The VIRTIS PDS/IDL software library (lecturePDS)

Issue 3.2

	NAME	FUNCTION	SIGNATURE	DATE
Prepared by	S. Erard	Archive Manager, Virtis VEx French team leader, Virtis Rosetta Planetary Science coord., OV-Paris		**8/12/2014**
Checked by	F. Capaccioni	PI, Virtis Rosetta		
	P. Drossart	coPI, Virtis VEx		
	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, Venus-Express, and other missions.

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.

GDL : Gnu Data Language
Open-source clone of IDL, in development.

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. In operation it used to decode telemetry files, check data consistency, quick look of the data and preliminary PDS formatting of data files.

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

SCET : SpaceCraft Elapsed Time
On board time. 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 from the TM, other than science data (in the present document, actually include information from both the housekeeping and event packets).

VEx : Venus-Express
Unofficial acronym for the mission / spacecraft.

VO : Virtual Observatory
An international community-based initiative aiming to allow global electronic access to astronomical data archives.

INTRODUCTION

What is it?

This library is intended to read a variety of PDS data files under IDL/GDL, in particular files containing Qube objects. It is particularly intended to read the files from the two VIRTIS experiments on board the Rosetta and Venus-Express missions, and should be used as the lower level layer in VIRTIS data processing software under IDL. From version 2.8, it is also intended to provide access to PDS data files in the frame of the EuroPlaNet-RI program, including through VO mechanisms. Version 3.0 supports a much broader scope of datasets.

This library is included in the VIRTIS Rosetta and Venus-Express archives at the PSA as the standard data access software, under the name LecturePDS. 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), with contributions from the VIRTIS teams. Port to GDL was done in the framework of EuroPlaNet-RI.

This manual describes basic VIRTIS data access. General users not interested in VIRTIS data may want to jump directly to the section “How to read other (non-Virtis) PDS files?”.

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

Distribution

The present distribution of the VIRTIS PDS library is intended for VIRTIS coIs /users to access VIRTIS data, and for users who need to access other 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 [version 1.6, April 2012]
AD4 VIRTIS Software Requirement Document [VIR-DLR-RS-003, version 2 draft 1, 6/12/2000]
AD5 VIRTIS VEx geometry files formatting [VIR-LES-SW-2268, version 1.2, July 2008]
AD6 VIRTIS Rosetta geometry files [VIR-LES-SW-2338, Draft 0.5, June 2012]
AD7 VIRTIS Rosetta EAICD v4.1 [VIR-INAF-IC-007, 24 / 3 / 2014]

RD1. Planetary Data System Standards Reference, version 3.8 [JPL Document D-7669, part 2, 27/02/2009]
RD2. ROSETTA Time Handling [RO-EST-TN-3165 Issue 1, 9 February 2004]

INSTRUCTIONS FOR USE

Installation

In the VIRTIS Rosetta and Venus-Express archives, the software is provided as a ZIP archive located in the DOCUMENT directory (according to PDS usage). Just copy the library in a directory included in your IDL path. In case of update, it is required to remove older versions from your path and restart IDL.

A directory tree can be added to the path by including the following lines in your IDL startup file (e.g., 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 standalone. In addition you may want to install the following open source libraries, which are available on the network (although they are not used by this library):
ASTRON (NASA IDL Astronomical library)
TEXTOIDL (by M. W. Craig, University of California)

The present version has been tested under IDL 7 and 8, but should be operational from IDL 5.6. From version 2.8, the library also runs under GDL, an open source clone of IDL. It requires GDL version ≥ 0.9.

How to read Virtis PDS files?

Any regular VIRTIS file 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 in batch mode, the option /SILENT inhibits all console messages.

