VIRTIS-VEX Data Workshop

R. Politi, G. Piccioni, D. Grassi, S. Erard

April 2, 2014

Contents

Contents iii

1 The LecturePDS toolkit 1
- 1.1 Installing the toolkit . . . 1
- 1.2 Using the toolkit . . . 1
  - Exercise 1 . . . 1
  - Exercise 2 . . . 3

2 Virtis-M Data Approach 5
- 2.1 Display the data . . . 5
  - Exercise 3 . . . 5
  - Exercise 4 . . . 6
  - Exercise 5 . . . 8
- 2.2 The Geometry Qubes . . . 8
  - Exercise 6 . . . 8
- 2.3 Data Projection . . . 10
  - Exercise 7 . . . 10
  - Exercise 8 . . . 13

3 Virtis-H Data Approach 15
- 3.1 Exercises . . . 15
  - Exercise 9 . . . 15
  - Exercise 10 . . . 15
- 3.2 Correction . . . 16
  - 3.2.1 Exercise 9-2 . . . 16
  - 3.2.2 Exercise 9-3 . . . 16
  - 3.2.3 Exercise 9-4 . . . 17
  - 3.2.4 Exercise 9-5 . . . 17
  - 3.2.5 Exercise 9-6 . . . 17
  - 3.2.6 Exercise 9-7 . . . 18
  - 3.2.7 Exercise 9-8 . . . 18
  - 3.2.8 Exercise 10-2 . . . 19
  - 3.2.9 Exercise 10-3 . . . 19
  - 3.2.10 Exercise 10-4 . . . 19
  - 3.2.11 Exercise 10-5 . . . 20
  - 3.2.12 Exercise 10-6 . . . 20

A Informations **23**
A.1 The Qube header 23
A.1.1 Header keywords 23
A.1.2 Header example 26
A.2 Geometry cubes 29

Chapter 1

The LecturePDS toolkit

1.1 Installing the toolkit

The Virtis team provides a library to access the data in PDS format. It runs under IDL and it opensource clone, GDL. It is also used to access the files under ENVI. The toolkit is in the Software directory of the VIRTIS-VEX dataset.

This tutorial is intended for use with the software documentation provided with the archive (VVX-LES-SW-2264).

- An older version of the library is included in the archive itself
- Updates are available here: http://voparis-europlanet.obspm.fr/othertool.shtml (current version is 3.1)
- Install the library somewhere on your disk and include it in your IDL path:

```
defsysv,'!VIR_DIR',"~/IDL/userlib/virtis"
!PATH = !PATH + path_sep(/search) +EXPAND_PATH('+'+!VIR_DIR)
```

1.2 Using the toolkit

The main program of the LecturePDS toolkit is virtisPDS.

```
cb=virtispds(dialog_pickfile())
```

The result *cb* is an IDL structure which content depend from the file type we are opening.

Exercise 1. *Opening a VIRTIS-M calibrated file and display the field informations of the structure.*

The main code for this purpose is:

```
cd, '~/virtis_data/MTP018/VIR0499/CALIBRATED' ; go to the data directory
cubeFile='VI0499_01.CAL'                      ; store the file name
qube=virtispds(cubeFile)                      ; open the cube
```

The output will be:

```
IDL> cd, '~/virtis_data/MTP018/VIR0499/CALIBRATED'
IDL> cubeFile='VI0499_01.CAL'
IDL> qube=virtispds(cubeFile)
Reading label VI0499_01.CAL
Number of objects found:            2

% VIRTISPDS: File in use: VI0499_01.CAL
** Structure <8f518bc>, 8 tags, length=125636164, data length=125636162:
   LABEL           STRING    Array[157]
   TABLE           FLOAT     Array[432, 256, 3]
   QUBE_NAME       STRING    Array[2]
   QUBE_DIM        LONG      Array[3]
   QUBE            FLOAT     Array[432, 256, 281]
   SUF_NAME        STRING    Array[3]
   SUF_DIM         LONG      Array[2]
   SUFFIX          UINT      Array[3, 281]
IDL>
```

In this case the content of the structure is shown on the standard output directly from **virtispds**. If you don't like print the information, for batch purpose, you can use the keyword **/Silent**.

```
IDL> qube=virtispds(cubeFile,/Silent)
IDL>
```

In this case to display the fields of the structure you must use the command **help**.

```
IDL> help, qube,/Structure
** Structure <7be9f8>, 8 tags, length=125636816, data length=125636810:
   LABEL           STRING    Array[157]
   TABLE           FLOAT     Array[432, 256, 3]
   QUBE_NAME       STRING    Array[2]
   QUBE_DIM        LONG      Array[3]
   QUBE            FLOAT     Array[432, 256, 281]
   SUF_NAME        STRING    Array[3]
   SUF_DIM         LONG      Array[2]
   SUFFIX          UINT      Array[3, 281]
IDL>
```

Exercise 2. *Open a calibrated and geometric file and display the fields.*

We will open the file VI0499_01.CAL and the correspondig geometric file VI0499_01.GEO.

```
IDL> cd, '~/virtis_data/MTP018/VIR0499/'
IDL>  cubeFile='VI0499_01.CAL'
IDL>  geoFile ='VI0499_01.GEO'
IDL>  qube=virtispds('CALIBRATED/' + cubeFile, /Silent)
IDL>  geo =virtispds('GEOMETRY/' + geoFile, /Silent)
IDL> help, qube, /Structure
** Structure <7d84a8>, 8 tags, length=125636816, data length=125636810:
   LABEL           STRING    Array[157]
   TABLE           FLOAT     Array[432, 256, 3]
   QUBE_NAME       STRING    Array[2]
   QUBE_DIM        LONG      Array[3]
   QUBE            FLOAT     Array[432, 256, 281]
   SUF_NAME        STRING    Array[3]
   SUF_DIM         LONG      Array[2]
   SUFFIX          UINT      Array[3, 281]
IDL> help, geo, /Structure
** Structure <7bf098>, 5 tags, length=9498064, data length=9498064:
   LABEL           STRING    Array[115]
   QUBE_NAME       STRING    Array[33]
   QUBE_COEFF      FLOAT     Array[33]
   QUBE_DIM        LONG      Array[3]
   QUBE            LONG      Array[33, 256, 281]
IDL>
```

Chapter 2

Virtis-M Data Approach

2.1 Display the data

In the next two exercises we will display two easy program to display the VIRTIS data.

Exercise 3. *Display a spectrum.*

```
1  Pro exercise3
2    sPath= '~/virtis_data/MTP018/VIR0499/'
3    sCubeFile='VI0499_01.CAL'
4    oCube=virtispds(sPath + 'CALIBRATED/' + sCubeFile, /Silent)
5    uSample=50
6    uLine=50
7    oPlt=Plot($
8      oCube.table[*,0,0], $
9      oCube.qube[*,uSample,uLine], $
10     XStyle=1, $
11     XTitle="Wavelength [$\mu m$]", $
12     YTitle=oCube.qube_name[0]+" ["+oCube.qube_name[1]+"]" $
13   )
14 End
```

The wavelengths of the spectrum is extracted from the first bottom plane in the *line 8* of the program.

```
8  oCube.table[*,0,0], $
```

The selection of the spectrum is done using the pixel coordinates *uSample* and *uLine* (unsigned integers) at the *line 9*.

