ppm

Precise Positioning Management

20xx

GNSS SENSOR SERIES REFERENCE MANUAL

English

Copyright

Printed in Germany.
No: 20xx GNSS sensor manual english
February 2013

Trademarks

PPM Products Limited Warrenty

All ppm global positioning system (GPS) receivers are navigation aids, and are not intended to replace other methods of navigation. Purchaser is advised to perform careful position charting and use good judgment.

READ THE USER GUIDE CAREFULLY BEFORE USING THE PRODUCT.

1. PPM WARRANTY

ppm warrants their GPS receivers and hardware accessories to be free of defects in material and workmanship and will conform to our published specifications for the product for a period of one year from the date of original purchase or such longer period as required by law.

THIS WARRANTY APPLIES ONLY TO THE ORIGINAL PURCHASER OF THIS PRODUCT.

In the event of a defect,ppm will, at its option, repair or replace the hardware product with no charge to the purchaser for parts or labor. The repaired or replaced product will be warranted for 90 days from the date of return shipment, or for the balance of the original warranty, whichever is longer. ppm warrants that software products or software included in hardware products will be free from defects in the media for a period of 30 days from the date of shipment and will substantially conform to the then-current user documentation provided with the software (including updates thereto). ppm's sole obligation shall be the correction or replacement of the media or the software so that it will substantially conform to the then- current user documentation.

ppm does not warrant the software will meet purchaser's requirements or that its operation will be uninterrupted, error-free or virus-free.

2. PURCHASER'S REMEDY

PURCHASER'S EXCLUSIVE REMEDY UNDER THIS WRITTEN WARRANTY OR ANY IMPLIED WARRANTY SHALL BE LIMITED TO THE REPAIR OR REPLACEMENT, AT PPM'S OPTION, OF ANY DEFECTIVE PART OF THE RECEIVER OR ACCESSORIES WHICH ARE COVERED BY THIS WARRANTY.

REPAIRS UNDER THIS WARRANTY SHALL ONLY BE MADE AT AN AUTHORIZED PPM SERVICE CENTER. ANY REPAIRS BY A SERVICE CENTER NOT AUTHORIZED BY PPM WILL VOID THIS WARRANTY.

3. PURCHASER'S DUTIES

To obtain service, contact and return the product with a copy of the original sales receipt to the dealer from whom you purchased the product.

ppm reserves the right to refuse to provide service free-of-charge if the sales receipt is not provided or if the information contained in it is incomplete or illegible or if the serial number is altered or removed. ppm will not be responsible for any losses or damage to the product incurred while the product is in transit or is being shipped for repair. Insurance is recommended. ppm suggests using a trackable shipping method such as UPS or FedEx when returning a product for service.Purchaser assumes the entire risk of using the software.

4. LIMITATION OF IMPLIED WARRANTIES

EXCEPT AS SET FORTH IN ITEM 1 ABOVE, ALL OTHER EXPRESSED OR IMPLIED WARRANTIES, INCLUDING THOSE OF FITNESS FOR ANY PARTICULAR PURPOSE OR MERCHANTABILITY, ARE HEREBY DISCLAIMED AND IF APPLICABLE, IMPLIED WARRANTIES UNDER ARTICLE 35 OF THE UNITED NATIONS CONVENTION ON CONTRACTS FOR THE INTERNATIONAL SALE OF GOODS.

Some national, state, or local laws do not allow limitations on implied warranty or how long an implied warranty lasts, so the above limitation may not apply to you.

5. EXCLUSIONS

The following are excluded from the warranty coverage:
(1) periodic maintenance and repair or replacement of parts due to normal wear and tear;
(2) batteries;
(3) finishes;
(4) installations or defects resulting from installation;
(5) any damage caused by
(i) shipping, misuse, abuse, negligence, tampering, or improper use;
(ii) disasters such as fire, flood, wind, and lightning;
(iii) unauthorized attachments or modification;
(6) service performed or attempted by anyone other than an authorized ppm Service Center;
(7) any product, components or parts not manufactured by ppm,
(8) that the receiver will be free from any claim for infringement of any patent, trademark, copyright or other proprietary right, including trade secrets
(9) any damage due to accident, resulting from inaccurate satellite transmissions. Inaccurate transmissions can occur due to changes in the position,health or geometry of a satellite or modifications to the receiver that may be required due to any change in the GPS. (Note: ppm GPS receivers use GPS or GPS+GLONASS to obtain position, velocity and time information. GPS is operated by the U.S. Government and GLONASS is the Global Navigation Satellite System of the Russian Federation, which are solely responsible for the accuracy and maintenance of their systems. Certain conditions can cause inaccuracies which could require modifications to the receiver. Examples of such conditions include but are not limited to changes in the GPS or GLONASS transmission.). Opening, dismantling or repairing of this product by anyone other than an authorized ppm Service Center will void this warranty.

6. EXCLUSION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES

PPM SHALL NOT BE LIABLE TO PURCHASER OR ANY OTHER PERSON FOR ANY INDIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO LOST PROFITS, DAMAGES RESULTING FROM DELAY OR LOSS OF USE, LOSS OF OR DAMAGES ARISING OUT OF BREACH OF THIS WARRANTY OR ANY IMPLIED WARRANTY EVEN THOUGH CAUSED BY NEGLIGENCE OR OTHER FAULT OF PPM OR NEGLIGENT USAGE OF THE PRODUCT. IN NO EVENT WILL PPM BE RESPONSIBLE FOR SUCH DAMAGES, EVEN IF PPM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
Some national, state, or local laws do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

7. COMPLETE AGREEMENT

This written warranty is the complete, final and exclusive agreement between ppm and the purchaser with respect to the quality of performance of the goods and any and all warranties and representations.
THIS WARRANTY SETS FORTH ALL OF PPM‘S RESPONSIBILITIES REGARDING THIS PRODUCT.
THIS WARRANTY GIVES YOU SPECIFIC RIGHTS. YOU MAY HAVE OTHER RIGHTS WHICH VARY FROM LOCALITY TO LOCALITY (including Directive 1999/44/EC in the EC Member States) AND CERTAIN LIMITATIONS CONTAINED IN THIS WARRANTY MAY NOT APPLY TO YOU.

8. CHOICE OF LAW.

This limited warranty is governed by the laws of Germany, without reference to its conflict of law provisions or the U.N. Convention on Contracts for the International Sale of Goods, and shall benefit ppm, its successors and assigns.
THIS WARRANTY DOES NOT AFFECT THE CUSTOMER'S STATUTORY RIGHTS UNDER APPLICABLE LAWS IN FORCE IN THEIR LOCALITY, NOR THE CUSTOMER'S RIGHTS AGAINST THE DEALER ARISING FROM THEIR SALES/PURCHASE CONTRACT.
Furtherher information concerning this limited warranty,
please call or write:

ppm GmbH
Grube 39a
D-82377 Penzberg
Phone: +49-8856-8030980
Mail: info@ppmgmbh.com

Introduction

Getting to know the 20xx GNSS receiver

Congratulations!
You have just acquired a new GNSS receiver from ppm!

The 20xx GNSS receiver is designed for precision GNSS applications. To facilitate additional calculations, e.g. in post-processing, any GNSS data, raw data and/or position data (e.g. NMEA) can be written to the internal SD memory.
The SD card is accessed using an USB port on the 20xx GNSS receiver; it shows up as a removable drive on your PC. For real-time applications GNSS data can be output on a selected port or written to a USB thumb drive. Corrections for DGNSS and RTK are available using the internal GSM radio (including Ntrip) or using an external device connected to a serial port.

The large number of ports (RS-232, USB, Event input, PPS output) allow to flexibly adapt the unit for any application. You can use it in vehicle dynamics, machine control, geomonitoring, etc.

A bi color LED shows the current status of the 20xx GNSS receiver.

The 20xx GNSS receiver supports custom configurations. These configurations are created on a PC and are transmitted to the unit using the USB port. Firmware updates can be installed using a serial port or by FTP connection to the unit.

Scope of this manual

This manual will help you to get familiar with your new GPS. We will accompany you, from unpacking to first use. This will enable you to quickly and successfully use the system.

Are you missing some piece of information? Do you have a comment regarding this manual?
Please provide your feedback to us by sending it to the following e-mail address using the reference "20xx manual":

info@ppmgmbh.de

Thank you!

The system

English

The table below provides an overview of the different 20xx GNSS receiver versions.

Model variants

Part number	Description	
	20xx L series	
ppm2011-L11	GPS L1 receiver	(NovAtel OEMStar)
ppm2011-L13	GPS+GLONASS L1 receiver	(NovAtel OEMStar)
ppm2022-L11	GPS L1 receiver	(NovAtel OEM615)
ppm2022-L12	GPS L1/L2 receiver	(NovAtel OEM615)
ppm2022-L13	GPS+GLONASS L1 receiver	(NovAtel OEM615)
ppm2022-L14	GPS+GLONASS L1/L2 receiver	(NovAtel OEM615)
ppm2022-L01	GPS L1 receiver	(Ashtech MB100)
ppm2022-L02	GPS L1/L2 receiver	(Ashtech MB100)
ppm2022-L03	GPS+GLONASS L1 receiver	(Ashtech MB100)
ppm2022-L04	GPS+GLONASS+Galileo L1/L2 receiver	(Ashtech MB800)
ppm2022-L24	GPS+GLONASS L1/L2 receiver	(Septentrio AsteRx2e)
	20xx S series (with internal memory)	
ppm2011-S11	GPS L1 receiver	(NovAtel OEMStar)
ppm2011-S13	GPS+GLONASS L1 receiver	(NovAtel OEMStar)
ppm2022-S11	GPS L1 receiver	(NovAtel OEM615)
ppm2022-S12	GPS L1/L2 receiver	(NovAtel OEM615)
ppm2022-S13	GPS+GLONASS L1 receiver	(NovAtel OEM615)
ppm2022-S14	GPS+GLONASS L1/L2 receiver	(NovAtel OEM615)
ppm2022-S01	GPS L1 receiver	(Ashtech MB100)
ppm2022-S02	GPS L1/L2 receiver	(Ashtech MB100)
ppm2022-S03	GPS+GLONASS L1 receiver	(Ashtech MB100)
ppm2022-S04	GPS+GLONASS+Galileo L1/L2 receiver	(Ashtech MB800)

English

Description of equipment 20xx -S

POWER + I/Os: System connector, SUB-D25.
The connecting cable 730124-X (available separately) offers 3 serial connectors (SUB-D9) and one power connector.
Once you connect the receiver to a power supply (9 to 32V) it boots up.
The pin assignment of the SUB-D25 is shown in the table on page 5.

USB: Mini-USB port for connecting to a PC, two features:

1. USB drive mode, enables data transfer.
   This mode is used to access the internal SD memory card.
2. USB communication mode.
   This mode is used to directly communicate with the internal GNSS board from the PC using USB.

GPS: TNC connector, used to connect a GPS antenna cable.

GSM: SMA connector, used to connect an external GSM antenna to the optional GSM radio.

Status LED:		
	GREEN blinking	system start-up
	RED blinking	no GNSS fix or GNSS fix, but no data is saved
	GREEN blinking	GNSS fix, data is saved
	quick, red blinking	read and write access to internal memory

Description of equipment 20xx -L

POWER + I/Os: System connector, SUB-D25.
The connecting cable 730124-X (available separately) offers 3 serial connectors (SUB-D9) and one power connector.
Once you connect the receiver to a power supply (9 to 32V) it boots up.
The pin assignment of the SUB-D25 is shown in the table on page 5.

USB: Mini-USB port for connecting to a PC, one function:

USB communication mode.
This mode is used to directly communicate with the internal GNSS board from the PC using USB.

GPS: TNC connector, used to connect a GPS antenna cable.

Status LED: GREEN blinking system start-up

GREEN blinking GNSS fix

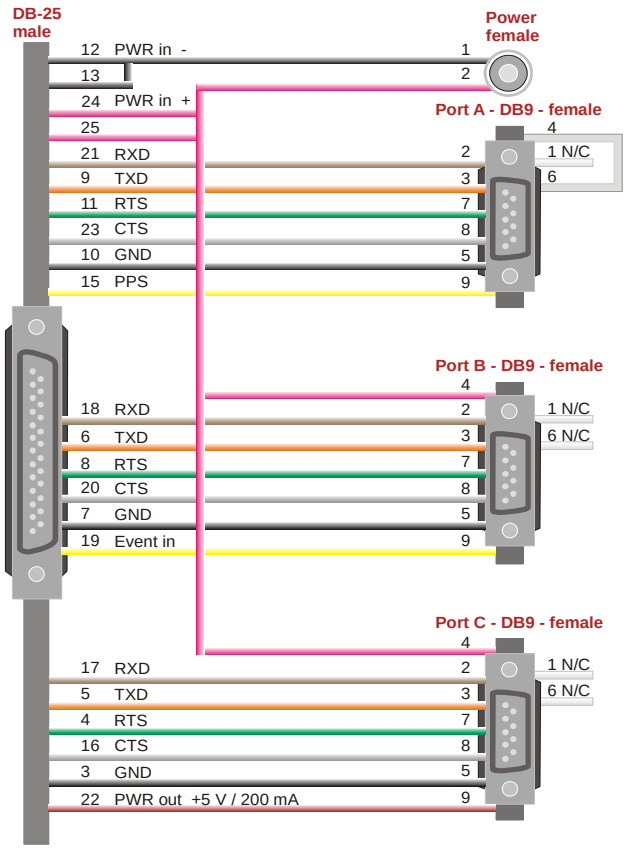
English

System connector 20xxS + 20xxL

RTS & CTS are internally connected for compatibility reasons.

n.c.	1	14	n.c.
GND	2	15	PPS out (3 Volts – 3 LV TTL)
GND	3	16	Com C CTS
Com C RTS	4	17	Com C RXD
Com C TXD	5	18	Com B RXD
Com B TXD	6	19	Event-in (3 Volts – 3LV TTL)
GND	7	20	Com B CTS
Com B RTS	8	21	Com A RXD
Com A TXD	9	22	n.c. L model: Power out 5V / 200mA
GND	10	23	Com A CTS
Com A RTS	11	24	Power in+ (10 to 32 VDC)
Power in– (GND)	12	25	Power in+ (10 to 32 VDC)
Power in– (GND)	13		

Optional system cable 730124-X



Back panel of 20xxS sensor

The illustration shows the optional 6pin connector used for digital I/Os

The back panel gives access to the optional GSM/GPRS radio's SIM module.

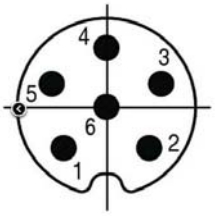
The SIM holder is protected by a water- and dust-proof cap.

SIM card

The illustration shows the SIM card ready for insertion:
Contacts on top, chamfered corner to the right. Press the SIM card into the holder to lock it in place. Press again. The card is unlocked.
Use tweezers to take the SIM card out completely.