Files supported by the routine are raw data files generated by the EGSE (after Nov. 2001) plus files produced by the regular pipelines, in particular those written with routines v_convlabel and v_geolabel (.QUB, .GEO, .PRE, .CAL)

Checking that data are from VIRTIS

If the file is not related to 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 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 describing the cube stored quantity 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 VIRTIS 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 Venus-Express/Rosetta calibrated cubes

VIRTIS 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 equally formatted as 2D cubes in output of virtispds — notice that the file itself contains a 3D PDS qube object 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 estimate (1-sigma)

result.qube_name: a 2-strings array providing the cube stored quantity 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: reconstructed 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 pixel:
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 stored quantity 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)

Beware that wavelengths and bandwidths may be provided in different units for M and H. This is documented in the PDS labels and in the EAICD documents (AD3 and AD7).

Other VIRTIS files

Other VIRTIS files include calibration files and special VIRTIS products, which can always be read with v_readpds. In some cases, virtispds.pro may also be used.
Some of these files have detached labels, in which case the label and data files may be located in different directories. The routine can be called with the name of either the label file or the data file in argument. If the working directory does not contain the data files, the path must be included in the file name.

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

TABLE data

The output structure to virtispds 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 to virtispds 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 to virtispds 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

Files read with v_readpds

The output to v_readpds is similar to the one of non-VIRTIS files, and is described below.

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 are flagged in the raw data cubes with the data type HK parameter set to value '2000'X (i.e., 2000 hexa = 8192 decimal)
They can be selected from raw data cubes with (beware of parentheses!):

```
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 most 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 the user 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.

In calibrated cubes, some codes are used to flag special cases e.g., dead pixels, saturated measurements… Theses codes have large negative values (close to -1000) which can affect scaling during a plot. The routine v_fcode replaces these values by NaN, which are filtered by PLOT and TV instructions. This routine can be applied only to arrays, not to structures:

```
d=virtispds('CTOB_05_0.CAL')
dd=d.qube
plot, dd(*, 100)        ; contains a saturation code
v_fcode, dd
loadct, 12
oplot, dd(*, 100), col=200
```

How to access Virtis operational parameters?

Observation parameters are stored in the labels and can be retrieved automatically. For instance, here is how to get the exposure time for Virtis-H (except in calibration sessions):

```
fp = v_pdspar(result.label, 'frame_parameter_desc')
ip = where(v_listpds(fp) EQ '"EXPOSURE_DURATION"')
Tint= (v_listpds(v_pdspar(result.label, 'frame_parameter')))(ip)   ; value
Tint = Tint(0)          ; turned to a scalar
Print, Tint
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 suitable 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,*,*)*where(result.suffix(m,*,*) LT 'FFFF'X)
```

HK values can be printed in hexadecimal encoding 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 Venus-Express. 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 these instrument parameters

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

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

A more direct way to retrieve the varying H exposure time in calibration sessions is:

```
hkparam = result.suffix
t_exp = v_compute_intnum(hkparam(33,0,*), hkparam(32,0,*))          ; in s
print, t_exp
```

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 can be called with two arguments. If present, the second argument is interpreted as an offset (in seconds) to be added to the first one. The result 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 Virtis 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:
When using raw data files prior to geometric computation (ie, in the pipeline), the UT estimate provided by v_scet2ut eventually relies on the time stamp provided in the TM packets. This estimate is only a rough approximation, and exact values can be derived only through the use of the Spice routines with appropriate Spice kernels. This is normally fixed when using data files from the archive.

Time from geometry files
The most reliable UTCs are available in the Venus-Express and Rosetta geometry cubes, where they are computed at each time step using Spice (reconstructed if needed). The formatting is different for the two missions, and for the H and M channels; the user is referred to the corresponding documentations (AD5 and AD6).

For Virtis-H Rosetta data: the scet is stored on two bytes in two consecutive planes (planes 22 and 23 for Rosetta):

```
geo = virtipds(file)
sz = size(geo.qube, /dim)
; SCET
ScetH = geo.qube(22,*,*) + geo.qube(23,*,*)/ 2d ^ 16d
ScetH = reform(ScetH, sz(1)*sz(2))
; UTC
```