Rember: the pixel (0,0) in IDL is at the bottom left corner of the image.

```
9  oCube.qube[*,uSample,uLine], $
```

As example at the *line 12* it is shown the use of the field *qube_name* of the data structure in which are stored the the name and the units of the data archived in the cube.

```
12 YTitle=oCube.qube_name[0]+" ["+oCube.qube_name[1]+"]" $
```

The output of the exercise is shown in Figure 2.1.

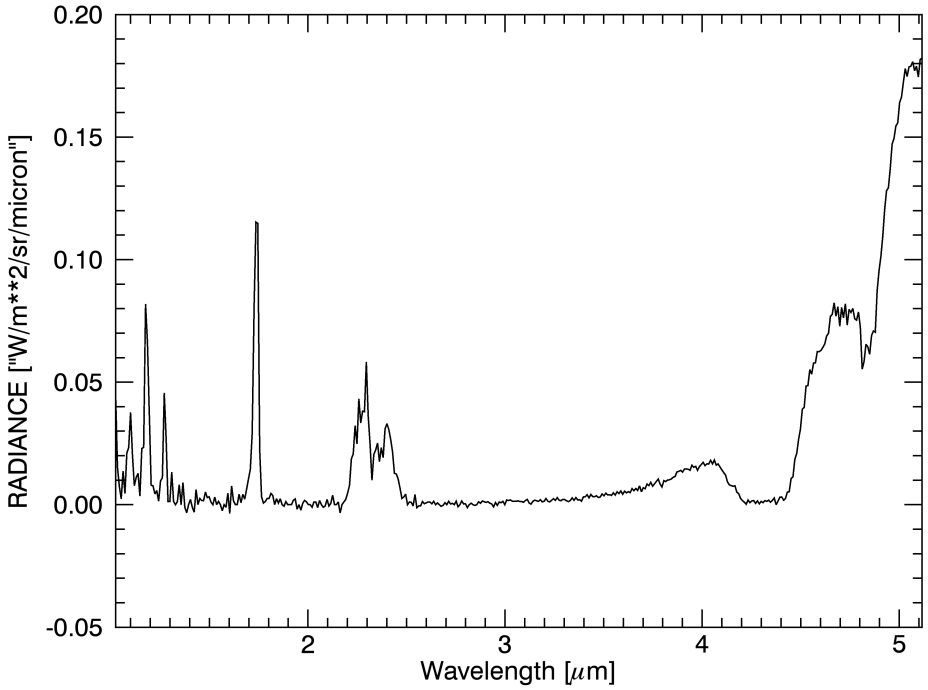

Figure 2.1: The output of the program exercise3.pro

Exercise 4. *Display an image.*

Now we will display an image from the same cube.

```
1  Pro exercise4
2    sPath= '~/virtis_data/MTP018/VIR0499/'
3    sCubeFile='VI0499_01.CAL'
4    oCube=virtispds(sPath + 'CALIBRATED/' + sCubeFile, /Silent)
5    uBand=380
6    oImg=Image( $
7      Reform(oCube.qube[uBand,10:*,*]), $
8      Min_Value=0, $
9      Dimensions=[ $
10       oCube.qube_dim[1], $
```

```
11        oCube.qube_dim[2]   $
12      ] $
13    )
14  End
```

As first step we need fix the band that we wont to display (*line 5*), storing the band number in the variable *uBand* (unsigned integer).

Remember: If you wont to display the wavelength corresponding to the band you can use the command:

```
1 Print,oCube.table[uBand,0,0]
```

In the *line 7* we transform a 3D object in a 2D matrix.

```
7 Reform(oCube.qube[uBand,*,*]), $
```

The first 8 samples of the CCD are blind, so are setted to negative values (-1004). For this reason at the *line 8*, we set to 0 the minimum value to display. To obtain an image with the rigth size we use the cube field *qube_dim* (*lines 10-11*) in which are stored the core dimensions of the cube.

In many cases in the last lines of the image there is a repetionsion of the scan of the provious area. In this case at the scanning mirror go back to the original position and go on, acquiring the same region of the first lines giving us information on the evolution of the scenario.

The output is shown in Figure 2.2.

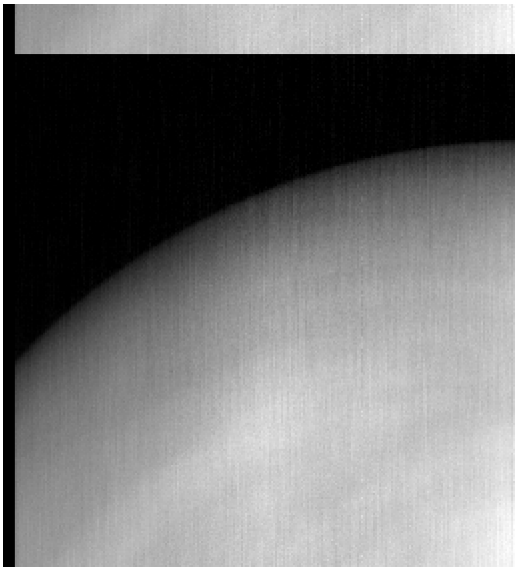

Figure 2.2: The output of the program exercise4.pro

Exercise 5. *Plot a full range spectrum.*

Similar to the exercise 3. In this case we open the cubes corresponding to channels M-VIS and M-IR.

```
1  Pro exercise5
2    sPath= '~/virtis_data/MTP010/VIR0268/'
3    sCubeFileIR='VI0268_04.CAL'
4    oCubeIR=virtispds(sPath + 'CALIBRATED/' + sCubeFileIR, /Silent)
5    sCubeFileVIS='VV0268_04.CAL'
6    oCubeVIS=virtispds(sPath + 'CALIBRATED/' + sCubeFileVIS, /Silent)
7    uSample=50
8    uLine=50
9    oPltI=Plot($
10     oCubeIR.table[*,0,0], $
11     oCubeIR.qube[*,uSample,uLine], $
12     XStyle=1, $
13     XTitle="Wavelength [$\mu m$]", $
14     YTitle="Radiance [$Wm^{-2}sr^{-1}\mu m^{-1}$]", $
15     Min_Value=0 $
16   )
17   oPltV=Plot($
18     oCubeVIS.table[*,0,0], $
19     oCubeVIS.qube[*,uSample,uLine], $
20     Min_Value=-0.1, $
21     Color='red', $
22     /Overplot $
23   )
24 End
```

At the *line 14* we do not use the internal units just for esthetic reasons.
The output is shown in Figure 2.3.

2.2 The Geometry Qubes

In this qubes are stored the geometric information relative to the data. the description of the fields is reported in the section A.2.

