Header” contains two variables, “Main_Product_Header” and “Specific_Product_Header”. The last one contains a struct called “Orbit_Information” that has variables “X_Position”, “Y_Position”, “Z_Position”, “X_Velocity”, … The additional variable “X_Position__attributes” contains “unit=”m””, which indicates the attribute “unit” with value “m” of the “X_Position”.

In the case of arrays with a variable number of elements, a variable “__attributes” will contain the text “count=…”, or ”num=…” indicating how many elements the array has. If the user modifies the array by adding or removing elements, the user must modify accordingly the count value.

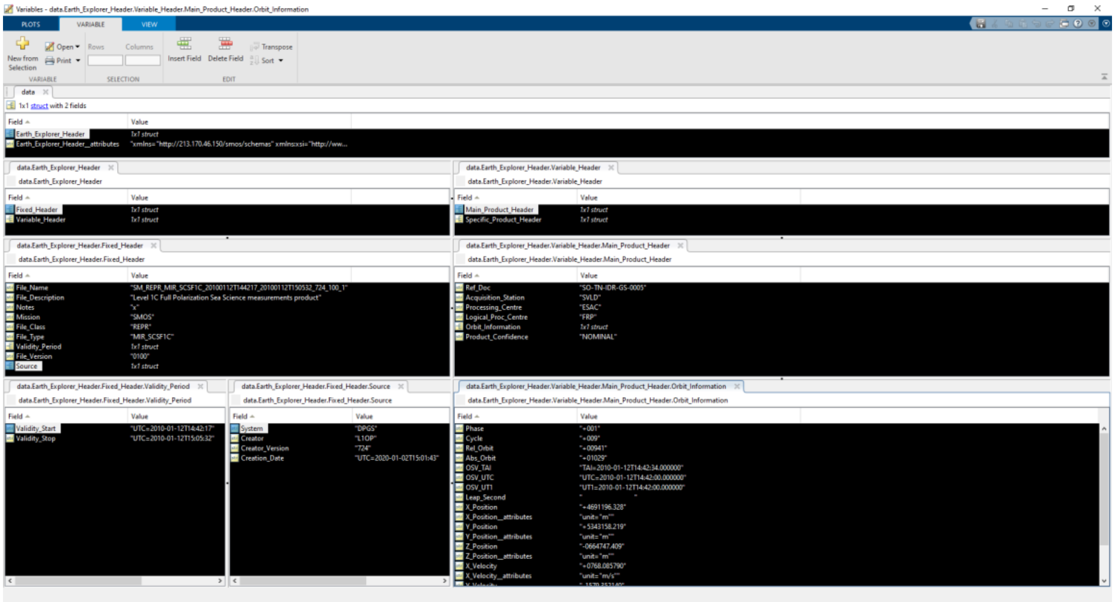

Figure 7-2 Example of a MIR SCLF1C HDR product read.

7.3 Considerations on the processing time

The software reads the SMOS products based on the schema definition. These definitions might not be optimized for the reading in Matlab for several reasons:

- A 3-Dimensional matrix of NxNxN elements can be defined in the schema as a 3-Dimensional array, or as vector of N elements, each of one is an array of N elements, each of one is an arrays of N elements. Both definitions describe the same structure in disk in the same exact order. The first case will be read all at once (1 measurement), whereas the second one will be read in a 2-Level for loop of N elements each one (N^(3-1) measurements). A M dimensional array of N elements will take N^(M-1) readings. If the dimension number increases, or the N number is large, the reading time can be dramatically large.
- When the array elements are complex structures, they must be read in sequential readings. For example, an element composed by (in order) a struct, a double, a struct, and a double will require 4 readings, whereas if it was stored as a struct, a struct, a double and a double could be read in 3 readings (the two last doubles could be read at the same time). L sequential readings in a M-dimensional array of N elements in each direction will take L·(N^(M-1)) readings.
- Some structs only contain 1 element. The code does not have the capability to optimize the readings. In this case the last level array cannot be read as one array but as N readings of single elements, thus the number of readings becomes N^M.
- The use of structs in Matlab adds a huge memory overhead compared with respect to using arrays and matrixes. For instance, the gridpoint list

in L1C products contains a field BTData, which contains multiple subfields. None of these subfields have sub-subfields, they all have numerical data. By converting all subfields in BTData to a matrix the memory usage would be reduced in a factor of 14. If not enough memory is available in the system, Matlab works with hard drive memory as RAM memory, and the process will slow down. Large and strongly nested products might reach this point.