```
; geo.qube(24,*,*) = # of days after origin, usually constant in a session
UTH = geo.qube(25,*,*)/10000./3600.     ; decimal hour of the day
print, d2dms(uth(412))     ; turn to h:m:s
```

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

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

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

where:

file_name is the name of the file. In case of detached label, it is safer to use the name of the label file (although the data file name also works). 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 suffixes in output when reading a qube

On output, result (and suf) are either structures or tables depending on the objects described in the label.

Several data objects can be described in the label, including several objects of the same type. The expected situation is to have each object described by a single name composed on the data type. Object names must be of the form PRE1_PRE2_TYPE where TYPE is a generic PDS type. For example, RED_IMAGE, SMALL_QUBE… are OK, but IMAGE_1… won't do. The tag names of the return structure are equal to the data object names. The option /ObjNum handles other cases, when the labels are ambiguous (containing several objects of the same type with generic name; this may occur in case of common detached labels for instance).

If only one image or one qube is present, result (and suf) are arrays of the corresponding size and dimensions. Table objects always produce a structure in output, because the columns may have different numeric types. If only one suffix (either sideplane or backplane) is associated to a qube, suf is a 3D array (usual case). If both types of suffix are included in the file, suf is a structure containing the two suffices (e.g., VIMS and OMEGA files). Beware that when using v_readpds to read VIRTIS qubes, the suffix is not converted to a convenient format (as the output of virtispds.pro is).

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 or FMT files, the data file must be located either in the same directory as the label, or in the working directory. If the files are located in different directories, try calling the routine from the directory containing the data files, and provide the complete label file name, including

path. If some files cannot be reached or cannot be read, the corresponding objects will be set to -1 in the output structure.

The library handles most PDS objects, although not all possible options are implemented. Besides, not all objects have been tested in details, and special problems may arise with some object types, in particular with tables (e.g., "container" objects are not supported) and images (e.g., "window" sub-objects are not supported).

Example output 1: simple qube

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 exposure times
tint =  v_listpds(v_pdspar(lbl, 'EXPOSURE_DURATION'))
print, 'Integ time SWL:',tint(0)
print, 'Integ time Vis:',tint(2)
bin =  v_pdspar(lbl, '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.

Example output 2: several images in a structure

The output variable contains all the main data objects described in the label, and therefore may be a complicated structure. The structure tag names are derived from the data object names in the label.

```
result = v_readpds(' AMI_LE1_R02069_00018_00018.IMG') ; AMIE raw image
help , result
** Structure <266ea88>, 3 tags, length=557060, data length=557060, refs=1:
  IMAGES          LONG                 2
  BROWSE_IMAGE    INT       Array[128, 128]
  IMAGE           LONG      Array[512, 256]
```

The structure tag names can be accessed this way:

```
        print, tag_names(result)
IMAGES BROWSE_IMAGE IMAGE
```

and the number of tags this way:

```
        print, N_tags(result)
```

The structure tags can also be named in order using the option ObjNum. This allows reading files that contain several objects with the same name (this is mostly intended to preserve compatibility with previous versions of the library, but can also be required with common detached labels):

```
        result = v_readpds(' AMI_LE1_R02069_00018_00018.IMG',/Objnum)
        help , result
** Structure <22f2c728>, 3 tags, length=557060, data length=557060, refs=1:
   IMAGES        LONG              2
   IMAGE0        INT       Array[128, 128]
   IMAGE1        LONG      Array[512, 256]
```

As discussed above, the result is a simple data object (not a structure) when only one image or one cube is included in the file.

Example output 3: table

The output variable is always a structure in this case, because the columns may have different data types. The output structure contains a vector of column names, and a table of structure containing the data columns.