The optional 6 pin connector on the right is used for digital I/Os.
(Binder Male socket front fastened dip 6-way PN: 09-3463-81-06)

The pin assignment is as follows:

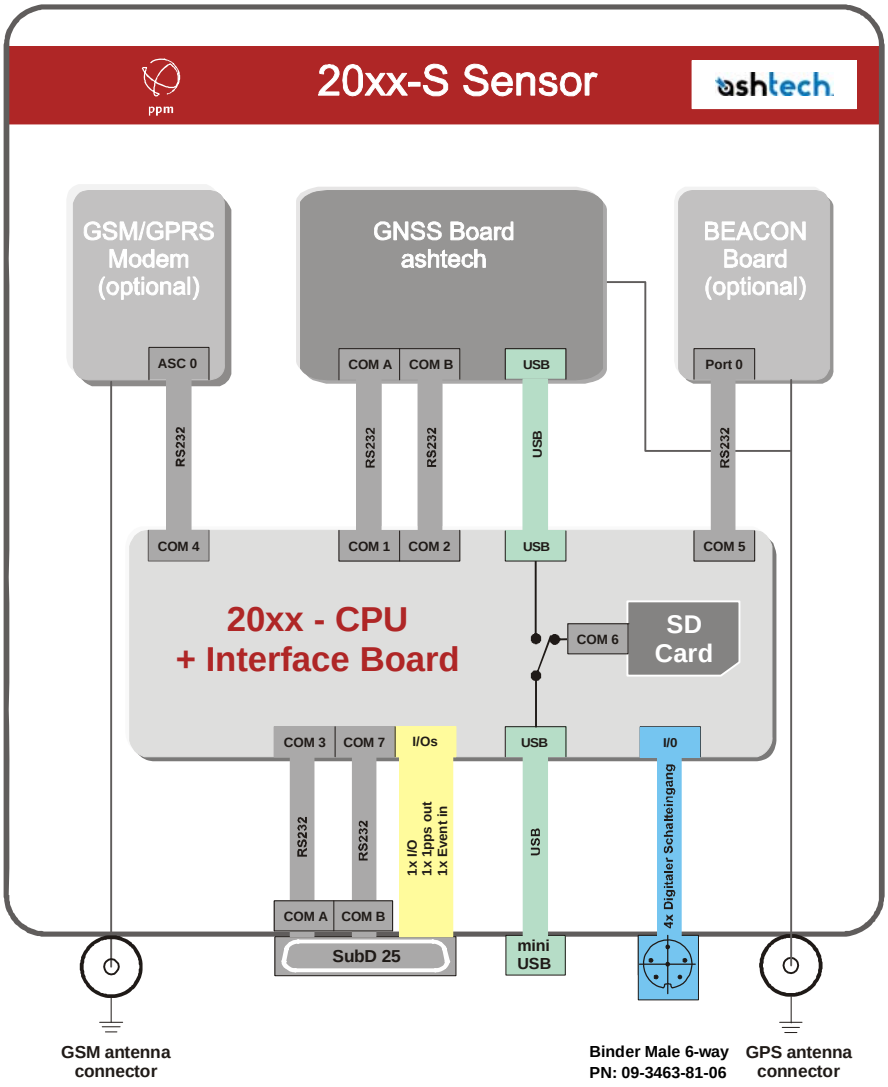


Pin	Description
1	can be monitored from ppmOS using peb.pf4
2	can be monitored from ppmOS using peb.pf5
3	can be monitored from ppmOS using peb.pf6
4	can be monitored from ppmOS using peb.pf7
5	can be monitored from ppmOS using peb.pf8
6	not used

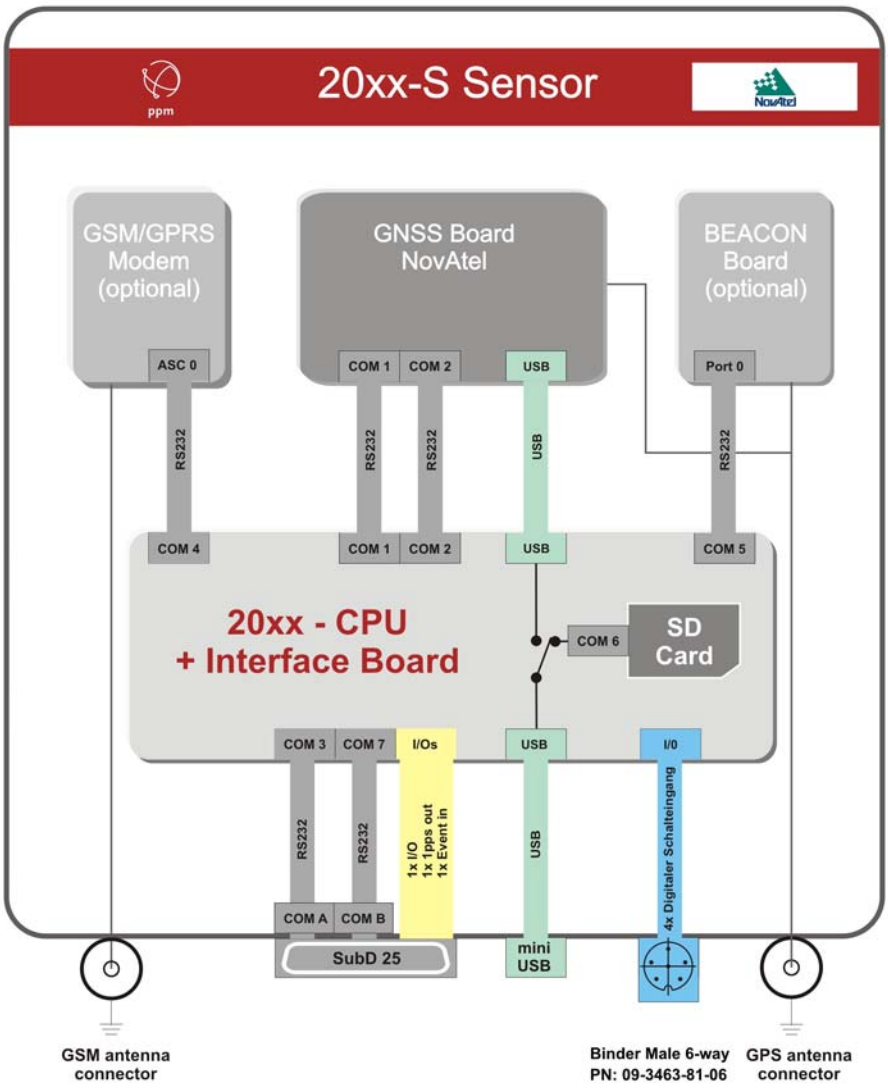
See „Script files“ on page 18 more information on using this connector.

English

Block diagramm Ashtech



Block diagram NovAtel



English

Power supply

Depending on the application, the 20xx GNSS receiver can connect to a power supply or a battery. The input range is 9 to 32 Volts.

Assignment on the system connector:

PIN 12+13: EXT_GROUND
(Power in– can be connected to GNS)

PIN 24+25: Power in + (+9 VDC to 32 VDC)

FUSE

We strongly encourage you to add a fuse to the power supply line.
The recommended fuse is slow-blow fuse, 1 Amp.

Optional accessories

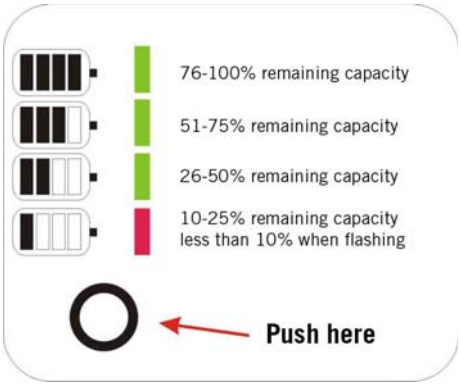
The following batteries are recommended by ppm GmbH (available separately):

BATTERY:

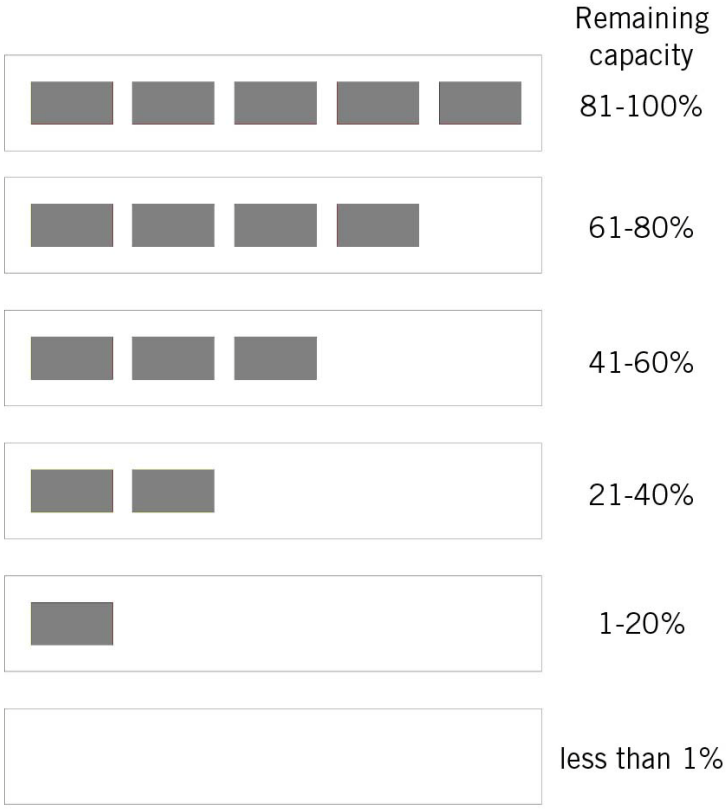
ppm GmbH recommends and sells lithium-ion batteries with varying capacity. These allow powering the full system for more than 24 hours.

The remaining capacity can be checked directly on the battery. Simply press the button.

The following indicators are used:



The 205x battery shows the capacity on a front-side display.



English

Power supply

BATTERY CHARGER:

The battery charger comprises the following components:

- Charging cradle, used to charge and calibrate the battery
- 3 plastics adapters, used to charge different types of batteries
  Use only the widest adapter.
- 1 power supply unit, 230 Volts, output 24 Volts, 2.5 Amp.
- IEC 60320-1 (C13) cable

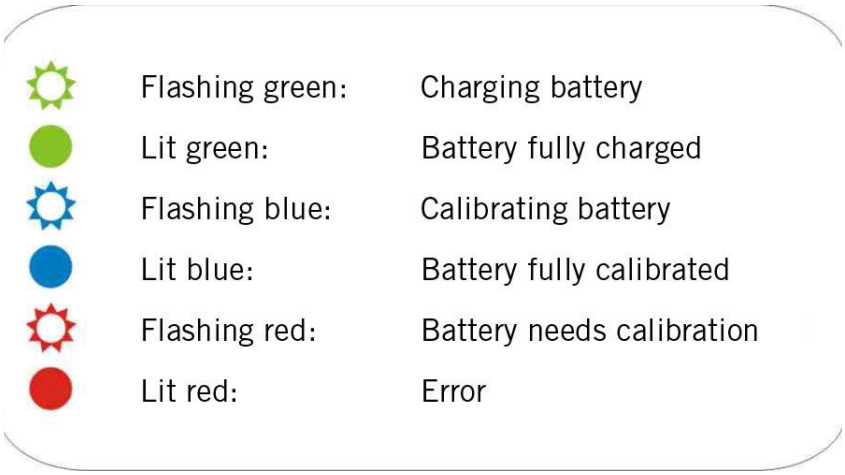
To charge, push the battery (contacts first) into the charging cradle. Plug the power cable into the cradle and the socket.

Charge duration:

Charge duration varies depending on the battery's residual capacity. The battery is equipped with an SMB (Smart Battery System). It will communicate with the charger and provide internally saved parameters to it. Thus, the charger can determine the optimal current and type for charging.
Charging a fully depleted battery takes approximately 4 hours.

Calibration:

The charger can be used to re-calibrate the battery. We recommend re-calibrating batteries once a month. To do so, insert the battery and then simply press the push button in front of the slot. Calibrating takes 12 to 15 hours.
The following LED indicators are available:



Safety notices on power supply

- Do not operate charging cradle or PSU near fluids or water. The enclosures are not waterproof.
- Do not open charging cradle and PSU. They contain no user-serviceable parts.
- Do not cover the ventilation openings. Keep the area behind the ventilation free. Otherwise the charger could overheat.
- Use only the provided PSU.
- Do not use or store the charger in places that can get hot.
- Note that the charger can be become very warm during calibration.

Using your 20xx GNSS receiver

The 20xx GNSS receiver is suitable for a wide range of applications, allowing a flexible and tailored configuration. Depending on the installed hardware and firmware options, you can configure several modes of operation.

To control those options and settings, every 20xx series receiver comes equipped with the **ppmOS** operating system.

ppmOS supports free programming and control of all functionality supported by the installed hardware and firmware.

With the system, configuration files and script files are differentiated.
Configuration files are used to store hardware settings; script files contain workflows and procedures.

Operating system ppmOS

ppmOS is an up-to-date, flexible operating system developed by PPM GmbH. It allows to run multi-tasking applications on simple microprocessors.
Within your 20xx GNSS sensor ppmOS is used to configure and control all system components.

Internal SD card memory

ppmOS supports SD memory cards using the two file systems FAT16 and FAT32.

Systems supplied with an internal 2 GB SD card by default use FAT16. The optional 8 GB SD memory card uses FAT32. No special requirements exist with regards to cluster size.
The internal SD card is designated as drive **c:**. If in USB drive mode, it can be formatted using a PC. Files stored on the internal SD card are limited to 8 characters for the file name and 3 characters for the extension.
File and folder names are not case-sensitive. The slash (/) is used as the delimiter in path names. By default, the following folders exist in ppmOS:

c:/sys
c:/data
c:/sms

You can create any number of additional folders and save data in them.

Control and configuration files

Sub-directory SYS

The files to control and configure the receiver must be placed in **c:/sys**.

Upon system boot, the 20xx GNSS sensor looks in **c:/sys** and runs specific script files in a pre-defined way. During this process system specific configuration files are read.

Configuration files

A maximum of 5 different configuration files is supported:

'**gps.cfg**', '**bestpos.cfg**', '**upload.cfg**', '**crontab.cfg**', and '**Ntrip.cfg**'.

Each of those configuration files is described in detail in the next section, 'Configuration files'.

The folder **c:/sys** includes all the executable script files as well.
Script files allow for time and event controlled functions and processes. Script files are handled in more detail in the section 'Script files'.

Sub-directory DATA:

Position and raw data is saved in the **c:/data** folder. Additional sub-directories are created for this data at runtime. These structure the logging into folders for year, month and day (e.g. c:/data/2011/09/30).

Sub-directory SMS:

SMS messages (texts) are saved in the **c:/sms** folder.

Information in configuration and script files is shown by line. Lines need to be ended using **LF** (0xa) or **CR LF** (0xd 0xa).
Empty lines and lines beginning with a hash (#) are ignored.
The **#** is used to introduce comments and notes. This allows you to show any information within the file itself or exclude a line from processing during testing (or if it is not needed constantly).

The key word ***exec*** at the start of a line takes the remainder of the line and runs it as a command.
This allows you to run any of the commands supported by ppmOS directly from the configuration file.

Configuration files

gps.cfg

This file lists all the commands required to configure the GNSS card within the 20xx GNSS receiver for the user-defined application.
See the GNSS manufacturer's manual for command structure and parameters. For example, you can determine if position output is in binary or an ASCII/NMEA format. Or you can set the output rate for PVT and raw data. Depending on the GNSS card used the maximum data rate is 50 Hz (50 position samples per second). Furthermore, you can select a setting for the Event-in lines.

