19. Data Simulation and Variance-Covariance Studies

19.1 Principles of Simulation

The simulation part of the Bernese GPS Software contains the program GPSSIM. GPSSIM may be used to generate simulated observations which may in turn be processed by the "normal" processing programs of the Bernese GPS Software. You cannot start this program using the menu system. It is necessary to prepare (using an ASCII editor) the input files (`GPSSIMI.INP, GPSSIMN.INP, GPSSIMF.INP`) and start the program manually – see Chapter 3. You will find the examples for the input files in the directory `X:[INX]`. According to the input specifications (see below) the following output files may be generated:

- code observations – code zero difference (header and observation) files in the Bernese format (binary),
- phase observations – phase zero difference (header and observation) files in the Bernese format (binary), and
- meteo files (in Bernese ASCII format – see Chapter 23).

In one program run observations for *one session* are generated. With several program runs you may generate an entire simulated campaign.

The following input files define the scope for your observation scenario:

- the standard orbit file (`*.STD`) is used to compute satellite positions,
- the broadcast orbit file (`*.BRD`) is used only to apply satellite clock corrections, and
- the coordinate file (`*.CRD`) gives the station positions.

The broadcast orbit file and the standard orbit file should cover (roughly) the same time interval. Observations may be generated only in the time span covered by the standard orbit file, and only for stations in the coordinate file.

GPSSIM may be used for two different purposes:

Pre-analysis: Given the satellite scenario, the network and the statistical a priori information, and the systematical errors, you may study the expected quality of the results depending on the processing strategy and options used (e.g. ambiguities free/fixed).

Research-oriented analyses: Questions concerning

- cycle slip repair (introduce gaps . . .),
- ambiguity resolution strategies for long baselines,
- orbit determination,
- . . .

may be studied.

If the program system is used for pre-analysis studies, the user will not be interested to go through the preprocessing part. Therefore the simulation program writes the true receiver clock information into the observation files and it includes the correct ambiguity parameters into the phase files. This information is preserved by program SNGDIF, the only program that has to be used before program GPSEST, if simulated data are processed by the Bernese GPS Software Version 4.0 . It is then possible to make at once tests with the "ambiguity fixed" mode of program GPSEST (assuming that you were able to resolve the ambiguities). By ignoring the ambiguity information in the files, GPSEST also allows you to make tests with the ambiguity-free mode. If you solve e.g. for ambiguity parameters and store them in the single-difference file headers, you know the true answer (all ambiguities should be zero). Because GPSEST internally works with double differences, identical ambiguity values for all ambiguities of an ambiguity cluster (see Chapter 15) is a correct answer, too. Of course you may use all programs of the Bernese GPS Software with simulated observations. Some analysts may like e.g. to remove cycle slips introduced by the simulation program with program MAUPRP. You may also be interested in studying the effect of using lower polynomial degrees in program CODSPP than in program GPSSIM, etc.

It is very easy to use the program GPSSIM for the present and past GPS satellite configuration. You simply start the simulation with a broadcast file corresponding to the time interval of interest. It is a good idea, but not mandatory, to use a broadcast file which was recorded at the same time of the year your new campaign will take place (the daily observation session will then take place at about the same time of the day). Then, by using programs BRDTAB and ORBGEN, you generate your standard orbit file. The next step consists of defining a ground truth by generating a coordinate file containing all stations of the campaign (not only of one session).

19.2 Models used for Simulation

19.2.1 Stochastic Properties

In the I-File (`GPSSIMI.INP` – see below) you may specify the following rms errors:

- rms for code observations (separately for L_1 and L_2)
- rms for phase observations (separately for L_1 and L_2)
- rms errors for meteorological conditions (pressure, temperature, and humidity)

Each simulated measurement will contain a normally distributed random error with the rms specified in the input file for the measurement type considered. This is true for phase, code and meteo measurements. The phase and code measurements generated will not contain any meteo errors (we assume that “natura (resp. troposphere) non facit saltas”). In addition to these random errors you may assign *systematic* errors to the troposphere measurements by specifying maximum biases for temperature, pressure and humidity measurements. The program then generates constant biases for the session but different for each station using a random number generator: It assigns to each station and to each measurement type a bias which is a random number uniformly distributed in the interval

$$I = \langle -B_{max}, +B_{max} \rangle,$$

where B_{max} is the maximum bias you specified in the input file for the measurement type considered. There is one more stochastic quantity, the random part of the free electron content of the ionosphere, which has to be mentioned. Since this is closely related to the discussion of the ionosphere model used, we will deal with this effect in the next section.

The random numbers are generated with the subroutines

– NORMAL	generating a normally distributed random number (given its mean and variance)
– RANDU	generating a uniformly distributed random number in the interval $\langle 0, 1 \rangle$

NORMAL calls RANDU several times. RANDU updates a positive `INTEGER*4` random variable which has to be initialized. It would have been possible to initialize this number in a “random way” (using e.g. the last digits of the system clock reading), but it seemed advisable to have this initialization under control by initializing this random number in the input file. We suggest that you use positive four digit integers for initialization. If you re-run the program with the same input specifications and the same initial random number, the same random errors will occur.

19.2.2 Deterministic Models

Satellite Orbits, Clocks, Coordinates

We mentioned already that the satellite positions are computed using a standard orbit, the satellite clocks using a broadcast file, the station coordinates using the file with station coordinates. Therefore the model for satellite orbits is the complex force model underlying program ORBGEN, the model for satellite clocks is purely deterministic (polynomial of degree 2 defined in the message). Moreover the user has to define the receiver clock behaviour (by a polynomial) in a file of the following kind:

```
RECEIVER CLOCKS AS POLYNOMIALS OF DEGREE N-1
POLYNOMIAL COEFFICIENTS IN SEC, SEC/DAY, SEC/DAY**2,...
 STATION        N     A0       A1       A2       A3       A4       A5
ZIMMERWALD      6 +0.016017-0.000123+0.000222-0.002000-0.012345+0.000555
CHASSERAL       6 -0.003002+0.001234-0.000000+0.010000-0.034556-0.023476
GENEROSO        6 +0.020000-0.010000+0.010000-0.005045-0.000345-0.123456
TITLIS          6 -0.013000+0.020000-0.001234+0.001000-0.006000-0.500000
```

If you are not interested in receiver clocks just state N=0 for each station.

Troposphere

Tropospheric refraction may be neglected or it may be modeled by the

- Saastamoinen Model
- Hopfield model (2 versions)

These models ask for (ground) meteo values for each station. These values are generated using the reference height and the temperature, pressure, and humidity at this reference height as specified in the constant file. This allows you e.g. to simulate data with a standard atmosphere different from that used by program GPSEST by using different constant files in programs GPSSIM and GPSEST.

Ionosphere

The "ionosphere truth" (deterministic part) is defined by the subroutine IONOS. The model underlying is the single layer model for the electron content (height of the layer=350 km). The number of electrons in the layer E is a function of local time only:

$$\begin{aligned} T_{loc} &= UT + \lambda \\ E &= \begin{cases} E_0 & \text{for } \; T_{loc} \in \langle 20^h, 8^h \rangle \\ E_0 + E_1 \, \cos\left(\dfrac{T_{loc} - 14}{12}\,\pi\right) & \text{for } \; T_{loc} \in \langle 8^h, 20^h \rangle \end{cases} \end{aligned}$$

where : E_0 is the night time electron content,
E_1 is the day time variation of the ionosphere.

E_0, E_1 are input parameters of program GPSSIM. No azimuth dependence is modeled here. In addition to this regular part an irregular part may be superposed: At epoch i the following term is added

$$\delta E_i = (E_{i1} \, \cos a + E_{i2} \, \sin a) \; d/200$$

where:		
	δE_i	is the random part of the electron content in the atmosphere
	d	is the distance in km between the station considered and the ionosphere reference station (specified in the input file)
	a	is the azimuth of the station considered from the ionosphere reference station
	E_{ik} , $k = 1, 2$	is the result of the following random walk: $E_{1k} = 0$, $\text{var}(E_{i+1,k} - E_{ik}) = I_i \cdot \text{var}_{1min}$, where I_i is the time between observations $i, i+1$ in minutes, var_{1min} is the variance of the difference of the electron contents at time t and $t+1$ minute. var_{1min} is an input variable of program GPSSIM.

This model certainly is a crude simplification for the random behaviour of the true ionosphere. It shares, however, important aspects with the real situation: (1) The irregular contribution increases linearly with the baseline length, for distances $>$ 200 km this distance dependence “stops”, (2) baselines which are close to each other geometrically will show similar random contributions. The strength of the irregular contribution may be tuned by an input parameter.

Cycle Slips

If you like preprocessing and its secrets you may introduce cycle slips in the simulation program. It is then really hard to tell whether you are looking at real or at simulated observations . . .

19.3 Essential Input Files

Option Input File (I-File):

Remarks:
- Campaign name and title line will show up in the output files. This helps to identify simulated files.
- Program SNGDIF will assume simulated data, if the receiver type starts with “`SIMULA`”. Therefore the string “`SIMULA`” is essential in the input part “`RECEIVER TYPE`”.
- In the troposphere model you may wish to assume identical biases for all station, you may even consider to skip the random number generator for the bias part by using the maximum biases. The troposphere biases actually applied are given in the program output.
- In the scenario input section you may specify a minimum elevation and the data sampling rate. For pre-analysis purposes we often use a very low rate (4 to 6 minutes) in order to save disk space. For other applications (e.g. investigations concerning minimum session lengths which allow you to resolve ambiguities) you may use short session lengths and high data rates (e.g. 3 sec).
- In the input section “cycle slips” you may specify how many cycle slips you wish to introduce into each file of the session. The program assumes that a slip may occur with the same probability at each epoch and for each satellite and that its size is uniformly distributed in the interval $I = \langle -slip_{max}, +slip_{max} \rangle$
- You may wish to apply identical slips in L_1 and L_2. These are particularly hard to discover. It might be of interest to look at the quality of results if, let us say, n slips of size 1 remain undiscovered in the data for $n = 1, 2, \ldots$.

```
GPSSIM: OPTION INPUT FILE                                  21-APR-90 13:02
----------------------------------------------------------------------
(REMARK: YES=1,NO=0 ; 2 EMPTY LINES AFTER EVERY INPUT GROUP)

CAMPAIGN NAME:
-------------
      ****************

--> : TESTV40
```

```
TITLE LINE:
----------
      *****************************************************

--> : SIMULATED DATA TO TEST VERSION 4.0

RECEIVER TYPE:
-------------
             *********

--> : SIMULA

SESSION DEFINITION:
------------------
             FROM                 TO
       YYYY MM DD HH MM    YYYY MM DD HH MM

--> :  1995 06 16 10 30    1995 06 16 14 00

                                                        ****

SESSION NUMBER                                 --> :   167
FILE NUMBER (SAME SESSION, SAME STATION)       --> :     1
TIME BETWEEN SUBSEQUENT OBSERVATIONS (SEC)     --> :   120
MIN. ELEVATION (DEGREES)                       --> :    20

TROPOSPHERIC MODEL:
------------------                                      *

NO METEO                            =  0
SAASTAMOINEN                        =  1
MODIFIED HOPFIELD (REMONDI)         =  2
SIMPLIFIED HOPFIELD (DAVIDSON)      =  3       --> :  1

                                                        ******

RMS ERROR FOR PRESSURE MEASUREMENT             --> :   2.0    MBAR
RMS ERROR FOR TEMPERATURE MEASUREMENT          --> :   1.0    DEG C
RMS ERROR FOR HUMIDITY MEASUREMENT             --> :   5.0    %

MAXIMUM PRESSURE BIAS FOR ONE STATION          --> :   3.0    MBAR
MAXIMUM TEMPERATURE BIAS FOR ONE STATION       --> :   2.0    DEG C
MAXIMUM HUMIDITY BIAS FOR ONE STATION          --> :  10.0    %

                                                        *

SAME BIAS FOR ALL STATIONS                     --> :  0
MAX.BIASES = ACTUAL BIASES FOR ALL STAT.       --> :  0

IONOSPHERIC MODEL:
-----------------                                       ***

NIGHT TIME ELECTRON NUMBER (IN 10**16)         --> :   20
DAY   TIME ELECTRON NUMBER (IN 10**16)         --> :   30
VARIANCE OF IRREG. CHANGE OF IONOSPHERE CONTENT
  IN 1 MIN. AT 200 KM FROM REF.SITE (IN 10**15)  --> :    1
                                                        ****************
REFERENCE SITE                                 --> :  ZIMMERWALD

STATISTICAL INFORMATION:
-----------------------                **.***

A PRIORI SIGMA CODE L1         --> :   1.00  M
A PRIORI SIGMA CODE L2         --> :   1.00  M
A PRIORI SIGMA PHASE L1        --> :  .003   M
A PRIORI SIGMA PHASE L2        --> :  .003   M
```

```
                                    ******

INITIAL INTEGER RANDOM NUMBER  --> :    7778

INTRODUCE CYCLE SLIPS:
--------------------                ******

NUMBER OF SLIPS PER FILE       --> :       0
MAXIMUM SIZE OF SLIP           --> :     100

SAME SLIPS IN L1 AND L2 ?      --> :       0
```

Session Definition File (F-File):

The file is truncated to the right: The definition of phase header and observation files is missing. The file defines code/phase header/obs file names (external) and meteo file names. It also relates the file to the station (name) with the additional information at the bottom of the file.

```
GPSSIM: OUTPUT FILE NAMES FOR CODE, PHASE, AND METEO
-----------------------------------------------------------------------..
CODE HEADER FILE NAME (OUTPUT)   CODE OBSERVATION FILE NAME (OUT) PHASE ..
******************************** ******************************** ******..

X:\CAMPNAM\OBS\CHAS1671.CZH      X:\CAMPNAM\OBS\CHAS1671.CZO      X:\CAM..
X:\CAMPNAM\OBS\GENE1671.CZH      X:\CAMPNAM\OBS\GENE1671.CZO      X:\CAM..
X:\CAMPNAM\OBS\TITL1671.CZH      X:\CAMPNAM\OBS\TITL1671.CZO      X:\CAM..
X:\CAMPNAM\OBS\ZIMM1671.CZH      X:\CAMPNAM\OBS\ZIMM1671.CZO      X:\CAM..


METEO FILE NAME (OUTPUT)
********************************

X:\CAMPNAM\ATM\CHAS1671.MET
X:\CAMPNAM\ATM\GENE1671.MET
X:\CAMPNAM\ATM\TITL1671.MET
X:\CAMPNAM\ATM\ZIMM1671.MET

STATION NAME      REC.UNIT  ANTENNA  OPERATOR NAME
****************    ****     ****    ****************

CHASSERAL             1      101    OPERATOR 1
GENEROSO              2      202    OPERATOR 2
```

Remarks: – The information on which types of observations should be created is located in the receiver file (see Chapter 23)
– If at least one of the meteo error parameters (rms or max. bias) is different from zero, the program will produce meteo measurement files. The names of these files have to be defined in the F-file above.

20. Services

In this chapter the options of Menu 5, as described below, will be discussed. The topics “POLE” and “COORDINATES” are described in Chapters 14 and 11, respectively, and will therefore not be discussed here.

```
 5                       SERVICES: OPTION MENU



 1      OBSERVATIONS    : Browse, Edit , Update and Graphic Obs. Files
 2      FILE HEADERS    : Change the Content of Obs. File Headers
 3 ..   RESIDUALS       : Browse and Check Residuals
 4 ..   COORDINATES     : Compare Coord. Files, Helmert Transformation
 5 ..   POLE            : Update and Extract Pole Information
 6 ..   EXTRACTIONS     : Generation of output summaries
 7 ..   BINARY <-> ASCII: Conversion Between Binary and ASCII Files
 8      DELETE FILES    : Delete GPS Specific Files
 9      JOB             : Processing of Job Output
```

20.1 Observations and File Headers

Since Menu 5.1 is frequently used one may also involve it by typing “OBS” after the prompt “Enter Selection”. This panel is used to manipulate the code and phase zero and single difference observation files. The menu program used here is called SERVOBS.

```
 5.1                       SERVICES: OBSERVATIONS

     B - Browse Observation File        M - Mark Observations or Satellites
     E - Edit Observation File          D - Delete Observation File
     G - Graphic of Observations        C - Create File Table
     2 - Split Observation File         A - Add Files to the File Table
     H - Edit Header File only          R - Reorder Files in File Table
     X - Exit

     Option:               >   <        (blank: Select option in file list)

     CAMPAIGN            > IGSA     <   (blank for selection list)

   Input File:
     MEASUREMENT TYPE     > PHASE  <    (CODE, PHASE, BOTH /options C,A,R/)
     DIFFERENCES          > ZERO   <    (ZERO or SINGLE)
     OBSERVATION FILE    > %%%%$D11 <   (blank for selection list)
```

In the upper half of this panel the options are listed with a short description.

The create (“C”) or add (“A”) options are used to either create a new observation file list from scratch or to update an existing observation file list. Under normal circumstances the observation file list will always be up-to-date because the two programs that generate observation files, RXOBV3 and SNGDIF, update the observation file list automatically. However, if files are copied or deleted manually the observation file list is *not* updated, which is why these two options exist. One should keep in mind here that the list of files which is shown when selecting an option will be based on the observation file list files and *not* on the actually available observation files. Individual observation file lists exist for the four different categories of observation files; zero-difference phase, zero-difference code, single-difference phase and single-difference code. The observation file lists are discussed in Chapter 23.

Apart from creating or updating observation file lists we may browse (“B”) or edit (“E”) the observation files. When browsing a file you can not alter the file, when editing it, you can. Editing an observation file is something which is necessary only in rare circumstances. Occasionally, one might want to change some information in the file header if it contains erroneous information. For that purpose the “H” option should be used since it is much quicker only to edit the header than to edit the entire file. When manually editing the observation file one should be very careful. One should e.g. not remove any satellites from the header file, nor change the number of satellites and the number of frequencies. Also do not change the ambiguity cluster information. One may change observation flags and remove observation epochs.

Program CHGHED, Menu 5.2, allows you to change the file header automatically. This is useful if a large number of header files has to be changed. Erroneous information may be stored in the file header if the RINEX files contained erroneous information or if the information in the translation tables used with RXOBV3 were in error. After running program CHGHED be sure to check the output of the program for the changes performed in the observation header files.

The graphics option (“G”) produces similar results as program RNXGRA: it produces a pseudo-graphic of each selected observation file. These pseudo-graphics may be generated for both, zero and single difference files. Another option is to split up the observation files in two parts (“2”). This is useful if you have data from a campaign with different start and end times. It will also be useful if for instance the antenna height was changed during the observation time span. However, we recommend to split up data files on the RINEX level rather than after the conversion to the Bernese format. Programs CCRINEXN and CCRINEXO are available for this purpose, see Chapter 7

With the mark option (“M”) we mark data in the observation files. This option may be used to “throw away” bad satellites or to remove bad data points using an “edit file”. The use of an edit file will be discussed later in this chapter. The format of the edit file is discussed in Chapter 23

The delete (“D”) option may be used to delete files. The files will be deleted from disk and removed from the observation file list. In Menu 0.1 you specify whether the deletion should be done with or without confirmation.