Exercise 6. *Display a Lat/Lon grid.*

As exmple of the use of the geometry qubes we will sovraimpose to the image a Lat/Lon grid.

```
1  Pro exercise6
2    sPath= '~/virtis_data/MTP018/VIR0499/'
3    sCubeFile='VI0499_01.CAL'
4    oCube=virtispds(sPath + 'CALIBRATED/' + sCubeFile, /Silent)
5    sCubeFileGEO='VI0499_01.GEO'
6    uBlind=8
7    uBand=380
8    oCubeGEO=virtispds(sPath + 'GEOMETRY/' + sCubeFileGEO, /Silent)
9    fMirrorCos= Reform(oCubeGEO.qube[32,6,*])
10   uIndex=Where( $
```

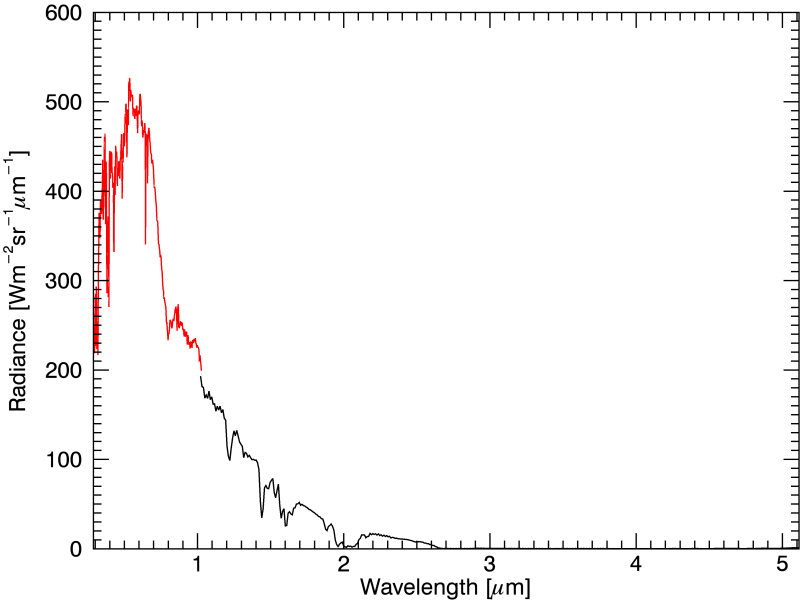

Figure 2.3: The output of the program exercise5.pro

```
11      fMirrorCos EQ Max(fMirrorCos) $
12    )
13    uStop=uIndex-1
14    fLon=Reform(oCubeGEO.qube[24,uBlind:*,0:uStop]* $
15      oCubeGEO.qube_coeff[24])
16    fLat=Reform(oCubeGEO.qube[25,uBlind:*,0:uStop]* $
17      oCubeGEO.qube_coeff[25])
18    oImg=Image($
19      Reform(oCube.qube[uBand,uBlind:*,0:uStop]), $
20      Dimensions=[ $
21        oCube.qube_dim[1]-uBlind,$
22        uStop $
23      ] $
24    )
25    oLat=Contour(fLat,/Overplot,C_Label_Show=1)
26    oLon=Contour(fLon,/Overplot,C_Label_Show=1)
27  End
```

In this case we do not display the blind part of the slit end the lines repetition at the end of the image. In order to exclude the blind samples we set, at *line 6*, the variable *uBlind* (unsigned integer) to 8, the number of the blind samples.

For the lines repetition we use the geometric file and estract the value of the cosine of the scan mirror angle. The value of the angle of the scan mirror is shown in figure 2.4 and is obtained by the command:

```
1   oTest=Plot(90-ACos(fMirrorCos/1000.)*180./!pi)
```

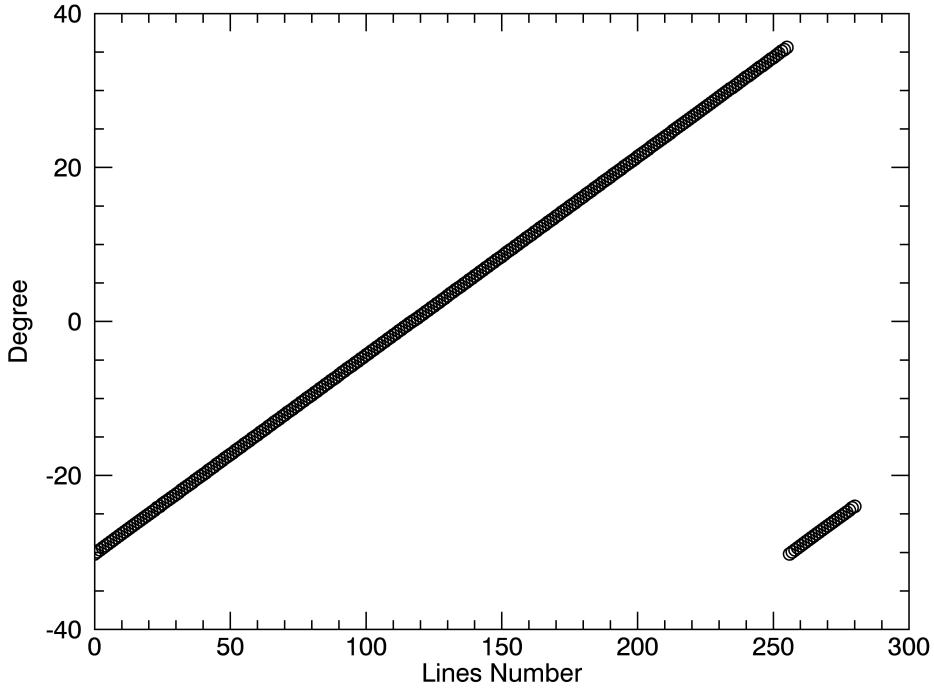

Figure 2.4: The positions of the scan mirror during the acquisition

We stop our image before the return to the start position of the scan mirror. The number of the last line is detected by the code:

```
9   fMirrorCos= Reform(oCubeGEO.qube[32,6,*])
10  uIndex=Where( $
11    fMirrorCos EQ Max(fMirrorCos) $
12  )
13  uStop=uIndex-1
```

The output is shown in Figure 2.5.

2.3 Data Projection

Exercise 7. *Footprint projection.*

Using the geometric information we can project to the planet the footprint of the qube.

```
1 Pro exercise7
2   sPath= '~/virtis_data/MTP018/VIR0499/'
3   sCubeFileGEO='VI0499_01.GEO'
4   oCubeGEO=virtispds(sPath + 'GEOMETRY/' + sCubeFileGEO, /Silent)
```

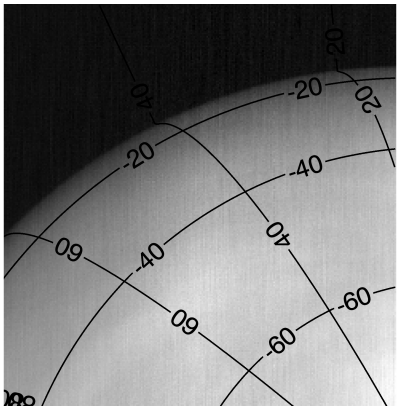

Figure 2.5: The output of the program exercise6.pro

```
  graph=map( $
    'PolarStereographic', $
    FILL_COLOR='light blue', $
    LIMIT=[-90,-180,0,180], $
    margin=[0.05,0.05,0.05,0.05] $
  )
  fLat=Reform(oCubeGEO.qube[25,*,*]*oCubeGEO.qube_coeff[25])
  uLatSz=Size(fLat,/Dimensions)
  fCubeTemp= Reform(oCubeGEO.qube[32,6,*])
  uIndex=Where( $
    fCubeTemp EQ Max(fCubeTemp), $
    uCnt $
  )
  If(uCnt EQ 1) Then uStop=uIndex-1 Else uStop=uLatSz[1]-1
  fLatNew=[ $
    Reform(fLat[0,0:uStop]), $
    Reform(fLat[*,uStop]), $
    Reform(fLat[uLatSz[0]-1,0:uStop]), $
    Reform(fLat[*,0]) $
  ]
  fLon=Reform(oCubeGEO.qube[24,*,*]*oCubeGEO.qube_coeff[24])
  uLonSz=Size(fLon,/Dimensions)
  fLonNew=[ $
    Reform(fLon[0,0:uStop]), $
    Reform(fLon[*,uStop]), $
    Reform(fLon[uLonSz[0]-1,0:uStop]), $
    Reform(fLon[*,0]) $
  ]
  oPlt=Plot( $
    fLonNew, $
    fLatNew, $
    Symbol='.', $
    Sym_Color='red', $
    /Overplot, $
    Line='none', $
    Color='red' $
  )
End
```

The output is shown in Figure 2.6.