Any periodic output signal (PPS pulse) is set here as well. The file is used to configure data output to a port besides logging it to the memory, too.

bestpos.cfg

This file contains the configuration for triggers, output and/or logging of the different measurements. It is required for any intermittent (long gaps) or event-triggered position outputs. These are used in geo-monitoring and other applications.

Examples

For NovAtel cards:

```
log com2 bestposb once
```

For Ashtech cards:

```
$PASHQ,POS,B
```

You can use ***exec*** lines in this file as well. Any line starting with `exec` is handed over to the command interpreter (shell) and executed immediately.

upload.cfg

Configuration file to connect to an FTP server via the cellular network

The internal GSM/GPRS radio modem (option) can be used to establish an Internet connection using the GSM/GPRS network. You will need a data-enabled SIM card and the log-in data for SIM card and FTP server (*not included*).

Simply enter the SIM PIN, the provider APN, the user name and – if required – the password in upload.cfg.

Data transfer uses the FTP protocol to facilitate transfers to FTP servers.
User name, password and the server's address are presented in the file using the keyword ***server***.
The server address must be entered as an IP address (four three-figure numbers, delimiter is dot).
You **must not** enter a URL (textual Internet address).

Example for upload.cfg

If you are using a SIM card issued by T-Mobile for use in Germany, the file will look like this:

```
UploadConfigFile v 0.1

#SIM card pin
PIN    1234

#Dial-in
apn    internet.t-mobile
user   td1
passwd td1

#FTP Server
server user_name:password@111.222.123.321
```

GPRS log-in credentials (as of December 2011)

Provider	State	APN Server	User	PW
T-Mobile	D	internet.t-mobile	td1	td1
Vodafone	D	web.vodafone.de		
O2	D	internet		
E-Plus	D	internet.eplus.de	eplus	internet
T-Mobile	A	gprsinternet	gprs	
Mobilkom A1	A	a1.net	ppp@a1plus.at	ppp
Orange Austria	A	web.one.at	web	web
Telering	A	web	web@telering.at	web
3-Drei-Hutchinson	A	drei.at		
Swisscom	CH	gprs.swisscom.ch	gprs	gprs
Orange	CH	internet		
Sunrise	CH	internet		

(without guarantee; please contact your provider for up-to-date information)

ntrip.cfg

Configuration file for Ntrip connections

The internal GSM/GPRS radio does support Ntrip connections.
Ntrip is a protocol used by several organizations (e.g. APOS, ASCOS, SAPOS, SWIPOS, …) to provide RTCM corrections for real-time DGPS or RTK operations.

The file ntrip.cfg contains the following:

```
#Configuration file to open an Ntrip Caster connection
#Note: Lines starting with exec are forwarded to the shell for execution

exec @echo reading from ntrip.cfg
srvtype socket
#address socktcp://<IP address>:<Port>
address socketcp://78.46.41.8:80
GET /<Mount Point> HTTP/1.0
User-Agent: Ntrip XS/1.14

Authorization: Basic <user_name>:<password>
```

The first two lines are comments. The line starting with ***exec*** is not used for configuration. Instead, it contains a command that is run by the command interpreter. In the example, a message is output to a potentially connected terminal. This message is used for testing purposes and allows monitoring the application's progress.

Required parameters are entered using **address socktcp://** and **GET /**. You need to enter the IP address of the Ntrip Caster or Server as well as the mountpoint.

User name and password **m u s t** be entered using base64 coding. You can create the coded output online at http://www.base64encode.org/.

If no Internet connection is available, you can get the coded output from the GNSS receiver.
To do so, simply connect the receiver to a PC and open a terminal application. Type the following instruction:

pob b64 user_name:passwort <ENTER>

The output contains the coded data.

Example

```
exec @echo reading from ntrip.cfg
srvtype socket
address socktcp://195.80.56.12:80
GET /LEIJ1 HTTP/1.0
User-Agent: Ntrip XS/1.14

Authorization: Basic TnRyaXB1c2VyOndlbGNvbWU=
```

where:

```
<IP address> :195.80.56.12
<Port>: 80
<Mount Point> : VRS_3
<user_name> : Ntripuser
<password> : TnRyaXB1c2VyOndlbGNvbWU=
```

After the TCP connection has been established by the GSM radio the data for GET, User-Agent and Authorization are transmitted to the Ntrip Server. Once they are accepted by the server, a continuous stream of data is received. The data normally is forwarded to the GNSS card.

For more details, see the command ***ntrip***.

English

CRONJOBS

crontab.cfg

Definition file for time-triggered workflows and processes

A cron manager within ppmOS allows you to create complex and time-dependent applications.

A cron manager is a software feature comparable to a diary; it allows to run certain commands at predefined times.

These commands or actions can be one-time or repeated. Conditional execution is supported as well. For example, you could check the state of an event line periodically. If a certain level (high or low) is found, action is taken.

Each action is a command within ppmOS.

You can combine commands within scripts.

All time-driven actions to be run after power on are listed within **crontab.cfg**. This file could be called the diary of your 20xx GNSS receiver.

If you are familiar with Unix or Linux, you might have heard crontab, cron manager and cron daemon before. However, unlike Linux this file is loaded upon execution of the 'cron init' command into a RAM resident table. Therefore, we recommend running the 'cron init' command from the autoexec.sh file. Linux simply monitors this file; no RAM resident table is used.
crontab.cfg within ppmOS has more features than crontab within Linux. For example, you can define times down to seconds or use extended expiration.

Up to 8 additional cron tasks are supported during runtime (limit induced by the controller RAM needed).

If a task is no longer needed it may be deleted. You can terminate cron tasks automatically after a certain number of repetitions.

Before we look into the crontab syntax have a look at this short example of a cron task (also known as a cronjob):

```
0,20,40 0 * * * 11-30 * gps bestpos
```

This line instructs the sensor to output 3 positions every full hour with a 20 second interval.

This is the syntax for periodic, nonperiodic or one-off actions:

Empty lines and lines starting with **#** are ignored.

Lines starting with ***exec*** are forwarded to the shell for execution.

Lines within crontab are interpreted as 'at time X, run command Y'.

The format of a line within crontab is as follows. The special characters used are described in the "ppmOS command reference".

Primary time specification:

```
<ss> <mm> <hh> <dd> <MM> <yy> <weekday> [-t{0|1|2}|-i] <command>
```

where:

<ss>	0..59 second
<mm>	0..59 minute
<hh>	0..23 hour
<dd>	1..31 day
<MM>	1..12 month
<yy>	0..99 year
<weekday>	0...6 weekday, 0 = Sunday, 1 = Monday
	any available command (resident or within a script saved to the SD card) Commands are copied to the console's in-pipe and processed within ppmOS in the same way as commands that have been received on port ComA-RS232.
[-t {o**|1|2}| -i**] <command>	Use -t0, -t1, or -t2 to execute a command in the respective task. This is useful if you expect longer execution duration and do not want to block the command interpreter. Use -i to execute a command immediately. The cron manager pauses during execution. This means that the "diary" cannot be accessed and no cronjobs can be executed.

The asterisk (*) is used as a wild card; it means "all possible parameters/numbers".

You can enter lists of numbers instead of only one, e.g. 2,4,6,8,9,11.

You can enter a range instead of one number, e.g. 4-10 or 58-2.

58-2 equals 58,59,0,1,2.

Numbers, lists and ranges can be entered in any order; they must not contain spaces or tab characters.

So far, we have covered primary time specifications. You can create more complex time specifications by adding a secondary specification with the seconds parameter. The seconds parameter and secondary specifications are not supported when using Linux.

Secondary time specification:

```
[[+<time offset>][/<time interval>[*<repetition counter>]][.]]
```

The secondary time specification follows BNF (Backus Naur Form). See the "ppmOS command reference" for details on the command syntax.

If the primary time specification is followed by a +<time offset> the first execution is delayed until <time offset> seconds after the primary specification.

Use a dot (.) after +<time offset> to run the command once and then delete the cronjob.

You can add /<time interval> after the seconds parameter.

This will run the command at the primary time specification and then every <time interval> seconds. This character is supported in all fields when using Linux.

It will replace any other primary time specifications.

You can add a *<repetition counter> to <time interval>.

This will run the command every <time interval> seconds for a total of <repetition counter> times. After this, the primary time specification is used again.

Use a dot (.) after the <repetition counter> to terminate the cronjob after the command has been executed <repetition counter> times.

Examples:

```
0 0 6,20 * * 11-30 * cp c:/sys/gps.cfg c:/bak/gps.cfg
```

Execute the copy command at 6:00 (am) and 20:00 (8 pm) on every day from 2011 to 2030. You should always specify the year, because at power-up the real-time clock (RTC) in ppmOS defaults to 1.1.1970 0:00:00. Once the first GNSS fix is available, the RTC will be set using UTC. You can see from this that the cronjob will only become active once the first fix has been achieved and the time has been set.

```
0 0 * * * 11-30 1-5 c:/sys/cmd_xyz
```

Run the script called c:/sys/cmd_xyz every full hour from Monday to Friday, starting in 2011.

```
*. * * * * 11-30 * gps log on
*+30. * * * * 11-30 * „date -s; gps log off ftp put @$(fname)
*+90. * * * * 11-30 * sleep 1710
```

This script starts 3 self-terminating cronjobs.

As soon as the RTC has been set, data logging starts (gps log on).

30 s later, two commands are executed. Those are shown in quotation marks „…“ and need to be delimited using a semicolon (;). Time and date are output, then the logging is stopped (gps log off). An additional command complements `gps log off`, to facilitate an FTP upload of the logging file that has just been closed.

Another 60 s later the processor enters sleep mode for 1710 s. Awaking from sleep mode requires a processor reset. The total cycle lasts for approximately 30 min. This is an approximate figure because the time required for GNSS initialization and first fix is not taken into account.

After sleep mode has ended (and the controller therefore has been reset), the file crontab.cfg is read again and the process starts over.

Entering a year parameter is necessary, because the RTC is set to 0:00:00 1.1.1970 after a reset. Once the first fix has been acquired the RTC is set according to UTC. If required, a firm time interval can be achieved by using the script firstfix.sh.

Refer to the command description `cron` for additional information.

Script files

Script files are used to control time and/or event driven workflows and processes.

autoexec.sh

This script file is executed after powering the receiver, after a reset or when exiting from the USB drive mode. This file lists all the commands required for a specific receiver application. Usually autoexec.sh will include at least the following commands:

cron rm all	*(delete any existing time-driven tasks)*
cron init	*(search for crontab.cfg, read it and execute any time-driven tasks)*
gps init	*(search for gps.cfg, read it and execute all commands used to configure the GPS card)*

firstfix.sh

This script file is executed after the first position fix.

It is executed only once, after the card does output the first valid position. Regardless of the contents of gps.cfg the GNSS card is configured to output and monitor the NMEA messages $GPRMC and $GPGGA on an internal port.

A position fix is considered valid if position, date and time are available, the validity flag (2nd parameter) is "A" and the message's CRC (check sum) is valid.

Only now data logging is possible. This is due to the fact that the file name is created from the current data and time. File names use one of 3 naming templates.

If you are using intermittent positions we recommend to setup time control (cron manager) within firstfix.sh.

English

event_hl
event_lh

These script files are executed by external events.
The file names event_hl and event_lh are exemplary only. Generally you should use speaking or meaningful file names referring to their function.

Receiver action can be triggered by up to 5 external and independent events. This is done by measuring changes in the voltage level of the following lines.

ppmOS is able to check the status (0 or 1) of an event input. Normally you will be interested in the transition of state. Therefore, a periodic check with subsequent processing is required.

Use the command 'peb' to check an event-in line.

```
peb pf.2
peb pb.4
peb pb.5
peb pb.6
peb pb.7
```

Following is an example to check event-in 1 using a script. All the script does is to output the status on the terminal. You could call this script 'show_ev1'.

```
@peb pb.4 >nul || echo Event-Pin 1 is high
@peb pb.4 >nul && echo Event-Pin 1 is low
```

To carry out an action once per transition:

The status check is controlled using a cronjob; this job needs to be swapped once the status changes. So, if the level is high, you need to check for low. As long as the level stays high you do not need to do anything. As soon as the status changes to low the level has changed. Now we need to carry out our action (could be a requesting a position fix from the GPS). At the same time the cronjob has to be swapped. We now have a low level and need to check for high.

The example here configures the 20xx sensor to output a position every 3 s while Event-in 1 is high. If Event-in 1 is low, the position is output every 60 s.

Initialization (include in firstfix.sh):

```
#run the relevant cron task for each level (high or low)
@peb pb.4 >nul || @c:/bin/event_lh
@peb pb.4 >nul && @c:/bin/event_hl
```

Contents of 'event_lh':

```
@echo Event pin is high
@cron put ix 5 * * * * * * * "@peb pb.4 >nul && @c:/bin/event_hl"
#Action: get a position every 3 s
@cron put ix 6 */3 * * * * * * @gps bestpos
```

Contents of 'event_hl':

```
@echo Event pin is low
@cron put ix 5 * * * * * * * "@peb pb.4 >nul || @c:/bin/event_hl"
#Action: get a position every 60 s
@cron put ix 6 */60 * * * * * * @gps bestpos
```

Data logging

Anything the micro controller receives on port **GPS2** can be written to a file and/or output on port ComB (25-pin SUB-D).

Anything the micro controller receives on port **GPS1** (NMEA GPGGA and GPRMC) is required for internal processing. You could configure and log additional NMEA messages on this port.

ppmOS supports SD memory cards using the two file systems FAT16 and FAT32. To format an SD card you need to connect the unit to a PC using USB. ppmOS implements FAT16 and FAT32 with short file names.

This means that the name can be up to 8 characters long. The extension is limited to 3 characters.

If the 20xx GNSS receiver is connected to a PC in USB drive mode the data rate is 3 to 14 MB/s.

If the 20xx GNSS receiver is connected to a PC in USB drive mode the data transfer to and from the SD card is performed using sector CR creation and checking.

If the 20xx GNSS receiver is not connected to a PC (ppmOS manages the SD card interface) the write and read rates are up to 1100 kB/s. Data rates are primarily affected by the type of SD card. The cluster size has only little effect. To allow such high data rates we use only industrial grade SD and SDHC cards with the 20xx GNSS receiver.

Note:

To achieve data logging with errors or gaps when writing a large amount of data (i.e. NMEA messages with 10 Hz or more) you have to delete any unneeded files on the SD card prior to the measurement. Reformat the SD memory card every 2 to 3 months to limit fragmentation. To do this, connect the 20xx GNSS receiver to a PC using the USB cable. In USB drive mode, save any important data from the SD card to the PC. Then format the card. Fast format will be fine. After formatting is complete, copy any required configuration files back to the SD card.

English

ppmOS: Command reference

ppmOS is a command line operating system, offering file-based and cooperative multitasking features. You can connect your 20xx GNSS receiver to a serial port (RS232) using a null modem cable. The parameters are 115200 baud, 8 bit, 1 stop bit, no parity, no XON/XOFF, no RTS/CTS.

A terminal application is used to control the receiver.
Make sure to operate the terminal in line mode, not in character mode, as ppmOS generates command line echoes, but no character echoes. We recommend writing and editing each command line in the terminal application. Then press the return key and ppmOS will generate the command line echo.

