ASTRONOMICAL INSTITUTE
UNIVERSITY OF BERNE

Bernese GPS Software Version 4.0

Edited by

Markus Rothacher and Leoš Mervart

Contributors:

G. Beutler, E. Brockmann, S. Fankhauser, W. Gurtner, J. Johnson,
L. Mervart, M. Rothacher, S. Schaer, T. Springer, R. Weber

September 1996

Documentation of the Bernese GPS Software Version 4.0
September 1996

Please send comments on the Bernese GPS Software and this manual as well as requests for copies to:

Dr. Markus Rothacher	Phone : ++ 41 31 631 85 91
Astronomical Institute	Fax : ++ 41 31 631 38 69
University of Berne	E-mail: rothacher@aiub.unibe.ch
Sidlerstrasse 5	
CH-3012 Berne	
Switzerland	

Contents

1. Introduction and Overview

1.1 Main Objectives and General Characteristics

In March 1988 the *Bernese GPS Software Version 3.0*, a new software tool based on its predecessor *Bernese Second Generation GPS Software* was completed. Since that time five major releases could be issued in order to take into account the rapid development in the field *high accuracy applications of the GPS*: Release 3.1 was issued in December 1988, release 3.2 in April 1990, 3.3 in May 1991, 3.4 [*Rothacher et al.*, 1993] in May 1993, and release 3.5 in February 1995.

The *Bernese GPS Software Version 4.0* is based on Version 3. Version 4.0 may be used to process any campaigns which were processed with Version 3 (downwards compatibility). The *new components* are:

- Completely revised orbit part. The full new Bern orbit model [*Beutler et al.*, 1994] is available in Version 4. The new model is a *generalization* of the one used in Software Version 3.
- The new processing program ADDNEQ may be used to combine normal equation systems that were generated with program GPSEST. ADDNEQ does *not* handle observations, only normal equation systems. This leads to a *dramatic improvement* for e.g. multi-campaign analyses or for the management of permanent arrays. New parameters (e.g. station velocities) may be set up in ADDNEQ.
- The ionosphere modeling part was completely revised [*Schaer et al.*, 1995], [*Schaer et al.*, 1996]. Version 4 allows it to produce *regional or global ionosphere models* which may be used to resolve the initial phase ambiguity parameters even on baselines up to 2000 km [*Mervart*, 1995]. The model parameters are established using the *double-difference phase observable.*
- Version 4 contains options to process special *kinematic data* in the post-processing mode.
- The *Bernese Processing Engine*, developed together with Ch. Rocken and J. Johnson from UCAR, is particularly well suited to process data from *permanent GPS arrays* in a completely automatic and a very efficient way.
- The documentation consists of *two components*: There are *help panels* accompanying (almost) every panel of the Version 4 menu system. In addition there is the *off-line documentation* focusing on theory, models, and on commented typical applications.
- Last but not least it should be mentioned that the *processing speed* of Version 4 could be improved by about an order of magnitude in programs GPSEST and ADDNEQ. Version 4 is designed to be used for big networks ($\geq$ 100 receivers), too.

What was said for Version 3 is of course also true for Version 4: **The Bernese GPS Software was developed as a tool for highest accuracy requirements**. Typical users of Version 4 (hopefully) are:

- Scientific users (research and education),
- survey agencies responsible for high-accuracy GPS surveys (e.g. first order networks),
- agencies responsible for big permanent GPS arrays,
- commercial users with complex applications looking for high accuracy, reliability, and high productivity.

The software tool is particularly well suited for

- rapid processing of small-size single and dual frequency surveys (typical example included in documentation),
- permanent network processing,
- ambiguity resolution on long baselines (up to 2000 km using high accuracy orbits),
- ionosphere and troposphere modeling,
- combination of different receiver types (antenna phase center calibrations),
- simulation studies,
- orbit determination and earth rotation parameter estimation,
- generation of so-called free network solutions.

General features of the software are:

- All principal observables recorded by high-accuracy geodetic GPS receivers may be processed (code and phase data on both carriers, see next section).
- Five different linear combination of L1 and L2 may be used (see next section).
- Data from various receiver types may be processed and combined in the same processing steps (this includes the establishment and the use of receiver-type specific antenna phase center variations).
- Single and dual frequency data may be processed in the same estimation step, ionosphere models may be used to *minimize the impact of ionospheric biases on coordinates.*
- The parameter estimation programs GPSEST and ADDNEQ may be used for baseline-, session-, campaign-, multiple-campaign processing. ADDNEQ makes it possible to produce many different complex solutions (e.g. annual coordinate- and ERP-solutions) using (e.g. daily) normal equation systems *without* actually reprocessing observations.
- A big number of different parameter types may be solved for simultaneously.
- Today the observations, broadcast ephemerides, etc. are handed over to the Bernese GPS Software *uniquely* in the RINEX format (the **R**eceiver **IN**dependent **EX**change format, see [*Gurtner*, 1994]. It may thus be said that data from all receiver types for which a RINEX interface exists, may be processed in the Bernese Software. Table 1.1 gives an impression of the receiver types which were successfully “handled” by the software.

Company	Receiver Type
Aerospace	Minimac 2816AT
Texas Instru.	TI-4100
Trimble	4000SX, SL, SLD, STD, STT, SSE, SSI, . . .
Wild/Magnavox	WM-101, WM102
Leica	SR299,SR399,. . .
Ashtech	Ashtech L-XII, LM-XII, P-XII, Z-XII, . . .
Osborne	Rogue, Minirogue, Turborogue

Table 1.1: Receiver Types Used by the Bernese GPS Software 1988–1996

Technically the *Bernese GPS Software Version 4* may be characterized as follow:

- High modularity on the program and subprogram level.
- Computer-system independence:
  - Standard Fortran 77 used (where possible).
  - All data files are accessed through a *translation table* containing the transformation from internal to external file names.

  There are implementations of the *Bernese GPS Software Version 4* on all of the major computer platforms (PC, VMS, UNIX).
- Consequent use is made of the PARAMETER Statement: easily adjustable maximal dimensions of arrays in *main programs* and *highest level subroutines.*
- No numerical constants in code: All values used are in files accessible to the user.
- The use of option input files allows a high degree of automation.
- Version 4 is *menu driven*: the user no longer actually has to edit any option input files. This job is taken over by the menu system. *User-friendliness* is the consequence.
- The menu system is actually written in FORTRAN 77, too. This makes the menu system portable, as well. There are successful implementations on *Personal Computers, Unix-, Linux-, and VMS-platforms.*

1.2 Program Structure and Functional Flow Diagram

The system contains more than 100 different program units (not counting the menu system). They are arranged logically in five different **Parts of the Bernese Software**:

Transfer Part: Generating Bernese Version 4.0 Format from RINEX data format (observations, broadcast information, meteorology).

Orbit Part: Generate source independent orbit representation (standard orbits), update orbits, generate orbits in precise orbit format, compare orbits, etc. .

Processing Part: Code processing (single station), dual frequency code and phase pre-processing, parameter estimation based on GPS observations (program GPSEST) and based on the superposition of normal equation systems (program ADDNEQ).

Simulation Part: Generate simulated GPS observations (code and/or phase, L1 or L1/L2) based on statistical information (rms for observations, biases, cycle slips).

Service Part: Edit/browse binary data files, compare coordinate sets, display residuals, etc. .

The **Bernese Processing Engine** is a tool on top of the five program parts which may be used to set up automated processing procedures which always have to be repeated, like e.g. the processing of data stemming from a permanent array. The processing strategy is *so to speak* set up once and for all *from the RINEX files to the final results* with all the programs necessary. It is even possible to set up a parallel processing on different machines.

The functional flow diagram for the normal use of the Bernese GPS Software Version 4 is given in Figure 1.1.

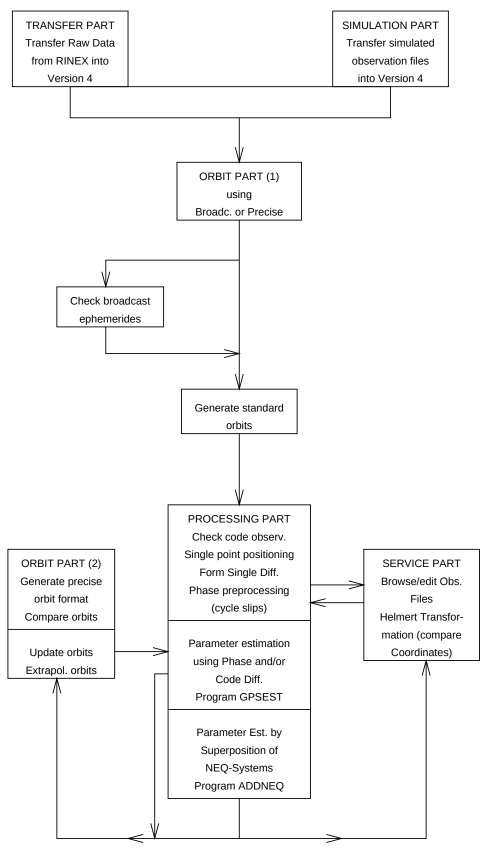


Figure 1.1: Functional Flow Diagram of Normal Processing in Bernese GPS Software Version 4.

The Processing Part may be considered as the **core** of the Bernese GPS Software Version 4. The *parameter estimation programs* GPSEST and ADDNEQ in turn are the *nucleus* of the Processing part. Let us give general characterizations for both parameter estimation programs.

General characterization of parameter estimation program GPSEST

- The *observation equations* for phase and/or code observations are set up, normal equations are built up and inverted, results are written into files, the (not inverted) normal equation system is written into an output file for later use.
- Consequent use is made of the *double difference observable* for code *and* phase observations.
- All results are based a *least squares estimation* where you have the possibility to model correlations correctly.
- There is a fully automatic book-keeping for the initial phase ambiguities. Ambiguities which were resolved once (e.g. in a baseline-mode) may be introduced as known in later program runs (e.g. in a session mode).
- There is no numerical integration in parameter estimation programs. This is done in program ORBGEN of the orbit part.
- Parameters which were estimated once (ambiguities, orbits, coordinates, troposphere, ionosphere) may be used as known a priori values in subsequent program runs.
- In Version 4 the program GPSEST is used to produce results for what we call a *session*, i.e. a time-interval covered by one contiguous satellite arc (very often one day).

General characterization of parameter estimation program ADDNEQ

- The *normal equation systems* as generated *either* by program GPSEST *or* by program ADDNEQ may be overlaid in program ADDNEQ. Result files are written as in program GPSEST.
- All results are again based on a *least squares estimation* where extensive use is made of methods related to *sequential least squares, Kalman filtering, etc.*.
- ADDNEQ offers unique options like the possibility to combine $n \geq 1$ one-day-arcs into an n-day-arc or to introduce new parameters like e.g. velocities for station coordinates for annual solutions.
- Weights which were added to the NEQ-systems in GPSEST may be removed or modified in ADDNEQ.
- The physical model may be changed in program ADDNEQ. It is e.g. possible here to add stochastic orbit parameters, to change the number of troposphere parameters (per station and day).
- Because there is no necessity to set up the observation equations, program ADDNEQ is very efficient. Long solution series may be produced in minutes.

The data files relevant to Version 4 may be divided into three groups:

General Information: Constants, geodetic datum, gravity potential of the Earth, Earth rotation parameters, etc. .

Campaign Specific Information: Observation files, orbits, coordinates, troposphere parameters, etc. .

Program Specific Information: Input option files, files with lists of file names (e.g. observation file names).

1.3 The Observables and Linear Combinations

In the Bernese GPS Software Version 4.0 we still primarily use the *double difference* as basic observable. This allows it to approximate the satellite clocks by a *single point positioning* (processing program CODSPP) *before* the precise parameter estimation in program GPSEST.

Phase and code observations may be used in the same program run which is why *weights* have to be introduced for different observables. The *weight ratio between code and phase observations* is an input variable. At present we use $\sigma(code) : \sigma(phase) = 1 : 10'000$ if the precise or $P - code$ is used, $1 : 100'000$ if the $C/A - code$ is used.

Five different linear combinations of the original observation (in L_1 and L_2) may be analysed in the processing programs (it is even allowed to use different linear combinations in the same program run) (see Chapter 9). Table 9.1 lists the different linear combinations and their properties.