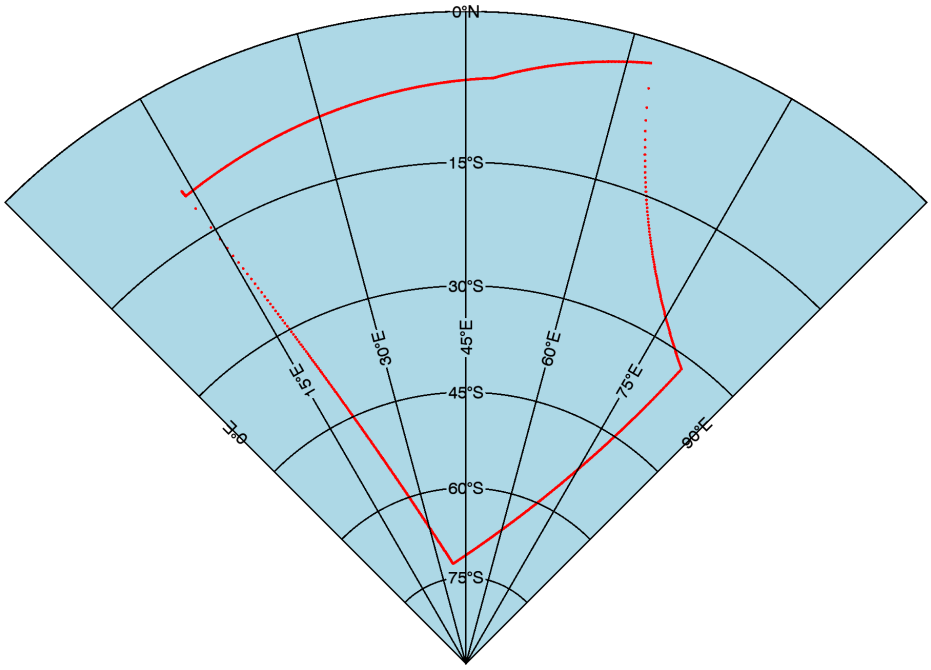

Figure 2.6: The output of the program exercise7.pro

Exercise 8. *Image projection.*

As last exercise we will project a band of the cube using the geometric information described in the prevous exercises.

```
Pro exercise8
  sPath= '~/virtis_data/MTP018/VIR0499/'
  sCubeFile='VI0499_01.CAL'
  oCube=virtispds(sPath + 'CALIBRATED/' + sCubeFile, /Silent)
  sCubeFileGEO='VI0499_01.GEO'
  oCubeGEO=virtispds(sPath + 'GEOMETRY/' + sCubeFileGEO, /Silent)
  uBlind=8
  uBand=380
  fMirrorCos= Reform(oCubeGEO.qube[32,6,*])
  uIndex=Where( $
    fMirrorCos EQ Max(fMirrorCos) $
  )
  uStop=uIndex-1
  fTempImage=Reform(oCube.qube[uBand,uBlind:*,0:uStop])
  fTempImage[Where(fTempImage LT 0)]=0.
  fLat=Reform(oCubeGEO.qube[25,uBlind:*,0:uStop]* $
    oCubeGEO.qube_coeff[25])
  uLatSz=Size(fLat,/Dimensions)
  fLon=Reform(oCubeGEO.qube[24,uBlind:*,0:uStop]* $
    oCubeGEO.qube_coeff[24])
  Device,decompose=0
  LoadCT,3
  Window,1,XSize=uLatSz[0]*2,YSize=uLatSz[1]*2
  WSet,1
  fImageCopyClipHigh=Max(fTempImage,Min=fImageCopyClipLow)
  fLatCenter=fLat[(Size(fLat))[1]/2,(Size(fLat))[2]/2]
  fLonCenter=fLon[(Size(fLon))[1]/2,(Size(fLon))[2]/2]
  fLonMin=Min(fLon)
  fLonMax=Max(fLon(Where(fLon NE 200.)))
  fLatMin=Min(fLat)
  fLatMax=Max(fLat(Where(fLat NE 200.)))
  Map_Set, $
    fLatMin, $
    fLonMin, $
    limit=[fLatMin,fLonMin,fLatMax,fLonMax], $
    /Albers, $
    /Isotropic, $
    /Horizon, $
    XMargin=5
  fImageCopy=BytSCL(fTempImage)
  bMap_v=Map_Patch( $
    fImageCopy, $
    fLon, $
    fLat, $
    XStart=Startx, $
    YStart=Starty, $
    XSize=sizex, $
    YSize=sizey)
  Map_Grid, latdel=10, londel=10,/Box_Axes,/Label
  Device,Decompose=1
```

```
51   LoadCT,0
52   Tv, bMap_v, Startx, Starty
53   Map_Grid, latdel=10, londel=10,/Label,Color=0
54 End
```

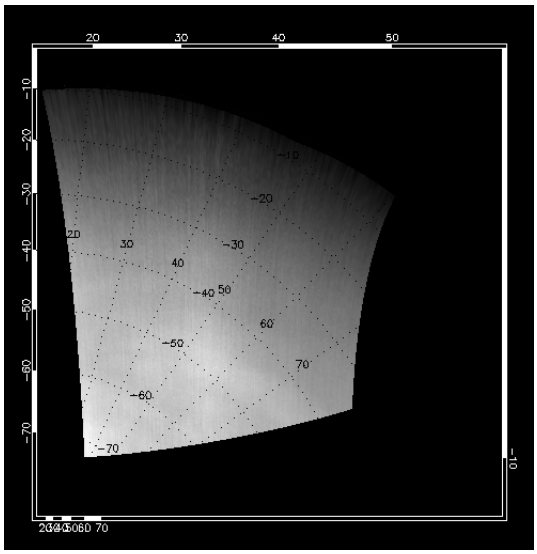

Figure 2.7: The output of the program exercise8.pro

Chapter 3

Virtis-H Data Approach

This part of the tutorial is based on traditional IDL 7 / GDL direct graphics.

3.1 Exercises

Exercise 9. *The 2.3 μm emission as a function of time*

1. Copy file VT1379_02.CAL from the archive in your working directory (orbit 1379, MTP49)
2. Read the file under IDL/GDL and print the label. Check the label information: what is the cube size? What is the date of acquisition? What is the integration time?
3. Load the wavelength vector, plot it in micron. Overplot the FWHM in nm
4. Plot the first spectrum of the cube. Interpretation?
5. Plot spectrum 100 of the cube. Interpretation?
6. Write a short procedure to plot independently the 8 orders of an arbitrary spectrum.
7. Find a measurement of emission in the 2.3 µm peak, plot it along the file. The astron library contains a useful function rdplot to read a graphic and identify wavelengths of interest.
8. Plot the 2.3 µm emission as a function of time

Exercise 10. *The 2.3 μm emission peak along the track*

1. Copy the corresponding geometry file VT1379_02.GEO from the archive in your working directory.
2. What are the frames present in the file?
3. Plot the emergence angle at the surface for each spectrum.