A number of commands are based on Linux. If you have any experience in using Linux, you will recognize syntax and usage similarities in the following descriptions. Usually you can request a short command line help by entering a command without any parameters or with the parameter **?**.

Usage conventions in the command reference

<term>	denotes a number, a character or a text input described by this term
[ ]	marks optional inputs
{ }	designates a choice; one of the options has to be selected
|	delimits lists of optional or compulsory entries

Special characters on the command line

@ or !	Use this as the first character on a command line. In this case, the standard command echo will be suppressed. (Note: You can turn all echoes off globally using the command 'set echo off' and turn them on again using 'set echo on'.)
;	Command delimiter to enable writing several commands on one line. Such lists of commands are **unconditional**.
||	Delimiter for a **conditional** list of commands. First, the command left of **||** is executed. If the return value does **not equal** 0 (usually the return value for an error in execution) the command left of the delimiter is executed.
&&	Delimiter for a **conditional** list of commands. First, the command left of **&&** is executed. If the return value does **equal** 0 (usually the return value for a correct execution) the command left of the delimiter is executed.
"..."	Characters and values are handled as one item. This allows the use of spaces or several commands. **time "csm c:/file_a >c:/file_a.csm"** The above example measures the time needed to create a checksum for the file called c:/file_a. Any output created by the command csm is written to the file c:/file_a.csm. When not using "..." any output – including that of the time command – would be written to the file.
>	This character sends output into a file; the file name has to be given as the next parameter. Existing files are overwritten. This output to file in ppmOS sends any outputs, that are output to the console interface are written into the file while the command is active. This means that outputs from any other task within this period are written to that file as well. A file name of **nul** means to send the output to nowhere (effectively producing no output at all).
>>	This character sends output into a file; the file name has to be given as the next parameter. Non-existing files are created. Existing files are appended to. A file name of **nul** means to send the output to nowhere (effectively producing no output at all).

Any other characters or words have to be written exactly as shown in the command syntax.

You can add any amount of space characters anywhere (beginning, between, after parameters).

Spaces or tabs are compulsory to delimit parameters from each other and the command.

Command overview

Command	Description
bd	set baud rate
cat	view ASCII file
cd	change directory
cmds	list commands
cp	copy file
cron	manage time-driven actions
csm	calculate check sum
date	set or show date and time
df	show free memory space on SD card
echo	output text to console
ftp	start FTP up- or download
gps	control GPS/GNSS functions
gsm	control GSM radio functions
loop	repeat command execution
ls	list directories available on SD card
mkdir	create directory on SD card
mv	rename file
ntrip	establish Ntrip Server/Caster connection
peb	output contents of 8-bit memory addresses
ptsk	log external events
Reset	reset processor
rm	delete file
set	show and modify environmental variables
sleep	switch processor to sleep mode
sms	send and receive text messages
stat	output software status
tail	show file contents, starting at the end
time	measure command runtime
ver	output software version
wait	delay command execution

Commands in detail

bd	Set baud rate
	Set, change or show communication baud rates.
Parameters	**{stdio|com{1|2|3|4|5|6|7}} [<baud>|-c <bd_ctrl>]** **{stdio|gps1|gps2|gsm1|bcn} [<baud>]-c <bd_ctrl>]**
{comA|comB|gps1|gps2|gsm1|bcn}	**ComA**, etc. denote the serial ports on the controller, linked to the modules (GNSS receiver, GSM radio or beacon receiver). **stdio** and **com3** refer to ComA (25-pin SUB-D), **com7** refers to ComB (25-pin SUB-D). **Attribution:** com1 - gps1 internal link to GNSS card com2 - gps2 internal data link to GNSS card com3 - comA or stdout interface to terminal com4 - gsm1 internal link to GSM radio (optional) com5 - bcn internal link to beacon receiver (optional) com6 SD card interface com7 - comB for free use on 25-pin SUB-D connector

[<baud>]	Baud rate setting, i.e. 4800, 9600, ..., 115200, 230400, 460800, 921600
	
	Please make sure to change the modules baud rate first, then the controller baud rate, if you change one of the command ports that links to a module (gps1, gsm1, bcn).
	
	Example: (NovAtel)
	
	`gps < com com1 230400 n 8 1 n off on`
	`wait 0.1`
	`bd gps1 230400`
	
	The wait command guarantees that the command for the card has been sent completely. Only now can the controller baud rate be changed.

	Enter **bd** without any parameter to show a list of current baud rates.

bcn	Configure beacon receiver (hardware option)
	You can purchase an optional beacon receiver with your 20xx-S sensor. On power-up, the sensor will check if a beacon receiver is available. If this is the case, an RMC NMEA message is passed to the beacon receiver intermittently.
Parameters	**< <cmd_to_beacon_receiver>** **| {@|!}<file_to_beacon_receiver>** **| bd [<baud>|-c <bd_ctrl>]** **| outport {gps1|gps2|stdout|nul|<hexadr>}** **| use_gprmc [on|off]** **| echo [on|off]** **| stat** **| -c <cmd>**
< <cmd_to_beacon_receiver>	Send data from the command line to the beacon receiver. This allows for advanced configuration of the beacon receiver. The prefixed < is used to route the output to the beacon receiver. This command implies 'bcm echo on' to show any outputs. Use 'bcn echo off' to turn those outputs off, if you do not require them.
{@|!} <file_to_beacon_receiver>	The content of specified file <file_to_beacon_receiver> is copied to the beacon receiver. The transfer takes place line by line. Lines starting with a hash (#) are comments. They are not copied over. Lines starting with exec are not copied, either. These are command lines executed by the CLI.
bd [<baud>|-c <bd_ctrl>]	Set or show the baud rate for the controller port that links to the beacon receiver. You can use non-standard baud rates. -c <bd_ctrl> outputs the value of the baud rate control register right away. The default setting is 4800 baud.
echo [on|off]	This sub-command allows activating or deactivating echoes to/from the beacon receiver.
stat	Display status information. Example: `bcn: bytes received: 83555, rxed GPRMC: 145, rtcm data: 83555 byte, tx 120 (diff 83435), inc 83555`
outport {gps1|gps2|std-out|nul|<hexadr>}	Set the output port for corrections. Default is **gps2**.
use_gprmc [on|off]	This sub command is used to turn on/off transmissions of $GPRMC to the beacon receiver.
-c <cmd>	This sub command is used to diagnose transmissions to and from the beacon receiver. Available options for <cmd> are 'stat' and 'cmds'. The command shows the number of bytes transmitted, any buffer overruns, as well as additional status information. Examples: bcn -c stat bcn -c cmds -u to show the frequency of received records

cat	View ASCII file
	Output an ASCII file to the terminal port (ComA).
Parameters	**[-fgets [-exec]] <file>**
-fgets	If used, the file is read line by line. Else it is read sector by sector.
-exec	All start of lines are scanned for 'exec'. If found, the following part of the line is executed.

cd	Change directory
	Set a path. In any subsequent commands the path is optional. However, the drive name (e.g. **c:**) has to be provided.
Parameters	**<path>**

English

cmds	List commands
	Lists all resident commands within the command interpreter in alphabetical order. Any scripts present on the SD card are not listed. Commands that have been removed using `ulink` are not listed, either. Alternatively, enter '?' to get the same output.
Parameters	**[-u] [<n_colums>]**
-u	Unsorted output. This option shows the number of calls per command since the last reset in front of the command name.
<n_colums>	Format the output in n_colums columns. Default is 3.

Example

```
>cmds
```

```
bcn         bd          cat
cd          cmds        cp
cron        csm         date
df          echo        find
ftp         gps         gsm
hexdump     loop        ls
mkdir       mv          ntrip
pe          po          prog
ptsk        Reset       rm
sector      set         sleep
sms         stat        tail
time        ulink       ver
wait        wrfile

38+1 commands
```

```
>cmds -u
```

```
  2 ver           3 cmds          0 find
  0 ulink        15 echo          0 pe
  0 po            0 time          0 loop
  2 date          1 set           0 bd
  0 stat          0 Reset         0 ls
  0 df            0 cp            3 cat
  0 sector        0 tail          0 csm
  0 wrfile        1 cd            0 mkdir
  0 rm            0 mv            0 hexdump
  0 prog          2 gps           0 ptsk
 12 gsm         745 sms           1 ftp
  0 ntrip         0 bcn

 63 (default)
38+1 commands
```

(default) stands for unknown command. In this case, 63 unknown commands were issued.

cp	Copy file
	Copy <srcfile> to <destfile>. <srcfile> and <desfile> can exist on different drives.
Parameters	**[-v] <srcfile> <destfile>**
-v	Verbose; used to show additional information. Every 1 MB the progress is output. On completion, the number of bytes copied is shown.

cron	Manage time-driven actions
	Use **cron** to manage the ppmOS diary (that is the cron manager).
Parameters	**ls** **| init** **| put [ix <ix>] <time_elements> [-t{0|1|2}|-i]** **<cmd> //-i means immediate else put in rx-pipe** **| put {@|!}<filen>** **| exec <ix>** **| rm {<ix>|all}** **| echo [on|off]** **| stat**
ls	List, output the current diary
put	Save a cron task. Optionally you can put the task in a specific position in the list using **ix <ix>**. The current version allows saving up to 8 cron tasks. See the **Crontab** description for more information on **<time_elements> [-t{0|1|2}|-i] <cmd>**.
put	{@|!}<filen> Load one or more cron tasks from the file named <filen>.
exec	<ix> Immediately run the command saved as cron task <ix>.
rm	{<ix>|all} Delete cron task <ix> (or all cron tasks).
echo	[on|off] Turn on/off reports while executing commands within cron tasks.
stat	Show information on cron manager. The following line is output: ***`cron: entries  0 (max. 8),executed 16, check cycle errors 0`*** *check cycle errors* shows how often the match time had been exceeded during testing. This can be the case if the cron manager is not run every second from within the task scheduler. The cron manager will simply run any due cron commands as soon as possible (delayed).

csm	Calculate check sum
	Create a simple 32-bit checksum for file <file>. The file is viewed as a number of 32-bit values in little endian.
Parameters	**<file>**

date	Set or show date and time
	Use `date` to set the real-time clock according to Unix time. This requires inputting the time in long format. Hours have to be given in 24 h format. Use `date` with no parameters to show date and time. `date ?` shows the command line help. The real-time clock is set automatically once the first GPRMC message is received on **gps1**. Thus, you will not normally set the clock.
Parameters	**[-s | -{c|n} <val>|<HH:MM:SS> <TT.MM.[JJ]JJ>]**

-s	Output date, time and uptime in short format.
	`>date -s`
	`date -s 8:55:37 03.07.12 1341305737.100 s 3332.889 s`

-c <val>	Calculate the date for the given value. You can enter a decimal or prefix 0x for a hexadecimal value. This is useful if you want to determine the date from a filename given in UTC-HEX. Example: *`>date -c 0x4e9450d3`* *`date: 14:21:07 11.10.11 unix 1318342867 0x4e9450d3, 284th day of the year, Julian Day 2455846`*
-n <val>	The clock is automatically set to the entered value (decimal or hexadecimal). Calculation follows the one for parameter **-c**. However, the time is set to this value.

df	Show free memory space on SD card
	Use `df` to show the free memory on the SD card as well as additional information (file system [FAT32 or FAT16], cluster size, card size, memory used, free memory). Depending on the card size the commands requires several seconds. You cannot run this command if files are opened. To do so, use the parameter **-s**. ppmOS can manage more than one SD card drive. Depending on your hardware version, the device will include one or more drives. Drive **c:** is always present.
Parameters	**[-s] [-i] {c:|d:|e:|f:}**
-s	Show SD card information, but do not determine the free memory.
-i	Identify SD card. This will run the same processes that happen after inserting a card (auto detect). You cannot change the SD card in your 20xx GNSS receiver at runtime.

Example

```
SD-card C:
FAT32, clustersize  32KB, size: 15515648 KB  used: 358464 KB  free: 15157184 KB
```

If one or more files are opened, the following is shown:

```
SD-card C:
FAT16, clustersize 32 KB, sect: 3979264, size: 1989248 KB files are open, can't
calc free space at this moment
```

echo	Output text to console
	Use this command to output text to the console port. Use the pipe (output routing, adding '>filename' or '>>filename' at the end) to write text into a file. We recommend prefixing `echo` with @. This suppresses the command line echo and outputs only the intended information.
Parameters	**<text>**

ftp	Start FTP up- or download
	Use this command to start an FTP upload (put) or FTP download (get) using the GSM radio. The source or target file exists in one directory of the file system. FTP address and credentials can be input using the command line or read from any info file. If you do not provide a parameter for <infofile>, the connection settings are reads from **c:/sys/upload.cfg** (if it exists). The info file is scanned for a line starting with **server**. Please note that GSM connections are not as stable and reliable as modern DSL lines. Therefore, you can set a number of repeat tries (**-try <n>**) for your upload. The GSM radio internal to the 20xx GNSS receiver allows for data rates in the range of 1.5 kB/s to 2.5 kB/s. Please take this into account when planning schedules (cron tasks).
Parameters	**[-v] put** **[-try <n>] [-rm[_anyway]] [-t[e] <tmot>]** **{{@|!} [<infofile>] | [<usr> [:<passwd>] {@|!}]** **<server>[[/path] [/<file>]]} [<local_file>]** **[-v] get** **-{a|i|d}} {{@|!} [<infofile>| [<usr>[:<passwd>]{@|!}]** **<server> [[/path] [/<file>]]} [<local_file>]**
-v	Verbose; output additional information during execution.
-try <n>	Number of tries to send data. No parameter = one try.
-rm	Delete the file once upload has successfully completed.
-rm_anyway	Delete the file even after an unsuccessful upload.
-a	ASCII download
-i	Binary download
-d	Download directory contents (ASCII list)
-t <tmot_s>	Try for this duration to read from the file (in case it is updated slowly from another process).
-te <tmot_s>	See '-t'; however, start reading from EOF.

Examples:

`ftp put ! c:/sms/smsprot.txt`

The SMS log file is transferred using standard settings defined in upload.cfg to the FTP server's root directory.

