The VIRTIS PDS/IDL software library

Issue 2.6.7

	NAME	FUNCTION	SIGNATURE	DATE
Prepared by	S. Erard	Archive Manager, Virtis VEx		**19/2/2007**
Checked by	A. Coradini	PI, Virtis Rosetta		
	P. Drossart	coPI, Virtis VEx French team leader, Virtis Rosetta		
	G. Piccioni	coPI, Virtis VEx		
Approved by				
Authorized by				

TABLE OF CONTENTS

Acronym and definition list

PDS : Planetary Data System
Standard definition scheme of data archives, including data file format, used by NASA for all planetary missions, and by ESA starting with the Rosetta mission.

FITS : Flexible Image Transport System
Standard data description system used for most telescopic and radiotelescopic images and data.

NAIF/SPICE : Navigation and Ancillary Information Facility / Spacecraft, Planetary ephemeris, Instrument pointing, C-matrix, Events
System used to describe and compute information such as spacecraft location and pointing direction. Supported by ESA for Rosetta and Venus-Express.

PSA : Planetary Science Archive
ESA's data archive for planetary missions.

IDL : Interactive Data Language
A commercial software commonly used in astronomy laboratories since the 1980's.

SI : Système International d'unités
International system of physical units.

UTC : Universal Time Corrected
Common time system, including leap seconds

EGSE : Electrical Ground Support Equipment
Ground-based segment of the VIRTIS instrument. Used to decode telemetry files, check data consistency, quick look to the data and PDS formatting of data files.

TM : TeleMetry
General name for data returned by the spacecraft/instrument. For VIRTIS, formatted as paquets including science data, housekeeping, event data, instrument parameters…

SCET : SpaceCraft Elapsed Time
Time marker of the TM packets. Present in the Data Field Header of each TM packet, and stored in the backplanes of the PDS files.

HK : HouseKeeping parameters
Instrument functioning parameters, other than science data (in the present document, actually include information from both the housekeeping and event packets).

INTRODUCTION

What is it?

This library is intended to read a variety of PDS data files under IDL, in particular files containing Qube objects. It is specially intended to read the files from the VIRTIS experiment on board the Rosetta and Venus-Express missions. It should be used as the lower level layer in VIRTIS IDL software.

This library is developed as a non-public software tool, and its distribution is so far restricted to the VIRTIS Rosetta and VEx teams. It is partly based on the SBNPDS 2.0 library and on the IDLASTRO library, and is maintained by S. Erard (LESIA, Observatory of Paris).

The virtispds.pro routine is a front-end interface specific to Virtis.
v_readpds.pro is a versatile PDS file reader, also usable with Virtis subsystem test files.

Distributions

The present distribution of the Virtis PDS library is intended for Virtis coIs /users to access Virtis data, and (hopefully) external, support, data written in PDS format. It is therefore limited to reading routines.
Additional routines are available on demand to Virtis coIs who need to produce PDS files of derived data.

Applicable / Reference Documents

The present software and documentation are responsive to the following documents:

AD1. VIRTIS data archive format [VIR-ORS-RS-1146, Version 3.4, 29/04/2002]
AD2. Update to VIRTIS Rosetta archive format, VIR-ORS-RS-2251 [issue 2.5, July 2006]
AD3. VIRTIS-VEx EAICD (from SOP-RSSD-TN-017) [draft 0.8, Nov. 2006]
AD4 VIRTIS Software Requirement Document [VIR-DLR-RS-003, version 2 draft 1, 6/12/2000]
AD5 VIRTIS geometry files formatting [VIR-LES-SW-2268, version 0.14, Nov. 2006]

RD1. Planetary Data System Standards Reference, version 3.6 [JPL Document D-7669, part 2, 01/08/2003]
RD2. ROSETTA Time Handling [RO-EST-TN-3165 Issue 1, 9 February 2004]

INSTRUCTIONS FOR USE

Installation

Just copy the library in a directory included in your IDL path. Also remove older versions from your path + restart IDL in case of doubt.

A directory tree can be added to the path by including the following lines in your IDL startup file (idl_start.pro):

```
defsysv,'!LIB_DIR', dirlib
!PATH = !PATH + Path_sep(/search) + EXPAND_PATH('+'+!LIB_DIR)
```

where dirlib is a string containing the path to a directory including the library (e.g. "~/IDL/userlib/" on Unix/Linux/MacOS X systems)

This library is normally standalone. In case of problems, you may want to install also the following open source libraries, which are available on the network:
ASTRON (NASA IDL Astronomical library)
TEXTOIDL (by M. W. Craig, University of California)

The present version is operational under IDL 5.5, but it is optimized for versions 5.6 and up.

How to read Virtis PDS files?

Any VIRTIS file generated by the EGSE (after Nov. 2001) can be read with:

```
result = virtispds('file_name')
```

where:
file_name is the name of the file that contains the label (for Virtis, this is the data file itself)
“result” is then a structure containing the label and data on output

Warning messages such as:

```
V_LISTPDS: WARNING - Value (2) is not a list
```

may be displayed during reading, with no consequence.

For automated processing, the option /SILENT filters all console messages.

If data are not from VIRTIS
The output structure contains no data, but includes the label to allow for automatic controlling:
result.label: label of the PDS file

The system variable !err is set to -1. This error code must be checked immediately after the call to virtispds (it is reset by any later IDL instruction). The standard reading instruction is therefore:

```
result = virtispds('file_name')
If !err EQ -1 then  message, 'Not a VIRTIS file'
```

Simple QUBE data

Virtis raw data and VVEx geometry files are formatted this way. The output structure is such that:

result.label: label of the PDS file
result.qube_name: a 2-strings array providing the cube name and unit
result.qube_dim: a 3-elt array providing the cube dimensions
result.qube: 3D data core of the qube
Size= (# of bands, # of lines, # of frames)
result.suf_name: list of the HK parameter names for H or M.
result.suf_dim: a 3-elt array providing the suffix dimensions
result.suffix: suffix of the data qube, reformatted.

For raw data files, the first dimension of the suffix contains a complete group of HK (82 for M, 72 for H, see Tables 1 and 2 below):
Size = (# of HK, # of HK structure/frame, # of frames)

For VVEx geometry files, the cube contains the geometry parameters (see Table 3 below), and no suffix is present:

result.qube_name: contains the list of geometric parameters stored in the cube core.
result.qube_coeff: contains the coefficients to apply to geometric parameters to get standard units (distances/elevations in km, angles/coordinates in °, local time in Venus hours).

VIRTIS VEx calibrated cubes

VVEx calibrated files include several PDS objects, and are different for H and M.