4. Plot surface elevation for each spectrum: along the line of sight, and below the clouds.

5. Plot the 2.3 µm emission level as a function of local time

6. Plot a rough visualization of the track on a sphere. Plot the 2.3 µm emission peak along the track.

3.2 Correction

3.2.1 Exercise 9-2

```
; with path on your local disk
fich = '/Volumes/Data3/VEx/orbites/VT1379_02.CAL'
sd=virtispds(fich)
print, v_eolpds(sd.label, /pr)

;Cube dimensions (H calibrated files are degenerated with only 2 dimensions):
help, sd.qube

;Acquisition date:
print, v_pdspar(sd.label, 'START_TIME')

;Integration time:
fp = v_pdspar(sd.label, 'frame_parameter_desc')
ip = where(v_listpds(fp) EQ '"EXPOSURE_DURATION"')
print, (v_listpds(v_pdspar(sd.label, 'frame_parameter')))(ip); in ms
```

3.2.2 Exercise 9-3

```
lam = sd.table(0,*) plot, lam, /xst
```

Wavelengths are stored from order 0 to order 7, in decreasing order. The funny shape is due to the overlap of consecutive orders.

```
FWHM = sd.table(1,*)
loadct, 12
oplot, FWHM*1000., col=200
```

Spectral resolution decreases along each order.

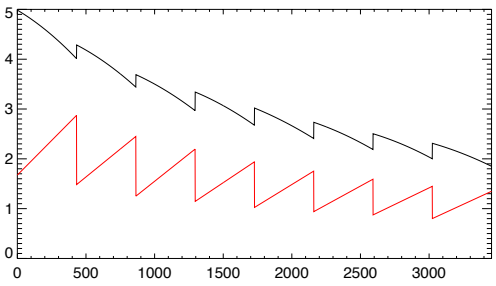

Figure 3.1: Plot for exercice 9-3 : Wavelength (black) and FWHM (red)

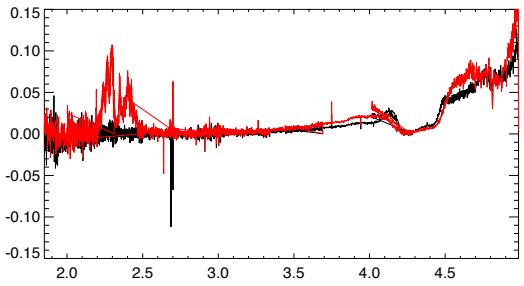

Figure 3.2: Plots for exercices 9-4 (spectrum 0 in black) and 9-5 (spectum 100 in red)

3.2.3 Exercise 9-4

```
plot, lam, sd.qube(*,0), /xst
```

This is a night side observation. Only the emission from the top of the atmosphere is visible; this is a relatively cold black body emission selectively absorbed by CO and CO_2.

The backward lines are due to orders overlap.

Negative levels are due to fluctuations in the signal and dark current. Occasional spikes are not filtered in calibrated data, to preserve possible narrow absorptions.

3.2.4 Exercise 9-5

```
plot, lam, sd.qube(*,100), /xst
```

The emission from the top of the atmosphere is still visible; in addition, the flux from the hotter atmosphere below finds its way out in specific spectral windows.

3.2.5 Exercise 9-6

```
Pro plotH, sd, Numspe
  lam = sd.table(0,*)
  cube = sd.qube
  WavelH = reform(lam, 432, 8)

  ind= array_indices(cube(0,*,*), Numspe)
  if size(ind, /dim) EQ 2 then $
    ind=[ind, 0] ; for cubes with 1 frame

  Spec = reform(cube(*,ind(1),ind(2)))
  Nchan = 432
```

```
12
13   loadct, 12
14   titx = 'Wavelength (micron)'
15   plot, wavelH, Spec, /nodata, /xst, xtit= titx
16   for i = 0, 7 do $
17     oplot, wavelH(*,i), Spec(Nchan *i:Nchan *(i+1)-1), col= i * 20 +10
18 end
```

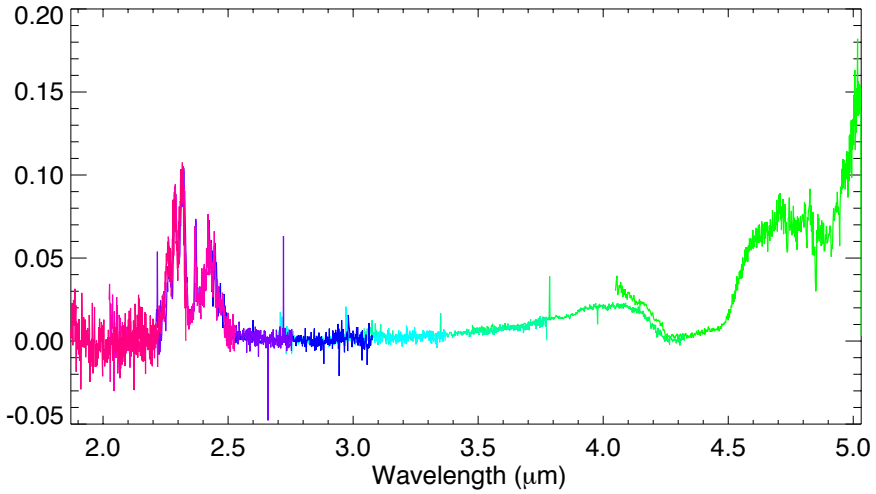

Figure 3.3: Plot for exercice 9-6 : 8 orders drawn independently

3.2.6 Exercise 9-7

The astron library contains a useful function rdplot to read a graphic and identify wavelengths of interest.

```
1 plot, lam, sd.qube(*,100), /xst
2 rdplot, /pr, /ful
```

The maximum emission is located near 2.3 µm, or indice 2800:

```
1 plot, sd.qube(2800, *)
```

3.2.7 Exercise 9-8

The Spacecraft Elapsed Time (scet) is contained in the cube suffix:

```
1 print, sd.suffix(*,0)
```

The information is encoded on 3 variables, which are translated by the routine `v_scet`.

```
Etime = v_scet(sd.suffix(0,*), sd.suffix(1,*), sd.suffix(2,*))
plot, Etime-Etime(0), sd.qube(2800, *), xtit= 'Time (s)'
print, fp
print, v_pdspar(sd.label, 'frame_parameter')
```

The scale is different from the previous plot, although the integration time is 1s. Successive spectra are acquired every 2.75 s (repetition time), and one dark frame is measured every 8 spectrum.

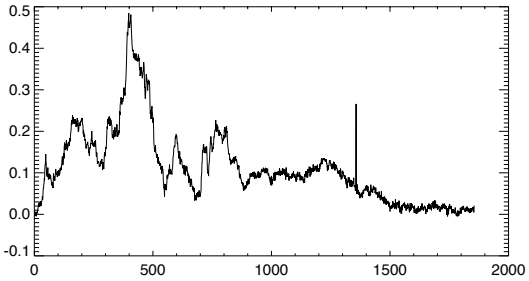

Figure 3.4: Plot for exercice 9-7 : 2.3 µm peak, along the file

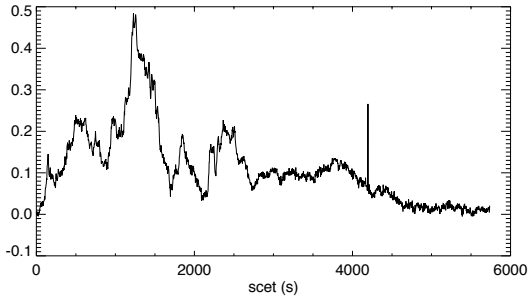

Figure 3.5: Plot for exercice 9-8 : 2.3 µm peak, as a function of time

3.2.8 Exercise 10-2

```
geo=virtispds('/Volumes/Data3/VEx/orbites/VT1379_02.GEO')
for i = 0, 40 do print, i, ' ',geo.qube_name(i)
```

Prints the frame names with indices. Units are described in the geometry file documentation. A conversion coefficient to more usual units is provided in the output structure.