1.4 Parameter Estimation

Geodesists are mainly interested in the site coordinates and their development in time. Orbit parameters and earth rotation parameters (ERPs) are of primary interest to agencies using the *Bernese GPS Software Version 4* for orbit determination. Recently the interest in *atmosphere modeling using the GPS* became an important issue: It is e.g. possible to extract (with a high time resolution) the precipitable water content over a station by subtracting from the estimated *zenith tropospheric delay* the *dry component of tropospheric refraction.* Other agencies see their primary interest in modeling the ionosphere using a regional or a global GPS tracking network. The software allows to estimate *ionosphere models of high quality*, too. Table 1.2, gives an overview of the active model parameters of Version 4.0.

Parameter	Description	Available in	
		GPSEST	ADDNEQ
Station Coordinates	Rectangular coordinates X, Y, Z in the ITRF (at present the ITRF93 is used). The results are also expressed in the (user defined) geodetic datum ($\lambda,\ \beta, h$).	Yes	Yes
Receiver Clocks	Absolute and relative receiver clock offsets may be estimated using the double difference observable with moderate accuracy (relative accurate to about 1μs, absolute to about $1ms$).	Yes	No
Station Velocities	In program ADDNEQ station velocities may be set up if a long time series of NEQ systems containing the same stations is available.	No	Yes
Epoch specific station coordinates	A set of station coordinates is assigned to each epoch (pseudo-kinematic surveys).	Yes	No

Parameter	Description	Available in	
		GPSEST	ADDNEQ
Ambiguities	One initial phase ambiguity parameter has to be assigned to each (linearly independent) double difference. If ambiguities were resolved once they may be introduced as known in subsequent program runs.	Yes	No
Antenna Phase Center Variations	Antenna phase center variations may be modeled using different techniques. Model parameters may be determined.	Yes	No
Receiver Antenna Offsets	May be estimated for antenna calibration experiments (together with phase centers) if antennas are rotated.	Yes	No
Orbit Elements	Osculating Orbital Elements at initial time t_0 of arc: semi-major axis a, eccentricity e, inclination i, right ascension of ascending node Ω, argument of perigee ω, and argument of latitude u_0 at time t_0.	Yes	Yes
Radiation Pressure Parameters	Radiation pressure parameters: A total of *nine* parameters per satellite and arc may be introduced, three in each of three orthogonal directions (direction sun-satellite, solar panels axis, and normal to the first two). Constant, sin-, and cos- terms with the argument of latitude $u(t)$ of the satellite as an argument may be introduced.	Yes	Yes
Pseudo-Stochastic Orbit Parameters	Velocity changes in pre-determined directions at user-defined epochs may be introduced for each satellite. Up to three directions (radial, along-track, out-of-plane) may be defined per (stochastic) epoch.	Yes	Yes
Earth Rotation Parameters	Polar motion (x and y components), $UT1 - UTC$, nutation in obliquity and in longitude may be modeled. Only drifts may be estimated for the latter three parameter types. High time resolution is possible.	Yes	Yes
Satellite Antenna Offsets	Such offsets may be assigned to different types of spacecrafts (Block I, Block-II satellites).	Yes	Yes
Center of Mass	The center of mass of the Earth may be estimated, which makes sense for global analyses.	Yes	Yes
Station Specific Troposphere Parameters	Time- and station-specific zenith delay parameters may be introduced and solved for. High temporal resolution is possible, as well as reduction of number of parameters in ADDNEQ.	Yes	Yes

Parameter	Description	Available in	
		GPSEST	ADDNEQ
Local Troposphere Models	In small survey areas (e.g. 10km$\times$10km) a height-dependent (polynomial) model for the tropospheric zenith delay may be established.	Yes	No
Ionosphere Models	Regional or global ionosphere single-layer models may be determined. The electron density in the layer is modeled using spherical harmonics.	Yes	No
Stochastic Ionosphere Parameters	Epoch- and satellite-specific ionosphere parameters may be introduced (together with a priori weights) to support ambiguity resolution.	Yes	No

Table 1.2: Parameters of the programs GPSEST and ADDNEQ in the Bernese GPS Software Version 4.0

Let us mention that ionosphere models based on the observations of one dual-band receiver only may still be established using the program IONEST.

1.5 Accuracy and Performance

It is interesting to ask what precision and what accuracy are achievable today with a good software tool. Table 1.3 tries to give an answer by listing the estimated formal and actually achieved accuracies. Formal uncertainties are always too optimistic. The *actual* coordinate accuracies either stem from *terrestrial* comparisons (e.g. for the Turtmann campaigns) or from comparisons with the other space geodetic techniques (VLBI and SLR). We must make a clear distinction between the activities before and after 1992: Up to this date it was necessary to establish the orbits for the campaigns of considerable ($\geq$ 10 km) size with the material of the campaign (e.g. EUREF-89), afterwards the results of the *International GPS Service for Geodynamics (IGS)* could be used.

The last entry in Table 1.3, CODE, stands for *Center for Orbit Determination in Europe*. CODE is one of at present seven IGS Analysis Centers. The Analysis Center is located at the *Astronomical Institute of the University of Berne*. It goes without saying that CODE uses the *Bernese GPS Software Version 4*. This particular application is a cornerstone for the development of the *Bernese Software*: it guarantees that *state-of-the-art* modeling techniques are used *and* that the software tool is very efficient. We believe that it is this combination of correct modeling and high efficiency which makes our software highly attractive.

It can be said that thanks to the IGS the *orbit error*, which was the dominant contributor to the error budget for large networks until 1992, became less and less important afterwards. After 1993 the IGS orbits allowed (and allow) to achieve sub-centimeter accuracy in the horizontal position and about 1 cm accuracy vertically even for regional networks with a diameter of several thousand kilometers, provided a good software system like the *Bernese GPS Software Version 4* is used.

Campaign	Year	Responsible Agency	# Sites	Size (km)	Accuracy (m) formal	Accuracy (m) actual
Ottawa	83	Canad. Energy, Mines, Resources	4	10 x 60	.004	—
Quebec	84	Energie et Re-sources, Quebec	16	2 x 3	.001	.001
California	84	JPL	2	140	.015	—
CERN	84	CERN	7	12 x 12	.002	.004
Alaska	84	U.S. NGS	8	800 x1600	.050	.070
HPBL	85	JPL	9 2	2000 x4200 240	.050 .020	.065 .030
Turtmann	85	Swiss Geodetic Commission	7	4 x 6	.002	.003
Iceland	86	UNAVCO	52	250 x 450	.020	—
Turtmann	86	Swiss Geodetic Commission	8	4 x 6	.002	.003
Alaska	86	U.S. NGS	8	800 x1600	.030	.030
Switzerland	87	Swiss Geodetic Commission	12	180 x180	.010	—
EUREF-89	89	EUREF Commission Commission	93	4000 x 2000	.020	.040
Turtmann	89	Swiss Geodetic Commission	8	4 x 6	.002	.003
Turtmann	91	Swiss Geodetic Commission	8	4 x 6	.001	.003
Turtmann	92	Swiss Geodetic Commission	8	4 x 6	.001	.003
Turtmann	93	Swiss Geodetic Commission	8	4 x 6	.001	.001
Turtmann	94	Swiss Geodetic Commission	8	4 x 6	.001	.001
GIG’91 (European Part)	94	JPL	14	3000 x 1000	.010	.020
GIG’91 (Global)	94	JPL	14	Global	.020	.050
Swiss Network	86-95	Federal Office of Topography	300	180 x 180	.001	—
COSMOS Japan	94-96	Geographical Survey Institute	100	200 x 200	.001	—
BiGG Japan	96	Geographical Survey Institute	600	1200 x 1500	.001	—
CODE	92-96	Center for Orbit Det. in Europe	75	Global	.001	.004

Table 1.3: Major Campaigns processed with the Bernese GPS Software

Admittedly the improvement of the orbit quality was not so important for campaigns of the Turtmann-type. Atmosphere modeling techniques (ionosphere and troposphere) and antenna phase center calibrations proved to be far more important, here. We refer to [*Beutler et al.*, 1995] for more information.

2. NAVSTAR GPS – Basic Facts

In 1973 the U.S. Department of Defence decided to establish, develop, test, acquire, and deploy a spaceborne Global Positioning System. The result of this decision is the present NAVSTAR GPS (NAVigation Satellite Timing And Ranging Global Positioning System). According to [*Wooden*, 1985]

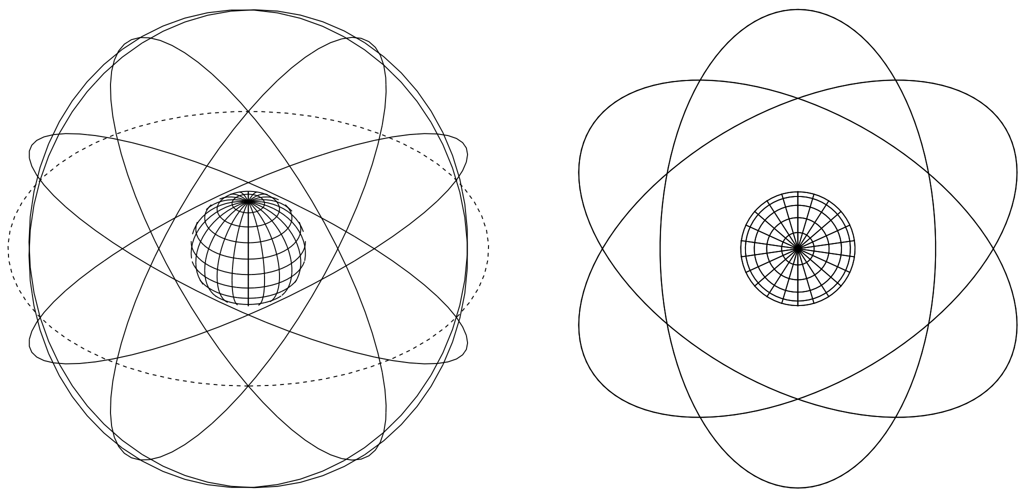
> "The NAVSTAR Global Positioning System (GPS) is an all-weather, space-based navigation system under development by the U.S. Department of Defence to satisfy the requirements for the military forces to accurately determine their position, velocity, and time in a common reference system, anywhere on or near the Earth on a continuous basis."

From this definition it is clear that the primary goals for developing the GPS were of a military nature. But the U.S. Congress has allowed the civilians to use this system with some restrictions. The civilian usage of the NAVSTAR GPS has developed enormously within the last decade. One of the most important events for the high accuracy civilian applications of GPS was the establishment of the *International GPS Service for Geodynamics (IGS)* – [*Mueller and Beutler*, 1992; *Beutler*, 1992].

There are several other Global Positioning Systems different to NAVSTAR GPS either operational or under development. However, NAVSTAR GPS has undoubtedly the greatest impact on the scientific community at present. Therefore we use the term GPS as a synonymous to NAVSTAR GPS. In this chapter we present some basic, but essential facts concerning the GPS.

2.1 GPS Satellites and their Constellation

The proposed constellation of the GPS was subject to several changes due to budgetary considerations. The present *full constellation* provides global coverage with four to eight simultaneously observable satellites above 15^o elevation. This is accomplished by 24 satellites. The satellites are located in six orbital planes in almost circular orbits with an altitude of about 20 200 km above the surface of the Earth, inclined 55^o with respect to the equator and with orbital periods of approximately 11 hours 58 minutes (half a sidereal day). Consequently almost the same Earth–satellite configuration repeats itself 4 minutes earlier every day. The GPS orbits are given in Figure 2.1. The first GPS satellite PRN 4 (Pseudo-Random Number – see below) was launched on 22 February 1978. PRN 4 was the first in a series of 11 so-called Block I satellites. The Block I satellites had an inclination of about 63^o with respect to the Earth's equator. The test configuration was optimized for the region of North America in the sense that four or more satellites could be observed for a considerable fraction of the day there. The test configuration was not optimal in other parts of the world. At present no Block I satellite is active any more.