```
        result = v_readpds('examplesPDS/psr1_01.lbl')
        help , result
        ** Structure <7700178>, 2 tags, length=219552, data length=219552, refs=1:
          COLUMN_NAMES    STRING    Array[27]
          TABLE           STRUCT    -> <Anonymous> Array[1]
        IDL> help, result.table
        ** Structure <185f208>, 28 tags, length=219120, data length=219120, refs=2:
          COLUMN_NAMES    STRING    Array[27]
          COLUMN1         LONG      Array[1072]
          COLUMN2         LONG      Array[1072]
          ...
          COLUMN27        DOUBLE    Array[1072]
```

The output can be reformed and simplified after reading if data types are consistent:

```
        names = result.column_names
        Ncol = N_elements(names)
        table = result.table.column1
        for j=2, Ncol do table = [[table],[result.table.(j)]]
        table = transpose(table)
```

In case of error

• Only regular Virtis *data* and *geometry* files (.QUB, .GEO, .PRE, .CAL) can normally be read with virtispds.
The VIRTIS archive includes other types of files, e.g. calibration images and tables. Only some of those may be read with virtispds, but all are supported by v_readpds.

• In case of detached labels, try to provide the name of *label* file.

• In case of I/O error when reading a detached label, check that you run the routine from the directory where the *data* file is located, and provide the whole path to the file (this may occur in particular with index files). If the error persists, try moving both the data and the label in the same directory.

DETAILS

Encoding of special information in Virtis files

Labels

• Acquisition parameters are usually constant during an 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 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).

-999.99	is used to flag an unknown / non-defined value where a digital value is expected. The standard PDS use of alphabetical codes (UNK…) would result in reading errors. Ex: DECLINATION when the spacecraft is not in inertial mode.
-1	for exposure time. It is used to flag non-constant exposure time during a sub-session. This is expected to occur only during calibration sessions. In this case, the actual values can be retrieved from the sideplane for every time step (see [AD 3]). Used in the first and third fields of FRAME_PARAMETER (resp. EXPOSURE_DURATION and EXTERNAL_REPETITION_TIME).
-1	for ORBIT_NUMBER. It is used to identify pre-orbital observations with Virtis VEx. The orbit number is encoded as 9999 in the corresponding file names. Orbit 0 corresponds to the Venus Orbital Insertion phase.
0	for SCIENCE_CASE_ID (Virtis VEx only). This extra Science Case code is used to flag data acquired when Venus is not observed. This includes calibration sessions and observations of targets different from Venus.

Raw data qube cores:

Data are stored in DN but may be negative, because a dark current estimate is already subtracted on board. The following codes are introduced through the standard PDS keywords for the Qube object:

-32768	Default value for NULL, and minimum valid value
32767	Maximum valid value, including saturation

Raw data qube suffix:

65535	(='FFFF' hexa) Flags missing HK measurements (this value is not normally used by any HK parameter)

Calibrated data qube cores:

Data are stored in physical units but may be negative, because of fluctuations on the dark current.
For Virtis VEx, the following codes are introduced through the standard PDS keywords for the Qube object:

-999	Minimum valid radiance value
-1004	No available data (including dead pixels)
-1003	Low representation saturation
-1002	Low instrumental saturation
-1001	High representation saturation
-1000	High instrumental saturation (physical saturation of the FPA)

Geometry files:

Different quantities are stored in the same qube core, in various physical units. The file structure is described in detail in document [AD 5]. Special codes used in this context are:

-20000	Missing value for elevation (missing in the GTDR)
100000	This offset value is added to surface elevation for limb observations. Larger values identify situations when the line of sight does not intercept the target. The offset must then be subtracted from the written value to retrieve the tangent altitude.
-2147483648	(='80000000' hexa) Quantity cannot be computed due to missing HK parameter.

Content of sideplanes in VIRTIS PDS raw data files