`ftp get ! c:/sys/autoexec.sh`

File autoexec.sh is transferred from the FTP server's root directory to c:/sys of the 20xx GNSS sensor. You can use '!' instead of '@'.
When transmitting commands by SMS you have to use '!'.

gps	Control GNSS functions
	Use this command to make any GNSS card settings and control position saving. The GNSS card is connected to the micro controller using two serial ports. gps always uses port ComA on the GNSS card. Any data on ComB of the GNSS card can be logged or forwarded. After the 20xx GNSS receiver is powered on, the GNSS card is initialized. The file **c:/sys/gps.cfg** is transmitted to the GNSS card. You can use this command for additional settings or actions at any point in time.
Parameters	**< <data_to_gps_board>** **| {@|!}<file_to_gps_board>** **| init [-i]** **| bd [{gps1 | gps2} [<baud> | -c <bd_ctrl>]]** **| set [filename_pattern {UTC_HEX|ASH_ATM|2LETTER_UTC <Letter1><Letter2>}]** **| bestpos** **| log [on [-append] [-container[<size]] [<file>] [-no_backup] [-dont_cp_cfg]]** **| log [off [<cmd_line>]] //in <cmd_line> the substring $(fname) will be replaced by the filename** **| echo [from|to] [on|off]** **| flush [cycle <flush_cyc_sec> | next <time_sec> |on|off|once]** **| stat** **| reply [on|com{1|2|3|4|5|6|7} |off]** **| -c <cmd>**
<<data_to_gps_board>	Send data from the command line to the GNSS card. This allows for advanced configuration of the GNSS card. The prefix **<** indicates that the data goes directly to the GNSS. This command implies 'gps echo on' to show any outputs. Use gps echo off to turn those outputs off, if you do not require them.
{@|!}<file_to_gps_board>	This sub-command is similar to the sub-command <data>. However, it does only transmit the contents of the file <**file_to_gps_board**> to the GNSS card. The transfer takes place line by line. Lines starting with a hash (#) are comments. They are not copied over. Lines starting with exec are not copied, either. These are command lines executed by the CLI.
init [-i]	Start initialization of the GNSS card. You have to call this sub-command explicitly. We recommend taking care of this in the file **autoexec.sh**. The script autoexec.sh is run after leaving the USB drive mode as well. Use **-i** to skip initialization (before gps.cfg).
bd [{gps1|gps2} [<baud> | -c <bd_ctrl>]]	Set or show the baud rate for the controller port that links to the GPS card. You can use non-standard baud rates. **-c** <bd_ctrl> outputs the value of the baud rate control register right away.

gps	Control GNSS functions
set [filename_pattern {UTC_HEX|ASH_ATM| 2LETTER_UTC <Letter1><Letter2>}]	Use this sub-command to set or show the file name pattern for GNSS logging. You can choose from three options: ***UTC_HEX*** The UTC at the start of logging is used to create the file name. UTC is shown as an 8-digit hexadecimal number. This guarantees that the alphabetical order conforms to the time order of file creations. The file name extensions comprise 3 letters. The GNSS card's serial number is used to create it. This allows attributing files to a specific 20xx GNSS receiver. **Example: 4e332b38.zjt** ***ASH_ATM*** The file name is compatible to the Ashtech Atom to Rinex converter. It starts with the letter 'G', followed by 2 digits for the hour, then 2 digits for the minute, one digit for the second (using letters) and 2 digits for the year. The file extension is the day of the year, starting with 1. **Example: G1501G11.776** ***2LETTER_UTC <Letter1><Letter2>*** The file name uses the two user-defined letters, followed by 2 digits for the day and 2 digits for the second. The file extension is created from the last figure of the year, one digit for the month (1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C) and the letter D (to indicate a data file). **Example: FP061233.15D** **Default is UTC_HEX.**
bestpos	The file **c:/sys/bestpos.cfg** is transmitted to the GNSS card. The transfer takes place line by line. Lines starting with a hash (#) are comments and are not transmitted. Lines starting with exec are not copied, either. These are command lines executed by the CLI. You can use the command line '**GNSS @c:/sys/bestpos.cfg**' for the same effect.

English

gps	Control GNSS functions
log [on [-append] [-container[<size]] [<file>] [-no_backup] [-dont_cp_cfg]]	This sub-command starts logging (on) or shows the current status (number of bytes logged). You can modify the logging using these parameters: **-append** Append data to the previous file or the file indicated by <file>. We recommend opening any container file using this parameter. **-container [<size>]** Data is written to a special container file. Container files have a pre-defined and fixed size. Once the EOF (maximum size) is reached, the oldest data is overwritten. This is called a ring buffer. Special file information kept within the first 512 bytes allows finding the logical start of file (as this will often not be the physical start). You can set the maximum size using <size>. If no <size> is given, a default value of 1.6 GB is used. The minimum size is 2 MB. If you open a file several times, the maximum <size> is used. This means the file can only grow in size, never shrink. This type of container does not exist within Windows and Linux. Therefore, a special program is used to read the definition data from the file. It recognizes a container file by scanning for the following 64-bit-signature at the file's beginning: 0x6b, 0x78, 0x3c, 0x59, 0x2d, 0x4e, 0x1f, 0x0a This is the same signature for every container file. It is followed by 32-bit numbers (little endian) showing the current end and beginning of file (= write/read offset), the maximum length, and a loop counter (showing the number of times the container has been fully written). All data starts after file offset 512. **<file>** Use <file> to name which file the data is written to. **-no_backup** No backup file is created on drive d:. Normally any GNSS data is written synchronously to the second SD memory card (d:), if it exists. **-dont_cp_cfg** No copy of the used file gps.cfg is created. If the parameter is missing, the GNSS configuration file c:/sys/gps.cfg for the current session is saved under the data file's name with the extension .cfg in the data directory. This allows you to check the settings used to create the file.
log [off [<cmd_line>]] //in <cmd_line> the substring $(fname) will be replaced by the filename	This sub-command stops logging (off) or shows the current status (number of bytes logged) if no parameter is given. **<cmd_line>** By adding this parameter, you can run the command line immediately after closing the data file. That means the command line is copied to the command pipe of the terminal port. Upon execution, the string **$(fname)** (if available) is replaced with the current file name. **$(fname)** can be used in any other command as well. Example of a typical command line: **ftp put @ $(fname)** This transmits the recently closed file to an FTP server.
echo [from|to] [on|off]	This sub-command allows activating or deactivating echoes to/from the GPS card. By default, the GPRMC message is activated on ComA of the GPS card, regardless of the logging configuration for ComB. It is used to detect the first position fix and monitor the validity (date, time validity flag) of the GPS fixes, and to drive the signal LED (red or green). The precise GPS time is used to set the receiver RTC regularly. The correction is output to the terminal port. If you do not require this output, you can turn it off using '**gps echo off**' or '**gps echo to off**'.

gps	Control GNSS functions
flush [cycle <flush_cyc_sec> | next <time_sec> | on | off | once]	Logged GPS data is written to the SD card in chunks of 512 bytes (sector). The directory entry is updated every 300 s (default). Should the unit fail, only the most recent <flush_cyc_sec> s of data is lost. Use this command to change those values. With <flush_cyc_sec> = 0 the directory entry is updated after every sector (512 bytes). Please note that frequent writing can lead to a faster aging of the respective sector. If you want to run an FTP upload while data logging is still in progress, you might need to change <flush_cyc_sec>. The upload can only transfer data that are shown to exist by the directory entry.
stat	Output duration of the different GPS states. Example: `gps: up 1245.222 s, no fix 29 s, gps 161 s, dgps 1056 s, station 0771` up shows the time since the last receiver reset no is the time that no valid position fix was calculated (accumulated) gps is the time that a valid position without DGPS was calculated (accumulated) dgps is the time that DGPS was used (accumulated) station is the DGPS station ID currently used (from $GPGGA)
reply [on|com{1|2|3|4|5|6|7} |off]	Forward data received by the GNSS card to a serial port. on forward to ComB (25-pin SUB-D), internal Com7 com{1|2|3|4|5|6|7} forward to the named controller port off turn off forwarding

-c <cmd>	This sub command is used to diagnose transmissions to and from the GPS card. Available options for <cmd> are 'stat' and 'cmds'.
	The command shows the number of bytes transmitted, any buffer overruns, as well as additional status information.
	
	Example:
	`gps -c stat`
	Show the reception buffer's size for port gps2 (in this example 2665 bytes).
	When using an Ashtech card, an output after initialization shows if all initialization steps were successful.
	(NAK received 0 or **N**ot **ac**knowledged 0, means, that no commands have been declined. This, in turn, means, that all commands have been accepted.)
	`com2 rxlvl 0 (2665), rec 512, ovfl rx 0 tx 0, wr to file 0 0 diff 512 0`
	`version received 1`
	`gps: position fix is valid`
	`RIO received 1`
	`ACK received 17`
	`NAK received 0`
	`PMGNGO received 2`
	
	Example:
	`gps -c cmds -u`
	Shows the frequency of received records

gsm	Control GSM radio functions
	This command controls the settings for the GSM radio. Using the GSM radio, you can transfer files to/from servers (FTP protocol, see command 'ftp'). Furthermore, you can receive real-time corrections from an Ntrip Caster using a TCP/IP connection (see command 'ntrip'). In the background, text messages (SMS) can be sent and received. However, the radio can only be used for one of the above. If it is busy it will be locked for any other requests. If a request is rejected, an error message is output, e.g.: `gsm: modem is locked from 'ntrip, request from 'sms ls' is rejected`

gsm	Control GSM radio functions
Parameters	**< <cmd_to_be_sent_to_mdm>** **| {@|!}<file_to_be_sent_to_mdm>** **| bd [gsm1 [<baud>|-c <bd_ctrl>]]** **| sms { cycle <poll_cyc_sec> | next <time_sec> | on | off | once }** **| stat** **| echo [lock] [on|off]** **| -c <cmd>**
<<cmd_to_be_sent_to_mdm>	Send data from the command line to the GSM radio. This allows for advanced configuration of the GSM radio. Use the prefixed **<** to control the GSM radio directly. This command implies 'gsm echo on' to show any outputs. Use '**gsm echo off**' to turn those outputs off, if you do not require them.
{@|!}<file_to_be_sent>	This sub command is similar to the sub command < <data>. However, it does only transmit the contents of the file <file_to_be_sent> to the GSM radio. The transfer takes place line by line. Lines starting with a hash (#) are comments. They are not copied over. Lines starting with exec are not copied, either. These are command lines executed by the CLI.
bd **[gsm1 [<baud>** **|-c <bd_ctrl>]]**	Set or show the baud rate for the controller port that links to the GSM radio. You can use non-standard baud rates. **-c <bd_ctrl>** outputs the value of the baud rate control register right away. The default setting is 57600 baud.
sms **{ cycle <poll_cyc_sec>** **| next <time_sec>** **| on | off | once | }**	Use this sub command to set the text message (SMS) functions of the GSM radio for reception of texts. Sending text messages is controlled using 'sms'. For more information on sending texts, see below. The GSM radio autonomously receives and saves text messages. The micro controller will check periodically for received text messages. This command is used to set the interval. **cycle <poll_cyc_sec>** Set the polling interval. The default value is 5 seconds. **next <time_sec>** This is the next time that the GSM radio is polled for received text messages. The time is shown based on UTC in Unix format (seconds). **on | off | once** You can poll the GSM radio never (off), regularly (on), or once (once) Please note that the radio will automatically receive and save text messages as soon as the 20xx GNSS receiver is turned on. The micro controller does only poll them from the GSM radio.
echo [lock] [on|off]	This sub-command allows activating or deactivating echoes to/from the GSM radio. Use the 'lock' parameter to turn on or off the message that is shown if the GSM radio currently is operating in Ntrip or FTP mode and cannot poll text messages.

stat	Show status information, especially on text message polling.
	Example:
	`gsm: bytes rec. on cmd: 6302, redir: 6302 (diff 0), dat: 0`
	`gsm: reentrance lock 0, echo 0`
	`sms: next 12:43:26 03.07.12 (4ff2e8ee), cycle 17 s, remaining 7 s, in progress 0`

gsm	Control GSM radio functions

-c <cmd>	This sub command is used to diagnose transmissions to and from the GSM radio. Available options for <cmd> are 'stat' and 'cmds'.
	The command shows the number of bytes transmitted, any buffer overruns, as well as additional status information.
	Examples:
	`gsm -c stat`
	`gsm -c cmds -u`
	`Shows the frequency of received records`

loop	Repeat command execution
	Run a command repeatedly. This can be a simple command, a script, or a number of commands within " ". Please note that during execution (especially of several loops with pauses) ppmOS is able to save additional commands in the reception buffer, but that the CLI will only run those once the current command is completed. You cannot stop a command before it is completed.
Parameters	**<nloops> [-t <millisec>] <cmd>**
<nloops>	Number of repetitions
-t <millisec>	Pause (integer milliseconds) between repetitions.

ls	List directories available on SD card
	Use this command to show directories and their contents. The output is unsorted. The first column does contain the file names. Then, the file length is shown (or **<dir>** for directories). The next columns show date and time of the last file modification as well as date and time of file creation. For GNSS data files, the difference between those times is the logging duration. For the file called 4E783E68.ZJT this duration is 3802 s. That translates to a data rate of 5.35 kB/s.
Parameters	**[-r [-f|"info text or cmd with %s as place holder"]] [-s] <path>**
-r	The directory as well as sub directories are shown (recursive search).
-f	A list of files including full paths is output. "info text or cmd with %s as place holder" This can be any string; '%s' is replaced by the respective file name. **Example:** `ls -r -f "ftp put -try 5 -rm ! %s" c:/data/2012/10 >ftpupld.sh` This will create a script to upload all files saves in October to an FTP server. Using the info text, the file list is a list of executable command lines.
-s	Only a summary is shown.

Example

```
>ls c:/data/2011/09/20

..                 <dir>   0:00:00 01.01.98   7:19:04 20.09.11
4E783E68.CFG        2285  18:27:46 19.09.11  23:46:18 15.09.11
4E783E68.ZJT    20335872   8:22:26 20.09.11   7:19:04 20.09.11
4E7852BC.CFG        2285  18:27:46 19.09.11  23:46:18 15.09.11
4E7852BC.ZJT     1628521   8:50:48 20.09.11   8:45:48 20.09.11
4E7865F4.CFG        2307  10:55:40 20.09.11  23:46:18 15.09.11
4E7865F4.ZJT     1677500  10:12:48 20.09.11  10:07:48 20.09.11
4E78695A.CFG        2307  10:55:40 20.09.11  23:46:18 15.09.11
4E78695A.ZJT     1652406  10:27:18 20.09.11  10:22:18 20.09.11
15.09.11
4E78F7FC.ZJT     1640957  20:35:52 20.09.11  20:30:52 20.09.11
4E78FB7C.CFG        2307  10:55:40 20.09.11  23:46:18 15.09.11
4E78FB7C.ZJT     1660162  20:50:48 20.09.11  20:45:48 20.09.11
4E78FE6A.CFG        2307  10:55:40 20.09.11  23:46:18 15.09.11
4E78FE6A.ZJT     1664344  21:03:18 20.09.11  20:58:18 20.09.11
4E7900C2.CFG        2307  10:55:40 20.09.11  23:46:18 15.09.11
4E7900C2.ZJT     1626885  21:13:18 20.09.11  21:08:18 20.09.11
4E790428.CFG        2307  10:55:40 20.09.11  23:46:18 15.09.11
4E790428.ZJT     1693687  21:27:50 20.09.11  21:22:48 20.09.11
4E7906FA.CFG        2307  10:55:40 20.09.11  23:46:18 15.09.11
4E7906FA.ZJT   178771935   6:36:14 21.09.11  21:34:50 20.09.11
   219050322 bytes in 28 files, 0 directories
```

mkdir	Create directory on SD card
	Use this command to create a path. All directories within <path> are created. '/' is used as delimiter between directory names. The path may partially exist.
Parameters	**[-s] <path>** **// -s single directory in <path> only**
-s	Use **-s** to only create the last directory. If you use -s and one of the directories within <path> (except for the last one) does not exist, an error message is shown. The command is not case-sensitive.

mv	Rename file
	Use mv (move) to rename a file. The renamed file is created in the same directory. The file name (<path>/<oldname>) can contain a drive letter and a path. The new file name <newname> must not contain a path. To specify a new path, you can use **cp** and **rm**.
Parameters	**<oldname> <newname>**

English

ntrip	Ntrip connections

	Use `ntrip` to configure the GSM radio for real-time corrections. A TCP/IP connection to an Ntrip Caster or Server is established. After the 20xx GNSS receiver has been authenticated, the Ntrip Caster will transmit GNSS corrections. The corrections received will be output to a user-selectable port. The default correction port is GNSS port GPSB (ComB). The GNSS card uses correction data to improve the position fix (if configured accordingly).
	A $GPGGA message is transmitted to the Ntrip Caster from time to time.
	