(a) Viewed from a Latitude of $\phi = 35^o$ (b) Viewed from a Latitude of $\phi = 90^o$

Figure 2.1: GPS Orbits (Earth and Orbital Planes in Scale)

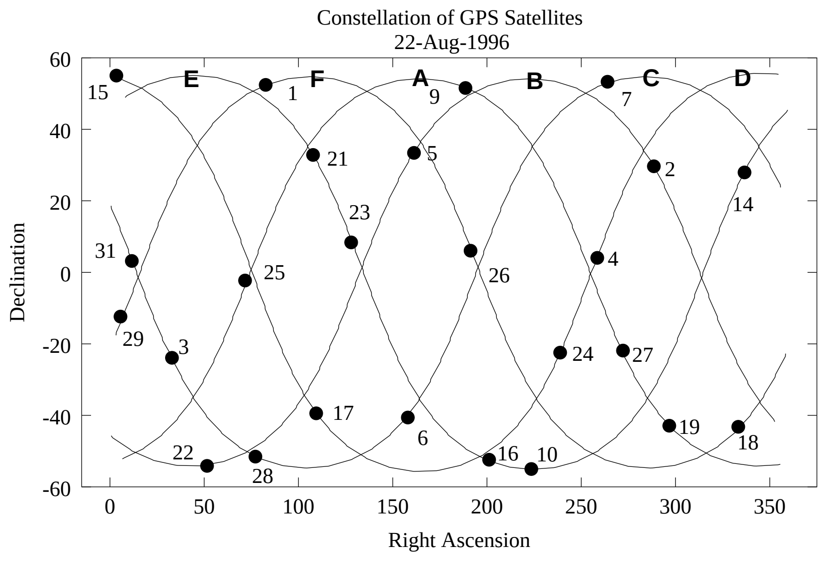


Figure 2.2: GPS Orbits (Planes A–F) in Non-Rotating Equatorial System

The GPS satellites provide a platform for radio transmitter, atomic clocks, computers, and various equipment used for positioning and for a series of other military projects (e.g. atomic flash detection). The electronic equipment of the satellites allows the user to operate a receiver to measure quasi-simultaneously topocentric distances to more than three satellites. Each satellite broadcasts a message which allows the user to recognize the satellite and to determine its spatial position for arbitrary time epochs. The satellites are equipped with solar panels for power supply, reaction wheels for attitude control, and a propulsion system for orbit adjustments.

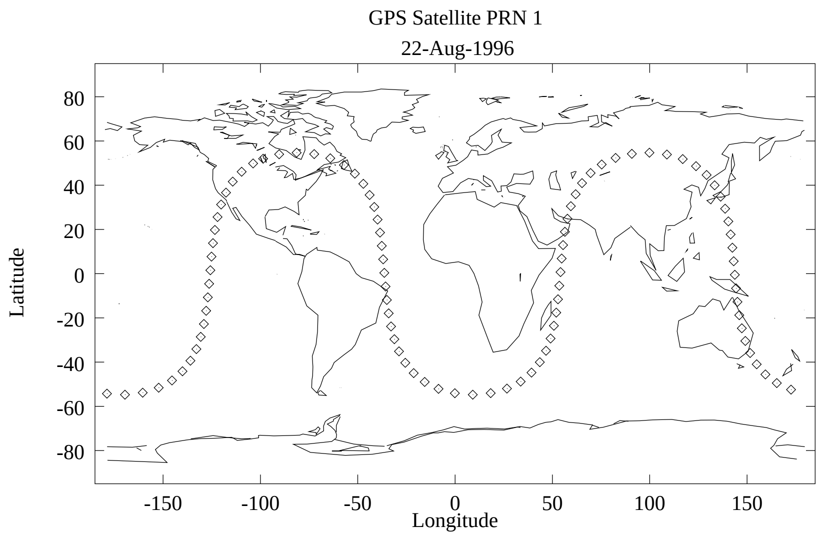


Figure 2.3: GPS Orbit in the Terrestrial System

The operational constellation is realized through the Block II and Block IIa satellites. The first Block II satellite was launched in February 1989. Today a full constellation of 24 Block II, Block IIa satellites is available.

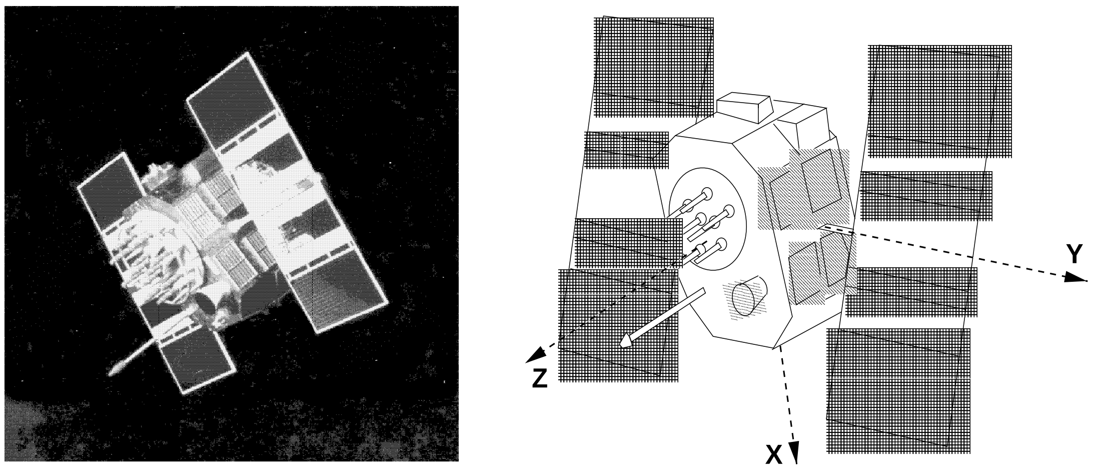


Figure 2.4: GPS Block II Satellite and Satellite-fixed Coordinate System

2.2 The Satellite Signal

All signals transmitted by the satellite (see Table 2.1) are derived from the fundamental frequency f_0 of the satellite oscillator.

Component	**Frequency** [MHz]		
Fundamental frequency	f_0	=	10.23
Carrier L_1	$f_1 = 154\, f_0$	=	1575.42 ($\lambda_1 \doteq 19.0$ cm)
Carrier L_2	$f_2 = 120\, f_0$	=	1227.60 ($\lambda_2 \doteq 24.4$ cm)
P-code $P(t)$	f_0	=	10.23
C/A-code $C(t)$	$f_0/10$	=	1.023
Navigation message $D(t)$	$f_0/204600$	=	$50 \cdot 10^{-6}$

Table 2.1: Components of the Satellite Signal

The two sinusoidal carrier frequencies f_1 and f_2 (corresponding wavelengths $\lambda_1 \approx 19$ cm and $\lambda_2 \approx 24$ cm) are modulated with the codes and the navigation message to transmit information such as the readings of the satellite clocks, the orbital parameters, etc. The so-called biphase modulation is used as shown in Figure 2.5:

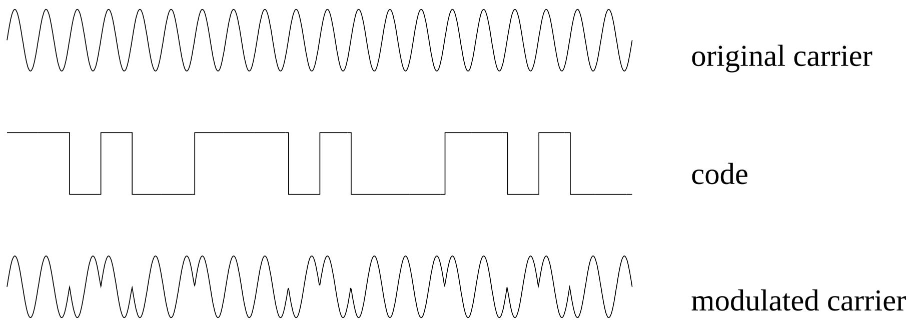


Figure 2.5: Biphase Modulation of the GPS Signal

The two codes $P(t),\ C(t)$ and the navigation message $D(t)$ consist of sequences with two states $+1$, -1, where according to [*Baueršíma,* 1982] the resulting signals may be described as

$$\begin{aligned} L_1(t) &= a_p\, P(t)\, D(t)\, \cos 2\pi(f_1 t) + a_c\, C(t)\, D(t)\, \sin 2\pi(f_1 t) \\ L_2(t) &= b_p\, P(t)\, D(t)\, \cos 2\pi(f_2 t) \end{aligned} \tag{2.1}$$

where a_p, a_c and b_p are the amplitudes of the signals which are not of interest in our context.

Pseudo-Random Codes

The two codes $P(t),\ C(t)$ consist of so-called pseudo-random noise (PRN) sequences. The generation of these sequences is based on hardware devices called tapped feedback shift registers. The C/A-code (Coarse-Acquisition or Clear-Access) is generated by the combination of two 10-bit tapped feedback shift registers where the output of both registers are added again by binary operation to produce the code sequence. A unique code is assigned to each satellite, the sequence has a length of 1023 bits and because of the basic frequency of 1.023 MHz it repeats itself every millisecond. The time interval between two subsequent bits ($\approx 10^{-6}$ s) approximately corresponds to 300 meters.

The generation of the P-code (Precise or Protected) is similar, but the length of the resulting sequence is approximately $2.3547 \cdot 10^{14}$ bits corresponding to a time span of approximately 266 days. The total code is partitioned into 37 one-week segments. One segment is assigned to each satellite (which defines the *PRN number* of the satellite). The P-code repeats itself every week. The time interval between subsequent bits is 10 times smaller than in the case of the C/A-code. Therefore the accuracy is approximately 10 times higher than for the C/A-code. The P-code may be encrypted. This procedure is called *Anti-Spoofing* (AS) and converts the P-code to the Y-code which is only usable when a secret conversion algorithm is accessible to the receiver.

The Navigation Message

The navigation message is 1500 bits long and contains information concerning the satellite clock, the satellite orbit, the satellite health status, and various other data. The message is subdivided into five *subframes.* Each subframe contains 10 words. The first word is the so-called *telemetry word* (TLM) containing a synchronization pattern and some diagnostic messages. The second word of each subframe is the *hand-over word* (HOW). This word contains also the so-called *Z-count* which gives the number of 1.5 s intervals since the beginning of the current GPS week. This number and the P-code give the reading of the satellite clock at signal transmission time. The first subframe contains various flags and the polynomial coefficients which define the satellite clock correction (see Table 2.2).

Parameter	**Explanation**
Code-Flag L_2	Indicator for C/A or P-code on L_2
Week No.	GPS week
L_2-P-Data-Flag	Indicator for data on L_2-P-code
SV-Accuracy (URA)	Measure for distance accuracy
SV-Health	Satellite health indicator
T_{GD}	Group delay difference L_1-L_2-P-Code
AODC	Age of clock data
t_{0c}	Reference epoch
$a_0,\ a_1,\ a_2$	Clock correction polynomial coefficients

Table 2.2: Broadcast Clock Parameters

The second and the third subframe contain the broadcast ephemerides of the satellite (see Table 2.3).

Parameter	**Explanation**
AODE	Age of ephemerides data
t_e	Ephemerides reference epoch
$\sqrt{a},\ e,\ M_0,\ \omega_0,\ i_0,\ \ell_0$	Keplerian parameters at t_e
$\mathrm{d}\,n$	Mean motion difference
$\mathrm{d}\,i$	Rate of inclination angle
$\mathrm{d}\,\Omega$	Rate of node's right ascension
$C_{uc},\ C_{us}$	Correction coeff. (argument of perigee)
$C_{rc},\ C_{rs}$	Correction coeff. (geocentric distance)
$C_{ic},\ C_{is}$	Correction coeff. (inclination)

Table 2.3: Broadcast Ephemerides

Using the broadcast ephemerides the Earth-fixed geocentric coordinates of the satellites may be computed according to the formulas given in [*Dierendonck et al.*, 1978]. The fourth and the fifth subframe contain data for military use, information on the ionosphere, and so-called almanac data (low-accuracy orbits of all the GPS satellites).

The GPS user may decide whether to use the broadcast ephemerides or the precise ephemerides (produced by the International GPS Service for Geodynamics) for processing. The broadcast ephemerides are available in real time, but they have an accuracy of "only" several meters. The precise ephemerides have an accuracy of several centimeters and they are available with a delay of about two weeks (see Chapter 7).

The satellite clock corrections are required for processing. The accuracy of this information in the broadcast message is artificially degraded for non-privileged users. This degradation is called Selective Availability (SA). However, the effect of SA is fully eliminated in geodetic applications when only relative positions of receivers are estimated. The IGS precise orbits contain highly accurate satellite clock corrections, too.