The format of the Virtis PDS archive is too tricky to be described here in details. It is completely described in the documents AD1/AD2, which apply to both Rosetta and Venus-Express.

Most raw 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 raw 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 Venus-Express 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 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. Incidence angle is equal to solar zenithal angle.
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. Incidence angle is equal to solar zenithal 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.)	J2000 reference frame
For M:	**1 supplementary plane**	
33	One frame-common plane	Provides 10 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 as 10,000 x seconds (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	In the instrument frame

	XY plane (counted from 0° at X axis)	
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 value contains the number of days since Jan. 1st, 2000, the second value contains the time of the day as 10,000 x seconds (starting from 0h)
37-38	Subspacecraft coordinates (longitude/latitude)	
39	Slit orientation	Relative to the pixel normal at footprint center
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.

Content of VIRTIS Rosetta geometry files

A similar scheme is currently being defined for Rosetta, starting with the Mars and Earth flybys. The present version provides support for Rosetta geometry files with a preliminary format, as defined in document AD6 (Table 4).

Table 4: contents of Virtis Rosetta geometric files, for observations intercepting the surface

Plane #	Parameter description	Comment
1-4	Longitudes of 4 pixel footprint corner points	Geometrical projection on the DTM
5-8	Latitudes of 4 pixel footprint corner points	Geometrical projection on the DTM
9-10	Longitude & latitude of pixel footprint center	Geometrical projection on the DTM
11-13	Incidence, emergence & phase at footprint center, relative to local normal	Angles relative to the DTM
14-15	Incidence & emergence at footprint center, relative to Mars reference ellipsoid	Not accounting for topography. Incidence angle is equal to solar zenithal angle.
16-17	Incidence & emergence at footprint center, relative to Mars centre direction	
18	Surface elevation (footprint center)	From DTM
19	Slant distance (line of sight from spacecraft to surface ellipsoid at pixel center)	Does not include topographic model
20	Local time at footprint center	
21-22	Right ascension and declination of pointing direction.	J2000 reference frame
For M:	**1 supplementary plane**	
23	One frame-common plane	Provides 10 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 as 10,000 x seconds (starting from 0h)
	5-6 Sub-spacecraft coordinates (longitude/latitude)	
	7-8 Sine and cosine of M mirror angle	Converted into sin/cos values from HK
	Sun direction: 9: angle between Sun direction and Virtis Z axis; 10: azimuth of Sun direction in instrument XY plane (counted from 0° at X axis).	
For H:	**9 supplementary planes**	
23-24	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)
25-26	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 as 10,000 x seconds (starting from 0h)
27-28	Sub-spacecraft coordinates (longitude/latitude)	
29	Slit orientation	Relative to the pixel normal at footprint center
30-31	Sun direction: 40: angle between Sun direction and Virtis Z axis; 41: azimuth of Sun direction in instrument XY plane (counted from 0° at X axis).	

VIRTIS VEx files names

The final file name scheme for Virtis Venus-Express 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
Preliminary geometry files (computed from predicted kernels) have extension .PRE. These files are normally replaced by *.GEO as soon as the reconstructed kernels are available.

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 H calibrated dark current files. Calibrated dark current files are provided (at least for H) so that a refinement of dark interpolation is possible independently of the complete calibration procedure.

VIRTIS Rosetta files names

The final file name scheme for Virtis Rosetta is described in more details in document AD7. In short:

Fn_scet.EEE
where:
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

n_scet is the onboard time of start acquisition (n = resync #, scet = integer part of the spacecraft elapsed time from the on-board clock, typically on 1+8 digits).