The reorder (“R”) option may be used to change the file order. By default we use the session number as first criterion to order the files followed by the station name and, finally, by the file sequence number. With the reorder option you may specify a different ordering.

Finally you exit this menu panel using the “X” option. You may also use “X” to exit when you are in an observation file selection list. Pressing “Q” when in an observation file selection list will put you back to the services panel.

20.2 Residuals

In Menu 5.3 you have available three different options to work with residual files. One may wish to look at the residuals as numbers printed on screen or in a file, to screen the residuals for outliers or to look at the residuals graphically.

```
 5.3                          RESIDUALS: OPTION MENU


  1       BROWSE RESID.    : Browse Residuals of GPSEST,MAUPRP,CODSPP
  2       CHECK RESIDUALS  : Check Residuals for Outliers
  3       GRAPHIC TOOL     : Display Residuals with GT
```

The “browse residuals” option allows you to inspect the residual files from either CODSPP, MAUPRP, or GPSEST using program REDISP. The residuals will either be displayed onto your computer screen or written into a file. The latter option is useful if you run into problems during processing. It will help you to identify which station might cause the problems. However, going through the data of a large number of stations “manually” is quite cumbersome. Therefore the residuals may also be screened automatically with the “check residual” option in Menu 5.3.2.

The “check residual” option will run program RESRMS. This program screens the residuals for outliers. Outliers are observations with residuals exceeding a threshold defined by the user. RESRMS creates a summary file and a so-called “edit file”, see Chapter 23 for a description of the format of the edit file. This edit file contains those points which were detected as outliers in the residual file(s). The file may be used with the SERVOBS program, Menu 5.1 option “M”, to remove these points from the actual observation files.

The “graphic tool”, GT, is currently only available on Unix systems. It has been developed by UNAVCO and is freely available. The version for VAX/VMS systems is almost ready and should be available soon. For more information please contact Chris Rocken at UNAVCO (e-mail rocken@unavco.ucar.edu). The graphic tool will give a (X-Windows) plot of the residuals and allows manual deletion of points, correction of cycle slips, and the setting up of new ambiguity parameters. These manual editing requests are also handle by means of an “edit file” and have to be applied to the observation files using the program SERVOBS.

20.3 Extractions

There are several extraction programs capable of summarizing the output files of the most important/critical programs of the Bernese GPS Software. They are very useful for automated processes (BPE) because they will summarize the relevant information and allow the user to verify easily whether the processing was successful or not. Extraction programs are available for programs CODSPP, ORBGEN, MAUPRP, GPSEST, and ADDNEQ.

```
 5.6     |                      EXTRACTIONS
---------+----------------------------------------------------------------
         |
---------+----------------------------------------------------------------
         |
  1      |  CODSPP OUTPUT    : CODSPP output summary             (CODXTR)
  2      |  DEFSTD OUTPUT    : DEFSTD output summary        (DEFXTR/DEFXTP)
  3      |  MAUPRP OUTPUT    : MAUPRP output summary             (MPRXTR)
  5      |  GPSEST/ADDNEQ OUT: General summary (several formats)    (GPSXTR)
         |
```

CODXTR

CODXTR, the extraction program for program CODSPP, produces a summary of all CODSPP output of the following form:

```
CODSPP EXTRACTION
-----------------
  23   4  GUAM 50501M002    OUT  19  96-08-06  6:26:30  96-08-06  7:07:30    83
   3   6  USUD 21729S007    OUT  19  96-08-06  7:02:30  96-08-06  7:59:30   115
   7   7  ALBH 40129M003    OUT  19  96-08-06  6:29:00  96-08-06  8:00:00   183
   8   8  ALGO 40104M002    OUT  19  96-08-06  6:39:30  96-08-06  8:00:00   162
  14  10  CHUR              OUT  19  96-08-06  6:26:30  96-08-06  8:00:00   188
  21  12  DRAO 40105M002    OUT  19  96-08-06  6:27:30  96-08-06  8:00:00   186
  23  13  FAIR 40408M001B   OUT  19  96-08-06  6:26:30  96-08-06  6:56:00    60
  24  13  FAIR 40408M001B   OUT  19  96-08-06  6:57:00  96-08-06  7:59:30   126
  26  14  FLIN              OUT  19  96-08-06  6:32:30  96-08-06  7:59:30   175
  28  15  GODE 40451M123    OUT  19  96-08-06  6:40:30  96-08-06  7:59:30   159
   4   2  KELY 43005M001    OUT  19  96-08-06  6:58:30  96-08-06  7:59:30   123
   7   3  NLIB 40465M001    OUT  19  96-08-06  6:26:30  96-08-06  7:59:30   187
  11   4  MDO1 40442M012    OUT  19  96-08-06  6:27:30  96-08-06  7:59:30   185
  14   5  PIE1 40456M001    OUT  19  96-08-06  6:27:30  96-08-06  7:59:30   185
  16   6  QUIN 40433M004    OUT  19  96-08-06  6:28:00  96-08-06  7:59:30   184
  19   7  RCM5 40499S018B   OUT  19  96-08-06  7:11:00  96-08-06  7:59:30    98
  22   8  THU1 43001M001    OUT  19  96-08-06  6:35:00  96-08-06  7:59:30   170
  23   9  WES2 40440S020    OUT  19  96-08-06  6:44:00  96-08-06  7:59:30   152
  28  10  WHIT              OUT  19  96-08-06  6:27:30  96-08-06  7:59:30   185
  30  11  YELL 40127M003B   OUT  19  96-08-06  6:27:30  96-08-06  6:56:00    58
  31  11  YELL 40127M003B   OUT  19  96-08-06  6:57:00  96-08-06  7:59:30   126

 61 FILES, MAX. RMS:   39.28 m for station: YAR1 50107M004
```

The summary file will contain at least one line, in the above example the last line. This last line of the summary gives the number of code files that were processed, the maximum rms, and the station for which this maximum occurred. The rms value should be around 25 meters, under SA conditions. Values up to 300 meters are still acceptable although they deserve special attention.

Furthermore it shows one line for each station–satellite combination where more than 50 points are considered as outliers. This is the case in the above example. The first two numbers one these "outlier" lines are not important, they reflect some internal CODSPP numbering. Following the two numbers is the station name the outlier identification and the satellite PRN number. The outlier identification will usually be "OUT". The line is ended with the start and end epochs of the outlier interval and the number of observations in this interval.

If a specific station shows up often, something is probably wrong with its data. If a certain satellite shows up frequently there is something wrong with the satellite. In the above example satellite 19 was marked for many sites. As the marked time intervals are during the day (UT) and not only towards the end of the day the outliers are not caused by a manoeuvre. In fact, on this particular day satellite 19 was scheduled for maintenance. Usually the satellite clock is reset in such cases. The outlier detection of CODSPP handles such problems and no action is required by the program user.

DEFXTR and DEFXTP

For program ORBGEN the extraction programs DEFXTR and DEFXTP may be used. We recommend to use only the latter of the two. The difference between the programs has to be seen in number and the format of the output files that are generated. We discuss only the program DEFXTP and the two output files associated with it, the normal summary and the weekly summary.

```
 ORBGEN.L97      # Sat.:  25 , # Eclipsing  9 , Max. Rms.:  0.38 for sat.: 10
 (DOY: 219)      Eclips. Sat. :    3   7  10  14  16  21  23  28  31
                 Min in eclips:   24  23  11  28  30  20  30  13  22
                 Rms          :   37  13  38  14  31  11  11  11   9

219  6  4278  4  3  4  7  3 31256  5 21  4  4 11  6  4 35  4  7  4  5  6  3  5
```

The normal summary contains 4 lines. The first line contains the output file name, the number of satellites, the number of eclipsing satellites, the maximum rms of the fit, and the satellite associated with this rms. The second line gives the (middle) day of year for the arc and a list of the PRN numbers for the eclipsing satellites. The third line gives the time interval (minutes) of the eclipse; the last line gives the rms of the fit for the eclipsing satellites. Usually the rms of eclipsing satellites is larger than the rms of non-eclipsing satellites.

The weekly summary produces just one line per ORBGEN output containing for the (middle) day of the year for the arc and the rms for all satellites. The satellite numbers are not printed but they are listed in numerical order. This output, if always appended to one file to give a long time series, is useful for generating plots.

MPRXTR

The extraction program MPRXTR, for program MAUPRP, produces a summary of the following type:

```
SUMMARY OF THE MAUPRP OUTPUT FILE
*********************************

SESS FIL OK?  ST1  ST2 L(KM) #OBS.  RMS   DX    DY    DZ   #SL #DL #MA  MAXL3      MIN. SLIP
---------------------------------------------------------------------------------------------
2191   1 OK   NCRO SKOU 1879  4819 0.012 0.001-0.020-0.034   8 940  85  0.030             85
2191   2 OK   NCHU NFLI  658  9422 0.018 0.009-0.017 0.057  24 486 108  0.011         283610
2191   3 OK   NCHU NTHU 2202  8485 0.012-0.006 0.004-0.039  25 882  95  0.014         283610
  .    . .     .    .    .     .   .     .     .     .      .   .   .   .                 .
  .    . .     .    .    .     .   .     .     .     .      .   .   .   .                 .
  .    . .     .    .    .     .   .     .     .     .      .   .   .   .                 .
2191  67 OK   EWTR EZIM  476  9820 0.007 0.032-0.003 0.054  31 907  49  0.013        1821677
2191  68 OK   EWTZ EZIM  476  8612 0.009 0.036-0.005 0.033   2 369  61  0.015        1990187
2191  69 OK   AYAR JISC 5937  1964 0.007 0.003 0.031 0.058  24 499  52  0.004        6384009
-------------------------------------------------------------------------------
Tot:  69                2228  6997 0.008                   295 575  70
```

The summary file contains one line per baseline processed. The line starts with the session, a file sequence number and the "OK" flag. If the baseline was not successfully cleaned the flag will be different from "OK" and the baseline should be removed. The extraction program will also (try) to identify which of the two stations of the baseline was causing the problems. Apart from the summary file, MPRXTR may also create a deletion list file, see Chapter 23, and a new baseline definition file. The "bad" single difference file(s) and, optionally, the bad zero difference file(s) will be listed in the deletion list file. This file may then be used by program DELFIL, Panel 5.8, to delete these files. If a zero difference file was removed a new baseline should be created to make sure that the network is complete. MPRXTR defines a new baseline which is listed in the baseline definition file. This file in turn may then be used by program SNGDIF, Panel 4.3, to actually create the new baseline. Make sure that you do not forget to run MAUPRP for the newly created baseline. Note that the deletion list file and the new baseline definition file are mainly meant to be used in an automated processing.

The next items in the line, after the OK flag, are the first 4 characters of the station names and the length of the baseline. Thereafter some information concerning the triple difference solution is given: the number of observations, rms, baseline change in the Cartesian X, Y, and Z components. The next three fields contain the number of corrected cycle slips, the number of deleted points, and the number of multiple ambiguities. Finally the maximum residual for the ionosphere-free linear combination is given followed by the smallest corrected cycle slip.

The last line of the summary gives the number of files, the mean baseline length of all baselines, the mean number of observations, the mean triple difference rms, the mean number of corrected cycle slips, the mean number of deleted points, and the mean number of multiple ambiguities. See Chapter 10 for more information about the program MAUPRP.

GPSXTR

Program GPSXTR consists of four different extraction routines for specific purposes. First there is the "general" extraction which works for both programs GPSEST and ADDNEQ. The other three extraction routines access GPSEST output files only. One is a coordinate summary based on a baseline-wise processing (one baseline per run). The second one creates statistics for the baseline-wise "QIF" ambiguity resolution runs, and the third one generates statistics for ionosphere estimation runs.

Let us include below an example for the "general" extraction output.

```
ADDNEQ.L98      Rms:  2.9 , # fil.:  69 , # obs.: 144888 , # par.:  2873
(DOY:   0)                  Max. correction in a:  -0.13 +- 0.01 for sat.:  7
                            Pole correction (middle of interval)
                                       in x(mas):  -2.90 +- 0.119
                                       in y(mas):   0.83 +- 0.101
                                  in t(sec*1/D6):  52.40 +- 5.05
```

The first line contains the output file name, the rms (mm) of the solution, the number of single difference files, the number of observations used, and the number of estimated parameters. The second line should first contain the day of year but this is not always correct as can be seen in the above example. If orbit estimation was performed it also lists the maximum change in the semi-major axis ("a") and the corresponding satellite. The last four lines summarize the ERP estimates if ERPs were estimated and no separate pole output file was generated.

Apart from this summary two one line summaries ("campaign format" and "weekly summary") may be generated containing the same information in a different format. These one line summary files are very convenient to generate plots. Furthermore a pole output summary, if pole estimation was performed, may be generated.

At CODE we normally check the processing quality by processing all baselines one by one with GPSEST. In this case GPSXTR should be used to generate a coordinate summary of the following type:

```
BASELINE  #OBS.  #AMB  RMS(MM)  L(KM)  DHT(M)    DH(MM) +-    DN(MM) +-    DE(MM) +-    DL(MM) +-
---------------------------------------------------------------------------------------------------
ABDR2181  1989    45     1.9   302.      510.1     5.0  2.5     0.3  0.4    -1.1  0.9    -0.8  0.9
ABQU2181  1632    47     2.0   956.     1074.0    -7.4  4.0    -0.3  1.0     1.8  1.4     0.1  1.1
ABWH2181  1506    47     2.0  1562.     1395.6    -5.4  3.8     4.5  1.3    -0.2  1.5     3.2  1.4
```

This file contains for each run of program GPSEST the name of the single difference file, the number of observations, the number of ambiguities, the a posteriori rms, the length of the baseline, the height difference between the two points and the change of the baseline coordinates in height, north, east, and length with respect to the a priori coordinates with the associated formal uncertainties.

When using the "QIF" strategy (baseline by baseline) to resolve the ambiguities one may generate a special output summary and, in addition, a file which lists the fractional parts of all the resolved ambiguities.

```
File        Length   #Amb  RMS0   Max/RMS L5 Amb  Max/RMS L3 Amb   #Amb  RMS0   #Amb Resolved
            (km)           (mm)   (L5 Cycles)     (L3 Cycles)            (mm)   (%)
--------------------------------------------------------------------------------------------
ABDR2191     301.8    102   1.7    0.250  0.058    0.090  0.032      10   2.1    90.2
ABQU2191     956.0    100   2.4    0.440  0.166    0.099  0.035      12   2.8    88.0
ALGD2191     776.5    102   2.6    0.405  0.107    0.093  0.034      14   2.9    86.3
   .           .        .    .       .      .        .      .         .    .       .
   .           .        .    .       .      .        .      .         .    .       .
   .           .        .    .       .      .        .      .         .    .       .
TAUS2191     987.2     76   2.6    0.440  0.161    0.100  0.040      12   3.3    84.2
WRZI2191     475.9     80   2.7    0.307  0.061    0.099  0.031      10   3.0    87.5
WZZI2191     475.9     78   2.9    0.357  0.068    0.098  0.030      10   3.1    87.2
--------------------------------------------------------------------------------------------
Tot:  42     912.5   4084   2.8    0.499  0.130    0.100  0.037     766   3.2    81.2
```

This summary contains for each run of program GPSEST the name of the single difference file, the length of the baseline, the number of ambiguities (L1 and L2) before resolving any ambiguity and the rms before resolving any ambiguity, the maximum fractional part and the rms for all resolved ambiguity fractional parts for both, the L5 and L3 linear combination, the number of unresolved ambiguities, the rms after ambiguity resolution. The percentage of resolved ambiguities concludes the line.

In the last line of the summary the number of baselines used, the mean length of the baselines, the total number of ambiguities before resolving, the mean rms before resolving, the maximal fractional part and rms of all ambiguities and all baselines for both, the L5 and L3 linear combination, the total number of ambiguities after resolution, the mean rms after resolution and the percentage of fixed ambiguities.

If program GPSEST was used to estimate an ionosphere model a separate ionosphere summary output can be generated.

20.4 Conversions

The observation files in the Bernese format are binary files which cannot be looked at directly and which cannot be transferred between different operating systems (e.g. VMS to UNIX). Therefore two conversion programs are made available: OBSFMT, to convert binary observation files to ASCII and the second, FMTOBS, to convert the ASCII observation files back to binary. These programs are also used when editing or browsing files in Panel 5.1

Similar conversion programs exist for the so-called standard orbit files, Chapter 23. These are mainly used to transfer standard orbits to a different operating systems. There is not much sense in editing standard orbit files.

20.5 Delete Files

The program DELFIL, Menu 5.8, is used to delete files stored in the campaign directories. This option was implemented for the BPE, but may also be used manually. A so-called deletion file, see Chapter 23, may be specified containing file names which should be deleted (including the use of wildcards and “$” variables). You may also specify which file types should be deleted and select specific files of this type from a selection list.

21. Bernese Processing Engine (BPE)

21.1 Introduction

In the last few years more and more permanent networks have been established in many different regions of the world. The amount of data collected by such networks day by day calls for a highly automated GPS data processing. Large campaigns are also being organized with a few tens to a few hundreds of sites occupied. For this type of tasks the *Bernese Processing Engine* (BPE) has been developed to allow a very automated GPS data processing. At present the BPE is only available for multi-tasking systems (UNIX and VAX, but not for DOS). It has been installed and running at the CODE Analysis Center of the IGS for the routine processing of the global IGS network for more than one year and it was also implemented at GSI (Geographical Survey Institute) in Japan to process on a daily basis the nation-wide Japanese network consisting of over 600 receivers.