For H calibrated cubes all spectra are grouped in a single dimension, whatever the acquisition mode is (ie, data in backup and nominal mode are formatted similarly as 2D cubes in the output of virtispds — notice that the file itself contains a 3D cubes with second dimension = 1). The output structure is such that:

result.label: label of the PDS file
result.column_names: names of following vectors
result.table: a 2D array containing the spectral table for every channel:
result.table(0,*) = wavelength
result.table(1,*) = bandwidth (FWHM)
result.table(2,*) = radiance uncertainty (1-sigma)
result.qube_name: a 2-strings array providing the cube name and unit
result.qube_dim: a 2-elt array providing the cube dimensions
result.qube: 2D data core of the qube (floats). Size=(# of bands, # of spectra)
result.suf_name: names of suffix parameters (SCET components)
result.suf_dim: a 2-elt array providing the suffix dimensions
result.suffix: Interpolated SCET. Size=(3, # of spectra)

For M calibrated data, the 3D format of the raw data is preserved. The output structure is such that:

result.label: label of the PDS file
result.table: a 3D array containing the spectral reference for every matrix spectel:
result.table(*,*,0) = wavelength

result.table(*,*,1) = bandwidth (FWHM)
result.table(*,*,2) = radiance uncertainty (1-sigma)
result.qube_name: a 2-strings array providing the cube name and unit
result.qube_dim: a 3-elt array providing the cube dimensions
result.qube: 3D data core of the qube (floats). Size=(# of bands, # of lines, # of frames)
result.suf_name: list of suffix parameters (SCET)
result.suf_dim: a 2-elt array providing the suffix dimensions
result.suffix: SCET for each frame. Size=(3, # of frames)

Other Virtis files include calibration files, which can also be read with virtispds.pro. Some of them have detached labels, in which case the label and data files may be located in different directories. The routine must be called with the name of the label file as argument.
If reading errors occur, one may try to include the complete path in the file name, or to call the routine from the directory containing the data file. In some cases, the routine v_readpds may provide better results.

```
dpix = virtispds('/VVEx/CALIBRATION/DEADPIXELMAP.LBL')
```

TABLE data
The output structure is such that:
result.label: label of the PDS file
result.column_names: a string array providing the names of the table columns
result.table: a 2D array containing the table

In particular, for calibrated spectrum (H individual spectra), the output structure is such that:
result.table: a 2D array containing the spectrum:
result.table(0,*) = wavelength
result.table(1,*) = intensity (radiance)
result.table(2,*) = uncertainty

IMAGE data
The output structure is such that:
result.label: label of the PDS file
result.nimages: number of images in the file
result.images: a 3D array containing all the images in the file, with size:
(# of images,# of columns,# of rows)

Mixed-object files
The output structure is a combination of the above formats. When several objects of the same type are present, an order number is appended to their tag, starting from 0 (with the exception of cubes, which are processed in one pass):
result.label: label of the PDS file
result.column_names0: relates to first table
result.table0

```
result.column_names1: relates to second table
result.table1
```

How to handle Virtis data cubes?

A frame is plotted with:

```
tvscl, result.qube(*,*,n)
```

A spectrum is plotted with:

```
plot, result.qube(*,p,n)
```

Successive H measurements in nominal mode are plotted at a given wavelength with:

```
plot, result.qube(Nlam,*,*)
```

Dark frames can be selected with (beware of parenthesis!):

```
idark = where((result.suffix(5,0,*) and '2000'X) NE 0)
plot, result.suffix(5,0,idark)
```

or alternatively can be filtered with:

```
Qdark = (result.suffix(5,0,*) and '2000'X)
NoDark = where(Qdark EQ 0)
plot, result.qube(25,0,nodark)
```

The label itself can be printed on screen with:

```
print, result.label                          ; messy on several systems
print, v_eolpds(result.label,/sil)           ; for a clean print on a MacOS 9
print, v_eolpds(result.label,/print,/sil)    ; for a clean print on Unix or MacOS X
```

Note: this example does not work with some versions of IDL (e.g., 5.5 for Mac OS9), because of an IDL bug in string printing. In case of problem, try:

```
for i=1,(size(result.label))(1) do print, result.label(i-1)
```

Raw data qubes and suffices may have their last dimension equal to 1 in some situations. Because of the way IDL handles structures, arrays extracted from such structures have only two dimensions. The fields QUBE_DIM and SUF_DIM allow to solve this issue with minor trouble:

```
f=virtispds('CTOB_05_0.QUB')
help , f.qube
<Expression>    INT       = Array[3456, 64]
s=reform(f.qube, f.qube_dim)
help , s
S               INT       = Array[3456, 64, 1]
```

The two spatial dimensions of Virtis H data cubes are only related to data processing in the main memory, however. Virtis H calibrated cubes only have two dimensions, one spectral and one spatial/temporal.

How to access Virtis operational parameters?

Observation parameter are stored in the labels and can be retrieved automatically. For instance, to get the exposure time for Virtis-H (assuming the final flight format):

```
fp = v_pdspar(result.label, 'frame_parameter_desc')
ip = where(v_listpds(fp) EQ '"EXPOSURE_DURATION"')
print, (v_listpds(v_pdspar(result.label, 'frame_parameter')))(ip)          ; value
print, (v_listpds(v_pdspar(result.label, 'frame_parameter_unit')))(ip)     ; unit
```

How to handle Virtis housekeeping parameters?

The housekeeping parameters are stored in the sideplane of the raw PDS qubes in a rather compact format to save disk space and memory. After a call to virtispds, they are accessible in the suffix tag of the result structure in a more siutable way. Successive HK structures are separated (instead of being packed as they are in the qube sideplane), but the values are still encoded according to the TM scheme. These quantities provide all the information required for data processing purposes, including calibration.

As mentioned above, the first dimension contains a complete group of HK (82 for M, 72 for H, see Tables 1 & 2 below):
result.suffix: suffix of the data qube, reformatted.
Size = (# of HK, # of HK structure/frame, # of frames)

The HK structure for a given spectrum is plotted with:

```
plot, result.suffix(*,0,p)      ; or
tvscl, result.suffix(*,*,p)
```

A given HK is plotted in sequence with:

```
plot, result.suffix(m,*,*)
```

For various reasons, some HK packets may be absent from the data stream. In this case, the HK structure is filled with "FFFF" hexadecimal values, which is a reserved code for VIRTIS HK (i.e., no HK can normally take this value). To filter missing HK codes when plotting the values, type:

```
plot, result.suffix(m,*,*)*(result.suffix(m,*,*) LT 'FFFF'X)
```

HK values can be printed in hexadecimal with:

```
print, result.suffix(*,*,*), format="(Z)"
```

A list of HK names in result.suffix can be printed with:

```
print, result.suf_name                  ; on Windows
print, result.suf_name+string(10B)      ; on Unix or MacOS X
```

For detailed HK monitoring, the library provides a way to convert HK parameters into physical values. Multiple HK are first split into independent quantities, then these quantities are converted using a mission-specific transfer function for Rosetta and VEx. The conversion is performed as follows:

```
ParTab = v_pdshk(result)
```

In output:

ParTab.values is an array of instrument parameters converted to physical units
ParTab.names is a list of the instrument parameters

The returned quantities are different from the HK parameters from virtispds, because single HK that encode several instrument parameters are split by v_pdshk. This routine, its subroutines and the parameter list are used on Otarie in Meudon to monitor HK values:

```
ParN = 58
print, ParTab.names(ParN)
print, sr.suf_name(ParN)       ; not the same thing...
plot, ParTab.values(ParN,*,*)*(ParTab.values(ParN,*,*)  LT 'FFFF'X)
```

How to handle time in Virtis files?