3.2.9 Exercise 10-3

```
plot, geo.qube(11,*)*geo.qube_coeff(11), ytit= 'Emergence angle (deg)'
```

Viewing geometry is not very far from nadir.

3.2.10 Exercise 10-4

```
plot, geo.qube(13,*)*geo.qube_coeff(13), ytit= 'Surface elevation (km)'
```

This quantity corresponds to the geometrical footprint intercepted by the line of sight. However, most of the signal originates from the top of the cloud layer, where it is refracted at the vertical.

The geometry files also provide the elevation of the footprint vertically below the cloud layer:

```
oplot, geo.qube(29,*)*geo.qube_coeff(29), col=200
```

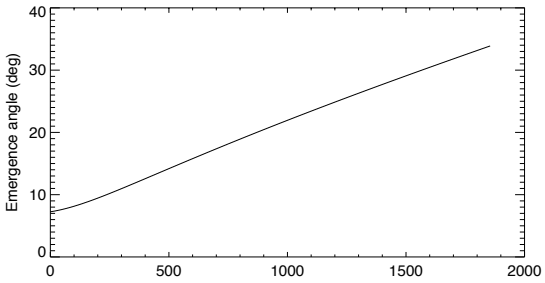

Figure 3.6: Plot for exercice 10-3 : emergence angle at the surface for each spectrum

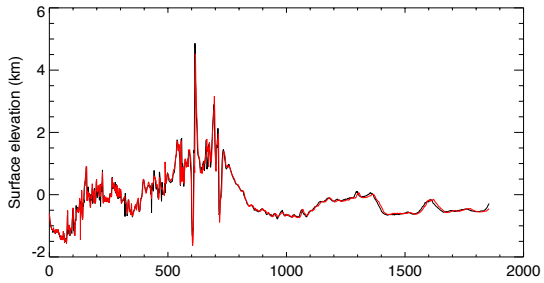

Figure 3.7: Plot for exercice 10-4 : surface elevation for each spectrum

3.2.11 Exercise 10-5

```
plot, geo.qube(15,*)*geo.qube_coeff(15), sd.qube(2800, *), $
  xtit= 'local time (h)'
```

3.2.12 Exercise 10-6

```
; track alone
map_set, /ortho, /iso, /grid, /label, /hor, 0, 180
plots, geo.qube(8, *)*geo.qube_coeff(8), $
  geo.qube(9, *)*geo.qube_coeff(9), $
  Psym=3, col= 200

; 2.3 um level along the track
map_set, /ortho, /iso, /grid, /label, /hor, 0, 180
plots, geo.qube(8, *)*geo.qube_coeff(8), $
  geo.qube(9, *)*geo.qube_coeff(9), $
  Psym=3, col=sd.qube(2800, *)*400
```

This method only uses the coordinates of the pixel center. More sophisticated maps can be plotted using all four corners of the footprint.

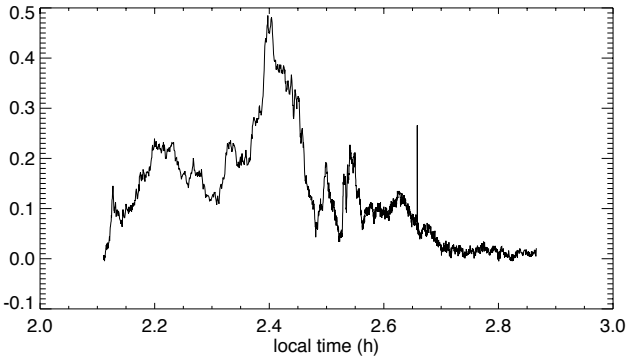

Figure 3.8: Plot for exercice 10-5 : 2.3 µm emission level as a function of local time

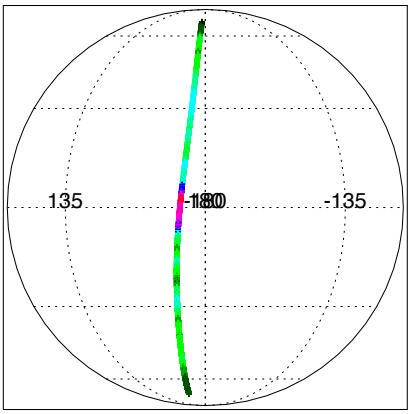

Figure 3.9: Plot for exercice 10-6 : rough visualization of the track on a sphere

Appendix A

Informations

A.1 The Qube header

A.1.1 Header keywords

The citated section in the table are refered to the document VVX-LES-IC-2269 available in the dataset archive, section document.

Table A.1: Geometry qube planes description

Keyword	Description
File Informations	
LABEL_REVISION_NOTE	References the version of the label template used when writing the file.
PRODUCT_ID	Provides the name of the data file, including extension, for further software processing.
RECORD_TYPE	Identifies the record characteristics of the file. Physical records are always fixed-length.
RECORD_BYTES	Identifies the number of bytes in each physical record of the data product file. Records length is always equal to 512 bytes.
FILE_RECORDS	Identifies the number of physical records in the file.
LABEL_RECORDS	Identifies the number of physical records that make up the PDS product label.
FILE_STATE	Is related to ISIS processing (not used).
Data Object Pointers	
Data Production	
PRODUCER_INSTITUTION_NAME	Identifies the institute in which the data product was generated. The possible values are LESIA (Observatoire de Paris, Meudon) and ISTITUTO NAZIONALE ASTROFISICA (INAF, Rome).

Table A.1: Geometry qube planes description - continue...

Keyword	**Description**
PRODUCT_CREATION_TIME	Provides the UTC of file generation as a standard ISO time string.
TELEMETRY_SOURCE_ID	Identifies the EGSE hardware used to generate the file (this information is relevant only for ground calibrations; the flight value is constant).
SOFTWARE_VERSION_ID	Tracks the processing applied to the current data product. It lists the software routines and versions used throughout the file history: on-board software version, low-level data handling to generate the PDS raw data file, plus calibration or geometry computation routines, and possibly intermediate reformatting routines.
SOURCE_PRODUCT_ID	Provides reference to an intermediate data product for internal check (EGSE output generated from the TM). This keyword is only present in raw data files.
Data Description	
The Planetary Science Archive of ESA implements the "Release" concept: data is delivered as units (releases), which can be updated (revision). The RELEASE and REVISION_ID data elements are included to handle the release concept.	
PRODUCT_TYPE	Identifies the product level. Possible values are EDR (raw data and geometry) or RDR (calibrated data). Derived products will use the value DDR.
PROCESSING_LEVEL_ID	Is accordingly 2 (raw data and geometry) or 3 (calibrated data).
STANDARD_DATA_PRODUCT_ID	Identifies data ("VIRTIS DATA") from geometry ("VIRTIS GEOMETRY") for use in further processing.
MISSION_PHASE_NAME	Indicates in which phase of the mission the data was acquired. It uses the values defined by the PSA (section 4.1 and Table 4.1).
CHANNEL_ID	Identifies the 3 channels of VIRTIS. Possible values are "VIRTIS_M_VIS", "VIRTIS_M_IR" and "VIRTIS_H" . A namespace is used here because the values are longer than allowed in the official definition of the PDS for this keyword.