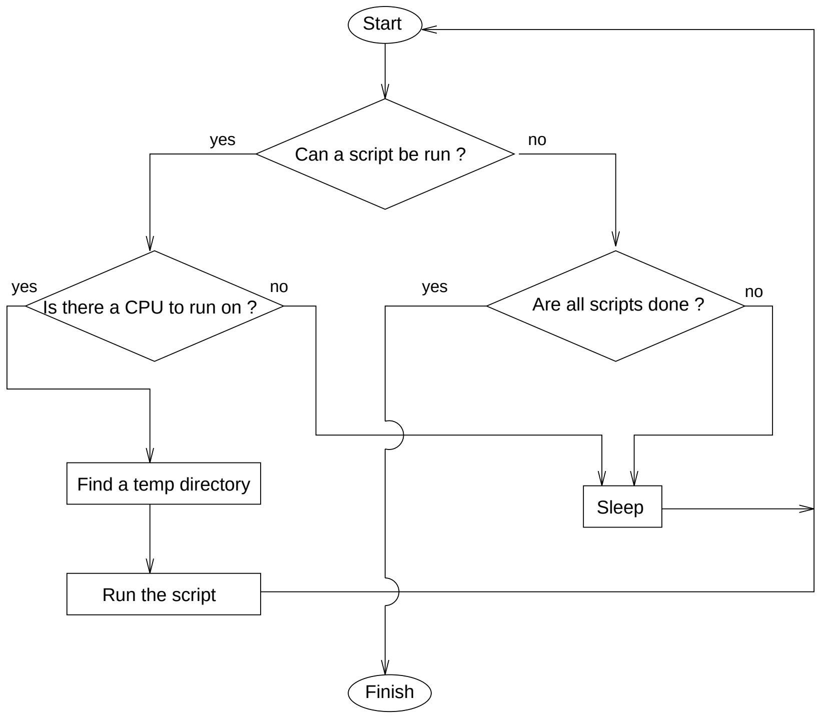


Figure 21.1: Process Control Script Flow Chart

The BPE is a system of programs, shell scripts, and control files designed to run and control Bernese GPS programs in an automated fashion. At the heart of the BPE is the Process Control Script (PCS). This script is responsible for starting and monitoring all the processes that are run by the BPE. The PCS can be given a list of various tasks to perform along with information about the interdependency of these tasks, such as which tasks need to wait for other tasks to finish, and the PCS will then start executing each task that is able to run. If a task cannot be run because it has to wait for another one to finish, the PCS will wait for the dependent tasks to finish. The PCS is also able to run more than one task at a time on several different CPUs, and can even divide a single task across different CPUs. A flow chart of what the PCS does is given in Figure 21.1.

The BPE provides a sturdy framework for writing shell scripts that can use the many different programs available in the Bernese GPS Software. It takes care of setting up environment variables, creating temporary work directories, error handling, logging, and interaction with the Bernese menu system. The actual body of a script that is run by the BPE can be very simple, such as calling one Bernese program, or quite complicated.

21.2 Getting Started

21.2.1 LOADGPS

As with the Bernese menu system, the BPE is started with the LOADGPS script. There are, however, some additional requirements on the LOADGPS script in order for the BPE to function correctly as listed below.

UNIX Version:

- The `SHELL` variable in LOADGPS must be set to a Bourne (`sh`) compatible shell. Shells such as the Bourne-Again shell (`bash`) can be used as long as they can interpret all standard `sh` commands.
- The LOADGPS script must be in the user's home directory and have execute permission on all hosts that are to be used by the BPE. This is because the BPE executes scripts by issuing a remote shell command to start the LOADGPS script. An alternative possibility is to make sure that the directory containing the LOADGPS script is part of the path. Depending on what shell the remote shell command actually starts up the path has to be set in a different login file (e.g. `.profile`, `.login`, `.cshrc`, ...).

VMS Version:

- The call of the command file `LOADGPS` has to be included in the user's `LOGIN.COM` (located in the user's home directory) to make sure that all symbols and logicals are correctly set.

21.2.2 LOADGPS Command Line Arguments

UNIX Version:

Normally the LOADGPS script is run without command line arguments. On UNIX systems there are two cases where the user wants to specify a command line argument. The first is when one of the directory variables in the LOADGPS script has been changed and the second is when the user wants to run a script automatically after the LOADGPS script starts. The second option is useful when the user wants to automatically start the BPE on some sort of schedule. The two options available are shown below. Only one of the options may be specified at a time.

`NEW`	:	This option will cause the LOADGPS script to re-set all directory links in the user directory structure to use the values specified in the LOADGPS script. This option should not be used if the user is currently running the BPE because programs and scripts that use the links that are re-set by this option may crash. THIS MUST BE DONE AFTER DIRECTORY VARIABLES ARE CHANGED IN LOADGPS!
`RUN_SCRIPT`	:	If this option is used, then LOADGPS will run the script named by the next argument. The script must be in the user's work area (`$U/WORK`). Example: `LOADGPS RUN_SCRIPT START_PCF` The above example will run the script `START_PCF` in the `$U/WORK` area after the LOADGPS script has finished.

VMS Version:

The command file `LOADGPS` is always run *without* command line arguments.

21.2.3 Getting Around

Once the LOADGPS script has been run, the user can use the directory variables that are defined in the script to navigate quickly through the directory trees. For example, to get to the work area, you can simply type:

```
erde[jjohnson]:V36-$ cd $U/WORK     (UNIX)
$   set default U:[WORK]     (VMS)
```

In the next sections we will not always include both the UNIX *and* the VMS directory and file names. VMS users should e.g. replace `$U/WORK` with `U:[WORK]`, `$P/CAMP/OUT` with `P:[CAMP.OUT]`, `$T/AUTO_TMP` with `T:[AUTO_TMP]`, etc.

The most important directories to know about are listed below:

`$U/WORK`	This is the user's main work area. This is where the Bernese menu system will write temporary files when the menu system is run interactively. Also, this is the directory where scripts that start up the BPE are normally kept. This directory is usually the starting point for all BPE and Bernese runs.

`$P/CAMP`	This directory is the top of a directory tree that contains all campaign-specific data of the campaign named `CAMP`. This includes raw and processed data, orbit data, and station information. The `P` variable (logical on VMS) is used here as an example, it could be any valid campaign variable set in the `LOADGPS` script.
`$P/CAMP/OUT`	One of the directories under the campaign directory tree is the `OUT` directory which holds all the output from Bernese and BPE programs.
`$T/AUTO_TMP`	The BPE creates a number of temporary files in this directory. It also stores log files (e.g. from remote shells) in this directory.
`$T/AUTO0001`	Under the `T` directory tree, there are a number of temporary directory structures that are created to run BPE scripts in. The names of these directories are `AUTOnnnn` where `nnnn` is a number. The BPE is able to run more than one script at a time and there will be at least one of these directories for each script the BPE is running. If there are no available directories when the BPE needs to run a script, then it will create a new one with a number one higher than the highest numbered `AUTO` directory.

21.2.4 The CPU File (PCFCTL.CPU)

The BPE is capable of running on a single host with one CPU, or on several hosts with multiple CPUs. In order to start jobs, the BPE must know which hosts it can use to run jobs on and how many jobs it can run on each host. This is all controlled by the file `PCFCTL.CPU` which is located in the `$U/WORK` directory. When the BPE is first installed, an example `PCFCTL.CPU` file is copied into `$U/WORK`. For example:

```
CPU      TRUE_NAME                      SPEED    NCPU SF   IDLE MAXJ JOBS
8******* 30**************************** 8******* 4*** 4*** 4*** 4*** 4***
CPU1     sakic                          FAST        4  100  100    2    0
CPUQIF   sakic                          FAST        4  100  100    2    0
```

Before you can start the BPE, you must edit the `PCFCTL.CPU` file so that it contains a valid CPU. Notice that the same CPU may be used twice. The fields of the `PCFCTL.CPU` file are:

`CPU` ...	This is the 'nickname' of the CPU that will be used in Process Control Files (PCF) to reference the CPU. The limit is 8 characters.
`TRUE_NAME` ...	This is the name of the host that is to be used in the `remsh` command. It must be a valid host name and can be up to 30 characters long.
`SPEED` ...	Reserved for future use, place the keyword FAST here. The field width is 8.
`NCPU` ...	The number of CPUs in the host. This value is not used by the BPE at this time and is reserved for future use. The field width is 4.
`SF` ...	Reserved for future use, place the value 100 here. The field width is 4.
`IDLE` ...	Reserved for future use, place the value 100 here. The field width is 4.
`MAXJ` ...	The maximum number of jobs the BPE is allowed to run on the CPU at a time. If the host is a single CPU computer, this number should be set to 2 or 3. If the host is a multi-CPU computer, then this number can be set higher. The field width is 4.

`JOBS` . . . This is the number of jobs that the BPE is currently running on the host. This value is maintained by the BPE and should be set to zero before starting any BPE jobs. The field width is 4.

The `PCFCTL.CPU` file is used by the BPE during run time in order keep track of how many jobs are running on each host. This means that the `PCFCTL.CPU` should not be edited while the BPE is running. Also, if the JOBS field in this filed is not zero when the BPE is started, then the BPE may not be able to start any jobs because it will think that too many jobs are running on all the hosts. There is a script named `CLEAN` in the directory `$X/EXE` than can be used to make sure the CPU file is clear before starting the BPE:

```
erde[jjohnson]:V36-$ . CLEAN    (UNIX)
$   @X:[EXE]CLEAN    (VMS)
```

The clean script should NOT be run if the BPE is running.

21.2.5 System Administration Notes

The BPE can be set up to run on a simple stand alone system or on a complicated network of computers. In the case of a stand alone system, there are only a few system administration details that are important:

UNIX Version:

- The user must have permission to execute a remote shell (`rsh` on most UNIX systems, `remsh` on HP-UNIX). To test if this is possible, the user can type the simple command "`rsh hostname ls`". If the user gets a response such as "`permission denied`", then he may need to consult his system administrator to enable remote shell capabilities. A common solution to this problem is to have a `.rhosts` file in the user's home directory that gives himself permission to execute remote shells.
- All required system environment variables must be set when the remote shell command is executed. Since the remote shell command is non-interactive, a different set of start up scripts may be started than when the user logs in with an interactive account. This can be confusing since a command may work for a user when the test is done in a login shell, but fails when the BPE issues the command from a script in a remote shell. A common problem is that the path to a command will not be found in a remote shell. Depending on what shell is actually used by the remote shell command you may have to define all start up variables in one of the login files (e.g. `.profile`, `.login`, `.cshrc`, ...). Another way to fix this problem is to place all start up variables required for the BPE into a file named `.profile.bernese` such as:

```
#
# setup command required for bernese
#
path=${path}:/usr/bin/extra_programs
#
DOT_PROFILE_BERNESE=TRUE
export DOT_PROFILE_BERNESE
```

and then add the following to the beginning of the LOADGPS script:

```
#
# make sure .profile.bernese has been executed
#
if [ "${DOT_PROFILE_BERNESE}" != "TRUE" ]
then
    . .profile.bernese
fi
```

The BPE has the ability to execute scripts on many different computers connected together by a network. There are some important restrictions for this configuration to consider:

UNIX Version:

- The LOADGPS script must be in the home directory on all hosts that are to be utilized. It is common in network systems to have one and the same home directory for a user independent of which host the user logs into, in which case the user only needs to maintain one LOADGPS script in that directory. On system where the user has a different home directory on each host, care must be taken to make sure a valid LOADGPS script is located in each of these directories. Another possibility is to make sure that the path that is set by the remote shell contains the directory with the LOADGPS script.
- There must be at least one common directory structure that is mounted on each of the hosts to be utilized to store campaign data (i.e. `$P` and `$Q`), temporary data (`$T`), and the user's work area (`$U`). The Bernese program and executable area (`$C`) may be in a directory local to each host if desired in order to minimize network traffic. On some systems, the exact path to mounted directories may be different on each system, which means that variables set in the LOADGPS may be host-specific. In the case where the user has the same home directory on all hosts, it is suggested that a separate LOADGPS file be maintained for each host and given the name LOADGPS.host. Then the LOADGPS script should be replaced with a simple script such as:

```
#!/bin/sh
#
# find out which host we are on
#
host=`/bin/hostname`
#
# execute the proper LOADGPS
#
if [ -f LOADGPS.${host} ]
then
    . LOADGPS.${host}
else
    echo LOADGPS.${host} not found
fi
```

- The Bernese and BPE programs may be compiled using shared libraries. This means that when a BPE program is executed, it must have access to these shared libraries. It is important to make sure that shared libraries are installed on all computers that are to run GPS programs. If you get errors when trying to run the programs that indicate that shared libraries cannot be found, you should consult your system administrator in order to correct the problem.

VMS Version:

The machines to be used by the BPE have to form a VMS cluster. The disks containing the directories for the campaign data (e.g. logical `P:`), the temporary area (`T:`), the user work area (`U:`), and the source code area (`C:`) have to be accessible from each of the machines.

21.3 Directory Structure

There are 4 directory structures in the Bernese/BPE system. Each of these directory structures can be located on different disks and the location of these directories are defined in the LOADGPS script. Three of the structures are fixed and generally do not change. These are the user's area (`$U`), program area (`$C`), and the temporary directory (`$T`). The fourth directory type is the campaign directory structure (`$P`, etc.) – see also Chapters 22 and 23. In addition to the directory structures themselves, there are two other items to be aware of. These are the directory variables (`$U`, `$T`, `$C`, `$P`, etc.) and directory links (links are not necessary and not present for VMS). The directory variables, which are defined in the LOADGPS script, are used by the user and BPE scripts to navigate to the base of each directory tree. On UNIX systems the links are found in the `WORK` directories under `$U` and the `$T/AUTOnnnn` directories. Directory links are used by the Bernese programs to find the location of directory structures. All of the Bernese programs expect to be started in a `WORK` directory that has these links set.

21.3.1 Directory Trees

An overview of the various directory structures used by the BPE is given in Figure 22.1 and Figure 23.1. You should keep in mind that *on UNIX systems* there are links defined (e.g. named `U:`) for each of these directory structures in the work area `$U/WORK` (pointing e.g. at the directory `$U`).

Notice that the directory link `U:` is recursive, i.e. that it points to an ancestor directory. This can cause problems if the user tries to copy the directory structure `$U` with the recursive option, for example:

```
erde[jjohnson]:V36-$ cd $U
erde[jjohnson]:V36-$ cp -r * ../BACKUP
```

will eventually fill up the disk. This is because the copy command follows links and will forever create directories like (assuming `$U` is `/home/user/GPSUSER`): `/home/user/BACKUP/WORK/WORK/WORK/WORK/WORK ...`

The `tar` command is safe, however, because it does not follow links and will instead copy the link information to the tar file. Thus, to safely back up the `$U` area, the user could do the following:

```
erde[jjohnson]:V36-$ cd $U
erde[jjohnson]:V36-$ tar cvf ../GPSUSER.BACKUP .
```

The `$T` directory structure deserves some additional comments. Notice that the `$T/AUTO0001` directory may have the `U:` notation in the BPE environment. This can cause confusion because the `$U` directory also has this notation.

The `AUTOnnnn` directories are used by the BPE to run scripts. Each script that the BPE runs requires its own work area because the Bernese programs expect to write temporary files in the `$U` area (`$U/INP`, `$U/PAN`, and `$U/WORK`). It also expects to find the input parameters in panel files in `$U/PAN`. So, in order to run more than one script at a time, it is necessary to have different directory structures for each script. When the BPE runs a script, it finds an available `AUTOnnnn` directory structure to run in and then re-sets the `$U` variable to that `AUTOnnnn` area. All the panel files in the `AUTOnnnn/PAN` area are deleted and new panel files for the specific script to be run are copied over. On UNIX systems the BPE also sets the links in the `AUTOnnnn/WORK` area. Thus, as far as the Bernese programs know, they are running in the standard `$U/WORK` area. So, the `$U` and `U:` directories are specific to each work area. The `U:` link in `AUTO0001/WORK` points to `AUTO0001`, `U:` in `AUTO0002/WORK` points to `AUTO0002` and so on. The user will never see the changing of `$U` to one of the temporary directories as this is all handled internally by the BPE. The `$U` variable defined at the Bourne shell prompt will always be the same.

The reason for this `AUTOnnnn` structure and the different `$U` variables is that each BPE script can have a user (work) area to run Bernese programs without overwriting input/output files from other scripts.

21.4 The Process Control Script (PCS)

The Process Control Script, or PCS, is the outer most script of the BPE. The script lives in the `$X/EXE` directory, which will be in the user's path after the LOADGPS script is run. The PCS script expects to be run from the user's `$U/WORK` area and takes as its basic parameters the name of a Process Control File (PCF), the session to run, and the year of the session. In general the user will start the PCS using the Menu 6.4.1 which allows him to specify all the parameters passed to the PCS script in input panels (see Section 21.7.5).

21.4.1 How the PCS Works

The main function of the PCS is to execute the scripts contained in a Process Control File (PCF). The PCF, which is covered in more detail in a later section (see Section 21.5), is a file that contains a list of scripts that are to be run. In addition, the PCF contains information about what input options to use for these scripts and the order in which to run the scripts. Most importantly, the PCF file tells the PCS script which scripts have to have finished in order to start any given script. The steps the PCS takes are outlined below:

(1) Validate input parameters. The PCS will check to make sure that:

- Session and year are defined and all parameters are valid.
- The PCF file is in the `$U/PCF` directory.
- All scripts in the PCF file are in `$U/SCRIPT`.

– All options directories in the PCF file are in `$U/OPT`.

If any of the above tests fail, the PCS will display an error message and terminate.

(2) Clear all protocol files. The PCS uses protocol files in order to track the status of the scripts in the PCF that are to be run. It uses these files to tell if they have been started and to tell when a script has finished. These protocol files are stored in two locations, `$T/AUTO_TMP` and `$P/CAMP/OUT`. The protocol files in `$T/AUTO_TMP` will be empty, they are only used to indicate that a script has been started. The protocol files in `$P/CAMP/OUT` will contain messages from the script and will only exist after the script has written information to them. The reason for empty protocol files in the `$T/AUTO_TMP` directory is that there can be a significant delay between when a script is started and when the script actually writes a message to the protocol file in `$P/CAMP/OUT`. After the PCS starts a script, it will check to see if there is another script to start, and if the PCS just checked for the existence of the protocol file for the script in `$P/CAMP/OUT`, it may not have been created yet and the PCS would think it needs to start the script again. Therefore, an empty protocol file is immediately created in the `$T/AUTO_TMP` area so that the PCS will know that the script has started.

(3) Check for errors in the protocol files. The PCS will scan all the protocol files generated for errors. If an error is found, then the PCS will print an error message and terminate. Any remote jobs that have been started and have not finished will still run, but new scripts will not be started.

(4) See if there is a script that can be run. The PCS goes through each script in the PCF file and checks to see if it has been started. If it has not been started, then it checks to see if the scripts that it needs to wait for have finished. If these scripts have finished, then the PCS will start the script and go to step 6. If the script has been started, or if the scripts it needs to wait for have not finished yet, then the PCS will move on to the next script until it either finds a script that is ready to run – in which case we go to step 6 – or there are no more scripts to check. If the PCS cannot find a script to run and not all the scripts have finished, then it goes to step 5. If all scripts have finished, then the PCS terminates.

(5) Sleep and then go back to step 3.

(6) Check the `PCFCTL.CPU` file (located in `$U/WORK`) for a computer to run the script on. The `PCFCTL.CPU` file specifies which computers the BPE is allowed to run scripts on and how many scripts at a time it can run on these computers. The PCS will look for an available computer and if it does not find one, it will go to step 5 (sleep a while, then check again). If a computer is found, the PCS increments the count for the number of jobs running on the computer in the `PCFCTL.CPU` file and then continues with the next step.

