Working with PcoGui FITS files

Kolja Glogowski

July 2, 2010

1 Overview

This text gives some hints on how to work with the FITS files that were created by the *PcoGui* program. It describes how to handle file names containing timestamps in IDL and introduces some functions (IDL and Python) for extracting the frame ID (or frame number) and the timestamp from images that were created by the PCO 2000 or PCO 4000 cameras.

This text and all scripts described in section 3 are available online at:

`http://www.kis.uni-freiburg.de/~kolja/camsys/`.

The IDL functions described in this text are already installed on the computer system at KIS.

2 How to handle file names in IDL

The *PcoGui* software creates files with names like

```
pco1_prefix_20101231-112233-123_suffix.fits
```

containing the name of the camera, a prefix, the file creation date/time and a suffix. In general the naming scheme is given by

```
[camera_][prefix_]YYYYMMDD-HHMMSS-ZZZ[_suffix].fits
```

where the prefix and suffix can be optionally assigned by the user and the additional camera prefix is only added by the software if the name of the camera is known (e.g. `pco1` and `pco2` at the VTT).

Working with files that have mandatory date and time fields in its name, may look a bit cumbersome at first, but it has some invaluable advantages compared to an ascending numbering scheme. For example:

- Each file name is unique
- You cannot accidentally overwrite an existing file
- All files with the same prefix are ordered chronologically
- You quickly determine the file's creation date and time by looking at its name

Aside from these benefits it isn't really that hard to work this kind of file names. In IDL you can use the function `FILE_SEARCH` to generate a list of files by calling it with a suitable file name mask. In the mask you can use the wildcard '`*`' which stands for any number of arbitrary characters, or the wildcard '`?`' that is interpreted as exactly one arbitrary character.

2.1 A simple IDL example

If you have recorded many files into the same directory and want to access all files that were created on june 29th 2010 between 10:00 and 10:59 you can simply create a chronologically sorted list of files using the following IDL command:

```
flist = file_search("*20100629-10*.fits")
```

for every day in june 2010 between 9am and 10am the command would be:

```
flist = file_search("*201006??-09*.fits")
```

You can now chronologically access each file in this list using `flist[0]`, `flist[1]`, ... or by writing a loop like:

```
for i = 0, n_elements(flist)-1 do print flist[i]
```

2.2 Another IDL example

Consider now the case that you have recorded some files in a first run with the file name prefix '`runa`' and some other files in a second run with the prefix '`runb`'. If we assume that all files contain one image with the dimension 1024x1024, then we can, for example, calculate the per-run-average of these images by the following code:

```
flista = file_search("runa_*.fits")
flistb = file_search("runb_*.fits")

a = fltarr(1024, 1024)
na = n_elements(flista)
for i = 0, na-1 do a += readfits(flista[i])
a /= na
```

```
b = fltarr(1024, 1024)
nb = n_elements(flistb)
for i = 0, nb-1 do b += readfits(flistb[i])
b /= nb
```

2.3 Using sub-directories

If you want to group some previously recorded files which have all the same prefix/suffix (or no prefix/suffix at all), you can also use sub-directories to separate some of these files and use for example

```
flista = file_search("seta/*.fits")
flistb = file_search("setb/*.fits")
```

to create the file lists in IDL.

2.4 Sorting numbers

A common problem is, that you may have created many files with different prefixes, where you have used only one digit to enumerate the first 9 files and two digits for the following files, e.g.

```
run1_, run2_, ..., run9_, run10_, run11_, ..., run20_
```

In this case the command

```
flist = file_search("run*.fits")
```

would result in the alphabetically sorted list:

```
run10_, run11_, ..., run19_, run1_, run20_, run2_, ..., run9_
```

which is probably not the way you want these files to be ordered. To solve this, you can call `FILE_SEARCH` twice and use the '?' and '??' wildcards respectively:

```
flist = [file_search("run?_*.fits"), file_search("run??_*.fits")]
```

This results in a list that is correctly ordered like the list at the beginning of this example.