For all these reasons some products might take long reading and writing times (writing times are always shorter than reading ones). Some schemas have been tweaked in order to speed up the reading and writing times as explained in section 7.4: AUX_SSSCLI, AUX_SSS, MIR_OSDAP2, MIR_OSUDP2, and MIR_SMUDP2. An indicative table for multiple products and auxiliary DBL files with their size and reading time can be found in Annex I: Processing Time.

7.4 List of bypassed products

In order to solve the issue mentioned in section 7.3 in some products, it is possible to bypass some fields and subfields in order to reduce the number of iterations when a loop is calling another loop.

For instance, imagine a vector of N elements “field” that has two elements, "subfield1” and “subfield2”. If each subfield has 1 subsubfield itself that are both basic data types (integers, doubles, char, etc…)., then we have:

Field.subfield1.subsubfield1
Field.subfield2.subsubfield1

When reading “Field”, the reader will see that it has two fields and will do a reading for each subfield, and this will be repeated N times (2N separate disk accesses). Instead, if we merge the subsubfields as:

Field.subfield1__subsubfield1
Field.subfield2__subsubfield1

Then, the reader will see a N-element array with 2 elements each one, which can be read sequentially in a single disk access.

Another possible bypass is to separate array variables into separate variables. For instance:

Field[N].subfield1
Field[N].subfield2

Become:

Field__1__subfield1
Field__1__subfield2
Field__2__subfield1
Field__2__subfield2
…
Field__N__subfield1
Field__N__subfield2

These methods have improved the reading time up to 150 times in some products, and made practical to read certain products for any processing in Matlab. Not all schemas are tweakable, or not in all cases the improvement is noticeable.

The following sections explain the bypasses implemented in order to speed up the reading of some of the most time-consuming products, and how variable names and structures have been changed.

Note that only the last version of the schema of the following cases have been tweaked.

7.4.1 AUX_SSSCLI

Tweaked file: DBL_SM_XXXX_AUX_SSSCLI_0002.binXschema.xml

The original Data_Set_Climatology_LUT_A field has the following structure:

Data_Set_Climatology_LUT_A.Gridpoint_ID_A
Data_Set_Climatology_LUT_A.Climatology_A.SSS_clim[34]

This has been bypassed to:

Data_Set_Climatology_LUT_A.Gridpoint_ID_A
Data_Set_Climatology_LUT_A.Climatology_A__SSS_clim[34]

This also applies to Data_Set_Climatology_LUT_D.

7.4.2 AUX_SSS

Tweaked file: DBL_SM_XXXX_AUX_SSS____0400.binXschema.xml

The original Data_Set_Climatology_LUT_A has the following fields:

Data_Set_Climatology_LUT_A.Grid_Point_ID_A

Data_Set_Climatology_LUT_A.Climatology_A[12].SSSa
Data_Set_Climatology_LUT_A.Climatology_A[12].SSSb
Data_Set_Climatology_LUT_A.Climatology_A[12].SSSa_quality
Data_Set_Climatology_LUT_A.Climatology_A[12].SSSb_quality

This has been bypassed to:

Data_Set_Climatology_LUT_A.Grid_Point_ID_A
Data_Set_Climatology_LUT_A.Climatology_A__1__SSSa
Data_Set_Climatology_LUT_A.Climatology_A__1__SSSb
Data_Set_Climatology_LUT_A.Climatology_A__1__SSSa_quality
Data_Set_Climatology_LUT_A.Climatology_A__1__SSSb_quality
Data_Set_Climatology_LUT_A.Climatology_A__2__SSSa
Data_Set_Climatology_LUT_A.Climatology_A__2__SSSb
Data_Set_Climatology_LUT_A.Climatology_A__2__SSSa_quality
Data_Set_Climatology_LUT_A.Climatology_A__2__SSSb_quality