EEE is the file extension: QUB for raw data, CAL for calibrated data files, DRK for H calibrated dark current files, GEO for geometry files (PRE for preliminary geometry files). Beware that corresponding files (e.g., H data and dark current) may have different root names.

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 other PDS data files

v_headpds.pro
    Low level routine to read a PDS label
v_pdsvalues.pro
    Low level routine to parse a PDS label, in test
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_fcode.pro
    Utility routine to filter special codes in calibrated data cubes
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. The encoding of SCET on 3 bytes may be specific to Virtis
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 VIRTIS 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, including M calibrated files (revised version, 2008). Also updates PDS labels from old EGSE format (VEx and Rosetta).
To be used with associated label templates. Only available to the team.
v_geolabel.pro
Formatting routine to write Virtis geometry files as computed by GeoVirtis (VEx) and GeoRos (Rosetta) libraries in Meudon.
To be used with associated label templates. Only available to the team.

Older routines, removed:
v_convlabel2.pro
Temporary version for M calibrated files (VEx and Rosetta).
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.

License

A (shorter) modification history

Version 3.2 (Aug 2014)

• Fixes to v_convlabel and v_geolabel to support recent updates in all Rosetta files (including added and removed keywords).
• Fix to v_compute_intnum (for long exposure times on Rosetta).
• Updated v_pdspar to remove quotes around critical keywords on option (new PSA requirements for Rosetta).
• Updated v_readpds to preserve last dimension for degenerated raw H cubes (x,x,1): nominal mode/64 spectra exactly (occurs on Rosetta).
• Added option in v_str2num to force numeric conversion in all cases + added routine v_pdsvalues to parse a complete label in one step (in test).

Version 3.1 (June 2013)

• Minor fixes to virtispds to support possible variations in Rosetta files, as well as older versions.
• v_geolabel and v_convlabel modified to update all geometry keywords in Rosetta raw data & geometry files + fix labels according to recent PSA requirements. Also handle possible changes in H pixel map + default to standard pixel map in flight for Rosetta.
• v_qubepds: Implemented qube suffices in BSQ mode (HST Mars).
• Minor fixes in doc as per Rosetta archive review + addition of special codes.

Version 3.0 (Nov. 2012)

1) Full support of multiple objects. Several instances of all main types are supported provided they have different names. Assumes standard PDS object names.
2) Improved support of specific objects and data sets, based on intensive testing (in particular for complex tables, arrays, and images). Checked under IDL 7/8 and GDL 0.9.2.
3) Updated doc.

• Modifications in v_readpds to handle several objects of the main types, as well as PDS' derived object names. Object names must be of the form PRE1_PRE2_TYPE where TYPE is a generic PDS type. For example, RED_IMAGE, SMALL_QUBE... are OK (but IMAGE_1... won't do) [all this is implicit in doc, and derives from PSA interpretation/requirements, and practice].
Can currently load images, qubes, tables, arrays, and collections. The default is to use object names in the output structure (option /ObjNum not set).
• Option /ObjNum maintains compatibility with previous calling routines, and also allows reading objects with duplicate names (like version 2.8 did).

If data consist in a single image or a single qube, the output is a simple array, for compatibility with previous versions. Support for object types with compound names (e.g. SPECTRAL_QUBE) is also implemented in principle.
• v_objdef output extended (additional structure tags provide specific name, generic object type, and object level).
• Routines v_imagespds and v_qubepds now read objects individually.
• In case of error when reading an object, the standard behavior is now to return -1 for this object and continue.

• v_btabpds now reads multi-element columns correctly from binary tables + handles mismatches in column number (e.g., M3).
• More or less dummy data types (string of bytes) are now supported in binary tables: “N/A” and “bit_string” + 6-bytes floats (not PDS compliant but used e.g. in M3 files).
• v_atabpds now fully supports multi-element columns in ascii tables.
• Implemented support of prefix and suffix in ascii and binary tables, available in output of v_readpds (e.g., Lunar Prospector RS).