(7) Run a script. The PCS comes to this step if it has found a script it can run (step 4) and there is an available computer to run the script on (step 6). The PCS will generate a header and tail script and copy them into the `$T/AUTO_TMP` directory. The header and tail scripts are attached to the beginning and end of the script to run. These scripts take care of the book-keeping required by the BPE, such as writing messages to the protocol file, and set up variables that define the parameters (such as the year and session) that are required by the script. The names of these files will be `HDRnnnnn` and `TL_nnnnn`, where `nnnnn` is a unique ID number that the PCS automatically generates. The PCS then generates a script that will be executed by an `rsh` command (UNIX) or a `submit` command (VMS) that will start the script in the PCF file. This file will be named `PCFnnnnn.RSH` (`PCFnnnnn.COM` on VMS) and will be copied over to the `$T/AUTO_TMP` directory. Finally, the PCS will start up the script by issuing an `rsh/submit` command. The `rsh/submit` command will run the LOADGPS script with special options that specify that the script `PCFnnnnn.RSH/PCFnnnnn.COM` should be run after the Bernese environment has been loaded. After the `rsh/submit` command has been issued, the PCS will go back to step 3.

The PCS process finishes once all scripts have been started and have finished, or if an error is found in one of the protocol files.

21.4.2 Format of Protocol Files

There are two protocol files that will be generated for each script that is started. One will be in the `$T/AUTO_TMP` directory and is used to tell the PCS that a script has been started. The other will be in the `$P/CAMP/OUT` directory and is used to report the status of the script (errors, completion status, etc.). The name of the protocol file has the following format:

`TTYYSSSS.PID` or `TTYYSSSS.PID_SUB`

Where:

`TT` ...	Task id, 2 characters, has to be used only, if more than one BPE runs using the same campaign and session(s) (default task id is `00`).
`YY` ...	Two digit year.
`SSSS` ...	The session number.
`PID` ...	The three digit process id of the script (defined in the PCF file).
`SUB` ...	The sub process id of the script. Parallel scripts can be run a number of times. Each separate run of a parallel script is given a sub process id, starting with 001 and incrementing by 1 each subsequent run. The `_SUB` is only present for parallel scripts.

The protocol file in `$T/AUTO_TMP` will be empty. The PCS only checks to see if this file exists in order to see if a script has been started. To see if a script has finished, the PCS will look through the protocol file in `$P/CAMP/OUT`. Here is an example protocol file from the campaign `OUT` directory:

```
erde[jjohnson]:V36-\$ cd $P/NOAA_FSL/OUT
erde[jjohnson]:V36-\$ cat 00950170.001
PROTOCOL FILE FOR BPE SCRIPT
----------------------------

SCRIPT NAME         : FTP_RNX
YEAR                : 95
SESSION             : 0170
CAMPAIGN            : NOAA_FSL
CAMPAIGN PATH       : P:/
OPTION DIRECTORY    : FTP_NOAA
PROCESS ID          : 001
SUB PROCESS ID      : 001
CPU                 : CPU1
PATH TO WORK AREA   : /u/erde1/rocken/GPSDATA/AUTO0001/

 DATE      TIME     STA PROGRAM  MESSAGE
----------------------------------------
 31-MAR-95 21:07:21 MSG FTP_RNX  PROCESS STARTED
 31-MAR-95 21:07:26 MSG FTPRNX   PROGRAM STARTED
 31-MAR-95 21:08:21 MSG FTPRNX   PROGRAM ENDED
 31-MAR-95 21:08:23 MSG FTP_RNX  PROCESS ENDED
---------------------------------------
```

The protocol file first displays the value of the variables that were set in the header script (`HDRnnnnn`). Below is a table of the items displayed:

`SCRIPT NAME`	...	This is the name of the script that was executed.
`YEAR`	...	The two digit year.
`SESSION`	...	The four character session.
`CAMPAIGN`	...	Name of the campaign.
`CAMPAIGN PATH`	...	The path to the campaign. For UNIX systems this shows the link name. For the above example, the campaign path is given as `P:/` which corresponds to the `$P` variable set in the LOADGPS script.
`OPTION DIRECTORY`	...	This is the name of the option directory used for input panel files. The option directories are located in the `$U/OPT` area.
`PROCESS ID`	...	This is the 3 digit process id. The process id for a script is assigned in the PCF file.
`SUB PROCESS ID`	...	The sub process id only has a meaning for parallel scripts. For parallel scripts, this indicates which run of the script the protocol file is for. For example, if a parallel script is split into 10 separate runs, then there will be 10 protocol files with sub process ids from 001 to 010.
`CPU`	...	This is the name of the computer (or the batch queue on VMS) that the script was run on. This is the abbreviated name. The full name of the computer (batch queue) can be found by looking at the `PCFCTL.CPU` file in `$U/WORK`.
`PATH TO WORK AREA`	...	This shows which `AUTOnnnn` directory was used as a temporary work area for the script.

Below the display of the variables, a chronology of which programs were run is shown. There may also be some additional warnings and/or error messages. The first two columns show the date and time the message was added to the protocol file. The third column shows the type of message. The possibilities are MSG (a simple message), WAR (a warning), and ERR (a fatal error). The fourth column shows the program or script the message comes from and the final column shows the message. The first and last message in a protocol file will be from the script being run, the first being PROCESS STARTED and the final PROCESS ENDED. The PCS looks for the message PROCESS ENDED to determine if a script has finished.

21.4.3 Log Files

When the PCS runs a script, it logs all the output from the rsh command (UNIX) or from the submitted job (VMS) in a log file in $T/AUTO_TMP. The name of the log file is the same as the protocol file except that the first character in the extension is the letter L. For example, here is the log file 00950730.L01 on a UNIX system:

```
--------
HDR00211
--------
HEADER: cd to /u3/jjohnson/GPSDATA/AUT00001/
HEADER: made directories and links
HEADER: in directory /u3/jjohnson/GPSDATA/AUT00001
HEADER: will delete old PAN files:
HEADER: copying PAN files from /u3/jjohnson/GPSUSER/OPT/FTP_BERN
-------------------

 29-MAR-95 12:10:31 MSG FTP_ORB  PROCESS STARTED

-------------------
-------------------------------
STARTING RUN_PGMS with FTPORB
-------------------------------
RUN_PGMS: COMFILE_2 exists and is: /u3/jjohnson/GPSDATA/AUT00001//WORK/
MENUBAT2.COM

-------------------

 29-MAR-95 12:10:38 MSG FTPORB   PROGRAM STARTED

-------------------
U:/WORK/FTPORB.COM started at : Wed Mar 29 12:10:39 MST 1995
will get cod07922.eph
Interactive mode off.
Local directory now /u3/jjohnson/GPSDATA/JJ_TEST/ORB
start
cod07922.eph
U:/WORK/FTPORB.COM   ended at : Wed Mar 29 12:10:54 MST 1995

 29-MAR-95 12:10:55 MSG FTPORB   PROGRAM ENDED

 29-MAR-95 12:10:58 MSG FTP_ORB  PROCESS ENDED
```

The log file will contain any output from the script that is not explicitly directed to the protocol file. Most importantly, if a shell/command file were to have a hard crash, such as from a program crashing with a segmentation fault, then the error message would appear in the log file rather than in the protocol file. This is because scripts have to explicitly write messages to the protocol file, and if the script itself crashes, there is no way for it to send a message to the protocol file.

21.5 The Process Control File (PCF)

The Process Control File (PCF) defines which scripts the Process Control Script (PCS) should run. In addition to listing the scripts to run, the PCF defines which scripts must wait for other scripts before they can be run and defines parameters that are to be passed into the scripts.

21.5.1 Linear PCFs

The basic PCF is linear. This means that each listed script is executed only once. It is still possible that processing will happen in parallel by having two different scripts running at the same time. Each script, however, is only run one time. For example, it may be possible to run an ftp script to get precise orbits at the same time a script is running to get IGS RINEX data.

A very simple PCF file is shown below (`GET_ORB.PCF`):

```
#
# Procedure Control File (PCF)
# All comment lines start with a #
# Comments:
#   GET_ORB.PCF -> a PCF file that will retrieve COD orbits from CODE
#
PID SCRIPT   OPT_DIR  CAMPAIGN CPU      P WAIT FOR....
3** 8******* 8******* 8******* 8******* 1 3** 3** 3** 3** 3** 3** 3** 3** 3**
001 FTP_ORB  FTP_BERN          any      1
#
# additional parameters required for PID's
#
PID USER         PASSWORD PARAM1   PARAM2   PARAM3   PARAM4   PARAM5   PARAM6
3** 12********** 8******* 8******* 8******* 8******* 8******* 8******* 8*******
#
# PCF Variables
#
VARIABLE DESCRIPTION                              DEFAULT          LENGTH
8******* 40************************************** 16************** 2*
#
# That's it
#
```

This is a simple PCF file that contains only one script to run. This PCF file will run a script to retrieve CODE precise orbits from the IGS Analysis Center CODE. Any line that starts with a `#` is a comment and comments can appear at any location in the PCF. The first two non-comment lines define the fields for the first section of the PCF. These two lines must be present. After the two header lines, the scripts that are to be run are listed. A description of the different fields is given below:

`PID`	...	Each script in the PCF is assigned a unique Process IDentification number. It is up to the author of the script to assign PID numbers to the script and the only restriction is that they must be unique 3 digit numbers. Usually the first script is given a process id of 001 and each subsequent script is given a PID one higher.
`SCRIPT`	...	This is the name of the script to run. The script must be located in `$U/SCRIPT`.
`OPT_DIR`	...	This is the name of the option directory that will be used to get panel files from. The option directories must be in `$U/OPT`.

`CAMPAIGN` ... This is the name of the campaign to process. Normally you will leave this field blank to be able to process any campaign using this PCF. If you specify a campaign name here it will take precedence over the campaign selected in Menu 6.4.1 when starting the BPE run.

`CPU` ... This is the name of the computer/queue that the script should be run on. This can be the 8 character name specified in the `PCFCTL.CPU` file (found in `$U/WORK`) or `ANY`. If the keyword `ANY` is used, then the script will be run on the first available computer found. You may also specify `FAST` or `SLOW`. In this case a computer with the keyword `FAST` or `SLOW` respectively in the `PCFCTL.CPU` in column `SPEED`, will be selected. Still another possibility is `IDLE` which means that a computer/queue is chosen where no other BPE script is running yet. The computer/queue given in the PCF file takes precedence over the `CPU` selection in Menu 6.4.1.

`P` ... This is the priority of the script. The number can range from 1 to 9. On UNIX systems a priority of 1 will cause the script to be run with the highest possible priority and a value of 9 will cause the script to be run with the lowest possible priority (`nice` command). On VMS different batch queues have to be selected for jobs that should run with different priorities.

`WAIT FOR` ... This field specifies which scripts must have finished before the script can be started. Up to nine scripts to be waited for can be specified.

The next section of the PCF file starts with two more header lines. If these additional parameters are not required for a script, then this section can be left empty. If additional parameters are required, then they will be listed in this section. The meaning of the different fields is listed below:

`PID` ... This is the PID of the script that the parameters belong to. This `PID` must be listed in the first section of the PCF file.

`USER` ... This field is reserved for future use and is not used at this time.

`PASSWORD` ... This field is reserved for future use and is not used at this time.

`PARAMn` ... Up to six parameters at a time can be specified to each script. There are some parameters that have a special meaning for all scripts and are listed below:

- `SKIP` ... If the keyword `SKIP` is passed to the script as `PARAM1`, then the script will not execute. It will write a message to its protocol file that it was skipped and then exit.
- `PARALLEL` ... This keyword is used for parallel scripts, which will be covered in detail in a later section.
- `$<extension>` ... This is used to generate unique file names to be used by parallel scripts. The PCS will automatically expand the `$` into a unique number and add the extension to it. For example, `$tmp1` would be expanded to a file name `019283.tmp1` (the 019283 will be different for different runs).

The third and final section of the PCF defines any special variables for the PCF. If the PCS is started from the Bernese menu system then a special panel will be created and presented to the user based on this information. The key words are:

`VARIABLE` ... This is the name of the variable that will be set in all scripts run by the PCF. These variables must start with the two characters `V_` and may then have up to six more characters. There are 7 special variables that can be set that correspond to variables in Panel 1.5.1 (see also the corresponding help panel and Chapter 3.8).

- `V_O` ... Sets the `$O` variable.
- `V_U` ... Sets the `$U` variable.
- `V_V` ... Sets the `$V` variable.
- `V_W` ... Sets the `$W` variable.
- `V_Z` ... Sets the `$Z` variable.
- `V_PLUS`, `V_MINUS` ... These two variables are used together to specify a range in the following Panel 1.5.1 variables:

`$SS1`	...	4-character session parameter `4-SESS1`
`$GW1`	...	4-character session parameter `4-WEEK1`
`$JRD1`	...	5-character session parameter `5-SESS1`
`$GDY1`	...	5-character session parameter `5-WDAY1`
`$JRSS1`	...	6-character session parameter `6-SESS1`

The five variables `V_x`, if defined in the PCF file, will be available in all scripts and the `$x` in all panels. For example, if the user were to set `V_O` to the value `XI`, then `$O` in a Bernese panel would be replaced with `XI` and the variable `$V_O` (in a UNIX script) resp. the symbol `V_O` (in a VMS command file) would be `XI`. The `V_PLUS` and `V_MINUS` variables will be concatenated together and appended to the session variables listed above.

`DESCRIPTION` ... This field can be up to 40 characters and defines the description the variable will be given.

`DEFAULT` ... This defines the default value the variable will have.

`LENGTH` ... This tells the Bernese menu system how many characters to allow for the variable. The maximum is 16.

NOTE:

For the 5 special `V_x` variables above, the length must be 2 because they are 2-character variables in the Bernese menu system.

EXAMPLE:

The following excerpt from a PCF would generate the following Panel 6.4.1–1.3 when starting the BPE with Menu 6.4.1:

```
VARIABLE DESCRIPTION                              DEFAULT          LENGTH
8******* 40************************************** 16************** 2*
V_O      EUROCLUS ORBIT FILE NAME                 R3                2
V_X      AMBIGUITY FREE  RESULTS                  EG                2
V_Z      AMBIGUITY FIXED RESULTS                  EQ                2
V_U      U ....                                   UU                2
V_V      V ....                                   VV                2
V_W      W ....                                   WW                2
V_PLUS   PLUS  DAYS                               +0                2
V_MINUS  MINUS DAYS                               -0                2
```

```
6.4.1-1.3           BPE SESSION PROCESSING:  SPECIAL PARAMETERS

  Special Parameter Setting:
    EUROCLUS ORBIT FILE NAME                 "O"       > R3 <
    AMBIGUITY FREE  RESULTS                  "X"       > EG <
    AMBIGUITY FIXED RESULTS                  "Z"       > EQ <
    U ....                                   "U"       > UU <
    V ....                                   "V"       > VV <
    W ....                                   "W"       > WW <
    PLUS  DAYS                               "PLUS"    > +0 <
    MINUS DAYS                               "MINUS"   > -0 <

  Control Process:
    SLEEP TIME    > 0      <    (in seconds, 0: default value used)
```

21.5.2 Parallel PCFs

Parallel PCFs are slightly more complicated that linear PCF files. They have the advantage, however, that they can split up a single task into multiple tasks, each of which can be executed on a separate computer. For example, to run the Quality Check program (QC) on all RINEX files, a linear PCF script could be written that would simply loop over all RINEX files for the specified year and session and run the QC program on each one of these files. This script would process one RINEX file after another in a linear fashion. With parallel scripts, it is possible to divide up the RINEX files into groups and have different computers work on different groups of RINEX files. Improvements in speed can be achieved even by running two groups of RINEX files on the same computer. This is because while one group is utilizing the disk I/O of the computer, the other can be utilizing the CPU and vice versa.

Parallel scripts require two scripts that work together in conjunction. The first script is the file script and its job is to either (a) create a list of files (e.g. observation files) to be processed one-by-one in parallel or (b) to divide up a list of files that are to be processed in groups. The second script is the script that actually does the work and will be executed once for each individual file in the file list in case (a) and once for each group of files generated by the file script in case (b).

Below is an excerpt from the example PCF file given in `$X/PCF` that shows the parallel script files for running the program GPSEST with the QIF ambiguity resolution strategy on each individual baseline of a session:

```
PID SCRIPT   OPT_DIR  CAMPAIGN CPU      P WAIT FOR....
3** 8******* 8******* 8******* 8******* 1 3** 3** 3** 3** 3** 3** 3** 3** 3**
:
018 GPSQIFAP EURO_QIF          any      1 017
019 GPSQIF_P EURO_QIF          any      1 018
:
PID USER         PASSWORD PARAM1   PARAM2   PARAM3   PARAM4   PARAM5   PARAM6
3** 12********** 8******* 8******* 8******* 8******* 8******* 8******* 8*******
018                       $tmp1
019                       PARALLEL $tmp1
:
```

The first section defines the option directories to use for the scripts and the order in which they must be run. In this case, the GPSQIFAP script must wait until script 017 has finished before it can start. Since the GPSQIFAP script only creates a list of baselines to be processed in parallel, it does not need any input options and instead of the option directory EURO_QIF an (almost) empty option directory (usually called NO_OPTS) could be specified. This option directory NO_OPTS is a directory only containing some panels that have to be present in any option directory (e.g. the default panels DAT0xxxx.PAN, the panel DAT151__.PAN, etc.) in the $U/OPT area. The GPSQIF_P script is the one that will do the work and the name of an option directory is passed to it that contains panel and option files that are needed in order to run GPSEST.

The bottom section of the PCF defines the parameters that are passed into the different scripts. The parameter $tmp1 is passed into GPSQIFAP as PARAM1. The $tmp1 variable is automatically expanded by the PCS into a file name that has a unique base name and the extension .tmp1 (01298.tmp1 for example). The GPSQIFAP script will write into the file $tmp1 one line for each baseline to be processed. The lines that GPSQIFAP writes into the $tmp1 file will be used by the PCS to generate parameters for the parallel script GPSQIF_P.

The keyword PARALLEL is specified for the GPSQIF_P script followed by the file name $tmp1. When the PCS detects the PARALLEL keyword, it will read the file specified in PARAM2, which is $tmp1 in this case, and read and remove the first line from this file. This line is then used to generate the PARAMn parameters that are actually passed into the GPSQIF_P script: the first item in the line will be passed to the script as PARAM1, the second as PARAM2, etc. The PCS keeps track of how many times it has executed the GPSQIF_P script and passes this number to the script as the subprocess id. This process of reading and removing the first line from the $tmp1 file continues until the $tmp1 file is empty, in which case the PCS will mark the script as being started. Once all of the scripts started by the PCS have finished, the script will be marked as done, and any scripts that have been waiting for the GPSQIF_P script to finish will then be able to run.

The protocol and log files for the GPSQIF_P will have an additional number added to the normal extension. For example, the first time the GPSQIF_P script is run, its protocol file name will be (assuming we are running session 1650 of the year 1996) 00961650.019_001 (in $P/CAMP/OUT) and 00961650.L19_001 for the log file (in $T/AUTO_TMP).