	You can forward the Ntrip data received to ComA by using the following commands in a script. You could even output GPS data on ComB and Ntrip data on ComA.
	
	`gps echo off`
	`gsm echo lock off`
	`ntrip -s open`
	`set ord com2 tee com3`
	
	If you want to provide external GPS corrections to the receiver, we recommend switching the sensor's USB port to RS232 mode. Now you can connect the GPS card directly to the external device using the USB port.

Parameters	**[-v|-s] open [{@|!}<ntrip_cfg_file>] [-outport {gps1|gps2|stdout|nul|<hexadr>}]** **| [-v] close** **| stat** **| tx_gpgga [on|off]** **| outport {gps1|gps2|stdout|nul|<hexadr>}**
[-v|-s] open [{@|!}<ntrip_cfg_file>] [-outport {gps1|gps2|stdout|nul|<hexadr>}]	The sub command opens the Ntrip stream. The GSM radio establishes a TCP/IP connection using the address from the optional file <**ntrip_cfg_file**>. Dialing in requires the file **c:/sys/upload.cfg**, containing SIM PIN and network provider information. If no <ntrip_cfg_file> is specified, **c:/sys/ntrip.cfg** is used. See above for more information on those files. **-v** Output additional diagnostic information during dial-in. **-s** Suppress status information 'tcp: got …' during transmission. Set this parameter if using '-outport stdout' or the output in gps2 is routed to stdout (to make corrections available externally). **-outport {gps1|gps2|stdout|nul|<hexadr>}** This optional parameter allows you to route the Ntrip corrections to an additional serial port. If the parameter is not used, gps2 is used as the output port.
[-v] close	The sub command closes the Ntrip Server connection.
stat	Output status information regarding **open** or **close** as well as on received and forwarded amount of data.
tx_gpgga [on|off]	Set and control transmission of $GPGGA to the Ntrip Caster (default is on).
outport {gps1|gps2|stdout|nul|<hexadr>}	Use this sub command to forward the corrections immediately to another serial port. The default port for correction is **gps2** (ComB on the GPS card). Forwarding can be changed as many times as required. Port **nul** forwards to nowhere. Specifying <hexadr> with prefix 0x does not affect the user. You have one more option for port forwarding. It is detailed in 'set iord …'.

ptsk	Log external events

	The command **ptsk** is used for a special type of logging. It performs time-dependent, single positions fixes using 'gps bestpos'. Depending on the level (high or low) of up to 5 PIO bits (available on 25-pin SUB-D connector and on a socket on the back of the data logger) additional records with GPS position fixes are written to the GPS log file.
	The process is initialized from autoexec.sh and controlled by cron tab entries and scripts. For each event-in line, the time duration between "Event-in is high" and "Event-in is low" are monitored and added up. If the event-in state changes (or 'ptsk put' is called), a record containing the current event-in state and the accumulated on-time is written in ASCII format to the GPS log file.
	This happens right before 'gps bestpos' is issued, to show the position of the change-of-state of the event-in line. Polling the event-in lines is time-driven, using the cron task '**ptsk chk**'.
	The record created comprises of 2 to 6 ASCII characters (CRLF terminated) in the following format:
	`ppm: 5 version 0.3`
	`mw 0 52 s`
	`m1 1 1232 s`
	`m2 0 0 s`
	`m3 1 466 s`
	`m4 0 230 s`
	The total number of lines following 'ppm: ...' is indicated by the number after 'ppm:'. The real information for the application is shown in the lines 'mw ...', 'm1 ...'. The number after 'mw' shows the logic state of the monitored event-in line. Then the accumulated on-time in seconds is shown (that is the time the logical state was 1).

Parameters	**init [-dont_use_comb] [<pos_interval_on_s> [<pos_interval_off_s>]] | chk | put | eof | rd | stat | echo [[chg] on|off]**
init [-dont_use_comb] [<pos_interval_on_s> [<pos_interval_off_s>]]	Initialize and set time for frequency of record output in case of no change. **-dont_use_comb** Do not use the event line on the 25-pin SUB-D connector, if this is needed for serial data input (RxD comB). **<pos_interval_on_s>** Frequency of output, if at least one event line is high. **<pos_interval_off_s>** Frequency of output, if all event lines are low.
chk	Check state of event inputs. You can process the return value using || or && (see command reference and special characters).
put	Write current state as ASCII record to GNSS log file.
rd	Output current state as ASCII records to the terminal port.
eof	Write ASCII record 'ppm: eof' to GNSS log file.
stat	Show time settings for record output
echo [[chg] on | off]	Output additional information to terminal port, output only for changes ('chg'), or turn off additional information.

peb	Output contents of 8-bit memory addresses
	Use this command to output the processor's RAM contents byte by byte hexadecimal. The memory addresses include the function registers of all on-chip modules within the processor.
Parameters	**[-p] <adr> [<cnt> [<bytes_per_line>]]** **| p{a|b|c|d|e|f}[.{0|1|2|3|4|5|6|7} | -m <mask>]** **<adr> [<cnt_hex> [<bytes_per_line>]]** **| p{a|b|c|d|e|f}[.{0|1|2|3|4|5|6|7} | -m <mask>]**
<adr>	Start address for memory output.
<cnt>	Number of bytes to output.
<bytes_per_line>	Number of bytes per line
p{a|b|c|d|e|f}[.{0|1|2|3|4|5|6|7} | -m <mask>]	Output of PIO ports of processor. You can request a bit position (.0 .1 .2 ... .7) or mask the port using a value (-m <mask>). The return value can be used for conditional command executions (see || and &&).

Example

```
>peb 0x2000 0x80
memb: 00002000 6e 66 78 00 f1 55 7a 7a 7a 00 2f 00 2c 00 00 6e nfx..Uzzz./.,..n
memb: 00002010 00 66 66 00 00 00 00 00 d9 2f ce 2a d1 2a d0 37 .ff....../.*.*.7
memb: 00002020 a8 3c ce 33 2c 00 2a 00 00 00 00 00 a0 40 d2 3f .<.3,.*......@.?
memb: 00002030 f8 3f d0 37 a8 3c ce 33 2c 00 00 00 00 00 33 47 .?.7.<.3,.....3G
memb: 00002040 c1 43 d1 2a d0 37 a8 3c ce 33 2c 00 70 75 00 67 .C.*.7.<.3,.pu.g
memb: 00002050 65 00 70 3a 00 64 69 66 66 20 74 6f 20 46 53 21 e.p:.diff to FS!
memb: 00002060 3f 00 6f 2e 6b 2e 00 20 09 00 2f 00 3e 00 3b 26 ?.o.k.. ../.>.;&
memb: 00002070 7c 00 2b 31 00 00 05 00 63 04 01 0c 06 00 06 03 |.+1....c.......
```

Reset	Reset processor
	Use this command to reset the micro controller and re-start processing. If you pass on no parameters, first all interrupts are locked, then all file buffers are written to the SD card, and finally a software reset is performed. For the micro controller used, a software reset is the same as a hardware reset.
Parameters	**[-nf] [{-j] [-w}]**
-nf	Files are not flushed; data loss can happen. The command is executed immediately.
-j	No software reset is performed. Instead, processing jumps to address 0 (start address used after a reset).
-w	No software reset is performed. The CPU enters an infinite loop. The active watchdog timer will, once it is no longer updated, reset the micro controller.
	After the reset several startup information is shown, including the cause for reset.

Example

```
ppmOS V 1.26 compiled Jul 3 2012 15:28:28 UART-MSPI (GCC 4.6.2)
copyright ppm GmbH 2011
----------------------------------------------------------------
Reset reason SWR (20) counts 70 0 RC32M
heap: 0x4433 free 7068 min 6738, SP 0x5fcf, txt 168916, dat 216, bss 3114, alloc 5937, ctor 4, dtor 2
Fs: [1] total 1142 (sizes Fs 8 DOS_t 1134 DIR_t 1130 FAT_t 1091 My_SPI 23, adr: 0x3e41 0 0 0)
----------------------------------------------------------------
```

The cause in above example has been a software reset. `counts` indicates the number of system starts. Here it means that the sensor has been started or powered 70 times. This value is incremented after each reset and saved in non-volatile memory.

rm	Delete file
	This command deletes the selected file from the file system (SD card). The 20xx GNSS receiver does not allow undoing deletions. If you want to recover an accidentally deleted file, you have to connect the 20xx GNSS receiver in USB drive mode to a PC and then recover the files using special recovery tools. Make sure that the 20xx GNSS receiver cannot write additional data to the SD card. To do so, you should cut the power supply immediately. Then, connect the 20xx GNSS receiver to a PC using a USB cable. Once the USB cable is connected, you can turn on the 20xx GNSS receiver's power supply. If you follow this routine, the 20xx GNSS receiver switches to USB drive mode right after the reset. In this mode, the SD card is locked for the micro controller.
Parameters	**<file>**

English

set	Show and modify environmental variables
	Use **set** to show or modify environmental variables and system settings.
Parameters	**echo** **[on|off] | iord {stdout|com{1|2|3|4|5|6|7} [[tee|com] [-a] <file>| nul | off]** **| fread_as_container [on|off]** **adc echo** **[on|off|esc] | adc cycle [<cyc_ns>]** **usb tgl** **[on|off] | stick | rs232 | mode [lo|hi] | rst [lo|hi]** **pwr [vcc3|gsm|sd] [on|off]**
echo [on|off]	Show or modify the echo flag. If the echo flag is set (on), all entered commands are output to the console port after reception (command echo). If the logger output on the console port is logged, you can use 'echo on' to create a full history including command inputs and outputs. Additional output is created. If you do not require this output, you can turn it off using 'set echo off'. For some commands (gps, gsm, cron) the output can be turned on or off using a specific sub command. Other commands (cp, ftp, ntrip) support a switch (-v) to determine the scope of the output.

[i]ord {stdout|com{1|2|3|4|5|6|7} [[tee|com] [-a] <file>| nul | off]	Forward 'ord' from one port to another or to a file. For compatibility reasons with previous software versions you can use 'iord' as well. However, 'ord' is more correct. Anything output from the controller to the selected port is either forwarded (= output elsewhere) or copied to another port or file.
	
	tee Output is copied (union tee) to another port or file in addition to the intended port.
	com Forward the intended port; the intended port has no output. Use 'set ord com3 com off' to re-activate outputs on com3 and allow a previously defined output to a file.
	
	Example:
	
	`set iord stdout tee -a c:/data/stdout.txt`
	Outputs of the terminal port are shown at that port and are appended to c:/data/stdout.txt.
	
	`set ord com2 tee stdout`
	Outputs to the GPS card (gps2, GPS corrections: Ntrip or beacon) are copied to the terminal port and can be made available to an external device.
	
	`set ord com2 tee stdout`
	`set ord stdout tee -a c:/data/stdout.log`
	Outputs to the GPS card (gps2, GPS corrections: Ntrip or beacon) are copied to the terminal port. Any output to the terminal port is additionally written to a log file, opened in append mode.
	
	`set ord com5 tee stdout`
	Any outputs to the beacon receiver are copied over the terminal port.
	
	`set ord com5 off`
	Forwarding or copying of outputs intended for the beacon receiver is stopped. Only the beacon receiver will receive those outputs.

fread_as_container [on|off]	This setting is used for diagnostics with container files and file-specific commands like 'cat', 'hexdump', etc. Default is 'on'.
adc echo [on|off|esc]	This setting is used for diagnostics of internal voltage and temperature monitoring. Use 'on' or 'esc' to show AD values; 'esc' is used for position-fixed outputs.
adc cycle [<cyc_ns>]	This setting modifies the internal AD converter rate. It is not needed in normal use. Default is 256 measurements per second.

set	Show and modify environmental variables

usb tgl [on|off] | stick | rs232 | mode [lo|hi] | rst [lo|hi]	This setting determines the logger behavior upon connection of a USB cable. There are two options that are alternated between for every new connection. The first connection mode is called USB drive mode. The internal SD card is displayed as a USB drive on the PC; data can be exchanged. The micro controller cannot access the internal SD card in this mode. The second connection mode is called RS232 mode. In this mode, one or more new RS232 ports (depending on the GPS card used) are shown on the PC). The PC is connected directly to the GPS card using those virtual RS232 ports (Vcom port – virtual com port). In this mode, the micro controller is not affected. The selected connection mode is saved permanently.
	
	Example:
	`set usb stick`
	The logger always starts in USB drive mode.
	
	`set usb rs232`
	The logger always starts in RS232 mode.
	
	`set usb tgl on`
	The mode alternates with every new connection.

pwr [vcc3|gsm|sd] [on|off]	Control of power supplies for internal modules. This sub command is not needed in normal use.

sleep	Switch processor to sleep mode
	Use this command to switch the micro controller to sleep mode. The sensor's current consumption goes down to a minimum of approximately 160 ƒÊA. The sleep time <sleep_sec> is specified in seconds. Once the sleep time has passed, a software reset is issued. Processing or initialization of the GPS board is re-started. If you issue the command repeatedly without a parameter, the processor load is shown. The value indicates the frequency used to poll the separate tasks for activities by the cooperative multi-tasking. A high number means a low load on the processor and vice versa. The exact value depends on the logger's specification and configuration. It cannot be converted to a load percentage. Pure interrupt load can be determined using '**time wait -l 1000000**'. A queue measurement starts, with a loop time of 1 µs. The processor will only process interrupts while this command is executed. From the time required, you can determine the CPU load of all interrupts.
Parameters	**[[-nc] <sleep_sec>]** **// -nc don't close any open file**
-nc	Any open files will not be closed before switching to sleep mode. This means that the switch to sleep mode is not delayed. However, data loss could happen. In normal use of the 20xx GNSS receiver this parameter is not used.

English

sms	Send and receive text messages

	Use this command to send text messages (SMS) using the GSM radio. Simply enter the phone number to use, followed by the text to send. All text between the number and the end of line will be included.
	If you enter 'POS' as the text to send, this will be replaced by the current GPGGA and GPRMC messages without the leading $.
	You can send a file or a 160 characters long sample from a file. When sending a partial file you can set the file offset of the sample.
	
	There is no support for text spanning several messages.
	
	The sensor can check if the GSM radio has received any text messages. Texts are received automatically by the radio; the logger's micro controller is not needed for that. If the logger is turned off any messages are stored with your service provider. The next time the logger registers within the GSM network any stored text messages are received.
	Use the command **sms -ls** to poll the GSM radio explicitly. By default this happens every 17 s. You can change the configuration using '**gsm sms …**'. If texts have been received they are read and passed to the command interpreter. You can use any command you could enter at the command line. However, not all characters are valid within text messages.
	As '@' is not supported, it can be replaced by '!'.
	