…

Data_Set_Climatology_LUT_A.Climatology_A__12__SSSa
Data_Set_Climatology_LUT_A.Climatology_A__12__SSSb
Data_Set_Climatology_LUT_A.Climatology_A__12__SSSa_quality
Data_Set_Climatology_LUT_A.Climatology_A__12__SSSb_quality

This also applies to Data_Set_Climatology_LUT_D.

7.4.3 MIR_OSDAP2

Tweaked file: DBL_SM_XXXX_MIR_OSDAP2_0402.binXschema.xml

This product has two different bypasses.

First, the original Available_Data inside SSS_MEASUREMENT_ANALYSIS originally contained:

Available_Data.Measurement_Data.Snapshot_ID
Available_Data.Measurement_Data.xi
Available_Data.Measurement_Data.eta
Available_Data.Measurement_Data.Meas_Flag

Available_Data.Diff_TBs.Diff_TB
Available_Data.Diff_TBs.Tb_gal_H
Available_Data.Diff_TBs.Tb_gal_V

This has been bypassed to:

Available_Data__Measurement_Data.Snapshot_ID
Available_Data__Measurement_Data.xi
Available_Data__Measurement_Data.eta
Available_Data__Measurement_Data.Meas_Flag

Available_Data.Diff_TBs__Diff_TB
Available_Data.Diff_TBs__Tb_gal_H
Available_Data.Diff_TBs__Tb_gal_V

Second, the original SSS_SWATH_ANALYSIS contained two fields, Grid_Point_Descriptors, Geophysical_Parameters_Prior, and Geophysical_Parameters_Post have been bypassed. Before, the SSS_SWATH_ANAYLISIS was structured as:

SSS_SWATH_ANALYSIS.Grid_Point_ID
SSS_SWATH_ANALYSIS.Latitude
SSS_SWATH_ANALYSIS.Longitude
SSS_SWATH_ANALYSIS.Grid_Point_Descriptors.(subfields)
SSS_SWATH_ANALYSIS.Geophysical_Parameters_Prior.(subfields)
SSS_SWATH_ANALYSIS.Geophysical_Parameters_Post.(subfields)

Now, the fields inside SSS_SWATH_ANAYLISIS are:

SSS_SWATH_ANALYSIS.Grid_Point_ID
SSS_SWATH_ANALYSIS.Latitude
SSS_SWATH_ANALYSIS.Longitude
SSS_SWATH_ANALYSIS.Grid_Point_Descriptors__(subfields)
SSS_SWATH_ANALYSIS.Geophysical_Parameters_Prior__(subfields)
SSS_SWATH_ANALYSIS.Geophysical_Parameters_Post__(subfields)

7.4.4 MIR_OSUDP2

Tweaked file: DBL_SM_XXXX_MIR_OSUDP2_0401.binXschema.xml

The following variables have been bypassed from:

SSS_SWATH.Geophysical_Parameters_Data.(fields)
SSS_SWATH.Product_Confidence_Descriptor.(fields)

To:

SSS_SWATH.Geophysical_Parameters_Data__(fields)
SSS_SWATH.Product_Confidence_Descriptor__(fields)

7.4.5 MIR_SMDUP2

Tweaked file: DBL_SM_XXXX_MIR_SMUDP2_0400.binXschema.xml

The following variables have been bypassed from:

SM_SWATH.Mean_Acq_Time.(fields)
SM_SWATH.Retrieval_Results_Data.(fields)
SM_SWATH.Confidence_Descriptors_Data.(fields)
SM_SWATH.Science_Descriptors_Data.(fields)
SM_SWATH.Processing_Descriptors_Data.(fields)
SM_SWATH.DGG_Current_Data.(fields)