If you want to process files in N groups (e.g. running program CODSPP with groups of files, i.e. running it once for each group of code observation files) the preparatory script has to generate N files (one for each group, let's call them group files) containing each a list of the files belonging to this group and one file (the $tmp1 file) containing the names of the N group files (one per line). The PCS then passes the name of the group file to the script to be run using the parameter PARAM1. The parallel script (the script that runs e.g. CODSPP) may then copy the group file to the "SELECTED" file (e.g. `$U/WORK/CDZFILE.SEL` for CODSPP), the file where the menu system saves the list of files selected the last time by the user (see 3.4.3). By setting the option “SELECTED” in the panel, where the names of the files to be processed have to be selected (e.g. `DAT42___.PAN` for CODSPP), exactly the files contained in the group file will be processed.

21.6 Running a PCS

We will now go through the process of running a simple PCS.

21.6.1 An Example PCF

The PCF file that we will use for this example is `T_PCF.PCF`:

```
#
# Process Control File
# Comments:
# Any line that starts with a # is a comment line
#
#
PID SCRIPT   OPT_DIR  CAMPAIGN CPU      P WAIT
*** ******** ******** ******** ******** * *** *** *** *** *** *** *** *** ***
001 T_SCRIPT NO_OPTS           any      1
002 T_SCRIPT NO_OPTS           any      1 001
#
# additional parameters required for PID's
#
PID USER         PASSWORD PARAM1   PARAM2   PARAM3   PARAM4   PARAM5
*** ************ ******** ******** ******** ******** ******** ********
#
# PCF Variables
#
VARIABLE DESCRIPTION                              DEFAULT          LENGTH
8******* 40************************************** 16************** 2*
V_0      TEST SETTING V_0                         HI                2
```

This PCF will run the script `T_SCRIPT` (which must be in `$U/SCRIPT`) twice. Since we have a wait PID for the second `T_SCRIPT`, it must wait for the first one to finish before it can start. If we did not have a wait PID for `PID 002`, then the BPE would start the second `T_SCRIPT` script right after the first one was started.

The option directory for the script `T_SCRIPT` has been selected to be `NO_OPTS`. In this option directory we only need to have a few basic panels that are required by the Bernese menu system. `T_SCRIPT` does not run any BPE programs and, in fact, will only echo some messages. At this point, we only want to show the flow of the BPE.

Here is the script `T_SCRIPT` for the UNIX case (for VMS please have a look at the scripts in `X:[USERSCPT]`):

```
#!/bin/sh
#
# T_SCRIPT
# ========
#
# An example script to show the BPE functioning
# ---------------------------------------------
#
# sh script file written by bds
# -----------------------------
# functions used by shell
do_rm( ) {
    if [ "$1" ]
    then
        if test -f `echo $1 | tr ' ' '\012' | head -1`
        then
            eval rm $1
        fi
    fi
}
#
toupper( ) {
    eval $1=`echo $2 | tr '[a-z]' '[A-Z]'`
    eval export $1
}
#
seterr( ) {
    if [ $? = 0 ]
    then
        ERRSTAT=OK
    else
        ERRSTAT=ERR
    fi
}
#
# -----------------------------
# SHELL STARTS HERE
# -----------------------------
#
# SHELL VARIABLES:
# ----------------
#
# YEAR    :  Year of the session to be processed (2 digit)
# SESSION :  Session number (4 characters)
# CAMPAIGN:  Campaign name
# CAMP_PTH:  Campaign path
# CAMP_DRV:  Drive letter for campaign (i.e. P)
# OPT_DIR :  Directory for panels
# PID     :  Process identification number (3 digits)
# SUB_PID :  Subprocess id (3 digits)
# PRT_FILE:  Protocol file name including path
# SCRIPT  :  Name of script
# TASKID  :  Task id of script, usually 00
# PRIORITY:  Priority of the script
# CPU     :  CPU the script is running on
# DAYYEAR :  Julian day of the year
# DAY     :  Day of the Month
# MONTH   :  Month, 1=JAN, 12=DEC
# GPSWEEK :  GPS week
# DAYWEEK :  Day of the week, 0=SUN, 6=SAT
# V_X     :  X Variable in DAT151__.PAN
# V_O     :  O Variable in DAT151__.PAN
# V_Z     :  Z Variable in DAT151__.PAN
# V_PLUS  :  Plus variable in DAT151__.PAN
# V_MINUS :  Minus variable in DAT151__.PAN
# V_x     :  User variable
# PARAMn  :  Script specific parameter, n is 1 thru 9
# U       :  Directory path to U:
#
# See if this shell is being run from the PCS
# script, or directly for testing.  If testing,
# the header script is not passed in as %1%
#
if [ "$1" = "" ]
then
#
#   set variables for testing here
#
    TEST_START_DIR=`pwd`
    cd $U/WORK
    TESTING="YES"
    export TESTING
else
    TESTING="NO"
    export TESTING
    . "$1"
    seterr
fi
#
# START THE MENU SYSTEM IN NON-INTERACTIVE MODE
# --------------------------------------------
. $X/SCRIPT/BEG_MENU
seterr
#
# SET VARIABLES IN DAT151__.PAN
# -----------------------------
. $X/SCRIPT/SET_SESS
seterr
#
# T_SCRIPT BODY
# -------------
#
# This is a simple test script that does nothing.
# It starts HERE
#
echo T_SCRIPT: Starting
echo SESSION IS ${SESSION}
echo V_O IS ${V_O}
echo T_SCRIPT: Ending
#
# It ends HERE
#
# --------------
# end the script
# --------------
. $X/SCRIPT/END_MENU
seterr
if [ "$TESTING" = "YES" ]
then
    cd "$TEST_START_DIR"
else
    . $X/SCRIPT/DO_TAIL
    seterr
fi
```

At first glance this script looks complicated. However, most of the script is a ’skeleton’ required by the BPE (all scripts have these start and end sections). What the script actually does is between the lines:

```
# It starts HERE
# It ends HERE
```

So, all this script does is echo some information (no GPS program is run). The output of the echo commands will be captured in the log files.

Scripts written for the BPE have to be standard Bourne shell scripts under UNIX, DCL command files under VMS. Notice that most of the above script consists of comments. The line

`. "$1"` (UNIX)

`$ @'P1' 'P1' 'P2' 'P3' 'P4' 'P5' 'P6' 'P7' 'P8'` (VMS)

executes the header script that the PCS generated (the PCS will pass the name of the header script as first parameter named `$1` under UNIX, `P1` under VMS). Then two scripts (`BEG_MENU` and `SET_SESS`) are run that will put the Bernese menu system into the non-interactive mode and set the campaign and session (which are defined in the header script). The last two lines in the script put the Bernese menu system back into interactive mode (script `END_MENU`) and run the script `DO_TAIL` that does some finishing tasks, such as writing the final messages into the protocol file. The `DO_TAIL` script will also execute the tail script that is generated by the PCS.

To start the PCF from the menu system, we select Menu 6.4.1 :

```
 6.4.1  |                 BPE: SESSION PROCESSING
--------+---------------------------------------------------------

    CAMPAIGN              > TST_CAMP <    (blank for selection list)

  Job Identification:
    JOB CHARACTER            >   <        (blank, or A..Z, 0..9)

  Input Files:
    PROCESS CONTROL FILE  > T_PCF    <    (blank for selection list)
```

We are using a fictitious campaign with the name `TST_CAMP` to demonstrate the menu steps. Next the following panel is displayed:

```
 6.4.1-1 |         BPE SESSION PROCESSING: INPUT OPTIONS
---------+---------------------------------------------------------

   Sessions Information:
     SESSION (START)      > 1110 <
     YEAR (START)         > 96   <
     NUMBER OF SESSIONS   > 1    <      (if negative: processing backwards)

   Task Identification:
     TASK IDENTIFICATION  > 00 <        (blank: 00)

   CPU/QUEUE Specification:
     CPU / BATCH QUEUE    > NO       <  (NO, or blank for selection list)

   Special Options:
     SPECIAL PARAMETERS   > NEW  <      (OLD.. NEW.. or ASIS)
     SKIP PROCESSES       > NO   <      (YES..  NO,  or ASIS)
     REMOTE SUBMIT        > NO   <      (YES..  NO,  or ASIS)
     DEBUGGING OPTIONS    > NO   <      (YES..  NO,  or ASIS)
```

We are thus processing the session 1110 of the year 96. Then the Panel 6.4.1–1.3 is displayed:

```
6.4.1-1.3          BPE SESSION PROCESSING:  SPECIAL PARAMETERS

   Special Parameter Setting:
     TEST SETTING V_0                        "0"      > HI <

   Control Process:
     SLEEP TIME    > 0      <    (in seconds, 0: default value used)
```

Here is where we are prompted for the `V_0` parameter. After we accept this panel, the BPE starts to run (whether the BPE runs in the foreground or background depends on the setting of the `JOB CLASS` option in Menu 0.1). First the PCS will read the PCF file specified and check to make sure that the option directories of all scripts are present. It will also do some basic error checking to make sure that the PCF file is consistent. Then the PCS will remove any protocol files for the year and session in the campaign OUT directory (`$P/TST_CAMP/OUT` in this case) and log files in `$T/AUTO_TMP`. The PCS will then check the status of the scripts and decide if a script needs to be run. Since the PCS just started, the scripts in the PCF file will show up as not being started or finished:

```
U:/WORK/PCS.COM started at : Wed Sep  4 19:45:15 MDT 1996
 PID STATUS IN PRCNXT
PID 001 T_SCRIPT NO_OPTS  STARTED = F ENDED = F
PID 002 T_SCRIPT NO_OPTS  STARTED = F ENDED = F
 WILL RUN PID  1
```

The status of which scripts are being run is listed. The displayed process list includes the PID of each script, the option directory for the script, whether or not the script as been started and whether or not the script has finished. There are three possible flags for `STARTED`: `F` (false) means that the script has not been started, `P` (partial) means that the script has been partially started (only for parallel scripts), and `T` (true) means that the script has been started. For `ENDED`, there are only two possibilities: `F` if the script has not finished and `T` if the script has finished. The above output tells us that no script has been started (or has finished). After the BPE starts the first script, we will see the lines change to:

```
PID 001 T_SCRIPT NO_OPTS  STARTED = T ENDED = F <
PID 002 T_SCRIPT NO_OPTS  STARTED = F ENDED = F
```

Note that any script that is running (`STARTED = T ENDED = F`) will have a "<" symbol to the right to allow the user to quickly see which script is running.

Since the second script in the PCF may only be started after the first one has finished, there is nothing to do for the PCS but to wait for it to finish. Thus the PCS sleeps for 30 seconds (default value). When it wakes up, it will again check the status of the scripts.

After the PCS has finished, we can take a look at the log files which will be in `$T/AUTO_TMP`:

```
sakic[jjohnson]:V40-$ cd $T/AUTO_TMP
sakic[jjohnson]:V40-$ ls -lt
total 6
-rw-r--r--   1 jjohnson users        847 Sep  4 19:46 PCFLOCK.NUM
-rw-r--r--   1 jjohnson users       1206 Sep  4 19:46 PCFLOCK.CPU
-rw-r--r--   1 jjohnson users        478 Sep  4 19:46 PCFLOCK.ATO
-rw-r--r--   1 jjohnson users        128 Sep  4 19:46 00961110.L02
-rw-r--r--   1 jjohnson users        128 Sep  4 19:45 00961110.L01
lrwxrwxrwx   1 jjohnson users         24 Aug 21 10:24 P: -> /home/jjo...
```

The files that start with `PCFLOCK` are temporary files used by the BPE. The two files that have the extension `.L??` are the log files and contain all the output generated by the script that would normally go to the screen. Since the `T_SCRIPT` echos information, we will find what the script echoed in these files. This is very useful for the debugging of scripts. If the script itself were to have an error, then this error would appear in the log file and not in the protocol output file (see below).

Here is the contents of `00961110.L02`:

```
sakic[jjohnson]:V40-$ cat 00961110.L02
HEADER: copying PAN files from /home/jjohnson/GPSUSER/OPT/NO_OPTS
T_SCRIPT: Starting
SESSION IS 1110
V_O IS HI
T_SCRIPT: Ending
```

If we go to the campaign `OUT` directory, we will see the protocol output files:

```
sakic[jjohnson]:V40-$ cd $P/TST_CAMP/OUT
sakic[jjohnson]:V40-$ ls -lt
total 2
-rw-r--r--   1 jjohnson users         584 Sep  4 19:46 00961110.002
-rw-r--r--   1 jjohnson users         584 Sep  4 19:45 00961110.001
```

These files contain information that is written by the basic BPE scripts:

```
sakic[jjohnson]:V40-$ cat 00961110.002
PROTOCOL FILE FOR BPE SCRIPT
----------------------------

SCRIPT NAME         : T_SCRIPT
YEAR                : 96
SESSION             : 1110
CAMPAIGN            : TST_CAMP
CAMPAIGN PATH       : P:/
OPTION DIRECTORY    : NO_OPTS
PROCESS ID          : 002
SUB PROCESS ID      : 002
CPU                 : CPU1
PATH TO WORK AREA   : /home/jjohnson/GPSTEMP/AUTO0001

 DATE      TIME     STA PROGRAM  MESSAGE
----------------------------------------
 04-SEP-96 19:46:00 MSG T_SCRIPT PROCESS STARTED
 04-SEP-96 19:46:03 MSG T_SCRIPT PROCESS ENDED
----------------------------------------
sakic[jjohnson]:V40-$
```

From the time tags of the start and end messages, we see that the script only took 3 seconds. When the last line of the protocol file is written (the `PROCESS ENDED` message), the PCS will know that the script has finished.

The `PATH TO WORK AREA` tells us where the BPE has set up a temporary work area to run BPE programs. Normally the Bernese menu system runs in the `$U/WORK` area and expects to find program input panels and other input files in directories under `$U`. Since the BPE can run more than one script and thus more than one Bernese program at a time, each script must have its own area for input files and panels so that programs running at the same time do not overwrite panels that are in use. The first thing the BPE does when it starts a script is to look for a free temporary directory with the name `$T/AUTOnnnn` where `nnnn` is a number. First it starts with `0001` and increments this number until it finds a free area. If it cannot find one, then it creates a new one. For the simple PCF file `T_PCF.PCF`, there is only one script running at a time, so there will only be one `AUTOnnnn` directory required, since the second run of `T_SCRIPT` can safely use the same area. From the protocol file we can see that the BPE used the directory `$T/AUTO0001`.

If we go to this directory (`$T/AUTO0001/PAN`), we can see the panel files that were used. We reproduce the panel `DAT151__.PAN` here:

```
1.5.1      PROCESSING: FILENAME PARAMETERS FOR AUTOMATIC PROCESSING

    Station Parameters:
      $STATION1 >                 <      $STATION2 >                 <
      ($i will be set to 2-char station abbrev, $STi to 4-char abbrev)

    4-character Parameters:
      $CD1      >               <        $CD2      >               <
      $CD3      >               <        $CD4      >               <

    3-character Parameters:
      $D1       > 111        <           $D2       >            <
      $D3       >            <           $D4       >            <

    2-character Parameters:
      $M   > 04       <        $O   > HI       <        $T   > 20       <
      $U   >          <        $V   >          <        $W   >          <
      $X   >          <        $Y   > 96       <        $Z   >          <

    6-character Session Parameters (+ - allowed):
      $JRSS1    > 961110              <  $JRSS2    > 961110              <
      $JRSS3    >                     <  $JRSS4    >                     <
      $JRS+1    > 961120              <  $JRS-1    > 961100              <
      $JD+2     > 96113               <  $JD-2     > 96109               <

      $GDY1     > 08496               <  $GDY2     > 08496               <
      $GD+1     > 08500               <  $GD-1     > 08495               <
      $GD+2     > 08501               <  $GD-2     > 08494               <

    4-character Session Parameters (+ - allowed):
      $SS1      > 1110          <        $SS2      > 1110          <
      $SS3      >               <        $SS4      >               <
      $S+1      > 1120          <        $S-1      > 1100          <
      $S+2      > 1130          <        $S-2      > 1090          <

      $GW1      > 0849          <        $GW2      > 0849          <
      $G+1      > 0850          <        $G-1      > 0849          <
      $G+2      > 0850          <        $G-2      > 0849          <
```

Notice that the BPE automatically filled in many of the `DAT151__.PAN` $-variables and the `$O` variable. All these variables may be used in the option panels.

21.6.2 Running Bernese Programs in BPE Scripts

In order to do something useful, scripts that are run by the BPE must be able to run Bernese programs. To do this, the user must run a special script `$X/SCRIPT/RUN_PGMS` and set the variable `PGMNAM`, where `PGMNAM` is the name of the Bernese GPS program to be run. For example, the following two lines would execute the Bernese program SNGDIF:

```
PGMNAM="SNGDIF"
. $X/SCRIPT/RUN_PGMS
```

Note that the `RUN_PGMS` script must be “sourced” (command “.”) and not executed as a separate shell.

For the VMS version the corresponding lines would be:

```
$ PGMNAM == "SNGDIF"
$ @X:[SCRIPT]RUN_PGMS
```

All the panels that are required by the program to be executed must be contained in the option directory that is specified in the PCF along with the script that is running the program. For example, if you want to run the program SNGDIF, you could have an option directory named:

```
$U/OPT/SDIF_NRM
```

(`SDIF_NRM` could stand for SNGDIF with NoRMal settings). In this directory all the panels that are used by SNGDIF would need to be present and filled out with all options that are needed.

Then in the PCF we would have:

```
:
PID SCRIPT   OPT_DIR  CAMPAIGN CPU      P WAIT FOR....
3** 8******* 8******* 8******* 8******* 1 3** 3** 3** 3** 3** 3** 3** 3** 3**
:
007 SNGDIF   SDIF_NRM          any      1 006
:
```

The names of the Bernese GPS programs may be found by looking at the settings in Menu 0.2, e.g. for the processing programs in Menu 0.2.4:

```
 0.2-4                  DEFAULTS: PROCESSING PROGRAM NAMES

    Preprocessing:
      CODE PREPROCESSING          > CODCHK <
      SINGLE POINT POSITIONING    > CODSPP <
      SINGLE DIFFERENCE FILES     > SNGDIF <
      OLD PHASE PREPROCESSING     > OBSTS1 <
      NEW PHASE PREPROCESSING     > MAUPRP <

    Processing:
      PARAMETER ESTIMATION        > GPSEST <
      IONOSPHERE ESTIMATION       > IONEST <
      ADD NORMAL EQUATIONS        > ADDNEQ <

    Path to the Programs         >  XG:/                     <
```

For most of the Bernese programs, the `PGMNAM` variable is set to the name of the GPS program. The name of the corresponding menu program – used to prepare the run of the GPS program – is then obtained by adding "`_P`" to the program name (e.g. CODSPP $\longrightarrow$ CODSPP_P). There are several programs, however, where the name of the menu program is different from the name of the GPS program after adding "`_P`":

```
PGMNAM="CCRINEXO" - Runs menu program CCRNXO_P and GPS program CCRINEXO
PGMNAM="CCRINEXN" - Runs menu program CCRNXN_P and GPS program CCRINEXN
PGMNAM="CCPREORB" - Runs menu program CCPREN_P and GPS program CCPREORB
PGMNAM="PRETAB"   - Runs menu program BRDTAB_P and GPS program PRETAB
PGMNAM="SATMRK"   - Runs menu program SERVOBS
```

21.6.3 The Panel Files

Before running the BPE all the options in the panel directories used by the BPE scripts have to be set correctly. For example if we want to run the program SNGDIF, we have to fill out all panels that are used by SNGDIF. The SNGDIF program is started using Menu 4.3 which means that all panels required by the SNGDIF program start with `DAT43xxx.PAN`. (Use the command “`=S`” after starting the Bernese menu system with “`G`” and enter the menu program name (e.g. SNGDIF_P) to obtain the menu and submenu options for a specific program).