	The logger can – within certain limits – be controlled using text messages. All received messages are logged in c:/sms/smsprot.txt. You can upload this file to an FTP server as shown below:
	
	Example:
	Uploading the text message log and creating a log of this upload
	
	`ftp put ! c:/sms/smsprot.txt >>c:/data/ftp.txt`
	`Example: Create and upload a list of directories`
	`ls c:/data/2012/06/14 >c:/data/ls120614.txt`
	`ftp put ! c:/data/ls120614.txt`
	
	Example:
	Downloading a new GPS configuration file
	
	`ftp get ! c:/sys/gps.cfg`
	
	Note:
	When entering multiple commands in one text message you need to include a LF at the end of every but the last command (press the enter key). A text may contain up to 160 characters. You can send several text messages to allow for more complex remote control.
	SMS use different character sets. Therefore we recommend to not using '@', '_', and other special characters.
	For ppmOS you can replace '@' with '!'. A '@' at the start of line cannot be replaced by this; however, you can simply leave it out, as it does not affect the execution of commands.

Parameters	**<nr> {<text>| POS |{@|!}<file> [<offset>]} | -ls**

sms	Send and receive text messages

	The 20xx GNSS receiver can – within certain limits – be controlled using text messages. All received messages are logged in c:/sms/smsprot.txt. You can upload this file to an FTP server as shown below:
	
	Example:
	Uploading the text message log and creating an FTP log of this upload
	
	`ftp put !c:/sys/upload.cfg c:/sms/smsprot.txt >>c:/data/ftp.txt`
	
	Example:
	Creating and uploading a list of directories
	
	`ls c:/data/2011/10/14 >c:/data/ls111014.txt`
	`ftp put !c:/sys/upload.cfg c:/data/ls111014.txt`
	
	Example:
	Downloading a new GNSS configuration file
	
	`ftp get -i !c:/sys/upload`

Note:	When entering multiple commands in one text message you need to include a LF at the end of every but the last command (press the enter key). A text may contain up to 160 characters. You can send several text messages to allow for more complex remote control.
Note:	SMS use different character sets. Therefore we recommend to not using '@', '_', and other special characters. For ppmOS you can replace '@' with '!'. A '@' at the start of line cannot be replaced by this; however, you can simply leave it out, as it does not affect the execution of commands.

stat	Output general software status
	Use this command to show general status information of the software.
Parameters	**[-f] //flush open files**
-f	Perform a flush (staging) of any data not yet saved. At the same time the directory entries of any opened files are updated. All logging uses RAM buffering. Only complete sectors are written to the SD card. The directory entry is only updated upon closing a file. When writing GPS data a directory flush is automatically performed every 32 KB or after a couple of seconds have passed.

Example

```
>stat
------------
heap: 0x542e free 2726 min 1464, SP 0x5ed4, txt 168954, dat 216, bss 3114,
alloc 10028, ctor 4, dtor 2
uart: rxlvl 1 (255) ovfl rx 0 tx 0, lines 337 esc_lines 0 unknown 1 intp_ovfl 0
open files: 2 max 4
C:4FF42B3F.J8G 2560 2560 77 00 1 11:38:55 04.07.12 11:38:40 04.07.12
C:STDOUT . 4266525 4266525 77 00 1 11:38:56 04.07.12 14:30:14 23.05.12
```

This shows that two files are opened. The maximum number of open files is 4. The first file has been opened at 11:38:40 and modified at 13:38:55; 2560 bytes have been saved. The full path is not shown. All the other output is used for diagnostics only.

tail	Show file contents, starting at the end
	Use this command to output the last 200 bytes of file <file> to the console port. An output forwarding is possible (see "Special characters on the command line" above). The command is most useful for ASCII files, e.g. to monitor a constantly growing log file.
Parameters	**[-c <bytes>] <file>**
-c <bytes>	Use this to select the number of <bytes> to output.

time	Measure command runtime
	Use this command to measure the runtime for any command or script. Time resolution is ca. 1 ms. This command is primarily intended for diagnostics during software development. You can determine write and read rates for SD cards with it. The command can be used recursively or in combination with loop.
Parameters	**<command>**

Example

```
>time wrfile -x512 -ma 10000 c:/data/testfile.txt
>wrfile -x512 -ma 10000 c:/data/testfile.txt
5120000 bytes written (should be 5120000) o.k.
cpu time 4.74804722.928710 s

>time csm c:/data/testfile.txt
>csm c:/data/testfile.txt
c:/data/testfile.txt len 5120000 csm 0xa22601f8
cpu time 4.0878918.734375 s
```

The example creates a file named c:/data/testfile.txt. The file has a size of 5120000 bytes. The micro controller needed 5.2 s at the current processor load. This leads to a write rate of 1078 KB/s. ppmOS achieves write rates of 800 KB/s to 1100 KB/s, depending on SD card type and manufacturer.

The second part of the example calculates a check sum for this file and measures the time required using `time`. So, the read rate is 1252 KB/s. ppmOS achieves read rates of 800 KB/s to 1250 kB/s, depending on type and manufacturer.

Note:

The read and write rates depend on the current defragmentation of the SD card. We recommend deleting or formatting SD cards before using them with the 20xx GNSS receiver. Make sure to first backup the folders sys, sms, and data!

ver	Output software version
	Use `ver` to show the ppmOS version. The command does not support any parameters.

Example

```
ppmOS V 1.26 compiled Jul 4 2012 1:48:23 PM UART-MSPI (GCC 4.6.2)
copyright ppm GmbH 2011
---------------------------------------------------------------
```

wait	Delay command execution
	Use `wait` to pause command execution for <delay>. <delay> without -l is given in s. You can enter a floating-point number (decimal point). If you specify -l the <delay> is given in microseconds. Without -l other tasks are processed. Use -l to pause task scheduling (however, interrupts are handled). Normally 'wait' is always used without -l.
Parameters	**[-l] <delay> // float sec, if -l simple delay loop in µs**
<delay>	<delay> indicates time in s. You can enter a floating-point number.

Example

>wait 3.5

```
start waiting 3500 ms
end wait
```

Command execution is paused for 3.5 s.

Configuration examples for sensors with NovAtel inside

This chapter contains a few configuration examples from different application areas. You can download these examples in the support area on our website **www.ppmgmbh.com** to modify them.

Configuration example 1

Determine a SBAS-based DGPS solution on the 20xx sensor and output this solution in NMEA format on a port.

We need two configuration files for this:

autoexec.sh

Commands	Comment
@echo-+-+-+-+-+-+-+-+-+--+	Outputs plus and minus signs to indicate the receiver startup.
@echo begin sys init	Outputs "begin sys init". This tells the user that the receiver is now processing the configuration files.
set pwr gsm off	Internal GSM radio switched off
gsm sms off	SMS call switched off
gps init	GPS Board initialization
cron rm all	All Cron Tab entries will be erased
gps log off	All open log files will be closed
@echo end sys init	Ouputs „end sys init“. This tells the user that the receiver has finished with processing the configuration files.

gps.cfg

Commands	Comment
GPS_Initialisation Novatel 0.1	Necassary identification line for NovAtel boards
	Optional: Commands for a reset and set to 'Factory Default':
com com1 9600 n 8 1 n off on	Set COM1 port of the interface board to 9600 baud
exec wait 0.5	System waits 0.5 sec to finish the command
exec gps bd gps1 9600	Set gps1 (=COM1) port of GPS board to 9600 baud
exec gps echo on	Terminal view switched on
freset command	Erase the existing configuration
exec wait 7	System waits 7 sec to finish the command
com com1 115200 n 8 1 n off on	Set COM1 port of the interface board to 115000 baud
exec wait 0.5	System waits 0.5 sec to finish the command
	Commands to configure the COM ports of CPU board and GPS board
exec gps bd gps1 115200	
com com2 115200 n 8 1 n off on	
exec wait 0.5	
exec gps bd gps2 115200	
exec bd com7 115200	
exec gps reply on	Acitivates data link from GPS board COM2 to 20xx-Sensor Port B
unlogall	All Logs switched off (f.e. raw data - or NMEA -messages)
ecutoff 10.0	Set elevations mask to 10°
ppscontrol enable negative 1.0 1000	Optional: Activiatation and configuration of 1PPS signal
log com1 rxstatuseventa onnew	Optional: Activiatation and configuration of Event-In
selectchanconfig 2	Optional: Configuration internal GPS channels, here: 12 x L1 GPS + 2 x SBAS !!! only valid for 2011-Sensor with internal NovAtel OEMStar board !!!

Commands	Comment
sbascontrol enable egnos	Configuration to use SBAS-corrections (here EGNOS)
	Note: all data on GPS port 2 will be transfered to 20xx Port COM B
	ASCII logs: Activation output NMEA and/or NovAtel ASCII messages
log com2 gpgga ontime 1	Turn on output NMEA-message GGA on Port B with 1 Hz output rate
log com2 gpvtg ontime 1	Turn on output NMEA-message VTG on Port B with 1 Hz output rate
log com2 gpgsv ontime 5	Turn on output NMEA-message GSV on Port B with 0,2 Hz output rate
log com2 bestposa ontime 0.1	Turn on NovAtel 'BESTPOS' ASCII message with 10Hz output rate
log comconfig	Output/View GPS COM-port configurations on 20xx-Sensor Port A
log loglist	Output/view all active logs on 20xx-Sensor Port A

Configuration example 2

Determine a Beacon-based DGPS solution on the 20xx sensor and output this solution in NMEA format on a port.

We need four configuration files for this:

autoexec.sh

Commands	Comment
@echo-+-+-+-+-+-+-+-+-+--+	Outputs plus and minus signs to indicate the receiver startup.
@echo begin sys init	Outputs "begin sys init". This tells the user that the receiver is now processing the configuration files.
set pwr gsm off	Internal GSM radio switched off
gsm sms off	SMS call switched off
gps init	GPS Board initialization
cron rm all	All Cron Tab entries will be erased
gps log off	All open log files will be closed
@echo end sys init	Ouputs „end sys init“. This tells the user that the receiver has finished with processing the configuration files.

gps.cfg

Commands	Comment
GPS_Initialisation Novatel 0.1	Necassary identification line for NovAtel boards
	Optional: Commands for a reset and set to 'Factory Default':
com com1 9600 n 8 1 n off on	Set COM1 port of the interface board to 9600 baud
exec wait 0.5	System waits 0.5 sec to finish the command
exec gps bd gps1 9600	Set gps1 (=COM1) port of GPS board to 9600 baud
exec gps echo on	Terminal view switched on
freset command	Erase the existing configuration
exec wait 7	System waits 7 sec to finish the command
com com1 115200 n 8 1 n off on	Set COM1 port of the interface board to 115000 baud
exec wait 0.5	System waits 0.5 sec to finish the command
	Commands to configure the COM ports of CPU board and GPS board
exec gps bd gps1 115200	
com com2 115200 n 8 1 n off on	
exec wait 0.5	
exec gps bd gps2 115200	
exec bd com7 115200	
exec gps reply on	Acitivates data link from GPS board COM2 to 20xx-Sensor Port B
unlogall	All Logs switched off (f.e. raw data - or NMEA -messages)
ecutoff 10.0	Set elevations mask to 10°
ppscontrol enable negative 1.0 1000	Optional: Activiatation and configuration of 1PPS signal
log com1 rxstatuseventa onnew	Optional: Activiatation and configuration of Event-In
interfacemode com2 rtcm novatel off	Configure GPS COM2 port to receive RTCM version 2.x correction data
psrdiffsource rtcm any	Specify the format of correction data, here: RTCM version 2.x
sbascontrol enable egnos	Activates EGNOS receiption and usage as backup system if no Beacon signals are available
	Note: all data on GPS port 2 will be transfered to 20xx Port COM B
	ASCII logs: Activation output NMEA and/or NovAtel ASCII messages

Commands	Comment
log com2 gpgga ontime 1	Turn on output NMEA-message GGA on Port B with 1 Hz output rate
log com2 gpvtg ontime 1	Turn on output NMEA-message VTG on Port B with 1 Hz output rate
log com2 gpgsv ontime 5	Turn on output NMEA-message GSV on Port B with 0,2 Hz output rate
log com2 bestposa ontime 0.1	Turn on NovAtel 'BESTPOS' ASCII message with 10Hz output rate
log comconfig	Output/View GPS COM-port configurations on 20xx-Sensor Port A
log loglist	Output/view all active logs on 20xx-Sensor Port A

firstfix.sh

Commands	Comment
	Note: The file 'firstfix.sh' is a script starting once after the first GPS position fix
@echo ----------	Outputs '----------' in terminal software view
@date -s	Output of current date and time (UTC time) in terminal software view
first	Internal status request to beacon board The answer will be not visible on teh commando line Received correction data will be transfered to the GPS board via the CPU

beacon.cfg

Commands	Comments
	Note: This file will configure the internal Marine Beacon board.
exec @echo reading beacon.cfg	Output/View of 'reading beacon.cfg' on port A
exec @echo ----------	Output/View of '----------' on port A
$GPMSK,314.5,M,100,M,0	Configuration to receive a Marine Beacon transmitter which is not stored here: the German station „Bad Abbach“
	Note: Alternative beacon stations within Germany: You can find the right values for your country here: http://www.trimble.com/findbeacon.asp
$GPMSK,293.5,M,100,M,0	Iffezheim
$GPMSK,302.5,M,100,M,0	Koblenz
$GPMSK,313.5,M,100,M,0	Mauken
$GPMSK,298.5,M,100,M,0	Helgoland
$GPMSK,303.5,M,100,M,0	Zeven
$GPMSK,308.0,M,100,M,0	Mohrdorf
exec @echo end beacon.cfg read	Output/View of 'end beacon.cfg read' on port A

Configuration example 3

Determine a NTRIP-based RTK solution on the 20xx sensor and output this solution in NMEA format on a port.