The library includes a versatile function to handle time conversions from ISO strings to vector format and back (and from Julian day to ISO string):

ISO time strings (such as START_TIME in labels) are standard strings with the form:
YYYY-MM-DDThh:mm:ss.fff
Vector format is an array containing (in the same order as ISO strings):
[Year,Month,Day,Hours,Minutes,Seconds.fractions]

The conversion function is called v_time.pro, and its basic use is straightforward:

```
        print, v_time('2005-05-16T01:26:20')      ; convert from ISO string
returns:        2005        5        16        1        26        20
        print, v_time([2005,5,15,23,50,20.2])     ; convert from vector format
returns:        2005-05-15T23:50:20.200
        print, v_time(2453506.55981482d)          ; convert from Julian day
returns:        2005-05-16T01:26:08.000
```

This function accepts a second argument. If present, it is considered as an offset (in seconds) to be added to the first argument. The resulting time is returned in the same format as the first argument:

```
        print, v_time([2005, 5, 15, 23, 50, 20.2], 620)
returns:            2005.00     5.00000     16.0000     0.00000     0.00000     40.2000
        print, v_time('2005-05-16T01:26:20', 50)
returns:        2005-05-16T01:27:10.000
```

All times in the TM data are provided as SCET (on-board time of acquisition, in S/C clock units). The timing of observations and HK acquisitions are available as SCETs in the sideplanes of the PDS cubes, and are encoded on 3 words (see Tables 1 & 2). The library provides a quick way to compare SCETs to ground based events given in UTC, such as TC timelines or telescopic observations:

```
lbl = v_headpds('FS535916.QUB')   ; get label for session of interest
scet = 68635016.143d              ; SCET of a particular TM event
print, v_scet2ut(lbl, SCET)       ; return corresponding UTC as an ISO string

im = virtispds('FS535916.QUB')
scet = im.suffix(0:2,0,0)         ; Acquisition time (SCET) encoded on 3 words
                                  ; (from PDS file sideplane)
sc = v_scet(Scet(0),Scet(1),Scet(2))      ; convert to # of seconds
print, v_scet2ut(im.label, SC)    ; return corresponding UTC as ISO string
print, v_scet(sc)                 ; reverts to 3-integer format
```

The SCET must be provided to v_scet2ut.pro as a long integer or as a double precision floating point (severe round-off errors will occur otherwise). The SCET must also be inside the session limits, or nearby, in order to get a correct UTC estimate (the relationship with UTC depends on the session, in particular on the Earth-S/C distance and on possible clock drift). See important note below.

v_scet2ut can also be called with no label in argument, in which case the SCET is considered as a number of seconds elapsed since launch in Earth's frame. Beware that in this case the result is *not* the UTC of a spacecraft event:

```
S_offset = 68635016.15d
print, v_scet2ut(S_offset)              ; Rosetta
returns:    2005-03-05T09:16:56.000
print, v_scet2ut(S_offset, /VEx)        ; VEx
returns:    2007-05-03T09:16:56.000
```

Important note concerning SCET to UT conversions:
Beware that the UT estimate provided by v_scet2ut eventually relies on the time stamp provided in the TM paquets. This estimate is only an approximation, and exact values can be derived only through the use of Spice kernels. Spice-derived UTC at moment of acquisition are available in the Virtis-VEx geometry cubes.

How to read other (non-Virtis) PDS files?

A variety of other PDS files can be read with a lower-level function, including VIRTIS test files produced during sub-system tests:

```
result = v_readpds('file_name', lbl, suffix=suf)
```

where:

file_name is the name of the file that contains the label (in case of detached labels, this is not the data file but the label file). This string may include a path to the file.
result is then a variable containing the data (in case of multiple detached labels, from all the data files described in the label)
lbl is an optional argument that contains the label in output
suf is an optional parameter that contains the suffix in output when reading a qube

On output, result (and suf) are structures if several data objects are described in the label. Beware that when using v_readpds, the VIRTIS suffixes are not translated to a convenient format (as does virtispds.pro).
If only one object is present, result (and suf) are arrays of the corresponding size and dimension. If only one suffix (either sideplane or backplane) is associated to a cube, suf is a 3D array (usual case). If both types of suffixes are included in the file, suf is a structure containing the two suffices (e.g., VIMS and OMEGA files).
All the data objects described in the label are stored in the output structure. In case of a detached label, this may include several data files. When working with detached labels, the data file must be located either in the same directory as the label, or in the working directory. In the files are located in different directories, the routine must be called from the data directory using the full label name, including directory path.
Most usual PDS objects are handled by the library, although not all possible options are implemented. Besides, not all objects have been tested in details (Arrays and Collections in particular have not been tested), and special problems may arise with some types of objects or some data files, specially Tables (e.g., external format definitions are assumed to be located in current directory, Container objects are not supported…).

The following set of instructions mimics the functioning of the routine lecomeg.pro used to read OMEGA/MEx data files, and produces the same output. It is about twice as fast as the improved version lecomeg2.pro (March 2004 version).

```
Nomfich = 'ORB0030_1'
  ; retrieve cube, suffixes and label
idat = v_readpds(Nomfich+'.QUB', lbl, suffix=suf)
sdat0 = reform(suf.B_suf)
sdat1 = suf.S_suf
suf = 0; save place
  ; parse integration times
tint =  v_listpds(v_pdspar(h, 'EXPOSURE_DURATION'))
print, 'Integ time SWL:',tint(0)
print, 'Integ time Vis:',tint(2)
bin =  v_pdspar(h, 'DOWNTRACK_SUMMING') ; parse line binning
print,'Binning:', bin
```

where:
Nomfich is the name root of the PDS file to read.
lbl contains the label in output.
idat contains the data qube (x, lambda, y)
sdat0 and sdat1 contain the backplane (dark current) and sideplane (housekeeping parameters) respectively.