If we want to use the option directory `SDIF_NRM` to hold options for SNGDIF, then we have to make a directory `$U/OPT/SDIF_NRM` and put the basic panels for the menu system (e.g. `DAT0*.PAN`, `DAT151__.PAN`, . . .) and the `DAT43*` panels in this directory. We can save some work by letting the Bernese menu system do this for us. If we have a PCF written that uses scripts and option directories corresponding to these scripts, then we may use Menu 6.1 to create the option directories and automatically copy all needed panels there. Menu 6.1 may also be used then to modify the options in these panels.

For example, add these lines to the simple script `T_SCRIPT` (in the `T_SCRIPT BODY` section):

```
PGMNAM="SNGDIF"
. $X/SCRIPT/RUN_PGMS
```

and then change `T_PCF.PCF` to be:

```
#
# Process Control File
# Comments:
# Any line that starts with a # is a comment line
#
#
PID SCRIPT   OPT_DIR  CAMPAIGN CPU      P WAIT
*** ******** ******** ******** ******** * *** *** *** *** *** *** *** *** ***
001 T_SCRIPT SDIF_NRM          any      1
#
# additional parameters required for PID's
#
PID USER         PASSWORD PARAM1   PARAM2   PARAM3   PARAM4   PARAM5
*** ************ ******** ******** ******** ******** ******** ********
#
# PCF Variables
#
VARIABLE DESCRIPTION                              DEFAULT          LENGTH
8******* 40************************************** 16************** 2*
V_0      TEST SETTING V_0                         HI                2
```

Now we have a PCF that uses a script that actually calls a Bernese program. Note that it is not very useful, but it will demonstrate how to setup option panels for BPE scripts.

Assuming that `$U/OPT/SDIF_NRM` does not exist yet, we can create a new option directory using Menu 6.1:

```
 6.1                         BPE: SELECT PCF FILE

 Input Files:
   PROCESS CONTROL FILE  > T_PCF    <    (blank for selection list)

 Input Option:                    (NEW, FIX, UPDATE or COPY existing Options)
   IOPT                  > NEW      <    (NEW, FIX, UPDATE, or COPY)
```

Then all scripts in the PCF will be searched to see if they run any Bernese programs. In our simple case we get:

```
 6.1-1                     BPE PROGRAM NAME SELECTION

     S      Program       Old Directory            New Directory

   >   <  > SNGDIF   <  > U:/OPT/BPE_PAN/      <  > U:/OPT/SDIF_NRM/     <
   >   <  >          <  >                      <  >                      <
```

This is telling us that the program SNGDIF is being run using the option directory `$U/OPT/SDIF_NRM`. The directory `$U/OPT/BPE_PAN` (under Old Directory) is the directory that will be used to get the first version of the panels from. In the normal software distribution the option directory `$U/OPT/BPE_PAN` does not exist. The user may create it and copy all the panels in his `$U/PAN` directory to the directory `BPE_PAN` or he may change the directory name given in the Panel 6.1–1 under "Old Directory" to e.g. `$U/PAN`. If we select SNGDIF (by putting an "`S`" in the first data field) then the directory `$U/OPT/SDIF_NRM` will be created for us and panels moved into that directory. After this we are prompted with the list of panels that go with SNGDIF:

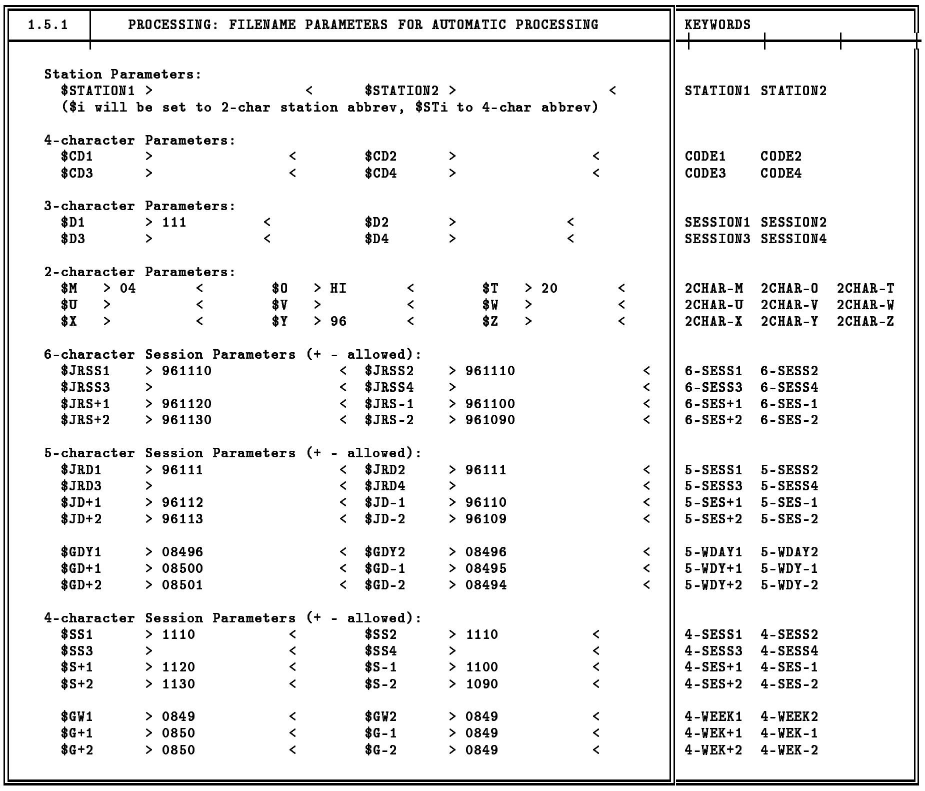
```
 U:/OPT/SDIF_NRM/                    SELECT PANELS TO EDIT FOR : SNGDIF

 DAT43___.PAN   4-SEP-96  PROCESSING: FORM SINGLE DIFF.
 DAT431__.PAN   4-SEP-96  FORM SINGLE DIFFERENCES: INPUT
```

We can edit any of these panels by selecting them (placing an "`S`" in the most left column). If we select both, first the panel `DAT43___.PAN`, then the panel `DAT431__.PAN` will be displayed to us for editing.

21.6.4 Panel Variables

In order to write generic scripts that will, for example, work with any given session, there must be some mechanism to automatically supply Bernese panels with information that will change. To do this, the BPE uses panel variables.

One of the first scripts that is executed in a BPE script is `SET_SESS`. This script will make entries into the Panel 1.5.1 and will set variables for the session being processed. Shown below is the panel with values that are automatically filled in by the BPE (for session 1110, day of year 111, year 1996 in this example):



```
 1.5.1        PROCESSING: FILENAME PARAMETERS FOR AUTOMATIC PROCESSING       KEYWORDS

Station Parameters:
  $STATION1 >                  <      $STATION2 >                  <       STATION1 STATION2
  ($i will be set to 2-char station abbrev, $STi to 4-char abbrev)

4-character Parameters:
  $CD1      >                <        $CD2     >                 <         CODE1    CODE2
  $CD3      >                <        $CD4     >                 <         CODE3    CODE4

3-character Parameters:
  $D1       > 111         <           $D2      >              <            SESSION1 SESSION2
  $D3       >             <           $D4      >              <            SESSION3 SESSION4

2-character Parameters:
  $M   > 04       <        $O   > HI      <       $T   > 20      <       2CHAR-M  2CHAR-O  2CHAR-T
  $U   >          <        $V   >         <       $W   >         <       2CHAR-U  2CHAR-V  2CHAR-W
  $X   >          <        $Y   > 96      <       $Z   >         <       2CHAR-X  2CHAR-Y  2CHAR-Z

6-character Session Parameters (+ - allowed):
  $JRSS1    > 961110              <  $JRSS2   > 961110              <   6-SESS1  6-SESS2
  $JRSS3    >                     <  $JRSS4   >                     <   6-SESS3  6-SESS4
  $JRS+1    > 961120              <  $JRS-1   > 961100              <   6-SES+1  6-SES-1
  $JRS+2    > 961130              <  $JRS-2   > 961090              <   6-SES+2  6-SES-2

5-character Session Parameters (+ - allowed):
  $JRD1     > 96111               <  $JRD2    > 96111               <   5-SESS1  5-SESS2
  $JRD3     >                     <  $JRD4    >                     <   5-SESS3  5-SESS4
  $JD+1     > 96112               <  $JD-1    > 96110               <   5-SES+1  5-SES-1
  $JD+2     > 96113               <  $JD-2    > 96109               <   5-SES+2  5-SES-2

  $GDY1     > 08496               <  $GDY2    > 08496               <   5-WDAY1  5-WDAY2
  $GD+1     > 08500               <  $GD-1    > 08495               <   5-WDY+1  5-WDY-1
  $GD+2     > 08501               <  $GD-2    > 08494               <   5-WDY+2  5-WDY-2

4-character Session Parameters (+ - allowed):
  $SS1      > 1110           <        $SS2     > 1110            <         4-SESS1  4-SESS2
  $SS3      >                <        $SS4     >                 <         4-SESS3  4-SESS4
  $S+1      > 1120           <        $S-1     > 1100            <         4-SES+1  4-SES-1
  $S+2      > 1130           <        $S-2     > 1090            <         4-SES+2  4-SES-2

  $GW1      > 0849           <        $GW2     > 0849            <         4-WEEK1  4-WEEK2
  $G+1      > 0850           <        $G-1     > 0849            <         4-WEK+1  4-WEK-1
  $G+2      > 0850           <        $G-2     > 0849            <         4-WEK+2  4-WEK-2
```

The `SET_SESS` script does not set *all* the values in `DAT151__.PAN`, but only those that are likely to be used by most of the scripts. This variable panel `DAT151__.PAN` is simply a means of setting some generic values to be passed into other Bernese panels. The names of the variables are shown in the main body of the panel and to the right (after column 80) the keyword for each variable is given. The values of the panel variables are shown in-between the angle brackets. In the above panel, only the PCF variable `V_O` (`$O`) has been set.

To use $-variables within Bernese panels, the variable name is used, `$CD1` for example. The keywords are used together with the script named `PUTKEYWE` to set the values in panel `DAT151__.PAN`. For example, the `$CD1` variable is set using the keyword `CODE1`. The script `PUTKEYWE` is covered in a following section.

The last few sets of variables have the annotation "`(+ - allowed)`". This option allows to specify a range of sessions. For example, it may be desirable to select files from more than just one session in a specific program run. A gliding comparison of coordinate files from e.g. the last 7 sessions using the program COMPAR is a possible example. In this case you may use e.g. "`1110 +0 -6`" for the value of `$SS1` etc. When the ± option is used the BPE will correctly adjust the day of the year and the year, accounting for transitions between the current and next or previous year. Remember that the format of a 4-digit session is to have the day of the year as the first three digits followed by the 1-character session identification within the day. To make use of the ± option you have to specify the variables `V_PLUS` and `V_MINUS` in the PCF file:

```
...
VARIABLE DESCRIPTION                              DEFAULT          LENGTH
8******* 40************************************** 16************** 2*
V_O      EUROCLUS ORBIT FILE NAME                 HI                2
V_PLUS   PLUS  DAYS                               +0                2
V_MINUS  MINUS DAYS                               -0                2
```

When starting the BPE using Menu 6.4.1 these variables will also be displayed to you (in Panel 6.4.1–1.3) and you may change their values. The variables `$JRSS1`, `$JRD1`, `$GDY1`, `$SS1`, and `$GW1` will then contain the ± values (e.g. "`961110 +1 -2` for `$JRSS1`, if `V_PLUS=1`, `V_MINUS=2`), whereas the variables `$JRSS2`, `$JRD2`, . . . will NOT include the ± parameters (e.g. "`961110`" only for `$JRSS2`).

If any of the special variables are set (`V_O`, `V_PLUS`, etc...) then the `SET_SESS` script will also set the appropriate variables in `DAT151__.PAN`.

21.6.5 The Clean Script

In the `$X/EXE` directory there is a simple script named `CLEAN`. This script can be used to delete any left-over temporary files that the PCS may have created on previous runs and will make sure that the `PCFCTL.CPU` is initialized to show that no jobs are running. The clean script can only be run if no PCS's are running. If it is run while a PCS is running, then the PCS will crash since temporary files that it is using will be deleted. To run the clean script, the user on a UNIX platform can use the Bourne shell dot command:

```
sakic[jjohnson]:V36-$ . CLEAN
```

On a VMS systems the corresponding command is:

```
@X:[EXE]CLEAN
```

21.6.6 Starting PCS From the Shell or System Prompt

As seen previously, the user can start the BPE using Menu 6.4.1. In some cases it may be desirable to start the BPE from a shell or shell script (UNIX) or from the system prompt (VMS). One example would be if you would like to run a BPE job every day at a given time. You could then set up a cron job that starts a script. Here is an example of starting the BPE from a shell:

First we change directory to the `$U/WORK` area, and then we start the PCS specifying the PCF file (here `GET_ORB.PCF`), the desired campaign (`NEW_CAMP`), the year (`95`), the session (`0710`), and possibly other parameters.

UNIX Version:

```
erde[jjohnson]:V36-$ cd $U/WORK
erde[jjohnson]:V36-$ PCS get_orb camp NEW_CAMP yr 95 ses 0710
```

VMS Version:

```
$ SET DEFAULT U:[WORK]
$ @X:[EXE]PCS.COM GET_ORB CAMP NEW_CAMP YEAR 95 SES 0101
```

Please be aware of the fact that PCF variables (variables defined in the third part of the PCF file) will not automatically be set when using the command above. You have to explicitly add these parameters when starting the PCS (e.g. by adding “`V_PLUS +2 V_MINUS -1`” to the PCS command above). On VMS the maximum number of parameters to be passed to a DCL command file is limited to 8. To overcome this limitation you may specify a file as one of the parameters. The file may then contain as many additional parameters as necessary. This might look as follows (this procedure is also used by the menu system on both, UNIX and VMS systems):

```
$ @X:[EXE]PCS.COM GET_ORB YEAR 95 SES 0101 + U:[INP]PCS.INP
```

The “+” indicates that the name of a file will follow containing additional input parameters for the PCS. The file `U:[INP]PCS.INP` in this case might have the following content:

```
CAMP
NEW_CAMP
V_O
HI
V_PLUS
+2
V_MINUS
-1
```

21.7 BPE Menu Items

We can access the BPE menu items through Menu 6:

```
  6      |                 BPE:  OPTION MENU
---------+-------------------------------------------------------------
 S:Y C:0 |
---------+-------------------------------------------------------------
         |
  0      |  PANEL UPDATE    : Update Panels for New Release
  1      |  PANEL EDITING   : Prepare Option Panels for BPE
  2      |  PREPARE RINEX   : Prepare RINEX Files for BPE
  3 ..   |  SPECIAL FILES   : Prepare Special Files for BPE
  4 ..   |  BPE PROCESSING  : BPE Processing (Session or Campaign)
  5 ..   |  BPE SERVICES    : BPE Service Programs
         |

            Enter Selection :
```

The menu items under this option allow the user to deal with panel files, PCF files, and to start BPE runs.

21.7.1 PANEL UPDATE

This option (Menu 6.0)can be used to update panels in one or more option directories. This is useful when the panels for the Bernese programs are changed and new keywords (input fields) are added to them (e.g. with a new release of the software or due to changes by the user). In this case, new complete panels will be created that retain the settings for pre-existing parameters with new parameters filled in from a master set of panels. This menu item is not thought to be used to change panel options for the BPE. It should only be used if menu programs and input panels have changed. To modify BPE options you should use Menu 6.1. The first menu that appears under the `PANEL UPDATE` option is shown below:

```
 6.0  |                     BPE: PANEL UPDATING

 Master Panels:
   PANEL DIRECTORY  > X:/PAN/              <
   MASTER PANEL     >          <               (blank for selection list)

 Panels to be Updated:
   PANEL DIRECTORY  > U:/PAN/              <   (NO or full directory name)
     or
   DIRECTORY LIST   > NO       <               (NO, blank for sel. list)

 Update Options:
   UPDATE/COPY      > UPDATE   <        (UPDATE or COPY)
   EXISTING/ALL     > EXISTING <        (EXISTING panels only, ALL panels)
```

Due to the fact that with the BPE a lot of panel option directories are created and used this tool is essential to maintain the hundreds of panels. Therefore you also have the possibility to update an entire list of panel directories in one run. An example of such a list of directories may be found in `$X/INX` with the name `EXAMPLE.UPD`. A detailed description of the options above may be found in the help panel `$X/HLP/DAT60___.HLP`.

21.7.2 PANEL EDITING

Below is the panel for `PANEL EDITING` (Menu 6.1):

```
 6.1  |                     BPE: SELECT PCF FILE

 Input Files:
   PROCESS CONTROL FILE  >          <    (blank for selection list)

 Input Option:                    (NEW, FIX, UPDATE or COPY existing Options)
   IOPT                  > NEW      <    (NEW, FIX, UPDATE, or COPY)
```

This menu item is the main tool to handle the BPE option panels. It allows the user to create new panel directories and to modify the options set in the individual panels in a user-friendly way. In addition to the name of the PCF file for which new panels have to be created or for which options in existing panels have to be modified, the user selects one of the following options: `NEW`, `FIX`, `UPDATE`, or `COPY`. The `IOPT` options will be explained below. Once a PCF file has been selected, all the scripts specified in the PCF will be searched for Bernese GPS programs that are to be run and the program names along with the option directories that will be used with each of the programs will be extracted. Below is an example of what the display might look like after a PCF file has been selected:

```
 6.1-1  |                BPE PROGRAM NAME SELECTION

    S      Program       Old Directory            New Directory

   >   <  > FTPRNX   <  > U:/OPT/FTP_OPTS/    <  > U:/OPT/FTP_OPTS/     <
   >   <  > FTPORB   <  > U:/OPT/FTP_OPTS/    <  > U:/OPT/FTP_OPTS/     <
   >   <  > CCRNXN   <  > U:/OPT/CCRX__ON/    <  > U:/OPT/CCRX__ON/     <
   >   <  > CCRNXO   <  > U:/OPT/CCRX__ON/    <  > U:/OPT/CCRX__ON/     <
   >   <  > CRDRNX   <  > U:/OPT/CRDRNX_1/    <  > U:/OPT/CRDRNX_1/     <
   >   <  > FTPRNX   <  > U:/OPT/GET_MISS/    <  > U:/OPT/GET_MISS/     <

U:/PAN/DAT611__.PAN                                               REPLACE
```

Depending on how large the screen is, the user may see more lines at a time than shown above. If not all lines are fitting on the screen, the missing lines can be viewed my moving the cursor to the bottom line and then pressing the down arrow key. The lines will then scroll up the screen until the end of the list is reached. The user places an “`S`” in the first column for each line that is desired. After all lines have been selected, the user types the continuation character (see Section 3.3.2).