2.3 Signal Processing

The receivers contain elements for signal reception and signal processing (antenna, pre-amplifier, radio frequency (RF) section, microprocessor, storage device, control device, and power supply). After signal input from the antenna, the signals are discriminated. Usually this is achieved through the C/A-codes which are unique for each satellite. The basic elements of the RF section are oscillators to generate a reference frequency, filters to eliminate undesired frequencies, and mixers. The *pseudorange measurements* are achieved as follows: A reference carrier is generated in the receiver and then modulated with a copy of the known PRN code. This modulated reference signal is then correlated with the received satellite signal. Neglecting the receiver and satellite clock errors (see Chapter 9) this correlation gives directly the travel time τ (or, multiplied by the velocity of light c, the so-called pseudorange $c\,\tau$).

The *phase measurements* are based on processing the reconstructed signal carriers. This signal is usually obtained by the code demodulation technique using the correlation between the received signal and the signal copy generated by the receiver. Other techniques must be used for the L_2 phase in C/A-code receivers or for both phases in the case of the codeless receiver. One technique is the so-called squaring technique, where the received signal is multiplied with itself and hence all "$\pm\pi$ modulations" are removed. The result is the unmodulated squared carrier with half the period. From this squared carrier a sine wave is derived with a wavelength of only half the wavelength of the original signal. Another possibility is the so-called cross-correlation technique.

The receiver records the signal at time t. This signal was transmitted by the satellite at time $t-\tau$ (see also Chapter 9). At time $t-\tau$ the phase of the satellite oscillator equals $\phi^i(t-\tau)$ and at time t the phase of the receiver oscillator equals $\phi_k(t)$. The receiver thus compares the following two signals:

$$y^i = a^i \ \cos\ 2\pi\phi^i(t-\tau) \quad \text{and} \quad y_k = a_k \ \cos\ 2\pi\phi_k(t)\ , \tag{2.2}$$

where a^i and a_k are the amplitudes of the signals. Multiplying these two signals we obtain:

$$y = y^i y_k = \frac{a^i a_k}{2}\Big\{\cos 2\pi\Big[\phi_k(t) - \phi^i(t-\tau)\Big] + \cos 2\pi\Big[\phi_k(t) + \phi^i(t-\tau)\Big]\Big\}\ . \tag{2.3}$$

After applying a low-pass filter, the high frequency part $\phi^i(t-\tau)+\phi_k(t)$ is eliminated and (compare Chapter 9)

$$\psi_k^i = \phi_k(t) - \phi^i(t-\tau) + n_k^i \tag{2.4}$$

may be measured. The accuracy of the phase measurements is about 1–3 mm, but the exact number of integer wavelength between the satellite and the receiver n_k^i is not known at the time of the first measurement. The unknown integer number of cycles n_k^i to be added to the phase measurement to get a pseudorange is called the *initial phase ambiguity* (see also Chapter 9). This phase ambiguity has the same value as long as the receiver keeps lock on the phase transmitted by the satellite.

3. The Menu System

3.1 Introduction

3.1.1 Bernese GPS Processing Programs

The processing part of the Bernese GPS Program System consists of more than 80 FORTRAN programs, designed for easy installation on a variety of computer types. Most of these programs run in a batch mode, i.e. they do not need any user interaction during execution time. All program options and lists of data (observation) files and auxiliary files (e.g. the file containing earth potential parameters) have to be prepared previously and made available to the program in special input files (the so-called `I-`, `F-`, or `N-`Files). In the following sections we will always refer to these programs as *processing programs.*

One of the major design principles was to stick to standard FORTRAN-77 features and to omit any system- or installation-dependent language enhancements or to rely on given filenames or device and directory structures. This is one of the reasons why the processing programs do not need any user interaction but fully rely on previously defined input option files. These input files were initially meant to be prepared "manually", therefore they were designed to be very descriptive.

However, the growing number and complexity of the processing programs and the rapidly increasing volume of data to be processed made it mandatory to develop tools to help the user to navigate through the various programs, to prepare the necessary input files, to handle the data and auxiliary files, and to keep track of the program outputs. Therefore we decided to develop another series of programs, the so-called *Menu System.*

3.1.2 The Bernese GPS Menu System for DOS, VMS, and UNIX

The Menu System is an independent program system for the preparation of the necessary input files and the management of all processing programs, data files, and program outputs.

It consists of

- a top-level program (MENU) that allows the user to select the specific task to be executed (e.g. to convert RINEX files to the Bernese binary observation files),
- a number of programs to allow the interactive preparation of the primary input files for the processing programs (principle: one preparatory program for each processing program),

- a number of programs for handling data files, program outputs, and Menu System options,
- a number of command procedures to ensure the proper sequence of program calls and to prepare the necessary job environments.

Most of the user interaction (selection of programs, files, and options) is done through mask-type input panels. These panels are simple ASCII files.

In order to have the possibility to run a series of processing programs together with the preparation of their option input files fully automatically, we also provide a special non-interactive mode of operation that does not display the input panel files but takes the current contents of the respective input fields in the panel files; we allow for special placeholders in the panel input fields that are automatically replaced by the actual values during execution of the preparatory programs.

Although we tried to follow the same design principles as for the processing programs, the Menu System is slightly computer-dependent, as there are several tasks being

- outside of the capabilities of standard Fortran (as e.g. the execution of system commands under the control of a Fortran program),
- dependent on the Operating System (as e.g. the creation of a list of existing files in a given directory),
- dependent on the terminal type used (for non-standard output as e.g. clear screen, cursor position, attribute settings).

Over the past years a Menu System has been developed

- for the DEC VAX and Alpha Computers running under the OpenVMS operating system respectively,
- for 386/486 Personal Computers running under DOS using a compiler (LAHEY) that creates code to be run under protected mode in order to overcome the 640 kBytes limitation for the program size,
- for the UNIX Operating System currently including the various flavors and varieties to be found on the Hewlett Packard workstations, the SUN workstations under Sun-OS and Solaris, the IBM workstations under AIX, LINUX on PC, and others (see Chapter 24, “Installation” for a complete list).

3.1.3 Structure of the Menu System

In order to gain a better overview of the various processing programs and their preparatory programs they have been grouped logically into a certain hierarchical structure. The top level is the major division into the first sub-levels, see Figure 3.1 .

```
0  DEFAULTS     : Defaults for Processing, Program and File Names
1  CAMPAIGNS    : Informations and Update of Campaigns
2  TRANSFER     : Data Transfer to Bernese Format; Simulation
3  ORBITS       : Orbit Computation, Check and Update
4  PROCESSING   : Preprocessing and Processing of Observations
5  SERVICES     : Service Programs
6  BPE          : Bernese Processing Engine
7  DOCU         : Documentation, Help Panels
9  USER         : Individual User Programs
```

Figure 3.1: Menu System Top Level

Each sub-level may again be divided into a number of lower-level branches until we arrive at the level of the actual preparatory programs.

Each branch is numbered from 0 to (maximum) 9, we address a specific branch by the series of the individual branch numbers, e.g.:

`2.7.3` denotes the second branch on the top level, then the seventh branch on the first sublevel, and then the third branch on the second sublevel.

```
top     first   second
         sub     sub

         1
 0       2
 1       3
 2  -->  4
 3       5
 4       6       1
 5       7  -->  2
                 3  -->  Program RNXMET_P
```

3.2 Starting the Menu System

3.2.1 Preparing the Environment (LOADGPS)

Before the MENU program can be run, the

- DOS batch file `X:\EXE\LOADGPS.BAT`, or
- VMS command file `X:[EXE]LOADGPS.COM`, or
- UNIX script file `LOADGPS`

must be invoked to perform the following tasks:

- substitution of the general GPS-Subdirectory by the drive name `X:` (and `$X` under UNIX),

- substitution of the selected RAM drive name by the name `Y:` and add `Y:` to the current path (DOS only),
- substitution of the user-dependent GPS-Subdirectory by the logical `U:` (VMS), `$U` and `U:` (UNIX), `U:` (DOS),
- set system parameters that are used by the LAHEY Operating System (DOS only),
- copy the Lahey Operating System to the RAM drive (if the drive is large enough, version 1.9) or load the OS386 Operating System (version 2.1) (DOS only),
- copy `.BAT` files from `X:\EXE` to the RAM drive `Y:` for easy access (DOS),
- define additional logicals and symbols (VMS),
- define additional variables (UNIX),
- define additional environment variables (DOS), such as `%RUN%` for the LAHEY run command.

This command procedure could be called automatically by `AUTOEXEC.BAT` (DOS), `LOGIN.COM` (VMS) or e.g. `.profile` (UNIX) at system start-up time (see also installation Chapter 24).

```
rem
rem  BATCH FILE TO PREPARE ENVIRONMENT FOR THE BERNESE GPS SOFTWARE
rem  --------------------------------------------------------------
rem                          Version 4.0
       set VERSION=40
rem
rem  OS/386 1.9 Users     : Remove 'rem' from line with "OS386-19" below
rem  OS/386 2.1 Users     : Remove 'rem' from line with "OS386-21" below
rem  Pharlap 386DOS Users: Remove 'rem' from line with "386DOS"   below
rem
rem    set DOSEXT=OS386-19
rem    set DOSEXT=OS386-21
       set DOSEXT=386DOS
rem
rem  Define below :
rem    the drive on which you have the PGM and LIB directories (set C)
rem    the subdirectory with the Menu System (subst drive X:)
rem    the ram drive to be used for the Menu System (subst drive y:)
rem
rem  device with source/obj/exe directories
       set C=C:
...
```

Figure 3.2: Preparing the Environment (DOS)

3.2.2 Calling the Menu System (G)

The command procedure `G.BAT` (DOS) / `G.COM`, executed by the symbol `G` on all platforms is used to start the Bernese GPS Software. It invokes the menu program MENU, the preparatory programs, and the processing programs.

The following example (Figure 3.3) of the DOS command procedure contains enough comments to explain its operations.

```
@echo off
rem
rem                 Batch Program to Run the Menuprogram and to
rem           Invoke the Preparatory Program and the Mainframe Program
rem                            Selected by the User.
rem   -------------------------------------------------------------
rem                                 Version 4.0
rem
rem  Parameter %1:  menu section to be executed first
rem  Write it to auxiliary file. It will be read back by MENU.
rem
     echo %1 >Y:\MENU.AUX
rem
rem  copy options file into working directory
rem
     if not exist Y:\MENU.OPT copy X:\SKL\MENU.OPT Y:\*.* >nul:
rem
rem  Call menu program to display menu panels
rem
:menulabel
     %RUN% %C%\PGM\MENU%VERSION%\EXP\MENU
rem
rem  If the user selected preparatory program was to be executed,
rem  MENU would create a command file MENUBAT1.BAT containing commands to
rem  execute the preparatory program.
rem
     if not exist Y:\MENUBAT1.BAT goto end
rem
     call Y:\MENUBAT1
rem
rem  If the user wanted the mainframe program to be automatically
rem  executed (no abort of preparatory program, processing default to
rem  SUBMIT=YES), the preparatory program would create a batch file
rem  MENUBAT2.BAT containing commands to execute the mainframe
rem  batch file.
rem
     if not exist Y:\MENUBAT2.BAT goto menulabel
rem
     call Y:\MENUBAT2
     goto menulabel
rem
:end
```

Figure 3.3: Menu System Startup File for DOS

`MENUBAT1.BAT` contains the command to execute the selected preparatory program *pgmnam*`_P`.

Example (DOS):

```
RUN386 C:\PGM\MENU40\EXP\DEFSTD_P
```

`MENUBAT2.BAT` contains the command to execute the selected processing program *pgmnam*.

Example (DOS):

```
CALL Y:DEFSTD.BAT
```

(Apart from the actual execution of the processing program, the command file DEFSTD.BAT performs some file handling as well. Details see 3.6.1).

The *VMS version* shows one major difference:

The actually used filenames for the option and auxiliary files of the program MENU and the `MENUBAT1` and `MENUBAT2` procedures are process name specific: Their original names are supplemented with the current process name as determined at startup by `LOADGPS.COM`. This means that one and the same user can run the menu program simultaneously from different processes (e.g. if he logs into the system several times under the same name). However, there may be conflicts between the different when accessing the same data files at the same time.