DETAILS

Special encoding of information in the PDS labels

Acquisition parameters are usually constant during a observing session, and are stored in the labels as a list introduced by the keyword FRAME_PARAMETER.
There is one situation in which parameters can vary during acquisition: EXPOSURE_DURATION and INTERNAL/EXTERNAL_REPETITION_TIME may take a series of values during calibration sessions. In this case, they are encoded as -1 in the label, and the current value has to be retrieved from the sideplane (see below).

Content of sideplanes in VIRTIS PDS raw data files

The format of the Virtis PDS archive is a little too tricky to be described here in details. It is completely described in the document AD1, which applies to both Rosetta and VEx.

Most data (except some H data in special acquisition modes) are stored in Qube objects. The HK are stored in the sideplane of the Qube, as a series of 2-byte parameters ("elemental structure"). The structure of this series is constant (for M and H separately), in all Virtis files. The following tables list the parameters in the sideplanes.
Parameters names are contained in result.suf_name in output of virtispds.
Missing HK measurements are encoded as "FFFF" hexadecimal, which is a reserved code for VIRTIS HK.

Table 1: Elemental HK structure for M files sideplane

Word number	Origin TM	Field in this TM	Word number in this TM *	Data Field
1	First Science reporting TM for this frame	Data Field Header	4	SCET data, 1st word
2	_	_	5	SCET data, 2nd word
3	_	_	6	SCET data, 3rd word
4	_	Science Data Header	9	Acquisition ID
5	_	_	10	Number of sub-slices + first serial number
6	_	_	12	Data Type
7	0	0		SPARE
8	VTM_ME_Default_HK_ Report (SID1)	Data Field Header	4	SCET periodic HK, 1st word
9	_	_	5	SCET periodic HK, 2nd word
10	_	_	6	SCET periodic HK, 3rd word
11	_	Source Data	10	V_MODE
12	_	_	11	ME_PWR_STAT
13	_	_	12	ME_PS_TEMP
14	_	_	13	ME_DPU_TEMP
15	_	_	14	ME_DHSU_VOLT
16	_	_	15	ME_DHSU_CURR
17	_	_	16	EEPROM_VOLT
18	_	_	17	IF_ELECTR_VOLT
19	0	0		SPARE
20	MTM_ME_General_HK _Report (SID2)	Data Field Header	4	SCET periodic HK, 1st word
21	_	_	5	SCET periodic HK, 2nd word
22	_	_	6	SCET periodic HK, 3rd word
23	_	Source Data	10	M_ECA_STAT
24	_	_	11	M_COOL_STAT
25	_	_	12	M_COOL_TIP_TEMP
26	_	_	13	M_COOL_MOT_VOLT
27	_	_	14	M_COOL_MOT_CURR
28	_	_	15	M_CCE_SEC_VOLT

29	0	0		SPARE
30	MTM_VIS_HK_Report (SID4)	Data Field Header	4	SCET HK, 1st word
31	_	_	5	SCET HK, 2nd word
32	_	_	6	SCET HK, 3rd word
33	_	Source Data	10	M_CCD_VDR_HK
34	_	_	11	M_CCD_VDD_HK
35	_	_	12	M_+5_VOLT
36	_	_	13	M_+12_VOLT
37	_	_	14	M_-12_VOLT
38	_	_	15	M_+20_VOLT
39	_	_	16	M_+21_VOLT
40	_	_	17	M_CCD_LAMP_VOLT
41	_	_	18	M_CCD_TEMP_OFFSET
42	_	_	19	M_CCD_TEMP
43	_	_	20	M_CCD_TEMP_RES
44	_	_	21	M_RADIATOR_TEMP
45	_	_	22	M_LEDGE_TEMP
46	_	_	23	OM_BASE_TEMP
47	_	_	24	H_COOLER_TEMP
48	_	_	25	M_COOLER_TEMP
49	_	_	26	M_CCD_WIN_X1
50	_	_	27	M_CCD_WIN_Y1
51	_	_	28	M_CCD_WIN_X2
52	_	_	29	M_CCD_WIN_Y2
53	_	_	30	M_CCD_DELAY
54	_	_	31	M_CCD_EXPO
55	_	_	32	M_MIRROR_SIN_HK
56	_	_	33	M_MIRROR_COS_HK
57	_	_	34	M_VIS_FLAG_ST
58	0	0		SPARE
59	MTM_IR_HK_Report (SID5)	Data Field Header	4	SCET HK, 1st word
60	_	_	5	SCET HK, 2nd word
61	_	_	6	SCET HK, 3rd word
62	_	Source Data	10	M_IR_VDETCOM_HK
63	_	_	11	M_IR_VDETADJ_HK
64	_	_	12	M_IR_VPOS
65	_	_	13	M_IR_VDP
66	_	_	14	M_IR_TEMP_OFFSET
67	_	_	15	M_IR_TEMP
68	_	_	16	M_IR_TEMP_RES
69	_	_	17	M_SHUTTER_TEMP
70	_	_	18	M_GRATING_TEMP
71	_	_	19	M_SPECT_TEMP
72	_	_	20	M_TELE_TEMP
73	_	_	21	M_SU_MOTOR_TEMP
74	_	_	22	M_IR_LAMP_VOLT
75	_	_	23	M_SU_MOTOR_CURR
76	_	_	24	M_IR_WIN_Y1
77	_	_	25	M_IR_WIN_Y2
78	_	_	26	M_IR_DELAY
79	_	_	27	M_IR_EXPO
80	_	_	28	M_IR_LAMP_SHUTTER
81	_	_	29	M_IR_FLAG_ST
82	0	0		SPARE

* According to document AD4

Table 2: Elemental HK structure for H files sideplane

Word number	Origin TM	Field in this TM	Word number in this TM *	Data Field
1	First Science reporting TM for this frame	Data Field Header	4	SCET data, 1st word