The first column is used to select panels to edit. The second column shows the name of the Bernese program. The third column shows a source option directory and the last column shows a destination directory. In the above example, the old and new directories are set to be the same directories. The use of the old and new directory will be explained in the following sections.

When a program is selected, all panels that are related to the program will be presented to the user. For example, if the `CCRNXN` line were selected, the user would be presented with the following:

```
U:/OPT/CCRX__ON/                 |      SELECT PANELS TO EDIT FOR : CCRNXN

DAT2562_.PAN  29-MAR-95  Concatenate RINEX Navigation Files  (Main Data Panel)
DAT25621.PAN  29-MAR-95  Concatenate RINEX Navigation Files: Input 1
```

All the panels that pertain to the CCRNXN program are displayed. The CCRNXN program corresponds to Menu 2.5.6.2. The menu program will offer the user all panels that start with `DAT2562`, since all of these panels will contain options for the program CCRNXN. In this case there are only two panels offered. For more complicated programs (e.g GPSEST), more panel names will be displayed.

You may now select each panel you wish to edit by placing an “`S`” in the most left column. All selected panels will then be offered to the user for editing, one after another. Once all panels have been edited, the complete list of panels will be shown again. At this point the user can either select more panels to edit, or exit by typing a “`Q`” in the first column.

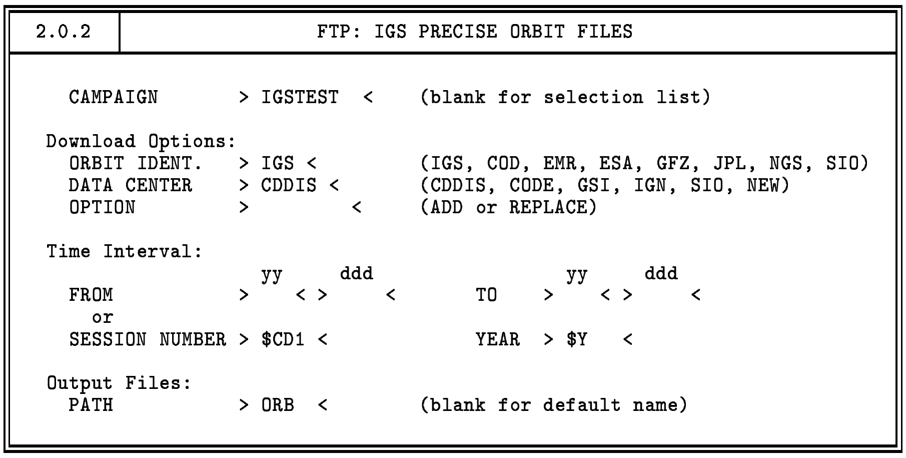
Panel Editing FIX Option

This option is used when the user wants to edit the options in existing panels. For this option, the Old Directory column will be the same as the New Directory column. If the directory name in the Old Directory column is changed, then the panels there will be copied to the New Directory before the panels are edited. Normally, however, the Old Directory column is not used with the `FIX` option.

If the options in one of the *default panels* have to be changed (e.g. the pole file in the Panel 0.3.1) the user has to edit the corresponding panel "manually" and cannot make use of Menu 6.1 (default panels are not displayed there).

Panel Editing UPDATE Option

When this option is selected, the panels in the New Directory will be updated with the panels in the Old Directory before panels are edited. The Old Directory column will be filled in with the value `U:/OPT/BPE_PAN`. Updating a panel means that any new fields in the source panel (Old Directory) are added to the target panel, but any selections existing in the target panel are left unchanged. For example, suppose a new DATA CENTER was added for the IGS precise orbit panel `DAT202__.PAN`:



```
2.0.2                    FTP: IGS PRECISE ORBIT FILES

   CAMPAIGN       > IGSTEST  <    (blank for selection list)

 Download Options:
   ORBIT IDENT.   > IGS <         (IGS, COD, EMR, ESA, GFZ, JPL, NGS, SIO)
   DATA CENTER    > CDDIS <       (CDDIS, CODE, GSI, IGN, SIO, NEW)
   OPTION         >        <      (ADD or REPLACE)

 Time Interval:
                    yy     ddd                 yy     ddd
   FROM           >    < >     <       TO    >    < >     <
     or
   SESSION NUMBER > $CD1 <             YEAR  > $Y   <

 Output Files:
   PATH           > ORB  <        (blank for default name)
```

Then the panel in the New Directory would be updated to show `NEW` as one of the options for the `DATA CENTER` field, but the actual value in this field would be left unchanged. If a completely new field is added to the panel, say `OPTION2` under `OPTION`, then this new line would be added to the panel in New Directory, and what ever selection is in the source panel would be copied over since there would be now pre-existing option in the target panel. This update of panels may be performed in a more general way using Menu 6.0.

Panel Editing COPY Option

This option will just copy the panels in the Old Directory over to New Directory. The Old Directory column is filled in with `U:/OPT/BPE_PAN` initially, but may be edited. After panels are copied, they are not offered for editing.

21.7.3 PREPARE RINEX

The `PREPARE RINEX` option is presently *only working on UNIX systems.* It allows the user to quickly look at RINEX observation files to check such parameters as antenna heights, receiver and antenna names, etc. The first panel that appears after selecting Menu 6.2 is:

```
 6.2                         BPE RINEX HEADERS: CHECK

   CAMPAIGN                  >NEW_CAMP  <   (blank for selection list)

 Input Files
   RINEX FILE                >          <   (blank: sel. list)
   A PRIORI COORDINATES      > NO       <   (blank: sel. list, NO: not used)
   ECCENTRICITY FILE         > NO       <   (blank: sel. list, NO: not used)

 Translation Tables
   STATION NAMES             > BPECAMP1 <   (blank: sel. list, NO: not used)
   RCVR / ANTENNA            > BPECAMP1 <   (blank: sel. list, NO: not used)
   ANTENNA HEIGHTS           > BPECAMP1 <   (blank: sel. list, NO: not used)

 Extension of Rinex Input Files (Wildcards allowed):
   EXTENSION                 > *O* <

 Summary File
   SUMMARY FILE              > NO       <   (NO: default name )
```

The parameters for this panel are described in the corresponding help panel.

After the above panel has been filled out, the user will be presented with a selection list of all RINEX files that were found given the specified parameters. For each RINEX file selected, header information will be extracted and presented to the user in a panel such as the following:

```
 6.2-1                      BPE RINEX HEADER CHECKING
                (Edit Old OR Enter New To Change Entries and Select)
                  (If New Filename Changes are applied to New File)
             (If New Stationname is shown Station is in Translation Table !!)

   S   Old Filename  Old Station Name  Old Ant Height       Receiver
   S   New Filename  New Station Name  New Ant Height       Antenna
                                             [m]

 > S <> FORT0900 <> FORTALEZA           <>  0.6430 <> ROGUE SNR-8000      <
 >   <>          <>                     <>         <> DORNE MARGOLIN T    <
 > S <> NLIB0900 <> NLIB                <>  0.1630 <> ROGUE SNR-8000      <
 >   <>          <>                     <>         <> DORNE MARGOLIN T    <
 > S <> PIE10900 <> PIETOWN             <>  0.1630 <> ROGUE SNR-8000      <
 >   <>          <>                     <>         <> DORNE MARGOLIN T    <
 > S <> ZIMM0900 <> ZIMM                <>  0.0000 <> TRIMBLE 4000SSE     <
 >   <>          <>                     <>         <> 4000ST L1/L2 GEOD   <
 >   <>          <>                     <>         <>                     <
```

The data extracted from the RINEX files will appear in pairs of lines. The first column allows the user to select a pair to be updated. The second column shows the name of the RINEX file, the third the station name in the RINEX file, the fourth column the antenna height, and the fifth column the receiver/antenna pair. If a translation name is found for the station name, then it will appear on the second line of the pair. The same is true for the antenna height. If the receiver/antenna pair is not found in the receiver/antenna translation table, then the character "R" will appear in the first column of the second line of the pair. The user has the option of editing any of the fields of the pair, although values in the second line for parameters two, three, and four will take precedence over changes in the parameters on the first line of the pair. An "S" appearing in the first column of either line of the pair will cause the RINEX file to be updated if either something was changed in the first line or there are new values in the second line. Note that all pairs are selected by default. If you do not want to change a RINEX file, then you must remove the "S" (replacing it with a blank) in the first column for the pair.

21.7.4 SPECIAL FILES

There are two menu options (Menu 6.3.1 and Menu 6.3.2) to prepare special files for the BPE processing. Both options are not fully implemented at present and should not be used. The special files that would be generated by the two options may as well be created just using an ordinary editor. The special file containing a list of stations for which RINEX files should be downloaded by ftp in connection with Menu 2.0.1 (presently only working on UNIX) just contains a list of the 4-character identifiers of the IGS sites you are interested in. An example of such an FTP station file is available in the directory `$X/INX` under the name `EXAMPLE.FTP`. It may be specified in Panel 2.0.1 for the input field "`STATION LIST`".

The second special file of importance for the BPE processing is a list of stations to be fixed or constrained in the programs GPSEST or ADDNEQ. More details on the format of this file type may be found in Chapter 23. To use such a file to fix station coordinates you have to specify "`SPECIAL_FILE`" in the Panel 4.5–1 (GPSEST) or Panel 4.8.1–1 (ADDNEQ) for the station(s) to be fixed and the name of the "fixed station" file in the Panel 4.5–1.5 or Panel 4.8.1–1.5 respectively. To *constrain* the station coordinates (the recommended procedure) you need to specify "`SPECIAL_FILE`" in Panel 4.5–2.4.B or Panel 4.8.1–1.7, respectively (again the name of the "fixed station" file is specified in Panel 4.5–1.5 or Panel 4.8.1–1.5). If the "fixed station" file contains a priori sigma values the corresponding station will be constrained, if no a priori sigmas are specified, the corresponding station will be fixed (see Chapter 23).

A third special file type is used in connection with the estimation of troposphere zenith delays for individual stations (see Chapter 12). A special file may be used to constrain the troposphere estimates of individual sites (see `$X/INX/EXAMPLE.SIG` for an example and Chapter 23) in the program GPSEST. You have to set the option "`SPECIAL_FILE`" in Panel 4.5–2.4.0 and the name of the special troposphere sigma file in Panel 4.5–1.6 to activate this special option.

21.7.5 BPE PROCESSING

To start BPE runs in a comfortable way you may use Menu 6.4.1. It allows you to set up a BPE run to process one or N consecutive sessions. The preparation of a BPE run with the menu system follows exactly the same principles as the preparation of e.g. a GPSEST run. The menu system generates – according to your selection of options – two script files (UNIX: `$U/WORK/PCS.COM` and `$U/WORK/PCS.CTL`, VMS: `U:[WORK]PCS.CO2` and `U:[WORK]PCS.CTL`, see also Chapter 3) and an option input file (`$U/INP/PCS.INP`). Whether the PCS (process control script) is running in the foreground or background and whether you can schedule the BPE run for a specific time depends on the setting of the "`JOB CLASS`" option in Menu 0.1. You may also use the command "`SJ PCS`" to directly start a BPE run that was previously prepared using the menu system (see Chapter 3).

If you intend to run several PCS (BPE runs) at the same time you may want to give the PCS a higher priority compared to the BPE scripts started by the PCS. This will make sure that the process control script gets the necessary CPU to efficiently control the status of the various BPE scripts instead of having to compete with all the other scripts running. The batch queue (priority) used by the PCS is defined in Menu 0.1, option "`JOB CLASS`".

When you want to run more than one PCS at the same time you have to choose a different job identification character in Panel 6.4.1, option `JOB CHARACTER`, for each BPE run. When two or more such PCS are even processing *data of the same session* you also have to specify another `TASK IDENTIFICATION` in Panel 6.4.1–1 for each PCS. The task identification is used to uniquely name the protocol files generated by the various scripts specified in the PCF file.

The menu options to be specified when starting a BPE run are described in detail in the corresponding help panels.

21.7.6 BPE SERVICES

The two menu items Menu 6.5.1 and Menu 6.5.2 are handy tools if you are interested in setting up a BPE run, where new stations (e.g. in a growing permanent network or at the very beginning of the processing of a new campaign) are automatically included in the processing. In this case the most difficult part is, that good a priori values for the *geocentric* station coordinates of the new sites have to be known. Especially for the station(s) you would like to fix or heavily constrain in the final solutions you have to dispose of good geocentric coordinates in the ITRF (see Chapter 11).

Menu 6.5.2 is used to check for a list of RINEX files (typically the RINEX files of one session) whether for each station with RINEX data there exist good a priori coordinates in the coordinate file specified by the user. If there are no sites in the session with good a priori coordinate information, the program CRDRNX (Menu 6.5.2) will generate a list of nearby IGS sites for which RINEX files should be downloaded. These additional RINEX files from IGS sites may then be used to form baselines and to determine a set of coordinates in the ITRF for all sites with coordinates of yet insufficient quality. The list of IGS sites may be used in connection with Menu 2.0.1 to download the RINEX data automatically.

After having transferred all the RINEX files of a session (including the additional IGS sites) into the Bernese observation file format, the program CRDCHK (Menu 6.5.1) may be used to generate a list of baselines suited to improve the a priori coordinates of the new sites. This list of baselines may finally be used as input into a script that does a baseline-wise processing from program SNGDIF up to program GPSEST to obtain good a priori coordinates.

21.8 BPE Scripts

The main function of the BPE is to run scripts. These scripts are written in the script language native to the platform the BPE is running on. For UNIX, BPE scripts are written using the Bourne shell (`sh`), for VMS they are written using the Digital Command Language (DCL). This section will describe how these scripts operate.

21.8.1 Skeleton Script

Although there is no real restriction on what is contained in a script that is run by the BPE, there are a few tasks that must be performed in any case. Even if the script does not run any Bernese program(s), the following skeleton structure should be used. This is because the script must make entries into the protocol file in order to let the BPE know that the script has finished. The basic skeleton for a UNIX script is shown below (comment lines start with the `#` character):

```
# The first task for the script is to execute the header script.  The
# name of the header script will be passed in as the first parameter.
# The header script will set variables and make entries to the protocol
# file.  It will also change to the temporary working directory.
#
# The variables that will be set are:
# YEAR         SESSION         CAMPAIGN
# CAMP_PTH     CAMP_DRV        OPT_DIR
# PID          SUB_PID         PRT_FILE
# SCRIPT       TASKID          PRIORITY
# CPU          STATUS          U
# UOLD
. $1
#
# Now we set up the Bernese menu system to run in non-interactive mode.
. $X/SCRIPT/BEG_MENU
#
# Set standard variables in the DAT151__.PAN.
. $X/SCRIPT/SET_SESS
#
BODY OF THE SCRIPT GOES HERE
#
# The Bernese menu system is taken out of interactive mode.
. $X/SCRIPT/END_MENU
#
# Terminate the script and update the protocol file.
. $X/SCRIPT/DO_TAIL
```

For the VMS system the corresponding skeleton script looks as follows:

```
$! The first task for the script is to execute the header script.  The
$! name of the header script will be passed in as the first parameter.
$! The header script will set variables and make entries to the protocol
$! file.  It will also change to the temporary working directory.
$!
$! The variables that will be set are:
$! YEAR         SESSION         CAMPAIGN
$! CAMP_PTH     CAMP_DRV        OPT_DIR
$! PID          SUB_PID         PRT_FILE
$! SCRIPT       TASKID          PRIORITY
$! CPU          STATUS          U
$! UOLD
$!
$   @'P1'
$!
$! Now we set up the Bernese menu system to run in non-interactive mode.
$   @X:[SCRIPT]BEG_MENU
$!
$! Set standard variables in the DAT151__.PAN.
$   @X:[SCRIPT]SET_SESS
$!
BODY OF THE COMMAND FILE GOES HERE
$!
$! The Bernese menu system is taken out of interactive mode.
$   @X:[SCRIPT]END_MENU
$!
$! Terminate the command file and update the protocol file.
$   @X:[SCRIPT]DO_TAIL
```

21.8.2 The RUN_PGMS Script

To run a Bernese program within a script/command file, the RUN_PGMS script must be used. This script is kept in the $X/SCRIPT area. To use the script, the user sets an environment variable named PGMNAM to the name of the program to be run, and then runs the RUN_PGMS script. Below is an example of running the Bernese program GPSEST:

UNIX Version:

```
PGMNAM="GPSEST"
. $X/SCRIPT/RUN_PGMS
```

VMS Version:

```
$ PGMNAM == "GPSEST"
$ @X:[SCRIPT]RUN_PGMS
```

The user then has to use Menu 6.1 to correctly set any input fields in the panels (in the panel directory to be used by the script) that may be required for program GPSEST (all panels named DAT45*.PAN).

21.8.3 The PUTKEYWE Script

When a script needs to update a value in a panel, it must use the PUTKEYWE script, which is in the $X/SCRIPT area. This script will replace data input fields in Bernese panels. Each panel input field is referenced by a keyword. The keywords are given on the far right side of the panel after column 80. Below is an example of using the PUTKEYWE script:

UNIX Version:

```
pp1=U:/PAN/DAT43___.PAN
pp2=STRATEGY
pp3=MANUAL
. $X/SCRIPT/PUTKEYWE
```

VMS Version:

```
$  pp1 == "U:[PAN]DAT43___.PAN"
$  pp2 == "STRATEGY"
$  pp3 == "MANUAL"
$  @X:[SCRIPT]PUTKEYWE
```

Three environment variables must be set before calling the PUTKEYWE script:

pp1 – The full path and name of the panel to update. On UNIX systems the linked directory name must be used, i.e. `U:`.

pp2 – Name of the keyword to update.

pp3 – Value to be placed in the data input field referenced by the keyword in `pp2`. If there is a space contained in the value, then double or single quotes must surround the value under UNIX.

21.9 BPE Special Programs

There are some utility programs available in the `$XB` directory that are run outside of the Bernese menu system. The use of these programs is described in the following sections.