3.3 Panels

There are three different panel types used by the Menu System:

(1) *Program panels* are used to navigate through the various levels of the Menu System and to select the actual preparatory program (or another program directly accessible through the Menu System like e.g. JOB).

(2) *Data panels* are displayed by the preparatory programs to select options, filenames, or other values and parameters that are used to prepare the basic input files for the processing programs.

(3) *Help Panels* contain information concerning the selected options and additional hints and tips for the programs.

All the panels are simple ASCII files with the following naming conventions:

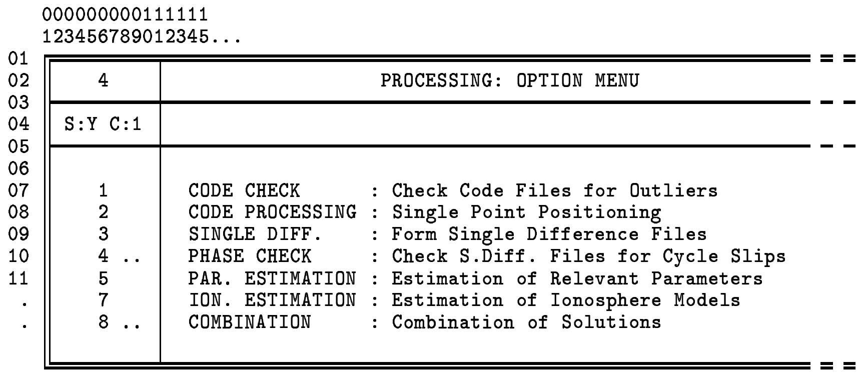
Program panels:	`PAN_____.PAN`: Top level panel
	`PAN27___.PAN`: Corresponds to selection 2.7
Data panels:	`DAT273__.PAN`: Primary panel of program 2.7.3
	`DAT2731_.PAN`: Subsequent data panel of 2.7.3
Help panels:	`DAT2731_.HLP`: Help panel for data panel 2.7.3-1

The data panels for the preparatory programs again show a hierarchical order according to the logical structure of the programs, similar to the structure of the Menu System. There is a maximum of 5 levels available in the panel naming (as the total of the program plus data panel levels). Unused levels are marked in the panel name with the "`_`" character.

The DOS version uses program, data, and help panels with graphic characters (ASCII values > 127) for the panel frames. For the VAX and UNIX version these characters have been replaced by standard characters like "-", "|", and "+".

3.3.1 Program Panels

The program panels are simplified panels that do not contain actual input fields. They are written to the screen by the MENU program using standard Fortran WRITE commands. The user input (usually the selected sublevel), entered after the prompt "`Enter Selection :`", is processed by MENU through the standard READ command.



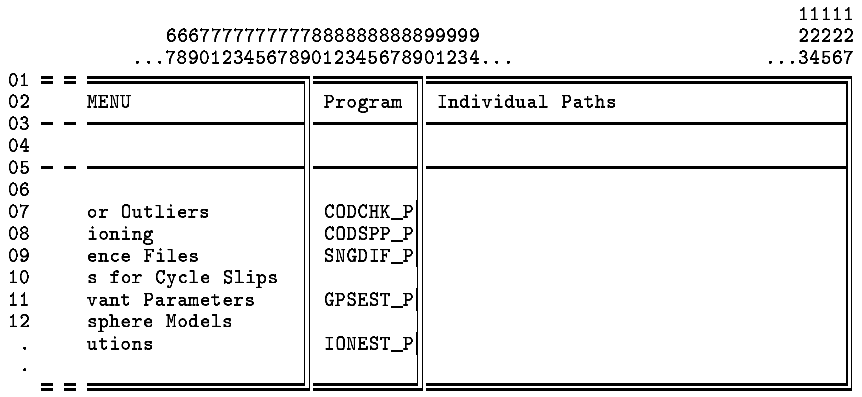
```
    000000000111111
    123456789012345...
01
02     4                    PROCESSING: OPTION MENU
03
04   S:Y C:1
05
06
07     1      CODE CHECK      : Check Code Files for Outliers
08     2      CODE PROCESSING : Single Point Positioning
09     3      SINGLE DIFF.    : Form Single Difference Files
10     4 ..   PHASE CHECK     : Check S.Diff. Files for Cycle Slips
11     5      PAR. ESTIMATION : Estimation of Relevant Parameters
 .     7      ION. ESTIMATION : Estimation of Ionosphere Models
 .     8 ..   COMBINATION     : Combination of Solutions
```

Figure 3.4: Left Part of Program Panel `PAN4____.PAN`

The second line contains the level of the panel and a panel description. The fourth line also contains two fields. The first one contains the current setting of the SUBMIT ("S") and JOBCLASS ("C") option (Menu 0.1, see also 3.5, PRCDEF), the second one is empty and may be used by the MENU program to display short messages.

The first six lines belong to the header of the panel.

The list of the branches (the panel body) starts at line 7. The branch numbers have to be in column 6. Two dots in the same field indicate that this branch will not yet call a preparatory program but will again split into various sub-branches.



```
                                                                  11111
           66677777777778888888888899999                          22222
        ...7890123456789012345678901234...                     ...34567
01
02     MENU                 Program     Individual Paths
03
04
05
06
07     or Outliers          CODCHK_P
08     ioning               CODSPP_P
09     ence Files           SNGDIF_P
10     s for Cycle Slips
11     vant Parameters      GPSEST_P
12     sphere Models
 .     utions               IONEST_P
 .
```

Figure 3.5: Right Part of Program Panel `PAN4____.PAN`

A blank line concludes the body of the panel. Including the blank line the body may have a length of up to 11 lines.

There may follow up to 10 additional lines.

Columns 79 and following contain additional information that is never displayed by the MENU program (see Figure 3.5).

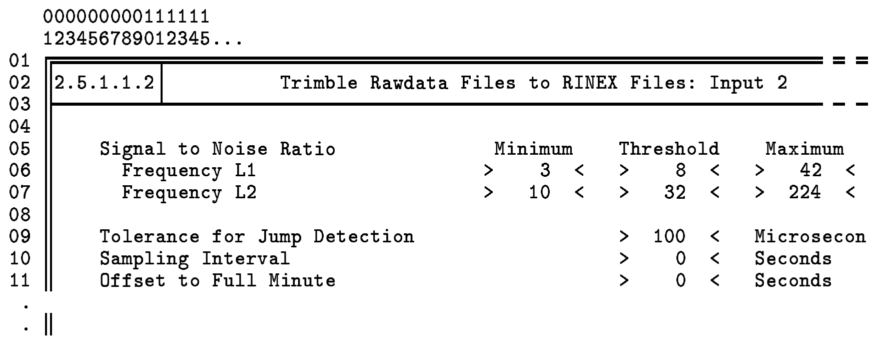
Columns 81 to 88 contain the name of the program that is to be called by the Menu System (usually the preparatory program). The default path to the executable is defined in `X:\SKL\MENU.OPT`.

Blanks in columns 81 to 88 indicate that there is no program to be called but a program panel of the next deeper level is to be displayed.

More details to the program panel fields is available in section 3.10.

3.3.2 Data Panels

Data Panels are ASCII files containing in columns 81 ff. additional information used by the programs that read and interpret the panel files.



```
   000000000111111
   123456789012345...
01 ╔══════════════════════════════════════════════════════════════════ = =
02 ║2.5.1.1.2│          Trimble Rawdata Files to RINEX Files: Input 2
03 ║─────────┴──────────────────────────────────────────────────────── ─ ─
04 ║
05 ║    Signal to Noise Ratio            Minimum    Threshold    Maximum
06 ║      Frequency L1                 >    3  <   >    8  <   >   42  <
07 ║      Frequency L2                 >   10  <   >   32  <   >  224  <
08 ║
09 ║    Tolerance for Jump Detection               >  100  <   Microsecon
10 ║    Sampling Interval                          >    0  <   Seconds
11 ║    Offset to Full Minute                      >    0  <   Seconds
 . ║
 . ║
```

Figure 3.6: Left Part of Data Panel

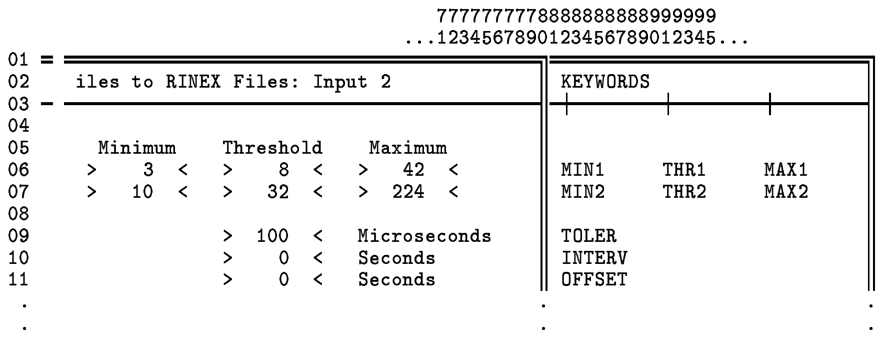
The data panel consists of a header of 3 lines and a body of an unlimited number of lines.

Each actual data input field is delimited by a right bracket (>) and a blank to the left end and a blank and a left bracket (<) to the right.

It is obvious that nowhere in the panel text any such brackets may be used, as they would be interpreted as a start or end of an input field.

At the right end of the data panel (see Figure 3.7), unique keywords for each input field have to be defined. There may be as many as 5 keywords (and corresponding input fields) in each line of the panel. The keywords start at cols. 82, 91, 100, 109, and 118, and may have a length of up to 8 characters.

The order of the keywords within the line has to correspond to the order of the respective input fields. Apart from this restriction the input fields may be positioned anywhere within the panel body. The access routines reading the panels read the input fields according to the corresponding keyword.



```
                                 7777777777888888888899999 9
                               ...12345678901234567890123 45...
01 =═══════════════════════════════════════╗╔═══════════════════════════╗
02   iles to RINEX Files: Input 2          ║║ KEYWORDS                  ║
03 -───────────────────────────────────────║║─┼────────┼────────┼───────║
04                                         ║║                           ║
05     Minimum    Threshold    Maximum     ║║                           ║
06    >    3  <   >    8  <   >   42  <    ║║ MIN1     THR1     MAX1    ║
07    >   10  <   >   32  <   >  224  <    ║║ MIN2     THR2     MAX2    ║
08                                         ║║                           ║
09                >  100  <   Microseconds ║║ TOLER                     ║
10                >    0  <   Seconds      ║║ INTERV                    ║
11                >    0  <   Seconds      ║║ OFFSET                    ║
 .                                        .                             .
 .                                        .                             .
```

Figure 3.7: Right Part of Data Panel

A string of 8 double frame characters (PC) or 8 dashes (-) in one of the keyword fields (i.e. the bottom line of the frame around the panel) will indicate to the FORTRAN access routines (`GETPAN.FOR`, `PTKEYW.FOR`) the end of the panel.

It is also allowed to repeat the same keywords in different lines (see comments in the subroutine GTKEYW), as it is done e.g. in the campaign definition panel `DAT11___.PAN`.

Interaction with the User

The interaction with the user is done by means of a system-independent Fortran routine (DSPPAN) and some system-dependent routines for cursor positioning, character attribute setting, character display, and keyboard operation. The interfaces to the system- dependent Fortran routines have been defined in a system-independent way, therefore the major routine DSPPAN is identical under DOS, VAX, and UNIX.

The next lower level consists of the routines DSPFLD (to display a range of characters at a defined position on the screen), CLRSCR (to clear the screen), and RDKYBD (to wait for and accept a single keystroke). DSPFLD and CLRSCR use `ANSI.SYS` (DOS) or VT200 type of escape sequences.

RDKYBD calls a Lahey routine (IXKEY from the Graphics Library) via routine INPCHR or, on the VAX system, routines using VMS-QIO system to determine what key has been entered and then defines a corresponding (system-independent) "action code".

On UNIX systems we use the curses library which provides special C-routines for terminal handling.

The relation between the system-dependent keycode or return code from the input routine and the corresponding "action code" is defined in a keycode table file, the name of which is handed over to the program in the file `Y:\KEYBOARD.TBL / U:[WORK]KEYBOARD.TBL / $U/WORK/KEYBOARD.TBL`. This file is prepared by the startup procedure `LOADGPS`, i.e. that is where the selection of the active keycode table is actually performed.