3 Extracting frame numbers and timestamps from images

When recording images with the PCO 2000 or PCO 4000 camera, the camera can write the frame number and a timestamp into the first line(s) of the image data. The way these informations are written, can be set to `ASCII` (human readable text, using a couple of image lines), `BINARY` (BCD encoded, using only the first 24 pixels of the first line), both at the same time or none at all. Because this is the only way to determine the frame ID and the frame's creation time, the *PcoGui* program always uses the `BINARY` or `ASCII+BINARY` mode. The *CamWare* program on the other hand, allows the user to disable these informations or to use the `ASCII`-only mode.

To extract the frame number or the frame's timestamp from IDL or Python you can use the IDL function `PCO_FRAME_ID` and `PCO_FRAME_TIME` or the Python functions `frame_id()` and `frame_time()` from the module `pcoframe`. The needed source files are available at

`http://www.kis.uni-freiburg.de/~kolja/camsys/`.

If you have used the *CamWare* program, this will of course only work, if you had enabled the `BINARY` or `ASCII+BINARY` mode.

3.1 Extracting frame numbers and timestamps using IDL

The IDL files needed are:

```
pco_decode_bcd.pro
pco_frame_id.pro
pco_frame_time.pro
```

The function `PCO_DECODE_BCD` is used by both other functions, so you need all three files. At KIS all files are already installed, so you only need to install them by yourself, if you want to use these functions with a local copy of IDL on your laptop.

3.1.1 Example

The following example IDL session shows how to extract frame informations from a FITS file that was recorded with the *PcoGui* program:

```
IDL> data = readfits("test_20100601-074713-343.fits")
IDL> help, data
DATA            INT       = Array[2048, 2048, 20]
IDL> print, pco_frame_id(data)
         1         2         3         4         5         6
         7         8         9        10        11        12
```

```
        13        14        15        16        17        18
        19        20
IDL> print, pco_frame_time(data[*, *, 0:4])
      2010         6         1         7        47        11     159248
      2010         6         1         7        47        11     611027
      2010         6         1         7        47        12      62807
      2010         6         1         7        47        12     514586
      2010         6         1         7        47        12     966366
IDL> print, pco_frame_id(data[*, *, 19])
        20
IDL> print, pco_frame_time(data[*, *, 19])
      2010         6         1          7       47        19     743057
```

3.1.2 Documentation

The following documentation was taken from the IDL files.

pco_frame_id:

```
NAME:
  PCO_FRAME_ID

PURPOSE:
  This function extracts the frame number from images that were recorded
  by the PCO 2000 or PCO 4000 camera.

INPUTS:
  data:  An image (2d-array with dimension WxH), a collection of images
         (3d-array with dimension WxHxN) or a single line (1d-array with
         dimension W) which contains the bcd-encoded frame id.

KEYWORDS:
  line:  The line of the image that contains the bcd-encoded data. The
         default is the last line (H-axis) of the array.
         If the frames were recorded by the PCO CamWare program, it may
         be necessary to set this keyword to 0.
  offs:  The offset (W-axis) to the beginning of the frame id bcd data.
         The default value is 0.

OUTPUTS:
  The frame id(s) of the input images.

NOTES:
```

```
  To use this function, the images must be recorded with the timestamp mode
  set to BINARY or BINARY+ASCII. This mode is always enabled for the
  PcoGui but can be turned off when using the CamWare program.
```