2	_	_	5	SCET data, 2nd word
3	_	_	6	SCET data, 3rd word
4	_	Science Data Header	9	Acquisition ID
5	_	_	10	Number of sub-slices + first serial number
6	_	_	12	Data Type
7	0	0		SPARE
8	VTM_ME_Default_HK_ Report (SID1)	Data Field Header	4	SCET periodic HK, 1st word
9	_	_	5	SCET periodic HK, 2nd word
10	_	_	6	SCET periodic HK, 3rd word
11	_	Source Data	10	V_MODE
12	_	_	11	ME_PWR_STAT
13	_	_	12	ME_PS_TEMP
14	_	_	13	ME_DPU_TEMP
15	_	_	14	ME_DHSU_VOLT
16	_	_	15	ME_DHSU_CURR
17	_	_	16	EEPROM_VOLT
18	_	_	17	IF_ELECTR_VOLT
19	0	0		SPARE
20	HTM_ME_General_HK _Report (SID3)	Data Field Header	4	SCET periodic HK, 1st word
21	_	_	5	SCET periodic HK, 2nd word
22	_	_	6	SCET periodic HK, 3rd word
23	_	Source Data	10	H_ECA_STAT
24	_	_	11	H_COOL_STAT
25	_	_	12	H_COOL_TIP_TEMP
26	_	_	13	H_COOL_MOT_VOLT
27	_	_	14	H_COOL_MOT_CURR
28	_	_	15	H_CCE_SEC_VOLT
29	0	0		SPARE
30	HTM_HK_Report (SID6)	Data Field Header	4	SCET HK, 1st word
31	_	_	5	SCET HK, 2nd word
32	_	_	6	SCET HK, 3rd word
33	_	Source Data	10	HKRq_Int_Num2
34	_	_	11	HKRq_Int_Num1
35	_	_	12	HKRq_Bias
36	_	_	13	HKRq_I_Lamp
37	_	_	14	HKRq_I_Shutter
38	_	_	15	HKRq_PEM_Mode
39	_	_	16	HKRq_Test_Init
40	_	_	17	HK_Rq_*Device*/On
41	_	_	18	HKRq_Cover
42	_	_	19	HKMs_Status
43	_	_	20	HKMs_V_Line_Ref
44	_	_	21	HKMs_Vdet_Dig
45	_	_	22	HKMs_Vdet_Ana
46	_	_	23	HKMs_V_Detcom
47	_	_	24	HKMs_V_Detadj
48	_	_	25	HKMs_V+5
49	_	_	26	HKMs_V+12
50	_	_	27	HKMs_V+21
51	_	_	28	HKMs_V-12
52	_	_	29	HKMs_Temp_Vref
53	_	_	30	HKMs_Det_Temp
54	_	_	31	HKMs_Gnd
55	_	_	32	HKMs_I_Vdet_Ana
56	_	_	33	HKMs_I_Vdet_Dig
57	_	_	34	HKMs_I_+5
58	_	_	35	HKMs_I_+12
59	_	_	36	HKMs_I_Lamp
60	_	_	37	HKMs_I_Shutter/Heater
61	_	_	38	HKMs_Temp_Prism
62	_	_	39	HKMs_Temp_Cal_S
63	_	_	40	HKMs_Temp_Cal_T

64	_	_	41	HKMs_Temp_Shut
65	_	_	42	HKMs_Temp_Grating
66	_	_	43	HKMs_Temp_Objective
67	_	_	44	HKMs_Temp_FPA
68	_	_	45	HKMs_Temp_PEM
69	_	_	46	HKDH_Last_Sent_Request
70	_	_	47	HKDH_Stop_Readout_Flag
71	0	0		SPARE
72	0	0		SPARE

* According to document AD4

Content of VIRTIS VEx geometry files

The Virtis VEx data set includes geometry files providing parameters that allow to project the data at Venus, plus viewing angles, surface elevation, etc… The format of the geometry files is described in details in the document AD5.

Geometry parameters are stored in Qube objects with no sideplane. The following table lists the parameters in the qubes. Parameters names are contained in result.qube_name in output of virtispds. The vector result.qube_coeff contains coefficients to convert geometric parameters into standard units (distances/elevations in km, angles/coordinates in °, local time in Venus hours).

Table 3: contents of Virtis VEx geometric files, for observations intercepting the surface

Plane #	**Parameter description**	**Comment**
1-4	Longitudes of 4 pixel footprint corner points	Geometrical projection on surface ellipsoid, with no correction for scattering or refraction
5-8	Latitudes of 4 pixel footprint corner points	
9-10	Longitude & latitude of pixel footprint center on surface ellipsoid	
11-13	Incidence, emergence & phase at footprint center, relative to Venus center direction	Angles at the reference surface, with no topography.
14	Surface elevation (footprint corners average)	From topographic model
15	Slant distance (line of sight from spacecraft to surface ellipsoid at pixel center)	Does not include topographic model
16	Local time at footprint center	
17-20	Longitudes of 4 corner points on cloud layer	Geometrical projection on reference cloud layer (60km)
21-24	Latitudes of 4 corner points on cloud layer	
25-26	Longitude & latitude of pixel center on cloud layer	
27-29	Incidence, emergence & phase, relative to local normal of cloud layer	Phase angle is the complement of the scattering angle
30	Surface elevation at the vertical of cloud layer intercept	From topographic model
31-32	Right ascension and declination of pointing direction (J2000 reference frame.)	
For M:	**1 supplementary plane**	
33	One frame-common plane	Provides 8 scalar quantities along the frame spatial dimension. The remainder is set to 0.
	1-2 Original data SCET from TM	The first value stores the SCET first two words (integer part), the second one stores the third SCET word (fractional part).
	3-4 UTC	Encoded UTC recomputed through the SPICE system. The first value contains the number of days since Jan. 1st, 2000, the second value

		contains the time of the day in ms (starting from 0h).
	5-6 Subspacecraft coordinates (longitude/latitude)	
	7-8 Sine and cosine of M mirror angle	Transformed into sin/cos values from HK.
	9-10 Sun direction: 9: angle between Sun direction and S/C Z axis 10: azimuth of Sun direction in the instrument XY plane (counted from 0° at X axis)	In the instrument frame
For H:	**7 supplementary planes**	
33-34	Original data SCET from TM	Interpolated for each spectrum in nominal mode. The first plan stores the SCET first two words (integer part), the second one stores the third SCET word (fractional part).
35-36	UTC	Encoded UTC recomputed through the SPICE system. The first plan contains the number of days since Jan. 1st, 2000, the second plan contains the time of the day in ms (starting from 0h).
37-38	Subspacecraft coordinates (longitude/latitude)	
39	Slit orientation	Relative to the pixel normal
40-41	Sun direction: 40: angle between Sun direction and S/C Z axis 41: azimuth of Sun direction in the instrument XY plane (counted from 0° at X axis).	First angle provides angle with Z (nadir) axis, second one provides the azimuth in (X,Y) plane (in the instrument frame).

• During limb observations surface elevation at the ellipsoid intercept (plane 14) is substituted by the tangent altitude (impact parameter above the surface) with the addition of a large offset (100,000 m). This offset is intended to select or filter limb observations easily. The 100,000 m offset must be subtracted from plane 14 to retrieve the tangent altitude.
• Surface elevation at the cloud layer intercept (plane 30) is maintained whenever possible. If the line of sight does not intercept the cloud layer, surface elevation is provided at the vertical of the tangent point. A surface elevation is therefore always available in the geometry cubes, although not necessarily below the tangent point.
• Angles and local time are computed at the intersection with the local vertical (tangent point).

• Geometric quantities that are constant in the time frame of a session are also provided in the label, through the following keywords: SOLAR_DISTANCE, SOLAR_LONGITUDE, SUB_SOLAR_LONGITUDE, SUB_SOLAR_LATITUDE.