21.9.1 GPSWIND

This program is used to set the time window for a session in the GPSEST panel `DAT4512_.PAN`. It will read from standard input the campaign name, the year and session (each on a separate line) and will update the panel `U:/PAN/DAT4512_.PAN` with the time window defined in the session definition file (`$P/TST_CAMP/DATPAN/DAT132__.PAN`, see Menu 1.3). Here is the UNIX example:

```
erde[jjohnson]:V40-$ cd $U/WORK
erde[jjohnson]:V40-$ cat U:/PAN/DAT4512_.PAN
```

```
 4.5-1.2 |             PARAMETER ESTIMATION: OBSERVATION WINDOWS

                   START DATE                    END DATE

              yy mm dd     hh mm ss        yy mm dd     hh mm ss

            >          < >          <   >          < >          <
```

```
erde[jjohnson]:V40-$ cat P:/TST_CAMP/DATPAN/DAT132__.PAN
```

```
  1.3-2  |               CAMPAIGNS: SESSION DEFINITION

 SESSION NUMBER          START DATE                    END DATE

     nnnn           yy mm dd     hh mm ss        yy mm dd     hh mm ss

   > ???0 <       >          < > 00 00 00 <   >          < > 23 59 59 <
```

```
erde[jjohnson]:V40-$ echo TST_CAMP > tmp.inp
erde[jjohnson]:V40-$ echo 95 >> tmp.inp
erde[jjohnson]:V40-$ echo 2120 >> tmp.inp
erde[jjohnson]:V40-$ XB:/GPSWIND < tmp.inp
erde[jjohnson]:V40-$ cat U:/PAN/DAT4512_.PAN
```

```
 4.5-1.2 |             PARAMETER ESTIMATION: OBSERVATION WINDOWS

                   START DATE                    END DATE

              yy mm dd     hh mm ss        yy mm dd     hh mm ss

            > 95 07 31 < > 00 00 00 <   > 95 07 31 < > 23 59 59 <
```

21.9.2 PRSLIN

The PRSLIN program in `$XB` may be used to get the root name of a file without the extension. This is useful to remove the path and extension from a file name. For example on a UNIX platform:

```
sakic[jjohnson]:V40-$ cd $U/WORK
sakic[jjohnson]:V40-$ echo P:/TST_CAMP/OUT/00961110.002 > inp
sakic[jjohnson]:V40-$ $XB/PRSLIN < inp > out
sakic[jjohnson]:V40-$ basename=‘cat out‘
sakic[jjohnson]:V40-$ echo $basename
00961110
sakic[jjohnson]:V40-$
```

21.9.3 PRSLINF

This program is similar to `PRSLIN` except that the extension of the file name is also returned. For example:

```
sakic[jjohnson]:V40-$ cd $U/WORK
sakic[jjohnson]:V40-$ echo P:/TST_CAMP/OUT/00961110.002 > inp
sakic[jjohnson]:V40-$ $XB/PRSLINF < inp > out
sakic[jjohnson]:V40-$ basename=‘cat out‘
sakic[jjohnson]:V40-$ echo $basename
00961110.002
sakic[jjohnson]:V40-$
```

21.9.4 QC

QC (Quality Check) is a program written by UNAVCO using Bernese subroutines that can be used to check the quality of RINEX data. Using this program along with the QCLIN program RINEX files can be filtered for obviously bad data. More information on the QC program may be found on UNAVCO's WEB page: `http://www.unavco.ucar.edu` in the `SOFTWARE` section. The QC program requires an input option file named `qc.inp`. It is convenient to place this file in an option directory such as `$U/OPTS/QC_OPTS` so that it will be available in `$U/PAN` when the BPE script is running. Here is an example `qc.inp` file:

```
INPUT PARAMETERS FOR GPS QC PROGRAM:

####
SET#:   1  (DEFAULT SET)
****************************************                     ***

MAXIMUM IONOSPHERIC RATE (L1)           CM/MIN              :300
REPORT DATA GAP GREATER THAN            MIN                 : 10
EXPECTED RMS LEVEL OF P1 MULTIPATH      CM                  :100
MULTIPATH SLIP SIGMA THRESHOLD          SIGMA               :  4
% INCREASE IN MP RMS FOR C/A & A/S                          :100
MOVING AVERAGE LEN FOR MP BIAS COR      POINTS              : 50
MINIMUM SIGNAL TO NOISE L1                                  :  0
MINIMUM SIGNAL TO NOISE L2                                  :  0
CONSIDER ORBIT FILE                     (Y=1,N=0)           :  1
OUTPUT PLOT DECIMATED BY FACTOR OF                          :  1
DETECT CYCLE SLIPS                      (Y=1,N=0)           :  1
ELEVATION CUTOFF FOR QC                 DEG                 : 15
COMPARISON ELEVATION                    DEG                 : 20
PREPARE PLOT FILE FOR MULTIPATH         (Y=1,N=0)           :  0
PREPARE PLOT FILE FOR L1 IONOSPHERE     (Y=1,N=0)           :  0
PREPARE PLOT FILE FOR L1 IONOSPHERE DOT (Y=1,N=0)           :  0
PREPARE PLOT OF SAT AZIMUTH ANGLES      (Y=1,N=0)           :  0
PREPARE PLOT OF SAT ELEVATION ANGLES    (Y=1,N=0)           :  0
PREPARE SLIP OUTPUT FILE                (Y=1,N=0)           :  0
PLOT FLAG FOR CLOCK SWITCH EPOCHS       (Y=1,N=0)           :  1
ECHO RINEX OBS HDR INTO SUM FILE        (Y=1,N=0)           :  1
CONSIDER TIME WINDOW FILE               (Y=1,N=0)           :  0
ORBIT DATA SAMPLE PERIOD                MIN                 : 10
DISPLAY WORKING PERCENTAGE (N=-1,Y=0 OR 6 (WRITE UNIT))     : -1

####
SET#:   2
****************************************                     ***

MAXIMUM IONOSPHERIC RATE (L1)           CM/MIN              :900
REPORT DATA GAP GREATER THAN            MIN                 : 30
EXPECTED RMS LEVEL OF P1 MULTIPATH      CM                  :100
MULTIPATH SLIP SIGMA THRESHOLD          SIGMA               :  5
% INCREASE IN MP RMS FOR C/A & A/S                          :200
MOVING AVERAGE LEN FOR MP BIAS COR      POINTS              : 50
MINIMUM SIGNAL TO NOISE L1                                  :  0
MINIMUM SIGNAL TO NOISE L2                                  :  0
CONSIDER ORBIT FILE                     (Y=1,N=0)           :  1
OUTPUT PLOT DECIMATED BY FACTOR OF                          :  1
DETECT CYCLE SLIPS                      (Y=1,N=0)           :  1
ELEVATION CUTOFF FOR QC                 DEG                 : 15
COMPARISON ELEVATION                    DEG                 : 25
PREPARE PLOT FILE FOR MULTIPATH         (Y=1,N=0)           :  0
PREPARE PLOT FILE FOR L1 IONOSPHERE     (Y=1,N=0)           :  0
PREPARE PLOT FILE FOR L1 IONOSPHERE DOT (Y=1,N=0)           :  0
PREPARE PLOT OF SAT AZIMUTH ANGLES      (Y=1,N=0)           :  0
PREPARE PLOT OF SAT ELEVATION ANGLES    (Y=1,N=0)           :  0
PREPARE SLIP OUTPUT FILE                (Y=1,N=0)           :  0
PLOT FLAG FOR CLOCK SWITCH EPOCHS       (Y=1,N=0)           :  1
ECHO RINEX OBS HDR INTO SUM FILE        (Y=1,N=0)           :  1
CONSIDER TIME WINDOW FILE               (Y=1,N=0)           :  0
ORBIT DATA SAMPLE PERIOD                MIN                 :  5
DISPLAY WORKING PERCENTAGE (N=-1,Y=0 OR 6 (WRITE UNIT))     :  0
```

An example BPE script (named `QC_SESS` and only available on UNIX systems !) that uses the QC program may be obtained on request. This script first runs the program QCWIND (see Section 21.9.6 to generate the file `qc.tim`. This will tell the program QC to window the RINEX data according to the session length. The file `qc.inp` is copied over from the option directory. Then the script looks for a suitable RINEX navigation file that can be used to get satellite rise and set times. The QC program is then run for all RINEX observation files that are in the campaign directory and belong to the session being processed. The script will check to make sure that the QC program has created an output file. If no output file is created, then the script will assume that there is something wrong with the RINEX files that caused QC to crash and will create an output file that will indicate bad data so that the QCLIN program (see next section) will mark the RINEX file as bad.

The output of QC will be files with the same name as the RINEX observation files except that the last letter of the file name will be an `S` instead of an `O` (to indicate a summary file).

21.9.5 QCLIN

The QCLIN program is used along with the program QC to find and mark bad RINEX files. QCLIN reads the summary files generated by QC and decides if the RINEX file should be used. There also exists a UNIX BPE script `CHECKSUM` which uses the QCLIN program, but it is not been included in the official Bernese distribution because it only works on UNIX systems. The script is available on request, however.

It will move any RINEX files that do not pass all the different quality tests to a file with the same name except that the last letter in the filename extension will be a `B` instead of an `O` (where `B` stands for “bad”). This will prevent the RINEX file from being used in subsequent processing steps.

21.9.6 QCWIND

This program is similar to the GPSWIND program: it reads the session panel `$P/CAMP/DATPAN/DAT132__.PAN` to create a time window file. In this case, the time window file is for the QC program.

The QCWIND program reads from the standard input the campaign name, the year and the session and writes to the standard output time window information that can be used by QC. The output of QCWIND is normally redirected to a file named `qc.tim`. Here is an example:

```
sakic[jjohnson]:V40-$ echo EUROCLUS > tmp1
sakic[jjohnson]:V40-$ echo 96 >> tmp1
sakic[jjohnson]:V40-$ echo 2900 >> tmp1
sakic[jjohnson]:V40-$ $XB/QCWIND < tmp1 > qc.tim
sakic[jjohnson]:V40-$ cat qc.tim
TIME WINDOW FILE FOR QC-PROGRAM
==============================

YYYY MM DD HH MM SS.SSS
----------------------
####
1996 10 16 00 00   .000  <-- FROM
1996 10 16 23 59   .000  <--  TO
sakic[jjohnson]:V40-$
```

21.10 BPE Example

Together with the Bernese GPS Software Version 4.0 a BPE example is distributed to allow the user to gain insight into the working of the BPE by looking at this concrete example.

As part of the installation procedure the example PCF file `DOCU40_1.PCF` as well as the corresponding example scripts and option directories are copied from the general `$X` area to the user-specific `$U` area (e.g. `$X/PCF/DOCU40_1.PCF` is copied to `$U/PCF/DOCU40_1.PCF`). After a successful installation (including the BPE part) it should be possible for the user to start the processing of the example campaign `DOCU40_1` (see Chapter 4) with the PCF `DOCU40_1.PCF` right away. The data files for the example campaign `DOCU40_1` are available through anonymous ftp at the AIUB (see Chapter 4). Please read the file `README.TXT`, available in the anonymous ftp directory `[BSWUSER.EXAMPLES]`, carefully before starting with the transfer of the example files. You should create a campaign (Menu 1.1) and the corresponding campaign directory and sub-directories (Menu 1.2) according to the steps explained in Chapter 4 before downloading the BPE example data.

The example PCF file is included here and the most important steps are described below:

```
#
# Procedure Control File (PCF)
# All comment lines start with a #
# Comments: An example PCF to process the example campaign DOCU40_1
#
PID SCRIPT   OPT_DIR  CAMPAIGN CPU      P WAIT FOR....
3** 8******* 8******* 8******* 8******* 1 3** 3** 3** 3** 3** 3** 3** 3** 3**
001 EURO_COP EUROCLUS          any      1
002 PRETAB   EUROCLUS          any      1 001
003 DEFSTD   EUROCLUS          any      1 002
004 RXOBV3   EUROCLUS          any      1 003
005 CODCHK   EUROCLUS          any      1 004
006 CODSPP   EUROCLUS          any      1 005
007 SNGDIF   EUROCLUS          any      1 006
008 MAUPRP   EUROCLUS          any      1 007
009 GPSEDT   EUROCLUS          any      1 008
010 ADDNEQ   EUROCLUS          any      1 009
011 COMPAR   EUROCLUS          any      1 010
012 GPSBAS   EURO_BAS          any      1 011
013 ADDNEQ   EURO_BAS          any      1 012
014 COMPAR   EURO_BAS          any      1 013
015 GPSEST   EURO_IOF          any      1 014
016 GPSEST   EUROFREE          any      1 015
017 COMPAR   EUROFREE          any      1 016
018 GPSQIFAP EURO_QIF          any      1 017
019 GPSQIF_P EURO_QIF          CPUQIF   1 018
020 QIFXTR   EURO_QIF          any      1 019
021 GPSEST   EURO_IOX          any      1 020
022 GPSEST   EURO_FIX          any      1 021
023 COMPAR   EURO_FIX          any      1 022
024 GPSEST   EURO_FXB          any      1 023
025 COMPAR   EURO_FXB          any      1 024
026 EURO_DEL EUROCLUS          any      1 025
```

```
#
# additional parameters required for PID's
#
PID USER         PASSWORD PARAM1   PARAM2   PARAM3   PARAM4   PARAM5   PARAM6
3** 12********** 8******* 8******* 8******* 8******* 8******* 8******* 8*******
018                       $tmp1
019                       PARALLEL $tmp1
#
# That's it
#
VARIABLE DESCRIPTION                              DEFAULT          LENGTH
8******* 40************************************** 16************** 2*
V_O      EUROCLUS ORBIT FILE NAME                 EI                2
V_X      AMBIGUITY FREE  RESULTS                  EG                2
V_Z      AMBIGUITY FIXED RESULTS                  EQ                2
V_U      INPUT SOLUTION TYPE (eg. R3)             R3                2
V_V      TESTING V                                XV                2
V_W      TESTING W                                XW                2
V_PLUS   PLUS  DAYS                               +0                2
V_MINUS  MINUS DAYS                               -0                2
```

The first section list scripts that are to be executed paired with the option directories that are to be used with the scripts. Then there is a section that shows the parameters that are to be passed to the scripts and finally there is a section that defines variables that are used in all scripts called by the PCF. All of the scripts that are shown in the PCF are located in `$U/SCRIPT` and the option directories are located in `$U/OPT`.

Notice that script 019 `GPSQIF_P` specifies a CPU specifically. This will be the host that will be used to run this script and an entry in the `$U/WORK/PCFCTL.CPU` file must exist with this CPU name, for example:

```
# Process control CPU information
CPU      TRUE_NAME                      SPEED    NCPU SF   IDLE MAXJ JOBS
8******* 30**************************** 8******* 4*** 4*** 4*** 4*** 4***
CPU1     sakic                          FAST        1  100  100    2    0
CPUQIF   sakic                          FAST        1  100  100    2    0
```

In the above case, `CPUQIF` and `CPU1` use the same host and specifying `CPUQIF` specifically in the PCF file seems to have no advantage. In fact, `CPUQIF` could be changed to `any` in the PCF file and the entry CPUQIF could be taken out of `PCFCTL.CPU` for the example CPU file shown above. However, there are cases where it may be desirable to specify a specific CPU for a script when more than one host is available for processing and one of the hosts is faster than the others. The `GPSQIF_P` script is very CPU intensive and we may want to make sure that the script is run on a fast CPU.

The processing starts very straight forward with the generation of a standard orbit (PRETAB and ORBGEN (the old DEFSTD)). Then the code and phase pre-processing is performed (RXOBV3, CODCHK, CODSPP, SNGDIF, MAUPRP).

This brings us to `PID 009`. This step is called `GPSEDT`, which means that the GPSEST residuals are checked in order to remove bad phase double difference observations. For this procedure the programs GPSEST, RESRMS, and SERVOBS are used: GPSEST is run in a baseline mode saving *both* the double difference L_3 residuals in residual files and the normal equations in NEQ files. The residual files are checked by program RESRMS followed by the program SERVOBS, which marks the detected outliers in the single difference files according to the edit file generated by RESRMS. The NEQ files are used with program ADDNEQ to get a network solution based on "uncleaned" (no RESRMS) data. This solution is labelled "EG0".

Now we are at PID 012 with the script called GPSBAS. Here the single difference files are processed baseline by baseline saving the NEQ files. These NEQs are combined with ADDNEQ to give a "cleaned" (after RESRMS) network solution. This solution is labelled "EGB".

At PID 015 we use GPSEST to estimate an ionosphere model to be used for the QIF ambiguity fixing in PID 018.

PID 016 is a GPSEST run using all single-difference files in one run with "CORRECT" correlations (see Panel 4.5–2). This solution is labelled "EG".

PID 018/019 are examples for a parallel processing. The ambiguity fixing using the QIF strategy with the ionosphere model from PID 015 is done baseline-wise and more than one baseline may be processed in parallel (depending on the entries in the CPU file $U/WORK/PCFCTL.CPU).

The GPSEST solutions of PID 021 and 022 are similar to those of PID 016, except that the ambiguities are now fixed (PID 021) and the elevation cut-off is set to 15 degrees (PID 022). The PID 021 solution is labelled "EQ", the PID 022 solution "EQB".

The following PCF variables are used for solution identifiers:

V_O ...	gives the first two characters of the standard orbit to be used. The orbit file name should be of the form $O_$Y$D1 The first script (EURO_COP) can be used to rename any input orbit file name to the naming convention given above. Currently the EURO_COP script does practically nothing (only deleting some files).
V_X ...	(EG above) is used to label all ambiguity-free results. The "0" and "B" are hard-coded in the option panels; "0" is used for results before RESRMS has been run; "B" used to identify correlations on the baseline level only.
V_Z ...	(EQ above) is used to label all ambiguity-fixed results. The (hard-coded) "B" is used to identify the 15 degrees elevation cut-off. The EG0 solution is also run with 15 degrees to make sure that the data is clean down to 15 degrees (of course also MAUPRP is run with 15 degrees cut-off). All other solutions (EGB, EG, EQ) use a cut-off angle of 20 degrees.

For those programs, where extraction programs are available, the extraction program is used (CODXTR, MPRXTR, GPSXTR, RESRMS summary) right after the program has been run. E.g. in the MAUPRP script not only MAUPRP but also the extraction program MPRXTR is run. The same is true for the programs CODSPP, GPSEST, and ADDNEQ. All these extraction files are concatenated in the EURO_DEL script to form a "nice" protocol/overview of the processing. Two example protocols are available in the OUT directory of the example campaign DOCU40_1.

The example PCF file thus covers quite a lot of programs and different applications of these programs. It shows an example of the parallel capabilities of the BPE (PIDs 018/019) and also covers the screening of residual files for outliers, the fixing of ambiguities, the use of baseline-wise and correct correlation and different elevation cut-off angles.