The keycode table on PC's is called `X:\SKL\PC_LAHEY.KBD`, for VT200 terminals on the VAX under VMS it is `X:[SKL]VMS_200.KBD`.

For Unix the definition of the keycodes depends on the use of the keypad() function in the `U_CUINIT.c` routine.

keypad active:	You will get terminal-independent return codes for each key. (Refer to curses manual)

keypad inactive: The return codes are terminal-dependent. (Refer to your terminal manuals)

There are various keycode tables available in the directory `$X/SKL` for different window environments and platforms (see command file `LOADGPS`).

The "action code" tells the calling program whether just a character key has been pressed or whether the user wants to execute some special action like moving the cursor to another input field, scrolling the screen up or down, delete characters or input fields, etc. The predefined action codes are linked to function keys or key combinations like Ctrl A or special keys like Enter or PgUp, etc. Pressing the help key (F1 on the PC, PF1 on the VT220) displays a help panel (see below) showing the possible actions and corresponding key strokes.

The most important key is the $<$ CONTINUE $>$ key to end the data entries and to continue with the program. We chose the left most key in the top row of the keyboard (ASCII value 96 decimal, key ‘) because it is comfortably placed and never used for anything else. On some keyboards the key might be found in a different place (!).

By pressing twice Esc you also invoke $<$ CONTINUE $>$ continue.

Of course you might easily re-program the key assignments according to your preferences by merely changing the key code tables. Please keep in mind that you might confuse other Bernese users, however. Use the program KBTEST to determine the key codes of your keyboard. To run the program, enter

```
RMENU KBTEST.
```

Pressing any key will immediately show the respective key code (in decimal) or, for function or other non-standard keys, maybe a series of key codes. These codes then have to be written into the active keycode table into the proper row.

The access to the data entered by the user into the data panel files is then performed by the routine GTKEYW with the data panel name and the keyword as input parameters and the string of the corresponding data field as output parameter.

3.3.3 Help Panels

Help panels are ASCII files, too. Basically there is a corresponding help panel with the same filename but the extension `.HLP` for every data panel. It is stored in the directory `X:\HLP / X:[HLP] / $X/HLP`.

Help panels are displayed by pressing F1 or PF1. (There exists also a help panel for the program panels, it can be displayed by entering "`=H`" in the MENU prompt).

Most of the help panels contain information and examples for all the input fields, sorted in the same order and flagged with the same short descriptive keywords (the uppercase description to the left of the input fields). When pressing the help key the corresponding section of the help panel should be displayed.

Help panels may contain links to other help or data panels or any other ASCII files. These links are embedded in brackets (e.g. `{DAT123__}` or `{U:[PAN]DAT123__.PAN}`). Default directory for the links is `X:[HLP]` etc. and default extension is `.HLP`. The linked file is displayed by moving the cursor to the link field and pressing the help key again. If the link points to a help panel F1 displays the help panel, but F2 displays the corresponding data panel in `U:[PAN] / $U/PAN / U:\PAN`.

Pressing the help key when a help panel is displayed and the cursor is outside of a link field will display a general help panel showing all special key functions used for data entry in the respective data panels.

3.4 Selections

This section summarizes the use of the menu and data panels and states what inputs are possible.

3.4.1 Menu Selections

After having started the Menu System using command "`G`" the top level (primary) program panel is displayed. Four types of input are possible:

(1) Selection of the menu item in the form `i.j.k`: The Menu System will first go to selection i of the *current* level, then to selection j of the next sublevel and then to selection k of the third level. Each sub-selection is separated from the others through a period. You can enter in one command as many sub-selections as there are actually defined. The selection always starts at the *current* level!

Examples: `1, 4.2, 2.7.1`.

(2) Selection of the menu item in the form `=i.j.k`:

The equal sign "=" tells the Menu System to first go up to the top level and then down to selection i of the primary panel, etc. Such a selection might be called an *absolute* selection. It is independent of the current position within the Menu System hierarchy.

(3) Special Selections `=H, =S, =P, =Q, =X`:

`=H` displays a help panel, either the program program panel (if it exists) or a general help panel showing the basic commands for the menu selection.

`=S` allows to search for the Menu System location of a specific program name. The Menu System will ask you for the program name. Wildcards are allowed, e.g. entering `GPS*` will result in

```
PROGRAM GPSEST_P : NR 5 IN PANEL 4
PROGRAM GPSXTR_P : NR 5 IN PANEL 5.6
```

`=P` will bring you to the primary program panel.

`=Q` will bring you up one level

`=X` will properly exit the Menu System and go to the Operating System prompt.

(4) Any other input

Any other input will be passed through to the Operating System and immediately be executed. Some commands are filtered and *not* passed through, such as `LOGOFF` to avoid menu scratch files hanging around.

The Menu System can also be started up by `G i.j.k` to immediately go to sublevel `i.j.k !`

3.4.2 Option Selections in Data Panels

Data panels contain special input fields for all user entries. The display routine only allows you to move the cursor within these fields and from one field to another. The usual keys (cursor keys, PgUp, PgDn, etc.) may be used (press twice the help key to get the general help panel with the key descriptions).

Immediately to the right of the input fields you usually find a list of the allowed entries as uppercase KEYWORDS within parentheses. Instead of entering the desired keyword manually you may also scan through the keywords with the space key if the cursor is positioned at the start of the input field (toggling).

Some options only include YES, NO, and ASIS. YES will bring a follow-up subpanel with options to choose from, NO will of course not make use of these options, ASIS ("as is") acts as YES but will *not* display these options and take the values as they were stored in the respective panel at an earlier time.

`=Q` brings you back to the previous data panel or to the first data panel of the preparatory program.

You may also enter `=X` into most of the input fields to immediately exit the current preparatory program and to go back to the Menu System.

`= i.j.k` will exit the preparatory program and go to sublevel `i.j.k` of the Menu System.

We recommend to enter such exit commands into the campaign field of the first data panel (because this field will be filled in automatically with the currently selected campaign name the next time you enter the program).

`^D` (Ctrl D) allows you to enter a command that is immediately passed to the Operating System.

3.4.3 File Selections

Many input fields concern names of files to be processed. Most of these fields only request the “stem” of the filenames, i.e. the filename extensions, the correct paths to the files being determined elsewhere. (Depending on what files are requested, the Menu System determines this additional information from the panels under Menu 0.3.)

You may either directly insert the correct name of the file (when just one file has to be processed) or you may have the Menu System display a selection list where you can mark all the files you want to process. (Usually the full names of the selected files will be passed to the processing program through the “F-file”, which is one of the three basic input files for the processing programs).

To generate a file selection table you may either leave the input field blank (which will display all available files of the correct file type and current campaign) or you may limit the number of displayed filenames by using a wildcard notation:

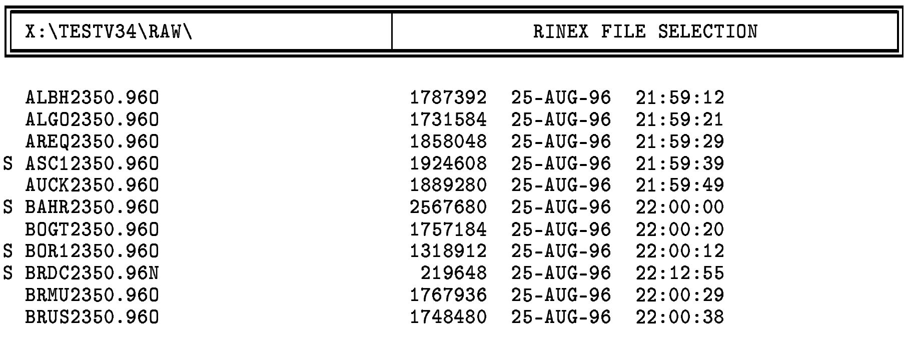
Examples:

Phase files: `ABCD*` will display all available files from station ABCD

Single Difference files: `????1234` will display all files from session 1234

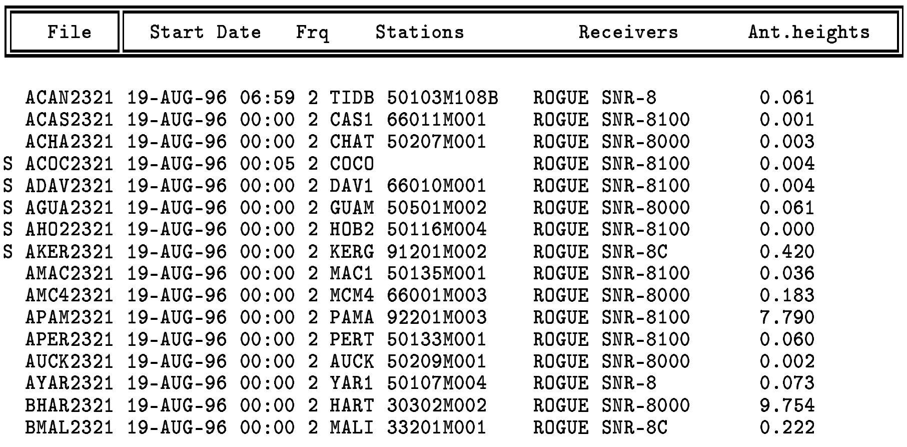
A question mark or percent sign (VMS) is a place holder for exactly one character. An asterisk may be replaced by any number of characters. (DOS version: Characters after an asterisk are allowed for zero-difference and single-difference files only!).

Files are selected by writing an “S” into the selection column of the selection table.



```
X:\TESTV34\RAW\                           RINEX FILE SELECTION

  ALBH2350.96O                     1787392  25-AUG-96  21:59:12
  ALGO2350.96O                     1731584  25-AUG-96  21:59:21
  AREQ2350.96O                     1858048  25-AUG-96  21:59:29
S ASC12350.96O                     1924608  25-AUG-96  21:59:39
  AUCK2350.96O                     1889280  25-AUG-96  21:59:49
S BAHR2350.96O                     2567680  25-AUG-96  22:00:00
  BOGT2350.96O                     1757184  25-AUG-96  22:00:20
S BOR12350.96O                     1318912  25-AUG-96  22:00:12
S BRDC2350.96N                      219648  25-AUG-96  22:12:55
  BRMU2350.96O                     1767936  25-AUG-96  22:00:29
  BRUS2350.96O                     1748480  25-AUG-96  22:00:38
```

Figure 3.8: Example: General File Selection Panel



```
   File    Start Date   Frq    Stations          Receivers      Ant.heights

  ACAN2321 19-AUG-96 06:59 2 TIDB 50103M108B   ROGUE SNR-8          0.061
  ACAS2321 19-AUG-96 00:00 2 CAS1 66011M001    ROGUE SNR-8100       0.001
  ACHA2321 19-AUG-96 00:00 2 CHAT 50207M001    ROGUE SNR-8000       0.003
S ACOC2321 19-AUG-96 00:05 2 COCO              ROGUE SNR-8100       0.004
S ADAV2321 19-AUG-96 00:00 2 DAV1 66010M001    ROGUE SNR-8100       0.004
S AGUA2321 19-AUG-96 00:00 2 GUAM 50501M002    ROGUE SNR-8000       0.061
S AHO22321 19-AUG-96 00:00 2 HOB2 50116M004    ROGUE SNR-8100       0.000
S AKER2321 19-AUG-96 00:00 2 KERG 91201M002    ROGUE SNR-8C         0.420
  AMAC2321 19-AUG-96 00:00 2 MAC1 50135M001    ROGUE SNR-8100       0.036
  AMC42321 19-AUG-96 00:00 2 MCM4 66001M003    ROGUE SNR-8000       0.183
  APAM2321 19-AUG-96 00:00 2 PAMA 92201M003    ROGUE SNR-8100       7.790
  APER2321 19-AUG-96 00:00 2 PERT 50133M001    ROGUE SNR-8100       0.060
  AUCK2321 19-AUG-96 00:00 2 AUCK 50209M001    ROGUE SNR-8000       0.002
  AYAR2321 19-AUG-96 00:00 2 YAR1 50107M004    ROGUE SNR-8          0.073
  BHAR2321 19-AUG-96 00:00 2 HART 30302M002    ROGUE SNR-8000       9.754
  BMAL2321 19-AUG-96 00:00 2 MALI 33201M001    ROGUE SNR-8C         0.222
```

Figure 3.9: Example: Observation File Selection Panel

The Menu System maintains separate files containing the latest file selection for various file types (e.g. phase single diff. files, file `U:[WORK]PHSFILE.SEL`) and replaces the user input in the data panel by `SELECTED` whenever the user interaction results in more than one selected file. (Exception: File selections containing filename parameters are *not* replaced!) Subsequent calls of the preparatory program will then access the previous file selections through the stored file selection list.