• v_imagepds now supports image prefix and suffix (e.g., HRSC). They are not available in output of v_readpds, though.
• v_imagepds now supports predefined display directions.
• v_qubepds now supports BSQ with sideplane or backplane, and Band/Line/Sample storage order (standard ISIS, e.g., Cassini UVIS) — only with no suffix.
• Improved file name support: always accepts to open files from another directory; supports spaces in file path on Mac (from drag/drop in Terminal).
• v_btabpds no longer tries to load label extensions (done in v_headpds) + fixed swap from LSB data.
• v_bmaskpds now handles extra spaces in binary masks (was an issue for AMIE).
• Array and Collection routines improved and much more robust.

3) Known limitations:
- Does not handle Qubes with more than 3 dim + suffix not implemented in BLS mode.
- Does not support Window sub-objects in images.
- Does not support Container objects in tables, nor bit_columns in binary tables (will quit).
- Implementation of Collection and Array objects still old fashioned, not checked in details.

Version 2.9 (Aug-Nov. 2012)

Development version, not released.

Version 2.8.2 (March 2012)

This is the last version that supports several objects of same type with the same name by default (pointers and descriptions are associated by order of appearance).

• Larger fix in virtispds to read geometric files:
Now always returns a 2D cube for H, Nplane x Nspectra (was Nplane x 1 x Nspectra or Nplane x Nspectra x1 depending on nominal or backup mode).

Impact to be checked in detail...
• Modif in v_pdspar to return correct object pointers in large files (type needs to be 'long')
• Modif in v_pdspar to support "=" separators in labels – relaxes an old constrain on label formatting, and allows access to more data sets. TBC, though: could cause major issues in some situations (fixed on Virtis).

Version 2.8.1 (Oct 2010)

Small fix in virtispds to read geometric files.

Version 2.8 (Sept 2010)

1) Now fully compliant with GDL, an open-source clone of IDL. GDL was updated in parallel to implement the IDL functionalities used by the library (GDL minimum required version = 0.9).
- Replaced findfile by file_search in all routines for GDL compatibility. This does not change compatibility with older versions of IDL (≥5.5 still required, ≥5.6 still favored).
- Data access + swap mechanism normalized for all object types. Removed last file accesses through ASSOC.
- Tested with examples distributed with readpds library from SBN, OK.

2) Other changes
- Fixed binary tables access
- Small fix in v_atabpds for non-stream mode (corrected an error in previous update).
- Added tag Qube_name for Virtis raw data files in output of virtispds
- Added support for ARRAYS and COLLECTIONS in v_readpds + fixes in dedicated routines (heavy changes for binary arrays)
- v_str2num now supports input strings using binary, hexa, and octal PDS encoding.
- Complete and standard implementation of bitmasking in v_bmaskpds (may impact AMIE reading).
- Fixed v_translatehk for VIRTIS-M mirror angle conversion.
- Implemented new format for M calibrated data files, which includes a single cube with sideplane & bottomplane (updated v_convlabel with v_convlabel2 mechanism).
- Updated virtispds with final Rosetta geometry file format.
- Overall revision of this doc.

Version 2.7.5 (Nov 2008)

1) Fixes to the main library:
- Minor fix in v_qubepds & v_imagepds to open detached labels.
- Both virtispds & v_readpds now first look for a detached label with the same name root. This allows to handle the data file when it is provided in argument, but does not prevent to read non standard files. Intended to support derived VEx data products.
- Changed parsing of numerical values in v_str2num. No longer uses IDL error handling system, but uses regular expressions instead. This is intended for portability to GDL.

- Implemented support for ascii files in stream mode in v_atabpds (one record = one line, variable length). Intended to read the MRO spectral library; works fine, but some files have incorrect data type in their labels.
- Replaced assoc by readu in v_atabpds for compatibility with GDL.

2) Fixes/modif to writing routines:
- Fixed padding of data area to nominal length for Qcalib mode in v_convlabel and v_convlabel2.
- Added path to new vvx archive server in Rome in v_convlabel and v_convlabel2.
- v_convlabel2 fixed to update M calibrated files from older (current) format to alternate one.