pco_frame_time:

```
  NAME:
    PCO_FRAME_TIME

  PURPOSE:
    This function extracts the date and time of recording from images that
    were recorded by the PCO 2000 or PCO 4000 camera.

  INPUTS:
    data:  An image (2d-array with dimension WxH), a collection of images
           (3d-array with dimension WxHxN) or a single line (1d-array with
           dimension W) which contains the bcd-encoded frame id.

  KEYWORDS:
    line:  The line of the image that contains the bcd-encoded data. The
           default is the last line (H-axis) of the array.
           If the frames were recorded by the PCO CamWare program, it may
           be necessary to set this keyword to 0.
    offs:  The offset (W-axis) to the beginning of the date/time bcd data.
           The default value is 4.

  OUTPUTS:
    The date and time when the input images were created. The format of each
    line of the returned array is:

        [ year, month, day, hour, minute, second, microsecond ]

  NOTES:
    To use this function, the images must be recorded with the timestamp mode
    set to BINARY or BINARY+ASCII. This mode is always enabled for the
    PcoGui but can be turned off when using the CamWare program.
```

3.2 Extracting frame numbers and timestamps using Python

The Python module needed is `pcoframe.py`.

3.2.1 Example

The following example Python session shows how to extract frame informations from a FITS file that was recorded with the *PcoGui* program:

```
>>> import pyfits
>>> import pcoframe as pco
>>> data = pyfits.open("test_20100601-074713-343.fits")[0].data
>>> data.shape
(20, 2048, 2048)
>>> pco.frame_id(data)
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20], dtype=int32)
>>> pco.frame_time(data[0:5])
[datetime.datetime(2010, 6, 1, 7, 47, 11, 159248),
 datetime.datetime(2010, 6, 1, 7, 47, 11, 611027),
 datetime.datetime(2010, 6, 1, 7, 47, 12, 62807),
 datetime.datetime(2010, 6, 1, 7, 47, 12, 514586),
 datetime.datetime(2010, 6, 1, 7, 47, 12, 966366)]
>>> pco.frame_id(data[19])
20
>>> pco.frame_time(data[19])
datetime.datetime(2010, 6, 1, 7, 47, 19, 743057)
```

3.2.2 Documentation

The following documentation was taken from the source file.

frame_id(data, line=-1, offs=0):

```
This function extracts the frame number from images that were recorded
by the PCO 2000 or PCO 4000 camera.

Arguments:
  data:  An image (2d-array with dimension WxH), a collection of images
         (3d-array with dimension WxHxN) or a single line (1d-array with
         dimension W) which contains the bcd-encoded frame id.
  line:  The line of the image that contains the bcd-encoded data. The
```

```
          default is the last line (H-axis) of the array.
          If the frames were recorded by the PCO CamWare program, it may
          be necessary to set this keyword to 0.
    offs: The offset (W-axis) to the beginning of the frame id bcd data.
          The default value is 0.

  Returns:
    The frame id(s) of the input images. The type of the result is an
    integer array (3d input-array) or an int value (2d input-array).

  Notes:
    To use this function, the images must be recorded with the timestamp
    mode set to BINARY or BINARY+ASCII. This mode is always enabled for
    the PcoGui but can be turned off when using the CamWare program.
```

frame_time(data, line=-1, offs=4):

```
  This function extracts the date and time of recording from images that
  were recorded by the PCO 2000 or PCO 4000 camera.

  Arguments:
    data: An image (2d-array with dimension WxH), a collection of images
          (3d-array with dimension WxHxN) or a single line (1d-array with
          dimension W) which contains the bcd-encoded frame id.
    line: The line of the image that contains the bcd-encoded data. The
          default is the last line (H-axis) of the array.
          If the frames were recorded by the PCO CamWare program, it may
          be necessary to set this keyword to 0.
    offs: The offset (W-axis) to the beginning of the date/time bcd data.
          The default value is 4.

  Returns:
    The date and time when the input images were created. The type of the
    result is a list of datetime objects (3d input-array) or a single
    datetime object (2d input-array).

  Notes:
    To use this function, the images must be recorded with the timestamp
    mode set to BINARY or BINARY+ASCII. This mode is always enabled for
    the PcoGui but can be turned off when using the CamWare program.
```