Table A.1: Geometry qube planes description - continue...

Keyword	**Description**
DATA_QUALITY_ID	Is an error flag set by the EGSE during low level processing (see section 3.3). A value of 0 indicates missing data paquets.
Science Operations	
TARGET_TYPE TARGET_NAME	Identify the target of the observation. They are usually set to PLANET and VENUS.
START_TIME STOP_TIME	Are the UTC of the first and last science data paquets received for the current sub-session. They are computed through the SPICE system from the OBT.
SPACECRAFT_CLOCK_START_COUNT SPACECRAFT_CLOCK_STOP_COUNT	are the corresponding OBT, from the TM. The format is described in section 3.2.2.
ORBIT_NUMBER	Is given at start time of the sub-session, from mission planning. The value to this keyword complies to the convention used at mission level, where orbit number changes at apocenter. This is different from the convention used to derive file names, and therefore this value may not match the directory and file names (see section 4.1). The value is encoded on 4 digits with initial zeros.
SCIENCE_CASE_ID	Refers to mission planning, and is retrieved from the ITL file. It is encoded as in Table 2.1, or set to 0 if value is missing/unknown (in particular for calibrated sessions, and whenever the target is not Venus).
OBSERVATION_TYPE	Was an early PSA requirement, but is not actually used.
SPACECRAFT_ORIENTATION SPACECRAFT_POINTING_MODE	Refer to spacecraft attitude, and are defined in the documents mentioned, located in the DOCUMENT directory.
The next keywords provide average geometrical quantities, as derived from the SPICE system:	
DECLINATION RIGHT_ASCENSION	provide the pointing direction in the J2000 reference frame (direction of the channel boresight). They are normally found in conjunction with the "INERT" value of the SPACECRAFT_POINTING_MODE keyword; in other situations they are set to -999.99.

Table A.1: Geometry qube planes description - continue...

Keyword	Description
MAXIMUM_LATITUDE	Specifies the northernmost latitude of the spatial area covered by the observation. Angles are expressed in degrees.
MINIMUM_LATITUDE	Specifies the southernmost latitude of the spatial area covered by the observation. Angles are expressed in degrees.
EASTERNMOST_LONGITUDE	Provides the maximum numerical value of planetocentric longitude of the spatial area covered by the observation, unless it crosses the prime meridian. Angles are expressed in degrees.
WESTERNMOST_LONGITUDE	Provides the minimum numerical value of planetocentric longitude of the spatial area covered by the observation, unless it crosses the prime meridian. Angles are expressed in degrees.
SLANT_DISTANCE	Provides the average distance, in kilometers, from the spacecraft to the center of the observation along the line of sight. The center of observation is either the intercept with Venus surface, or the tangent point for limb observations.