- Update of v_geolabel both to support Rosetta/Steins flyby and to implement last PSA specifications.

Version 2.7.4 (July 2008)

1) Modification in v_atabpds to handle files in STREAM mode (used for corrupted H darks lists + required for multiple ascii tables).
2) v_convlabel, v_convlabel2 and v_geolabel modifed to pad the data area with 0 ascii (file length now exactly equals record_length * record_number). OK when reading.

Version 2.7.3 (Feb. 2008)

1) virtispds updated to filter some versions of the geometry files for H in nominal mode. Files generated by the new EGSE with geovirtis up to 3.2 have incorrect SCET interpolation, and therefore all quantities are computed at the wrong time, which is sensitive for limb studies (they are shifted by up to 1 repetition time). Trying to load such files will result in message requesting to download updated versions.

2) Implemented support for a possible future format of M-calibrated cubes (to meet a possible PSA/PDS request for Rosetta):
- v_qubepds updated to read cubes with bottomplanes in BIL and BIP storage (still no support for BSQ with suffix). Now allows to have different suffixes using different byte numbers (but stored in the same SUFFIX_BYTE area, along PDS guidelines).
- virtispds updated to handle this new format, in addition to the current one. The outputs are identical.
- v_convlabel updated to write M calibrated files with a bottomplane instead of a calibration cube. This is done by another version, currently called v_conlabel2; the older format is still written by v_convlabel. The SCET backplane is now stored in long integers (with only two bytes used out of 4 — this is required by the unique SUFFIX_BYTE keyword, and the usual ambiguity on PDS specifications).

3) Added Table 4 (contents of Rosetta preliminary geometry files)

Version 2.7.1 (Dec. 2007)

1) Geometry fixes/updates
• Tentative fix for incorrect VEx geometry files in virtispds (local time correction for files generated after 1/5/2007 [with SPICE N61] + geovirtis up to version 2.x).
• virtispds now supports Rosetta preliminary geometry files (names and coefficients), as defined for Mars and Earth flybys.
• v_geolabel updated to write Rosetta geometry files from the Georos library (Mars and Earth only so far).

2) Various
• Fixed v_atabpds when table name is provided in addition to column names (GEO_VENUS.LBL as written with PSA's geolib).
• updated v_fcode to filter all special codes in VEx M calibrated files, and to substitute any arbitrary value to special codes.
• updated v_listpds to handle non-ordered PDS lists (provided in round brackets). Used to handle SOFTWARE_VERSION_ID keyword.
• Updated VEx time origin in v_scet2ut (still provides only a rough conversion from SCET to UTC).
• Fixed Table 3 in this doc.
• Added support for Rosetta calibration scheme in v_pdshk.
• Various modifications in v_geolabel and v_convlabel to make them consistent with the VEx PSA archive, first delivery.
• Checked to handle VIMS cubes properly.

Version 2.7 (June 2007)

1) Support for new EGSE files and actual archive labels (after major EGSE update):
• Update of Virtis HK handling routines, adapted to new EGSE file labels (Florence).
• Updated v_convlabel and v_geolabel to handle new EGSE files + inserted onboard software reference in labels, and handles first VEx on-board software update (Rosetta reference to be inserted).
• Added routine v_fcode to filter special codes in calibrated data.

2) Various fixes and improvements:
• First CVS version in LESIA (from 6/6/2007)
• Tentative fix for a rare IDL-OS issue when writing updated/calibrated/geometry files.
• Cryptic fix in v_imagepds to handle errors on open file smoothly (required for M ITF files on Windows systems).
• Several fixes in v_convlabel and v_geolabel, including a fix for line length (Florence).
• Fixed v_scet2ut to parse SCET correctly when reset number is present.

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).
• Added reverse 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 packets and suffixes) to number of seconds (as double precision floating point).
• Removed routine v_timepds, never used (from SBNPDS library).

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).

Oct/Dec 2000 - Version 2.1

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

- Fixed identification of EOL markers in label.
- 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).

30/09/1999 Version 1.0 for Virtis

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