Special Selection Characters

If the selection table is containing the names of ASCII files you may look at the contents of the files by typing `B` (for “browse”) into the selection column. The action is immediate, no continuation character has to be entered.

`Q` brings you back to the previous panel, `X` leaves to the Menu System.

`^D` allows you to enter a command that is immediately passed to the Operating System.

Exception: `^D` and subsequently `S ALL` selects *all* files in the selection list. (In fact *c* `ALL` marks all files with the selection character *c*: Some menu programs like OBS Menu 5.1 or JOB Menu 5.9 use other selection characters, as well).

`^D` followed by `RESET ALL` clears all current selections.

Filename Parameters

Panel 1.5.1 allows you to define values for so-called filename parameters that can be used to facilitate routine or semi-automatic operation. More information is available in section 3.8.

Example:

`????$SS1` will include into the selection list all available files of the session $SS1 only. The value of $SS1 has to be previously defined in Panel 1.5.1.

3.5 Special Menu Commands

In order to simplify the use of some frequently used menu programs special commands, to be entered either at the Operating System prompt or as input to the MENU program after the "`Enter selection: `" prompt, have been defined. You find the corresponding command files in the directory `X:[EXE] / $X/EXE / X:\EXE`.

PRCDEF

`PRCDEF` (for *processing defaults*) calls the menu program DSPDPAN to display Panel 0.1. In this panel you may set some processing default parameters, such as SUBMIT and JOBCLASS (see Section 3.6.1) .

SUBMIT defines whether (after the preparation of all processing program option) input files the processing program shall be called or whether the Menu System should go back to the program panel level *without* executing the processing program.

On multitask systems JOBCLASS defines the processing mode *foreground* or *background*:

- JOBCLASS = 0 denotes foreground processing.
- JOBCLASSES 1 to 4 define different batch processing queues in VMS (logicals `BATCH1, ..., BATCH4`, their actual values have to be defined on the system level, e.g. `DEFINE BATCH1 SYS$BATCH`) or priorities in UNIX (an example for a command script is given in Section 3.11.2).
- Negative values -1, ..., -4 call the same batch queues 1 to 4, but with "timed" execution (you will get a panel displayed where you can enter the actual execution time).

The only meaningful value for JOBCLASS under DOS is 0!

`PRCDEF` is identical with `G 0.1` on the system level or `=0.1` on the menu level, with the exception that after having processed Panel 0.1, the former command brings you back to where you were, whereas the latter will bring you to the Menu 0.

JOB

`JOB` calls the menu program JOB. It corresponds to `G 5.9` or `=5.9` respectively, again with the difference of where you will go to after completion of the `JOB` command.

`JOB` allows you to handle the job outputs of the processing programs (browse, edit, print, copy to another file, delete) if you had them stored into a job output file (and not just displayed on the screen, see Panel 0.1). More information is available in section 3.6.2.

OBS

`OBS` calls the menu program SERVOBS. It corresponds to `G 5.1` or `=5.1` respectively, again with the difference of where you will end up after command completion.

`OBS` allows you to browse, edit, display, or split the binary observation files or to mark/unmark observations in these files (see Chapter 20, "Services").

ERR, ERRDEL

Most processing programs may generate error messages that are stored into the SYSERR output file. SYSERR is the internal name, the actual name of the error message file is defined in Panel 0.3.1. Usually the file is named `U:[WORK]ERROR.MSG`.

`ERR` is a symbol or command file that allows you to look at this error message file. `ERRDEL` browses *and deletes* the file.

The Menu System displays a message in the message field of the program panels, whenever (and as long as) an error message file was created.

More details are available in section 3.7.

SJ

`SJ` *pgmnam* (SJ for *Submit Job*) allows you to execute the processing program *pgmnam* using previously defined option files. This command has to be used whenever SUBMIT (see Panel 0.1) is set to NO. If you change some information directly in the primary option input files (N- I- or F-files), you may also reprocess *pgmnam* using `SJ` (see also section 3.6.1).

3.6 Job Submission and Job Output Handling

3.6.1 Submit Jobs

The preparatory program *pgmnam*_P prepares the command file *pgmnam*`.BAT` (DOS) or *pgmnam*`.COM` (VMS and UNIX) containing the actual commands to properly call the processing program *pgmnam*. If SUBMIT (see Panel 0.1) is set to NO, the Menu System does not immediately execute this command file.

If the program crashes you might want to see the error messages again; or you would like to execute the program again after having "manually" changed some input parameters directly in the `N-`, `I-`, or `F`-files (something an advanced user might want to do!).

Under DOS you may directly execute the program anytime later by just typing *pgmnam*.

Under VMS and UNIX you execute the program by typing `SJ` *pgmnam*.

Why this difference? In the multitask Operating Systems VMS and UNIX you have the possibility to execute the processing programs in the background, selected by the JOBCLASS option in Menu 0.1, details see Section 3.5:

The command file *pgmnam*`.COM` is always created by the preparatory program irrespective of the current setting of the SUBMIT and JOBCLASS processing defaults. It contains the actual call of the executable file *pgmnam*`.EXE` (VMS) or *pgmnam* (UNIX) and the command to create a copy of the N-File in the current directory under the name of *pgmnam*`N` (which is *hard-wired* into all the processing programs!).

In order to execute *pgmnam*`.COM` properly, i.e. according to the JOBCLASS parameter, the Menu System prepares an additional command file *pgmnam*`.CTL` containing the proper commands. You call this file by typing:

`SJ` *pgmnam*

(VMS: `SJ` is a symbol that points to a command file `X:[VMS]SUBJOB.COM`, UNIX: `SJ` is an executable script file, see Section 3.11.3.)

Whenever SUBMIT is set to NO, the Menu System will remind you how to execute the processing program manually with a message in the message field of the menu panels.

3.6.2 Job Output

The primary job output may either be written to the standard output (which is the screen in interactive processing (JOB CLASS = 0) or the LOG file in VMS batch processing) or routed to a special job output file (which is the recommended practice). Values are defined Panel 0.1 or by PRCDEF.

The name of the job output is *pgmnam*`.L`*nn*, e.g. `GPSEST.L15`. *pgmnam* is the name of the processing program (e.g. `GPSEST`), and *nn* is a number between 00 and 99. The Menu System automatically increases the number within the valid range (it keeps track of the number to be used in the file *pgmnam*`.J` in the campaign `.OUT` subdirectory).

The job output of most of the programs are stored into the campaign `.OUT` subdirectory, i.e. they are campaign-dependent, and you may retain up to 100 job outputs per program and campaign. (Very few programs store their output into a directory that may be selected by Menu 0.1).

If 100 different job outputs are not enough you could ask the Menu System to use three digits *nnn* in the job output filename extension (also in Menu 0.1). The filename will then look like *pgmnam.nnn.*

Menu 5.9 or `JOB` allows you to handle the job output of the processing programs (browse, edit, print, copy to another file, delete). You may also reset the job output counter to any valid number by the same program.

3.7 Error Handling

3.7.1 Error Message File

Most processing programs may generate error messages and warnings that are stored to their SYSERR output file. SYSERR is the internal name *hardwired* in the processing programs. The actual (external) name of the error message file is defined in Panel 0.3.1, usually the name is `U:[WORK]ERROR.MSG`. The same Panel 0.3.1 allows you to specify whether you really want to store the messages into a file or whether you prefer to have them included into the standard job output (background operation only, in foreground the messages are *always* written to the special file).

The Menu System displays a message in the message field of the program panels, whenever (and as long as) this error message file exists.

You may look at and delete the error message file using the commands `ERR` and `ERRDEL` respectively (see Section 3.5).

3.7.2 Return Codes

The Fortran subroutine EXITRC calls, depending on the Operating System and where available, a routine EXIT with an exit status value that may signal to the Operating System an error condition. More details may be found as comments in the source code of EXITRC and also in the Fortran Users Guides for the use of EXIT and its parameter.

These error conditions could be used and tested in command procedures for the semi-automated processing (see Section 3.8). They are extensively used by the Bernese Processing Engine BPE (see Chapter 21).

3.8 Semi-Automated Processing

Usually the processing follows the same pattern. In the case of a single session we distinguish the following steps:

- Import of the date (RINEX –> Bernese),
- Import of the orbits (precise –> standard),
- Computation of receiver clocks,
- Network definition (–> single difference file creation),
- Check of phase data (cycle slip detection and repair),
- Double-difference processing,
- Interpretation of the results.

Every single step usually consists of two parts: The preparation of the major input files through interactive option and file selection, and the execution of the corresponding processing programs.

The majority of the selected options in the data panels do not change from one session to the next. What *is* changing is the session number, hence the names of the observation and orbit files.

Provided there are no major decisions to be taken between the steps, an automated sequencing through the steps of e.g. one session is fairly easy to realize:

(1) We have to prepare the data panels beforehand.

(2) We have to provide the possibility to introduce certain parts of the data panel input fields as variable parameters, the values of which are determined at run time only ($\rightarrow$ *filename parameters*, see below).

(3) We have to provide the possibility to run the preparatory programs without actual user interaction.

(4) We have to prepare special command files that contain the calls to both the preparatory programs and the processing programs in the proper order.

Ad (1):

The easiest way to prepare the panels properly is to actually run a sample session. In order to prevent overwriting the prepared panels it might be a good practice to save these panel files into a separate directory and to copy them into the user directory `U:[PAN]` as part of the automated procedure.

Ad (2):

All those input fields that ask for file names (and some others as well, see help panel of Menu 1.5.1) accept the so-called filename parameters, i.e. a code starting with a dollar sign, e.g. `$CD1`. These codes are place holders, have the same length (i.e. number of characters) as the values they stand for, and they will be replaced by the actual values at run time.

The values are defined in Menu 1.5.1, either interactively or by means of an auxiliary program PUTKEYW, see below.

Example: `????$SS1`

The four question marks (VMS: percent signs) stand for any four characters, e.g. the code for the stations in observation files. The filename parameter `$SS1` will be replaced by the session number, previously defined in Menu 1.5.1. When calling the preparatory program interactively you would get a file selection table containing all existing observation files of the current campaign and the defined session, i.e. all the files you actually want to process.

Some of the parameters allow the assignment of up to three different values (e.g. $SS1 may have the values 4321 and 4323 and 4325, so that the files of these three sessions would be selected), some of the parameters allow for ranges (4325 + 4 - 3, i.e. all sessions between 4322 and 4329), details are found in the help panel of menu Menu 1.5.1.

Ad (3):

The subroutine DSPPAN that displays the data and file selection panels determines the current mode of operation by checking the existence of a "flag file" `U:[WORK]IAMODE.`*proc_id*. This file has to be created (*and deleted!)* by the general command file (see item (4)). *proc_id* is the current process identification under VMS (symbol define by `LOADGPS`) to allow for more than one simultaneous automated processing (which has to be handled *very* carefully to prevent undesired interferences through panel or other files!). Under UNIX and DOS *proc_id* is set blank.

Whenever a file selection panel has to be prepared by the Menu System in the non-interactive mode, it will automatically select all the files contained in the selection list. Therefore you have to use a combination of wildcard characters (* and ?) and filename parameters to get the correct automatic file selection.

Ad (4):

The sample command file

`X:[EXE]AUTOSESS.COM / $X/EXE/AUTOSESS / X:\EXE\AUTOSESS.BAT`

shows the major steps to be performed for the automated processing of one session.

The program PUTKEYW, called by the command file `X:[EXE]PUTKEYW.COM`, etc., allows to directly write an option value into a specific input field. It reads from the standard input in a first record the panel file name and in the second record the value and the keyword name of the input field (i.e. the name to be found in the panel file in columns 81ff.). All inputs are non-quoted ASCII strings, blanks act as delimiters.

The Bernese Processing Engine BPE (see Chapter 21) makes extensive use of these automation techniques.