To:

SM_SWATH.Mean_Acq_Time__(fields)
SM_SWATH.Retrieval_Results_Data__(fields)
SM_SWATH.Confidence_Descriptors_Data__(fields)
SM_SWATH.Science_Descriptors_Data__(fields)
SM_SWATH.Processing_Descriptors_Data__(fields)
SM_SWATH.DGG_Current_Data__(fields)

8 Annex I: Processing Time

Table I shows the reading and writing times for some v724 DBL products and auxiliary files. The experiment was conducted with a NMVe M.2 SSD and Matlab 2022b. The time has been measured with the products in section 7.4 bypassed. In grey, the products requested in the WP. In dark grey, the largest product found for that type of requested product in the WP.

Table I Reading and Writing time for some 724 DBL products and auxiliary files with Matlab 2022b

File Type	Version	Size		Reading Time [s]
AUX_BNDLST	300_003_3	13,03112	Mb	2,525
AUX_BNDLST	303_004_3	1,6266708	Mb	0,43
AUX_BNDLST	303_002_3	2,3531685	Mb	0,469
AUX_BNDLST	303_004_3	1,627182	Mb	0,4
AUX_BNDLST	303_004_3	6,0496254	Mb	0,966
AUX_BNDLST	303_004_3	13,283119	Mb	2,48
AUX_BSCAT	300_003_3	812,84375	kb	0,276
AUX_DFFFRA	001_007_3	640,59343	Mb	1881,581
AUX_DFFLAI	600_001_3	216,82907	Mb	64,15
AUX_DFFLMX	001_006_3	40,260595	Mb	38,433
AUX_DFFSNO	201_001_5	40,260595	Mb	20,248

AUX_DFFSOI	001_003_3	499,33864	Mb	370,575
AUX_DFFXYZ	001_003_3	428,71125	Mb	56,951
AUX_DGG	300_003_3	40,000267	Mb	5,373
AUX_DGGFLO	600_100_1	35,000164	Mb	8,686
AUX_DGGRFI	600_100_1	90,000374	Mb	15,616
AUX_DGGROU	600_100_1	65,000278	Mb	21,498
AUX_DGGTFO	600_100_1	65,000278	Mb	21,491
AUX_DGGTLV	600_100_1	65,000278	Mb	21,823
AUX_DGGXYZ	001_004_3	40,000267	Mb	5,969
AUX_DISTAN	001_011_3	37,500029	Mb	6,887
AUX_DTBCUR	700_001_1	117,43477	Mb	27,15
AUX_DTBCUR	699_100_1	117,43477	Mb	31,582
AUX_DTBXY	699_100_1	23,119062	Mb	221,707
AUX_DTBXY	699_200_1	107,65217	Mb	771,289
AUX_ECMCDF	001_003_3	4,8334808	Mb	2,997
AUX_ECMCDF	001_003_3	4,8334808	Mb	2,595
AUX_ECMWF	400_001_3	79,472204	Mb	14,191
AUX_ECOLAI	305_006_3	1,2808141	Gb	43,275
AUX_FOAM	001_011_3	84,504696	Mb	2,489
AUX_FRSNEL	720_001_3	1,1723633	Mb	0,428
AUX_GAL2OS	001_016_3	526,96549	Mb	8,717
AUX_GALAXY	300_004_3	15,853287	Mb	0,607
AUX_GALNIR	300_003_3	7,9266434	Mb	0,437
AUX_GAL_OS	001_011_3	27,751514	Mb	0,693
AUX_GAL_SM	001_003_3	7,9266434	Mb	0,5
AUX_LSMASK	300_003_3	15,000172	Mb	4,092
AUX_MASK	300_002_3	12,500162	Mb	2,814
AUX_MN_WEF	001_002_3	25,748047	kb	0,292
AUX_MOONT	300_002_3	32	b	0,287
AUX_OTT1D	699_100_1	261,03906	kb	0,471
AUX_OTT1F	699_100_1	1,0166855	Mb	0,533
AUX_OTT2D	699_100_1	261,03906	kb	0,438
AUX_OTT2F	699_100_1	1,0166855	Mb	0,47
AUX_OTT3D	699_100_1	261,03906	kb	0,427
AUX_OTT3F	699_100_1	1,0166855	Mb	0,498
AUX_PATT	720_004_3	1018,2118	Mb	42,453
AUX_RFI	300_003_3	12,500162	Mb	2,981
AUX_RGHNS1	001_016_3	1,0714302	Mb	0,469
AUX_RGHNS2	001_013_3	71,341988	Mb	1,549
AUX_RGHNS3	001_016_3	190,51129	Mb	3,184
AUX_SGLINT	001_012_3	319,17489	Mb	9,698
AUX_SOIL_P	001_002_3	106,79538	Mb	55,051
AUX_SSS	001_014_3	380,00029	Mb	815,931