VIRTIS VEx files names

The final file name scheme for Virtis VEx is described in more details in document AD3. In short:

VFxxxx_nn.QUB **raw data files**
where:
V is a literal “V” character
F is the FPA/transfer mode identifier (one alpha character):
H: H image transfer mode (backup observation mode)
S: H spectrum transfer mode (including dark files in nominal mode)
T: H "64-spectra" transfer mode (nominal mode)
I: M-IR
V: M-Vis

xxxx is the orbit number coded on exactly 4 digits to encompass the duration of the extended mission.
nn is the subsession ID (ID of file produced by this FPA for this orbit).

VFxxxx_nn.GEO **geometry files**
Name root is identical to the corresponding data file described above

VFxxxx_nn.EEE **calibrated data files**
Name root is identical to the corresponding raw data file described above
EEE: Extension is "CAL" for calibrated data files, or "DRK" for calibrated dark current files. Calibrated dark current files are provided (at east for H) so that a refinement of dark interpolation is possible independently of the complete calibration procedure.

Contents of the library

The current version includes the following routines:

virtispds.pro
    Front-end interface to read VIRTIS data
    Returns (label + data objects + HK) in a structure
v_readpds.pro
    Front-end interface to read most common PDS data files

v_headpds.pro
    Low level routine to read a PDS label
v_imagepds.pro
    Low level routine to read all (binary) image objects. Returns either an array or a structure
v_qubepds.pro
    Low level routine to read all (binary) Qube objects and their suffices. Returns either arrays or structures.
    Supports PDS non-conformities in some data sets.
v_atabpds.pro
    Low level routine to read an ascii table object. Returns a structure with columns
    Replaces v_tascpds.
v_btabpds.pro
    Low level routine to read a binary table object. Returns a structure with columns
    Replaces v_tbinpds.
v_btabvect.pro
    Utility routine to extract a column from a binary table
v_arascpds.pro
    Low level routine to read an ascii array object
v_arbinpds.pro
    Low level routine to read a binary array object
v_colaspds.pro
    Low level routine to read an ascii array collection object
v_colbipds.pro
    Low level routine to read a binary array collection object

v_getpath.pro
    Utility routine to get directory and file name from a path string (works from IDL 5.4 and up)
v_eolpds.pro
    Utility routine to convert end-of-line markers from LF only to CR-LF, and reverse
v_pdspar.pro
    Utility routine to get the value corresponding to a PDS keyword from a label
v_listpds.pro
    Utility routine to extract a single value from a PDS value list
v_str2num.pro

Utility routine to return the numeric value of a string

v_typepds.pro
Utility routine to identify the IDL type of a variable from its PDS keywords definition

v_objpds.pro
Utility routine to get definition lines in label and pointer to the object

v_pointpds.pro
Utility routine to parse filename and offset from PDS data pointer

v_bmaskpds.pro
Utility routine to apply bit masks to binary data

v_swapdata.pro
Utility routine to convert from MSB or LSB encoding to host encoding.
Replaces v_msbtohost.

v_vaxtoieee.pro
Utility routine to convert floats from Vax to IEEE encoding, and optionally VAX (LSB) integers to host encoding.
Replaces v_conv_vax_unix.

The following routines are used independently to handle time information:

v_scet.pro
Utility routine to convert SCET (on-board time) between 3-words integer format and number of seconds

v_scet2ut.pro
Utility routine to convert SCET into UTC (ISO format) within a session (first order estimate)

v_time.pro
Utility routine to convert a time string from ISO format to numerical vector and back, and to compute time offsets

The following routines are used independently to handle housekeeping parameters:

v_pdshk.pro
Utility routine to split HK values from the sideplane and convert them into physical units (M and H channels, Rosetta and VEx)

v_translatehk.pro

v_hk_names.pro

v_transfunchk.pro

v_compute_intnum.pro

v_compute_scet.pro (similar to v_scet.pro)
Lower level routines for HK parameters handling

The following routines are not included in the basic distribution, but are available on demand to Virtis data producers:

v_convlabel.pro
Formatting routine to write Virtis calibrated files and to update PDS labels (VEx and Rosetta)
To be used with label templates

v_geolabel.pro
Formatting routine to write Virtis VEx geometry files
To be used with label templates

Older routines, removed:

v_checktime.pro
Utility routine to check Virtis session limits (UTC/SCET in label against SCET in sideplane)

v_tascpds.pro
Low level routine to read an ascii table object. Returns a structure with columns.

v_tbinpds.pro
Low level routine to read a binary table object. Returns a structure with columns.

tirtispds.pro
Front-end interface to read VIRTIS data, temporary version for initial calibration files
(corrects a byte order problem in the initial EGSE software. No longer useful after November 2001)

v_timepds.pro
Utility routine to extract a time string from a label, and possibly convert it. Replaced by v_time.pro.

v_conv_vax_unix.pro
Utility routine to convert floats and integers from Vax to IEEE encoding. Replaced by v_vaxtoieee.pro.
v_msbtohost.pro
Utility routine to convert from MSB encoding to host encoding (i.e., to LSB if needed). It was called v_frommsb.pro in previous versions. Replaced by v_swapdata.pro.

Virtispds.doc This doc.

A (shorter) modification history

Version 2.6.7 (Feb 2007)

- Fix for "2D" cube format (H calibrated cubes and suffices) in virtispds.
- Fixed v_qubepds and v_imagepds to close files properly when multiple objects with attached label (calibrated Virtis-M files).
- New scheme to load files with detached label from another directory, and to support case variations in file names. Currently implemented for tables (handles Virtis calibration functions), images and cubes. Use new function v_getpath for IDL ≥ 5.5.
- Various fixes in object routines for non-Virtis files.
- v_qubepds adapted to read qube objects/pointers with derived names (such as REF_QUBE...). This is not yet supported by virtispds or v_readpds.
- Fixed v_convlabel and v_geolabel for last label update. Includes the writing of both M maximum temperatures in M calibrated labels.
- Updated HK routines (resynchronized with Otarie).
- Update/correction of this doc.

Version 2.6.6 (Nov. 2006)

- Implemented M calibrated file format in v_convlabel and virtispds + this doc.
- Fixed v_qubepds for suffices of multiple qubes (M calibrated files).
- Fixed file_records in v_geolabel for both modes (corrects files writing) + added option DEBUG in virtispds to check file length in labels.
- Fixed v_convlabel and v_geolabel for H spectrum transfer mode + H calibrated files (channels x 1 x spectral acquisitions)
- Documentation update for latest geometry files content.