A.1.2 Header example

```
PDS_VERSION_ID       = PDS3
LABEL_REVISION_NOTE  = "SE, 30/09/2010"

     /* File format and length */
PRODUCT_ID     = "VI0499_01.CAL"
RECORD_TYPE    = FIXED_LENGTH
RECORD_BYTES   = 512
FILE_RECORDS   = 245951
LABEL_RECORDS  = 12
FILE_STATE     = CLEAN

     /* Pointers to data objects */
^HISTORY     = 13
OBJECT       = HISTORY
 DESCRIPTION  = "Reserved area for ISIS compatibility"
END_OBJECT   = HISTORY
^QUBE        = 14

     /* Producer information */
PRODUCER_ID                = VEX_VIRTIS_TEAM
PRODUCER_FULL_NAME         = "DROSSART-PICCIONI"
PRODUCER_INSTITUTION_NAME  = "ISTITUTO NAZIONALE ASTROFISICA"
PRODUCT_CREATION_TIME      = 2012-10-26T08:39:33.00
TELEMETRY_SOURCE_ID        = "VIRTIS_EGSE_3"
SOFTWARE_VERSION_ID        = {"VirtisVEX SW v.2.60", "EGSE_SOFT_7.0",
 "PDS_CONVERTER_7.0","EGSE2PSA_CONVLABEL_1.2.1", "GEOVIRTIS_2.0",
 "V_GEOLABEL_3", "GEOVIRTIS_3.3", "V_GEOLABEL_3", "GEOVIRTIS_4.0",
 "V_GEOLABEL_4", "UPDATEGEOLABEL_2.0", "V_GEOLABEL_5", "V_CALIBM_VEX_2.2",
 "V_CONVLABEL_2.0"}
```

```
     /* Data description parameters */
DATA_SET_NAME         = "VENUS EXPRESS VENUS VIRTIS 2/3 V3.0"
DATA_SET_ID           = "VEX-V-VIRTIS-2/3-V3.0"
RELEASE_ID            = 0001
REVISION_ID           = 0000
PRODUCT_TYPE          = RDR
PROCESSING_LEVEL_ID   = 3
STANDARD_DATA_PRODUCT_ID  = "VIRTIS DATA"
MISSION_NAME          = "VENUS EXPRESS"
MISSION_ID            = VEX
INSTRUMENT_HOST_NAME  = "VENUS EXPRESS"
INSTRUMENT_HOST_ID    = VEX
MISSION_PHASE_NAME    = "PHASE 5"
PI_PDS_USER_ID        = "DROSSART-PICCIONI"
INSTRUMENT_NAME       = "VISIBLE AND INFRARED THERMAL IMAGING SPECTROMETER"
INSTRUMENT_ID         = "VIRTIS"
INSTRUMENT_TYPE       = "IMAGING SPECTROMETER"
^INSTRUMENT_DESC      = "VIRTIS_EAICD.TXT"
VEX:CHANNEL_ID        = "VIRTIS_M_IR"
PHOTOMETRIC_CORRECTION_TYPE  = NONE
DATA_QUALITY_ID       = 1
DATA_QUALITY_DESC     = "0:INCOMPLETE ; 1:COMPLETE"

     /* Science operations information */
TARGET_TYPE            = "PLANET"
TARGET_NAME            = "VENUS"
START_TIME             = 2007-09-01T15:56:51.214
STOP_TIME              = 2007-09-01T16:09:12.377
SPACECRAFT_CLOCK_START_COUNT  = "1/00079027006.08105"
SPACECRAFT_CLOCK_STOP_COUNT   = "1/00079027747.24379"
ORBIT_NUMBER           = 499
VEX:SCIENCE_CASE_ID    = 3
VEX:SCIENCE_CASE_ID_DESC  = "Please refer to VEX_SCIENCE_CASE_ID_DESC.TXT"
OBSERVATION_TYPE        = "NULL"
SPACECRAFT_ORIENTATION        = (0.0677,0.0246,0.9974)
^SPACECRAFT_ORIENTATION_DESC  = "VEX_ORIENTATION_DESC.TXT"
SPACECRAFT_POINTING_MODE      = "MOSAIC"
^SPACECRAFT_POINTING_MODE_DESC  = "VEX_POINTING_MODE_DESC.TXT"
DECLINATION            = -999.999
RIGHT_ASCENSION        = -999.999
MAXIMUM_LATITUDE       = -2.56273
MINIMUM_LATITUDE       = -72.5244
EASTERNMOST_LONGITUDE  = 81.4561
WESTERNMOST_LONGITUDE  = 13.8593
SLANT_DISTANCE         = 69260.8

     /* Instrument status */
INSTRUMENT_MODE_ID     = 19
^INSTRUMENT_MODE_DESC  = "VIRTIS_EAICD.TXT"
INST_CMPRS_NAME        = "REVERSIBLE"
INST_CMPRS_RATE        = "N/A"
INST_CMPRS_RATIO       = 2.79700

VEX:VIR_IR_START_X_POSITION  = 1
VEX:VIR_IR_START_Y_POSITION  = 7
VEX:SCAN_MODE_ID          = 1
SCAN_PARAMETER        = (-30.2126,35.621,0.2582,1)
SCAN_PARAMETER_DESC   = ("SCAN_START_ANGLE","SCAN_STOP_ANGLE",
 "SCAN_STEP_ANGLE","SCAN_STEP_NUMBER")
SCAN_PARAMETER_UNIT   = ("DEGREE","DEGREE","DEGREE","DIMENSIONLESS")
FRAME_PARAMETER       = (0.3,1,2.5,20)
FRAME_PARAMETER_DESC  = ("EXPOSURE_DURATION","FRAME_SUMMING",
 "EXTERNAL_REPETITION_TIME","DARK_ACQUISITION_RATE")
FRAME_PARAMETER_UNIT  = ("S","DIMENSIONLESS","S","DIMENSIONLESS")
EXPOSURE_DURATION_DESC  = "-1: many exposure times (calibration sessions)
Values are available in sideplane. See EAICD.TXT"
MAXIMUM_INSTRUMENT_TEMPERATURE  = (86.9011,152.6780,152.1670,75.218,173.60)
INSTRUMENT_TEMPERATURE_POINT    = ("FOCAL_PLANE","TELESCOPE", "SPECTROMETER",
"CRYOCOOLER","VIS_FOCAL_PLANE")
```

```
INSTRUMENT_TEMPERATURE_UNIT     = ("K","K","K","K","K")

     /* Pointer to navigation data files*/
SPICE_FILE_NAME  = {"NAIF0009.TLS","PCK00008.TPC","DE405.bsp",
"CLOUD-VENUS.ker","venus_cloud.bsp","VEX_070904_STEP.TSC","VEX_V09.TF",
"VEX_VIRTIS_V03.TI","ORHV_____________00030.bsp",
"ORVV__070901000000_00143.bsp","ATNV_P051109051109_00138.bc",
"VEX_VIRTIS_070825_070922_01.bc"}

     /* Data Cube keywords */
OBJECT  = QUBE
 AXES                        = 3
 AXIS_NAME                   = (BAND,SAMPLE,LINE)
 CORE_ITEMS                  = (432,256,281)
 CORE_ITEM_BYTES             = 4
 CORE_ITEM_TYPE              = "REAL"
 CORE_BASE                   = 0.0
 CORE_MULTIPLIER             = 1.0
 CORE_VALID_MINIMUM          = -999
 CORE_NULL                   = -1004
 CORE_LOW_REPR_SATURATION    = -1003
 CORE_LOW_INSTR_SATURATION   = -1002
 CORE_HIGH_REPR_SATURATION   = -1001
 CORE_HIGH_INSTR_SATURATION  = -1000
 CORE_NAME                   = RADIANCE
 CORE_UNIT                   = "W/m**2/sr/micron"

 SUFFIX_BYTES                = 4
 SUFFIX_ITEMS                = (1,0,3)

 BAND_SUFFIX_NAME            = "SCET"
 BAND_SUFFIX_UNIT            = DIMENSIONLESS
 BAND_SUFFIX_ITEM_BYTES      = 2
 BAND_SUFFIX_ITEM_TYPE       = MSB_UNSIGNED_INTEGER
 BAND_SUFFIX_BASE            = 0.0
 BAND_SUFFIX_MULTIPLIER      = 1.0
 BAND_SUFFIX_VALID_MINIMUM   = 0
 BAND_SUFFIX_NULL            = 65535
 BAND_SUFFIX_LOW_REPR_SAT    = 0
 BAND_SUFFIX_LOW_INSTR_SAT   = 0
 BAND_SUFFIX_HIGH_REPR_SAT   = 65535
 BAND_SUFFIX_HIGH_INSTR_SAT  = 65535

 LINE_SUFFIX_NAME            = ("WAVELENGTH","FWHM","UNCERTAINTY")
 LINE_SUFFIX_UNIT            = ("MICRON","MICRON","W/m**2/sr/micron")
 LINE_SUFFIX_ITEM_BYTES      = 4
 LINE_SUFFIX_ITEM_TYPE       = "REAL"
 LINE_SUFFIX_BASE            = 0.0
 LINE_SUFFIX_MULTIPLIER      = 1.0
 LINE_SUFFIX_VALID_MINIMUM   = "NULL"
 LINE_SUFFIX_NULL            = "NULL"
 LINE_SUFFIX_LOW_REPR_SAT    = "NULL"
 LINE_SUFFIX_LOW_INSTR_SAT   = "NULL"
 LINE_SUFFIX_HIGH_REPR_SAT   = "NULL"
 LINE_SUFFIX_HIGH_INSTR_SAT  = "NULL"
END_OBJECT  = QUBE

END
```

A.2 Geometry cubes

Table A.2: Geometry qube planes description

Plane	Parameter Description	Comment
0 - 3	Longitudes of 4 pixel footprint corner points	Geometrical projection on surface ellipsoid, with no correction for scattering or refraction.
4 - 7	Latitudes of 4 pixel footprint corner points	
8 - 9	Longitude & latitude of pixel footprint center on surface ellipsoid	
10 - 12	Incidence, emergence & phase at footprint center, relative to Venus center direction	Angles relative to the reference surface (not accounting for topography). Incidence angle is equal to solar zenithal angle.
13	Surface elevation (footprint corners average)	From topographic model.
14	Slant distance (line of sight from spacecraft to surface ellipsoid at pixel center)	Does not include topographic model.
15	Local time at footprint center	
16 - 19	Longitudes of 4 corner points on cloud layer	Geometrical projection on reference cloud layer (60km)
20 - 23	Latitudes of 4 corner points on cloud layer	
24 - 25	Longitude & latitude of pixel center on cloud layer	
26 - 28	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.
29	Surface elevation at the vertical of cloud layer intercept	From topographic model
30 - 31	Right ascension and declination of pointing direction.	J2000 reference frame
Virtis-M :		
32	One frame-common plane	Provides 10 scalar quantities along the frame spatial dimension. The remainder is set to 0.
	0 - 1 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).
	2 - 3 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).
	4 - 5 Sub-spacecraft coordinates (longitude/latitude)	
	6 - 7 Sine and cosine of M mirror angle	Converted into sin/cos values from HK multipled by 1,000.

Table A.2: Geometry qube planes description - continue...

Plane	Parameter Description	Comment
	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).	
Virtis-H : 9 supplementary planes		
32 - 33	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)
34 - 35	UTC	Encoded UTC recomputed through the SPICE system
36 - 37	Sub-spacecraft coordinates (longitude/latitude)	
38	Slit orientation	Relative to the pixel normal at footprint center
39 - 40	Sun direction:	
	39 : angle between Sun direction and Virtis Z axis	
	40 : azimuth of Sun direction in instrument XY plane (counted from 0Â° at X axis)	