AUX_SSSCLI	001_002_3	234,73759	Mb	5,43
AUX_SUNT	300_002_3	32	b	2,141
AUX_SUN_BT	001_001_3	45,648438	kb	0,333
AUX_VTEC_C	311_001_3	151,63672	kb	2,402
AUX_VTEC_P	311_001_3	151,63672	kb	2,78
AUX_VTEC_R	320_001_3	151,63672	kb	2,506
AUX_WEF	001_003_3	16,748047	kb	0,358
MIR_AFWD1A	720_001_3	5,2543535	Mb	1,635
MIR_ANIR1A	720_001_3	5,5380859	kb	0,603
MIR_BWLD1C	724_100_1	258,61914	kb	13,416
MIR_BWLF1C	724_100_1	1,6902714	Mb	53,919
MIR_BWSD1C	724_100_1	611,39062	kb	30,866
MIR_BWSF1C	724_100_1	3,1022034	Mb	98,754
MIR_CRSD1A	724_100_1	640,75781	kb	0,867
MIR_CSTD1A	724_100_1	524,26855	kb	0,466
MIR_FTTD	730_001_3	220,70898	kb	0,501
MIR_FTTF	730_001_3	375,53906	kb	0,517
MIR_GMATD	600_007_3	18,313597	Gb	893,595
MIR_JMATD	600_009_3	1,3305198	Gb	62,498
MIR_OSDAP2	699_100_1	52,91247	Mb	67,357
MIR_OSDAP2	699_200_1	388,41801	Mb	352,527
MIR_OSUDP2	700_100_1	6,22579	Mb	2,548
MIR_OSUDP2	700_200_1	25,301603	Mb	6,713
MIR_SCLD1C	724_100_1	14,536347	Mb	31,131
MIR_SCLF1C	724_100_1	112,4705	Mb	104,38
MIR_SCSD1C	724_100_1	35,545432	Mb	56,452
MIR_SCSF1C	724_100_1	197,89247	Mb	183,937
MIR_SCSF1C	724_300_1	533,26562	Mb	462,36
MIR_SC_D1A	724_100_1	14,058028	Mb	3,753
MIR_SC_D1B	724_100_1	7,4974442	Mb	3,745
MIR_SC_F1A	724_100_1	92,20768	Mb	18,283
MIR_SC_F1B	724_100_1	49,003426	Mb	21,516
MIR_SMDAP2	700_001_1	221,98973	Mb	808,428
MIR_SMDAP2	700_100_1	76,898928	Mb	302,699
MIR_SMUDP2	700_100_1	9,5618315	Mb	3,492
MIR_TARD1A	724_100_1	7,3733768	Mb	1,844
MIR_TARD1B	724_100_1	3,9323807	Mb	2,156
MIR_TARF1A	724_100_1	6,0364466	Mb	1,529
MIR_TARF1B	724_100_1	3,1977615	Mb	1,806
MIR_UAVD1A	724_100_1	674,12109	kb	1,155
TLM_MIRA1A	724_100_1	5,2464218	Mb	39,454