Version 2.6.5 (July 2006)

Various:

- Fixed geometry vectors in virtispds (coefficients)
- virtispds and v_convlabel now support calibrated H files with SCETs in backplane.
- virtispds now returns a 2D cube for H calibrated spectra (#channel, #spectrum).
- Updated documentation (geometry planes + file names).

Version 2.6.4 (June 2006)

Full support for geometry files:
• Modified virtispds to return a list of parameters + scaling coefficients if reading a geometry cube.

Support for calibrated files:
• Fixed v_convlabel to write H calibrated cubes, preliminary version with real data only, no suffix.
• Fixed binary object functions to swap IEEE floats if required (v_imagepds, v_qubepds, v_btabpds). Allows to read standard floats on Intel platforms (calibrated files).

Version 2.6.3 (June 2006)

• Fixed v_btabpds for column number (to read actual H calibrated files).
• Minor fix in v_qubepds to return a scalar value (0b) if no suffix is present (previously returned an ambiguous structure).
• Addedreverse conversion in v_scet.pro: SCET conversion from double precision to 3-word format is now revertible — but output format is purposely different from input format.

Version 2.6.2 (April 2006)

• Added field QUBE_DIM and SUF_DIM in output structure of virtispds (+ small fix in v_readpds). This is used to pass qube dimensions, which are reformed when the qube is extracted — cubes with only one frame end up with 2D only.

• Small fix in v_imagepds: no longer tries to read from label directory if detached label (only in current directory). Previous scheme required IDL 6.0 + produced errors in some situations. Therefore, the data file must be located in the current directory.

March 2006 - Version 2.6.1

1) General improvements
• Now supports qubes in files compressed with gzip (not zip), no extension assumed. Replaced assoc by readu in v_qubepds to allow reading of gziped files with no time penalty + changes in v_headpds to open gziped labels correctly.
• Now supports table format definition in external files + any external definition in labels. Modified v_headpds to recursively include external files provided through ^STRUCTURE pointer. Currently requires the external files to be located in the same directory.
• Improved file search in v_headpds and other routines (relax cases under Unix)
• Optimized v_qubepds and v_readpds for memory usage and speed.
• Checked the library is actually running under IDL 5.5.

2) Fixes
• Solved rare problem with binary input when several variable types are involved, depending on dimensions. Fixed in v_qubepds, v_imagepds, and v_btabpds.
• Fixed file handling (now closes all units correctly).

Feb. 2006 - Version 2.6

1) Improved VIRTIS files support:
• Can now read any combination of images, cubes and tables. In particular files including multiple tables [required to store H calibration data used by calibration software].
• Updated Otarie HK parsing routines (F. Henry)
• Modified file selection in virtispds again: now only accepts plain Virtis files, generated through the OBDH. Those include VVEx-M ground calibration files, which use non-compliant INSTRUMENT_ID. Returns error code + label alone otherwise.

2) Several fixes in subroutines. Most notably:
• Object routines modified to run under IDL 5.5 (recent versions required IDL 6.0).

Dec 2005 - Version 2.5.2

1) Improved VIRTIS files support:
• virtispds can now read calibrated H spectra and calibrated qubes (as generated by Otarie with v_convlabel) + at least some files from subsystem tests (with non-compliant instrument names).
• v_convlabel writes a new file from data/suffix + original label, using label templates. Currently writes calibrated H spectra and qubes, and updates older files labels [this routine is useful only to data producers and is available on request].
• Included routines to split HK parameters and convert them to physical values (Otarie routines by Florence Henry).
• Added procedure v_checktime to monitor time limits in Virtis sessions.
• Added default time origin for VEx flight data in v_scet2ut — seems a bit off, though...
• Checked time accuracy in v_scet2ut, v_scet... at least 0.001s if argument is provided as double or string (~1s if provided as float).
• virtispds is now callable from another routine.

2) Handling of new (future) VIRTIS labels/formats:
• virtispds and v_readpds can now read files including several types of objects (required for future calibrated Virtis files).
• Turned on standard table reading. Fixed v_tascpds and v_tbinpds to read extracted H spectra and calibrated H qubes: EOL markers, offset, data conversion... Still requires DATA_TYPE which is optional.
• Handles resynch # as prefix in S/C clock count.
• Check and fixes for new labels (July then Dec. 2005); seems OK for all 3 channels.

3) Improved image/qube support (26 Oct 2005):
• v_imagepds can now read both browse_image objects and image objects, either together or independently.
• v_imagepds and v_qubepds now skip other types of objects mixed with images/qubes. Can now read uncompressed Clementine images, TES, Viking mosaics & images, Voyager, Near

images... + Virtis calibrated files [Problem came from object reading functions in original SBNPDS 2.0]
• Now supports basic bit masking, through v_bmaskpds.pro. Reads e.g., AMIE images properly.

4) Internal cooking
• Modified data pointer parsing in v_imagepds, v_qubepds, v_tbinpds and v_tascpds.
• v_eolpds now quietly handles outputs from headpds.pro in the original SBNPDS library — this is useful e.g., to print AMIE labels properly. Use either /silent to filter all messages, or /continue to print all formatting errors (line length is checked only when option /print is set).
• v_swapdata now filters byte type (which is not supported in swap_endian_inplace).
• v_scet2ut now accepts scet provided as strings (output from v_pdspar).
• Extensive fix in v_pdspar to parse any kind of multiline values. No longer relies on delimiters; accepts multiline values starting with an empty line (required for Virtis H pixel map coefficients); returns line # of END tag. New option /NoNumeric prevents string conversion (used to preserve unit after value, or to preserve bitmask strings). Now returns empty string (rather than 0) if keyword is not present.
• slibpds removed (no longer uses Virtis routines).

June 2005 - Version 2.5

1) Improvement in label parsing:
• Update/fix to v_pdspar:
- Modified so that it can handle keywords introduced by namespaces (e.g. "ROSETTA:" in flight labels). Default is to filter namespaces before search (match required on keyword only). Option /NAMESPACE forces complete match.
- Fixed numerical conversions (uses type of first keyword occurrence only).
- Now filters all internal (not compliant) spaces from keywords before comparison (allows use with VTL FITS headers...).
- Now returns a scalar if only one occurrence of keyword is found.
• Extensive rewriting of original v_str2num to handle string conversions correctly — it just didn't work. Now supports long integer types, but never returns bytes (complete mess if they are converted back to string afterwards).
• Fix in v_listpds: now returns input string (+ error code) if not a list.
This allows a go at default IDL conversions in the calling routine. Also handles lists followed by a measurement unit.

2) Improvements in data conversions:
• Update in v_typepds.pro: now supports all usual PDS variable types (all except Vax F and G, and bitstrings). Output types are updated (now identifies specifically PC_REAL type and complex types).
• Replaced v_msbtohost by v_swapdata.pro: now a procedure, converts from either MSB or LSB to host. Uses modern swapping routines. Supports all data types (not only integers), all data structures and all platforms automatically, faster and less demanding with IDL ≥ 5.6. Replaced option to simulate other architectures by forced swap.