3.9 Calling Programs Without the Menu System

There are some processing programs not included in the Menu System, such as the simulation program GPSSIM. Of course you may still run these programs, but you have to prepare their primary options files (the I-, N- and F-files) manually. There is a command procedure RUNGPS available (under VMS the symbol `RUNGPS` calls the command file `X:[EXE]RUNGPS.COM`), that displays the proper I-, N-, and F-file using the standard editor for modification, copies them to the working directory, and finally calls the processing program.

Example: `RUNGPS GPSSIM`

Sample I-, N-, and F-files for these programs are available in the directory `X:[INX]`, etc, under the names *pgmnam*`N.INP`, *pgmnam*`I.INP`, and *pgmnam*`F.INP`, respectively. You have to copy them to `U:[INP]` etc.

In order to merely execute a preparatory program *pgmnam*`_P` or a processing program without any additional file copying etc. you may use the command procedures (VMS: symbols) `RMENU` and `RMAIN`, respectively.

Examples: `RMENU GPSEST_P`, `RMAIN GPSEST`

The former command should be used at the *Operating System prompt* if the preparatory program runs into a non-recoverable execution error to better see the system error messages.

The latter command might be used to run a processing program in the foreground to observe any runtime system error messages.

3.10 User-Specific Additions to the Menu System

With the exception of the primary program panel *selection 0* is never used by the Menu System. Therefore the user has the possibility to add his own sublevels to any one of the program panels. The escape to user program panels from the primary panel goes through selection 9.

These are the steps you have to follow to add user-specific program panels and programs:

(1) Add the new selection (0) into the program panel of your choice to escape to a new sublevel.

(2) Prepare a new program panel, the name of which is `PAN`*ijk*`0_.PAN` in the directory `U:[PAN]` etc, if you added the selection to panel `PAN`*ijk*`__.PAN`. Please keep in mind that there are 5 possible levels for both the program and the data panels. The easiest way to prepare a new panel is to copy an existing one to the new name and modify it accord to your task.

(3) Include into the new panel any selections according to your needs. Make sure that the columns 81ff. are empty if the selection points to a new program panel or that they contain the name of and the path to the program to be called (see below for details).

(4) Put your program executables into the proper directories.

(5) After successful tests copy the new panels from `U:[PAN]` to `X:[PAN]` and, if appropriate, to other user panel directories.

We recommend to escape to user program panels and programs through selection 9 of the primary program panel only, because user additions to existing panels will not be preserved after version updates. In any case you should carefully backup your own program panels before any version updates.

Let us give here some more rules and details to be observed (see also Section 3.3.1 and Figures 3.4 and 3.5):

Program Calls

Columns 81 to 88 contain the name of the program that is to be called by the Menu System (usually the preparatory program). The default path to the executable is defined in `X:\SKL\MENU.OPT`.

Blanks in columns 81 to 88 indicate that there is no program to be called but a program panel of the next deeper level is to be displayed.

Individual Paths

Columns 91 to 125 may contain an individual path to the executable, overriding the default in `MENU.OPT`. The current Menu System version does not make use of these individual paths. So, most program panels do not contain information in columns 90 to 127.

Addition of Non-Lahey Programs (DOS) or Calls with Symbols (VMS)

An exclamation character (!) in column 80 tells the program to send whatever text there is in cols. 81 to 88 to the Operating System, preceeded by any individual path found in the same line in cols. 91 to 125. Calls to programs or command files using predefined symbols (VMS) or execution of .COM or .EXE files in DOS may be included this way into the Menu System.

Calls to batch files (.BAT in DOS), command files (.COM in VMS) or script files (no extension in UNIX) are performed by putting an at sign (@) into col. 80. Either the individual path (if found) or the default path defined in `MENU.OPT` will be used to call the file defined in cols. 81 to 88 (preceeded by the proper command CALL in DOS or "@" in VMS).

Display of Data Panels

A percent character (%) in col. 80 tells the MENU program to call a special program to display the data panel the name of which is stored in cols. 81 to 88. The name of the display program (DSPDPAN) is specified in `MENU.OPT`. This option is mainly used in Menu 0.

An "&" in col. 80 tells the MENU program to call a special program to display the data panel the name of which is stored in cols. 81 to 88. The name of the display program (UPDTPAN) is given in `MENU.OPT`. The difference to the % in col.80 is: Here, it is possible to repeat or delete lines in the data panel, used e.g. to add or remove sessions in the session definition panel in Menu 1.3.

3.11 Technical Details

3.11.1 Command files *pgmnam*.BAT (DOS); *pgmnam*.COM (VMS and UNIX)

The command file *pgmnam*`.BAT` or *pgmnam*`.COM`, which is created by the preparatory program *pgmnam*_P irrespective of the current setting of the SUBMIT and JOBCLASS processing defaults, contains the actual call of the executable file *pgmnam*`.EXP` / *pgmnam*`.EXE` / *pgmnam* and the command to create a copy of the N-File in the current directory under the name of *pgmnam*`N` (which is hardwired into all the processing programs!).

The command file also deletes the auxiliary file *pgmnam*`.SCR` if it already exists. At runtime LAHEY would run into an error if a previously existing auxiliary file that had been written with formatted records would now be opened for unformatted access (with `STATUS='UNKNOWN'`).

Examples for such a command file to call a processing program are given Figures 3.10 (DOS), 3.11 (VMS), and 3.12 (UNIX).

```
@ECHO OFF
IF EXIST X:\OPT\DEFSTD.SCR DEL X:\OPT\DEFSTD.SCR >NUL:
COPY X:\INP\DEFSTDN.INP DEFSTDN >NUL:
RUN386 C:\PGM\MAIN40\EXP\DEFSTD
IF EXIST X:\OPT\DEFSTD.SCR DEL X:\OPT\DEFSTD.SCR >NUL:
DEL DEFSTDN >NUL:
```

Figure 3.10: DOS Example: `Y:DEFSTD.BAT`

```
$ IEXIT == "1"
$ SET DEFAULT U:[WORK]
$ COPY U:[INP]DEFSTDN.INP DEFSTDN.DAT
$ ASSIGN/USER_MODE SYS$COMMAND SYS$INPUT
$ R XG:DEFSTD.EXE
$ IF $SEVERITY .NE. 1 THEN IEXIT == "3"
$ DELETE DEFSTDN.DAT;
$ AUXFIL=F$SEARCH("U:[WORK]DEFSTD.SCR;*")
$ IF(AUXFIL .NES. "") THEN DELETE U:[WORK]DEFSTD.SCR;*
$ EXIT 'IEXIT'
```

Figure 3.11: VMS Example: `U:[WORK]DEFSTD.COM`

```
cd $2
cp U:/INP/DEFSTDN.INP DEFSTDN
echo "U:/WORK/DEFSTD.COM started at : `date`" >U:/WORK/DEFSTD.LOG
time C:/PGM/MAIN35/EXE/DEFSTD
echo "U:/WORK/DEFSTD.COM   ended at : `date`" >>U:/WORK/DEFSTD.LOG
echo >$1
tail -1 U:/WORK/DEFSTD.LOG >$1
if test -f U:/WORK/DEFSTD.SCR
then
  rm U:/WORK/DEFSTD.SCR
fi
```

Figure 3.12: UNIX Example: `$U/WORK/DEFSTD.COM`

3.11.2 Command file *pgmnam*.CTL

The VMS and UNIX versions (see subroutine `VMS_CREBAT.FOR`, `U_CREBAT.f)` additionally create a file *pgmnam*`.CTL` containing the commands to either call *pgmnam*`.COM` in the *foreground*:

VMS Example: `$ @U:[WORK]DEFSTD.COM`

UNIX Example:
```
cons=`tty`
sh U:/WORK/DEFSTD.COM ${cons} $U/WORK >>U:/WORK/DEFSTD.LOG &
```

or to submit the program to a batch queue into the *background*:

VMS Example:
```
$ SUBMIT/QUEUE='BATCH1'/NOPRINT/NOTIFY-
        /LOG=U:[WORK]DEFSTD.LOG U:[WORK]DEFSTD.COM
```

UNIX Example:
```
cons=`tty`
nohup nice -${PRIO1} sh U:/WORK/DEFSTD.COM ${cons} $U/WORK
                                >>U:/WORK:/DEFSTD.LOG &
```

according to the processing default in Menu 0.1.

3.11.3 Command file SUBJOB.COM (VMS); SJ (UNIX)

The *pgmnam*`.CTL` file may be executed manually by means of a symbol `SJ` (standing for "`@X:[EXE]SUBJOB` ") or a shell script `SJ` (for UNIX), e.g.: `SJ DEFSTD`.

VMS Example: `$ @U:[WORK]'P1'.CTL`

UNIX Example:
```
cd $U/WORK
sh $1.CTL
```

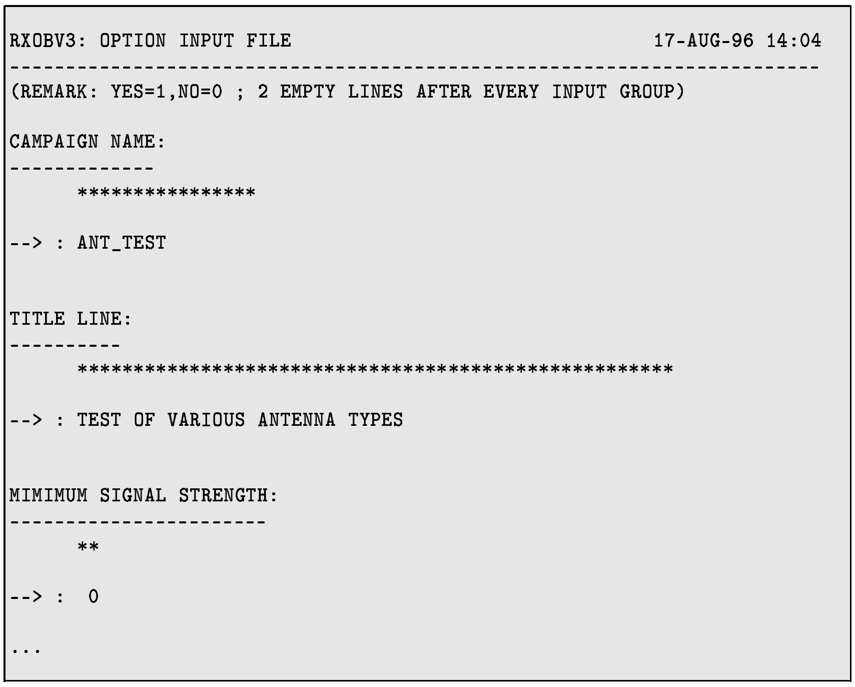
3.11.4 Skeleton Files

The I- and F-files are pure ASCII files with a fairly complicated structure. Especially the I-files contain a lot of explanatory text. The properly formatted READ statements for these files are coded in the processing programs. In order *not* to have to depend too much on the actual format of these files in the coding of the preparatory programs we use so-called skeleton files (to be found in the directory `X:[SKL]`, etc) containing the proper structure with all the explanatory text. The fields where the actual data has to be written, are indicated with strings of percent characters. A subroutine (FILSKL) takes the proper skeleton panel, fills in the actual options values and creates the corresponding options file.

The directory `X:[SKL]`, etc also contains similar skeleton files used for the run time creation of some data input panels. *The skeleton files are not to be modified!*

```
RXOBV3: OPTION INPUT FILE                                %%%%%%%%%%%%%%%
------------------------------------------------------------------------
(REMARK: YES=1,NO=0 ; 2 EMPTY LINES AFTER EVERY INPUT GROUP)

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

--> : %%%%%%%%%%%%%%%%


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

--> : %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


MIMIMUM SIGNAL STRENGTH:
-----------------------
      **

--> : %%

...
```

Figure 3.13: Sample Skeleton File for I-File (extract)



```
RXOBV3: OPTION INPUT FILE                                17-AUG-96 14:04
------------------------------------------------------------------------
(REMARK: YES=1,NO=0 ; 2 EMPTY LINES AFTER EVERY INPUT GROUP)

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

--> : ANT_TEST


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

--> : TEST OF VARIOUS ANTENNA TYPES


MIMIMUM SIGNAL STRENGTH:
-----------------------
      **

--> :  0

...
```

Figure 3.14: Sample I-File (extract)