• Replaced v_conv_vax_unix by v_vaxtoieee.pro: now a procedure, converts VAX floats to IEEE encoding + optionally VAX integers from LSB to host encoding. Uses modern conversions, faster and standard. Beware that floats conversion is not reversible.

3) Support for time management:
• Added routine v_time.pro to convert time between ISO strings and numerical vectors, and to manage offsets provided in seconds.
• Added routine v_scet2ut.pro to convert SCET in UTC in the frame of a given session (uses RSOC time stamp from START_TIME in the PDS label). If no label is provided, count from Rosetta reference time (2003-1-1); this assumes SCET is actually an offset in UTC, and does not account for S/C clock drift or variations of S/C-Earth distance.
• Added routine v_scet.pro to convert SCET from 3-integer storage (in TM paquets and suffixes) to number of seconds (as double precision floating point).
• Removed routine v_timepds, never used (from SBNPDS library).

April 2005

• Checked with first VIRTIS-Rosetta flight data, OK (old labels, though).
• The library now parses integration time correctly from OMEGA data files (OMEGA files now use PDS/PSA compliant labels).

September 2004 - Version 2.4.1

• Minor update to v_pdspar to read labels containing no space separator. This, to handle some non-compliant M ground calibration files from Virtis-VEx (where keyword/value separator is '=' instead of ' = ').
• Routinely used for OMEGA flight data files, fast.

July 2004 - Version 2.4

• Added option /Print to v_eolpds. This performs a clean print of a label on Unix systems, and allows to check label line lengths.
• Added support for PDS spectral library (OMEGA-CRISM) in slibpds (uses modified v_tascpds). Handles beta version from Dec 2003, which is not PDS-compliant.
• Clarify history / version numbers of this library.

October 2003

• Update after playing with VIMS and OMEGA calibration cubes:
1) Fixed bug that returned only sample suffix when both suffixes types and a data core are present (VIMS and OMEGA files). Both suffixes are now returned in a single structure in this case.

2) Small fix in v_qubepds related to Band Suffix type reading.
3) v_pdspar updated to read multiline lists of values, and to filter in-line comments.

• Check that the library can read Viking, MOC, MOLA and THEMIS images, VIMS, OMEGA, VIRTIS, NIMS, ISM. THEMIS Qubes include tables with external description, and therefore are not handled. TES tables not handled.

October 2002

Dirty fixes after playing with THEMIS qubes:
• v_readpds no longer tries to read Table, Spectrum, Serie or Palette objects. The reason for this is twofold: THEMIS qube files (which include a table) do not provide the INTERCHANGE_FORMAT keyword; the original v_tascpds does not handle ^structure definitions, but requires DATA_TYPE which is optional. This may work with some data files, but this is certainly non-standard.
• v_qubepds now supports qubes in BSQ format, but assumes that no suffix is present. Does not require the SUFFIX_ITEM keyword in this case.
• v_qubepds now supports spectral_qubes objects.

June 2002

Fixed a small problem in v_conv_vax_unix.pro that precluded compilation on DecAlpha systems.

January 2002 - Version 2.3

1) Fixed a problem in v_qubepds and virtispds that returned 2D qubes or suffixes in some situations. Now, core and suffix are always 3D IDL arrays for Virtis, whatever the dimensions of the data.

2) Added instructions for use in the present doc.

October 2001 - Version 2.2

Adapted during Virtis ground calibrations in Orsay (together with EGSE PDS writer).

√ **virtispds.pro** completely rewritten. Now reformats qubes: returns label, data, suffix and HK list in a structure. HK are reformatted as elemental structures.
√ tirtispds.pro: same as above + corrects byte ordering in the data core (corrects a bug in the initial calibration EGSE software).
√ v_frommsb.pro renamed v_msbtohost.pro for consistency.

√ Small fix in v_qubepds.pro to return suffix formatted as qubes in all situations (previously, dimensions with only 1 element were degenerated, and the suffix was declared as a 2D array).

April 2001

√ v_headpds.pro modified to read long labels (> 32767 bytes) in several passes.

March 2001

√ Small modification in v_qubepds.pro to read a data qube with no suffix. The PDS doc does not tell if this is standard or not, but VIRTIS-H simulation files written by vv_writepds.pro can actually contain one qube with no suffix.
This is indicated as:

SUFFIX_BYTES = 2
SUFFIX_ITEMS = (0,0,0)

(both lines are mandatory in the label)

√ Added function V_eolpds.pro to manage EOL markers in label strings.

Oct/Dec 2000 - Version 2.1

The present version includes:

v_conv_unix_vax.pro - adapted from Astron

• Suffixes dimensions are always read in the same direction whatever the Qube order:
SX = backplane length
SY= sideplane length
SZ = bottomplane length
(they are inverted afterwards if needed).

	BIP (ISM)	BIL (VIMS)	BSQ
Sx	Band	Sample	Sample
Sy	Sample	Band	Line
Sz	Line	Line	Band

√ Changed integer conversion routines.
√ Adapted to read an empty Qube core.
√ Can read VIMS Qube files (flight data) and ISM qubes.

Changes after first comments from LESIA team (Nov. 2000):

- Fixed identification of EOL markers in label.
- Fixed MSB to LSB conversion in v_imagepds.
- Identified problem in VIRTIS_H_DM and VIRTIS_H_YACADIRE test files (images): integers are written as LSB, but described as UNSIGNED_INTEGER (implicitly MSB) in the label. This was transparent before, because the original imagepds.pro didn't performed MSB to LSB conversion. Added non-conformity processing: Forces integer type to LSB in VIRTIS_H written before Jan. 2001 (actually, the very first files for engineering model could be MSB, to be checked – files themselves cannot be modified, they're too widespread).
- Added support for VIMS files first processed with ISIS: they contain two suffixes, backplane and sideplane.

20 / 09 / 2000 Version 2.0

Complete rewriting, adapted from SBNPDS 2.0 (from PDS Small Bodies Node)

√ Offsets to objects in files are fixed.
√ Conversion of data to host encoding (MSB to LSB and vice versa, Vax-float to IEEE, but not the other way round).
√ Read Qube objects (routine v_qubepds). Reads and returns suffixes, with some limitations (see routine header).
√ Reads images properly (original version mixed up Image and Image_Histogram).

15 / 10 / 1999

VIRTISPDS.pro: upper level routine, transforms output in order to get a structure when only one object is found in the file. Added by Yann Hello.

30/09/1999 Version 1.0 for Virtis

Adapted from readPDS.pro 1998 version (from PDS Small Bodies Node).