We need four configuration files for this:

autoexec.sh

Commands	Comment
@echo-+-+-+-+-+-+-+-+-+--+	Outputs plus and minus signs to indicate the receiver startup.
@echo begin sys init	Outputs "begin sys init". This tells the user that the receiver is now processing the configuration files.
gsm sms off	SMS call switched off
gps init	GPS Board initialization
cron rm all	All Cron Tab entries will be erased
gps log off	All open log files will be closed
@echo end sys init	Ouputs „end sys init“. This tells the user that the receiver has finished with processing the configuration files.

gps.cfg

Commands	Comment
GPS_Initialisation Novatel 0.1	Necassary identification line for NovAtel boards
	Optional: Commands for a reset and set to 'Factory Default':
com com1 9600 n 8 1 n off on	Set COM1 port of the interface board to 9600 baud
exec wait 0.5	System waits 0.5 sec to finish the command
exec gps bd gps1 9600	Set gps1 (=COM1) port of GPS board to 9600 baud
exec gps echo on	Terminal view switched on
freset command	Erase the existing configuration
exec wait 7	System waits 7 sec to finish the command
com com1 115200 n 8 1 n off on	Set COM1 port of the interface board to 115000 baud
exec wait 0.5	System waits 0.5 sec to finish the command
	Commands to configure the COM ports of CPU board and GPS board
exec gps bd gps1 115200	
com com2 115200 n 8 1 n off on	
exec wait 0.5	
exec gps bd gps2 115200	
exec bd com7 115200	
exec gps reply on	Acitivates data link from GPS board COM2 to 20xx-Sensor Port B
unlogall	All Logs switched off (f.e. raw data - or NMEA -messages)
ecutoff 10.0	Set elevations mask to 10°
ppscontrol enable negative 1.0 1000	Optional: Activiatation and configuration of 1PPS signal
log com1 rxstatuseventa onnew	Optional: Activiatation and configuration of Event-In
	Configuration positioning mode here: RTK with NTRIP via internal GPRS radio
interfacemode com2 rtcmv3 novatel off	Configuration GPS COM2 port to receive RTCM version 3.x correction data
rtksource rtcmv3 any	Activation of RTK calculation with RTCM version 3.x correction data
psrdiffsource auto any	Activation of DGPS positioning mode as first backup here: all correction data formats
sbascontrol enable egnos	Activation of EGNOS receiption and usage as second backup

Commands	Comment
	Note: all data on GPS port 2 will be transfered to 20xx Port COM B
log com2 gpgga ontime 0.1	ASCII logs: Activation output NMEA and/or NovAtel ASCII messages
log com2 gpvtg ontime 1	
log com2 gpgsv ontime 5	Turn on output NMEA-message GGA on Port B with 1 Hz output rate
log com2 bestposa ontime 0.1	Turn on output NMEA-message VTG on Port B with 1 Hz output rate
log comconfig	Turn on output NMEA-message GSV on Port B with 0,2 Hz output rate
log loglist	Turn on NovAtel 'BESTPOS' ASCII message with 10Hz output rate

firstfix.sh

Commands	Comment
	Note: The file 'firstfix.sh' is a script starting once after the first GPS position fix
@echo ----------	Outputs '----------' in terminal software view
@date -s	Output of current date and time (UTC time) in terminal software view
	Creating a cron job
cron put */33 * * * * * * ntrip open	Cron job opens every 33 seconds the file ntrip.cfg

upload.cfg

Commands	Comment
	Note: This file configures the internal GSM/GPRS module
UploadConfigFile v 0.1	Activation SIM card in 20xx receiver
exec @echo reading from upload.cfg	Output „reading from upload.cfg“ on port A (view in terminal software)
PIN 1234	Pin code for SIM-card Note: We recommend to deactivate the PIN code. But please do not remove this line.
apn internet.t-mobile	APN adress of cellphone provider (here T-Mobile D) (see also GPRS access data on page 15)
user td1	
passwd td1	
exec @echo reading upload.cfg finished	Output „reading upload.cfg finished“ on port A (view in terminal software)

ntrip.cfg

Commands	Comment
	Note: This file configures the NTRIP connection
`exec @echo reading from ntrip.cfg`	Output „reading from ntrip.cfg“ on Port A (view in terminal software)
`address socktcp://195.200.71.81:2101`	IP adress of NTRIP provider Note: Use IP adress only, no URL adress !!
`GET /VRS_3_BY HTTP/1.0` `User-Agent: NTRIP XS/1.14`	Define mountpoint
`Authorization: Basic` `TXVzdGVybWFubjp3ZWxjb21l`	Username and passwort with base64 code
`exec @echo ntrip.cfg scan completed`	Output „ntrip.cfg scan completed“ on port A (view in terminal software)

Configuration example 4

Determine a 20xx sensor should store PVT (Position-Velocity-Time) - data on the internal memory.

We need three configuration files for this:

autoexec.sh

Commands	Comment
`@echo-+-+-+-+-+-+-+-+-+--+`	Outputs plus and minus signs to indicate the receiver startup.
`@echo begin sys init`	Outputs "begin sys init". This tells the user that the receiver is now processing the configuration files.
`set pwr gsm off`	Internal GSM radio switched off
`gps init`	GPS Board initialization
`cron rm all`	All Cron Tab entries will be erased
`gps log off`	All open log files will be closed
`@echo end sys init`	Ouputs „end sys init“. This tells the user that the receiver has finished with processing the configuration files.

gps.cfg

Commands	Comment
`GPS_Initialisation Novatel 0.1`	necassary identification line for NovAtel boards
	Optional: Commands for a reset and set to 'Factory Default':
com com1 9600 n 8 1 n off on	Set COM1 port of the interface board to 9600 baud
exec wait 0.5	System waits 0.5 sec to finish the command
exec gps bd gps1 9600	Set gps1 (=COM1) port of GPS board to 9600 baud
exec gps echo on	Terminal view switched on
freset command	Erase the existing configuration
exec wait 7	System waits 7 sec to finish the command
com com1 115200 n 8 1 n off on	Set COM1 port of the interface board to 115000 baud
exec wait 0.5	System waits 0.5 sec to finish the command
	Commands to configure the COM ports of CPU board and GPS board
`exec gps bd gps1 115200`	
`com com2 115200 n 8 1 n off on`	
`exec wait 0.5`	
`exec gps bd gps2 115200`	
`exec bd com7 115200`	
`exec gps reply on`	Acitivates data link from GPS board COM2 to 20xx-Sensor Port B
`unlogall`	All Logs switched off (f.e. raw data - or NMEA -messages)
`ecutoff 10.0`	Set elevations mask to 10°
ppscontrol enable negative 1.0 1000	Optional: Activiatation and configuration of 1PPS signal
log com1 rxstatuseventa onnew	Optional: Activiatation and configuration of Event-In
	Note : all data activated on GPS COM2 port will be stored on the SD card if command `gps log on` was executed
	ASCII logs: Activation output NMEA and/or NovAtel ASCII messages

Commands	Comment
log com2 gpgga ontime 0.1	Turn on output NMEA-message GGA on Port B with 1 Hz output rate
log com2 gpvtg ontime 1	Turn on output NMEA-message VTG on Port B with 1 Hz output rate
log com2 gpgsv ontime 5	Turn on output NMEA-message GSV on Port B with 0,2 Hz output rate
log com2 bestposa ontime 0.1	Turn on NovAtel 'BESTPOS' ASCII message with 10Hz output rate
log comconfig	Output/View GPS COM-port configurations on 20xx-Sensor Port A
log loglist	Output/view all active logs on 20xx-Sensor Port A
	Define pattern for file naming
exec @gps set filename_pattern UTC_HEX	File name: UTC as 8 digit hex number and extension with 3 letters with serial number of GPS board
exec @gps set filename_pattern ASH_ATM	Note: alternative: File name fits to the Ashtech Atom naming to convert to Rinex
exec @gps set filename_pattern 2LETTER_UTC TB	Note: alternative: File name starts with 2 letters (free of choice) following 2digits for the day and 2 digits for the seconds

firstfix.sh

Commands	Comment
	Note: The file 'firstfix.sh' is a script starting once after the first GPS position fix
@echo ----------	Outputs '----------' in terminal software view
@date -s	Output of current date and time (UTC time) in terminal software view
cron rm all	Deletes all existing CRON Tab entries
gps log on	Opens once a new log file after the first GPS position fix immediate
cron put 0 0 * * * 12-30 * "gps log off; gps log on -dont_cp_cfg"	Note: alternative: Define the peride for execution of the cron job, here very full hour: cron put 0 0 * * * 12-30 * Close the current file: gps log off Opens a new file: gps log on -dont_cp_cfg

Configuration example 5

Determine a 20xx sensor should store raw dat (to be converted in Rinex format) data on the internal memory.

We need four configuration files for this:

autoexec.sh

Commands	Comment
@echo-+-+-+-+-+-+-+-+-+--+	Outputs plus and minus signs to indicate the receiver startup.
@echo begin sys init	Outputs "begin sys init". This tells the user that the receiver is now processing the configuration files.
set pwr gsm off	Internal GSM radio switched off
gps init	GPS Board initialization
cron rm all	All Cron Tab entries will be erased
gps log off	All open log files will be closed
@echo end sys init	Ouputs „end sys init“. This tells the user that the receiver has finished with processing the configuration files.

gps.cfg

Commands	Comment
GPS_Initialisation Novatel 0.1	Necassary identification line for NovAtel boards
	Optional: Commands for a reset and set to 'Factory Default':
com com1 9600 n 8 1 n off on	Set COM1 port of the interface board to 9600 baud
exec wait 0.5	System waits 0.5 sec to finish the command
exec gps bd gps1 9600	Set gps1 (=COM1) port of GPS board to 9600 baud
exec gps echo on	Terminal view switched on
freset command	Erase the existing configuration
exec wait 7	System waits 7 sec to finish the command
com com1 115200 n 8 1 n off on	Set COM1 port of the interface board to 115000 baud
exec wait 0.5	System waits 0.5 sec to finish the command
	Commands to configure the COM ports of CPU board and GPS board
exec gps bd gps1 115200	
com com2 115200 n 8 1 n off on	
exec wait 0.5	
exec gps bd gps2 115200	
exec bd com7 115200	
exec gps reply on	Acitivates data link from GPS board COM2 to 20xx-Sensor Port B
unlogall	All Logs switched off (f.e. raw data - or NMEA -messages)
ecutoff 10.0	Set elevations mask to 10°
ppscontrol enable negative 1.0 1000	Optional: Activiatation and configuration of 1PPS signal
log com1 rxstatuseventa onnew	Optional: Activiatation and configuration of Event-In
	Note : all data activated on GPS COM2 port will be stored on the SD card if command gps log on was executed
	ASCII logs: Activate message types for raw data outpute
log com2 rawephemb onnew	Activates the GPS ephemerides raw data on GPS COM2 port with the output rate: on new

Commands	Comment
log com2 glorawephemb onnew	Activates the GLONASS ephemerides raw data on GPS COM2 port with the output rate: on new
log com2 rangecmpb ontime 10	Activates code and phase raw data outpunt on GPS COM2 port with the output rate: 0.1Hz
log com2 ionutcb onnew	Activates the Ionospheric raw data on GPS COM2 Port with the output rate: on new
log comconfig	Output/View GPS COM port configurationen on 20xx-sensor port to view it in terminal software
log loglist	Output/View all active logs on 20xx-sensor port A to view it in terminal software
	Define pattern for file naming
exec @gps set filename_pattern UTC_HEX	File name: UTC as 8 digit hex number and extension with 3 letters with serial number of GPS board
exec @gps set filename_pattern ASH_ATM	Note: alternative: File name fits to the Ashtech Atom naming to convert to Rinex
exec @gps set filename_pattern 2LETTER_UTC TB	Note: alternative: File name starts with 2 letters (free of choice) following 2digits for the day and 2 digits for the seconds

firstfix.sh

Commands	Comment
	Note: The file 'firstfix.sh' is a script starting once after the first GPS position fix
@echo ----------	Outputs '----------' in terminal software view
@echo Start Data Storage	Output 'Start Data Storage' to view it in terminal software
@date -s	Output of current date and time (UTC time) in terminal software view
cron rm all	Deletes all existing CRON Tab entries
gps log on	Opens once a new log file after the first GPS position fix immediate
cron put 0 0 * * * 12-30 * "gps log off; gps log on -dont_cp_cfg; c:/sys/rnx_data.sh"	Define the peride for execution of the cron job, here very full hour: cron put 0 0 * * * 12-30 * Close the current file: gps log off Opens a new file: gps log on -dont_cp_cfg Call a script file to execute GPS status messages for für RINEX files: c:/sys/rnx_data.sh
gps <log com2 bestposb once	Call a position data (for RINEX header): BestPos data, interval: once
gps <log com2 versionb once	Call receiver information (type, serial number,..., for RINEX header): BestPos data, interval: once
gps echo off	Switch off GPS echo (switch off output of any GPS data on terminal port)

rnx_data.sh

Befehle	Kommentar
`gps <log com2 bestposb once`	Call a position data (for RINEX header): BestPos data, interval: once
`gps <log com2 versionb once`	Call receiver information (type, serial number,..., for RINEX header): BestPos data, interval: once
`gps echo off`	Switch off GPS echo (switch off output of any GPS data on terminal port)

Configuration example 6

Determine a 20xx sensor should store raw dat (to be converted in Rinex format) data on the internal memory and send them to a ftp server.

We need five configuration files for this:

autoexec.sh

Commands	Comment
@echo-+-+-+-+-+-+-+-+-+--+	Outputs plus and minus signs to indicate the receiver startup.
@echo begin sys init	Outputs "begin sys init". This tells the user that the receiver is now processing the configuration files.
gsm sms off	SMS call switched off
gps init	GPS Board initialization
cron rm all	All Cron Tab entries will be erased
gps log off	All open log files will be closed
set usb stick	
@echo end sys init	Ouputs „end sys init“. This tells the user that the receiver has finished with processing the configuration files.

gps.cfg

Commands	Comment
GPS_Initialisation Novatel 0.1	Necassary identification line for NovAtel boards
	Optional: Commands for a reset and set to 'Factory Default':
com com1 9600 n 8 1 n off on	Set COM1 port of the interface board to 9600 baud
exec wait 0.5	System waits 0.5 sec to finish the command
exec gps bd gps1 9600	Set gps1 (=COM1) port of GPS board to 9600 baud
exec gps echo on	Terminal view switched on
freset command	Erase the existing configuration
exec wait 7	System waits 7 sec to finish the command
com com1 115200 n 8 1 n off on	Set COM1 port of the interface board to 115000 baud
exec wait 0.5	System waits 0.5 sec to finish the command
	Commands to configure the COM ports of CPU board and GPS board
exec gps bd gps1 115200	
com com2 115200 n 8 1 n off on	
exec wait 0.5	
exec gps bd gps2 115200	
exec bd com7 115200	
exec gps reply on	Acitivates data link from GPS board COM2 to 20xx-Sensor Port B
unlogall	All Logs switched off (f.e. raw data - or NMEA -messages)
ecutoff 10.0	Set elevations mask to 10°
ppscontrol enable negative 1.0 1000	Optional: Activiatation and configuration of 1PPS signal
log com1 rxstatuseventa onnew	Optional: Activiatation and configuration of Event-In
	Note : all data activated on GPS COM2 port will be stored on the SD card if command gps log on was executed
	ASCII logs: Activate message types for raw data outpute

Commands	Comment
log com2 rawephemb onnew	Activates the GPS ephemerides raw data on GPS COM2 port with the output rate: on new
log com2 glorawephemb onnew	Activates the GLONASS ephemerides raw data on GPS COM2 port with the output rate: on new
log com2 rangecmpb ontime 10	Activates code and phase raw data outpunt on GPS COM2 port with the output rate: 0.1Hz
log com2 ionutcb onnew	Activates the Ionospheric raw data on GPS COM2 Port with the output rate: on new
log comconfig	Output/View GPS COM port configurationen on 20xx-sensor port to view it in terminal software
log loglist	Output/View all active logs on 20xx-sensor port A to view it in terminal software
	Define pattern for file naming
exec @gps set filename_pattern UTC_HEX	File name: UTC as 8 digit hex number and extension with 3 letters with serial number of GPS board
exec @gps set filename_pattern ASH_ATM	Note: alternative: File name fits to the Ashtech Atom naming to convert to Rinex
exec @gps set filename_pattern 2LETTER_UTC TB	Note: alternative: File name starts with 2 letters (free of choice) following 2digits for the day and 2 digits for the seconds

firstfix.sh

Commands	Comment
	Note: The file 'firstfix.sh' is a script starting once after the first GPS position fix
@echo ----------	Outputs '----------' in terminal software view
@echo Starte die Datenaufzeichung	Output 'Start Data Storage' to view it in terminal software
@date -s	Output of current date and time (UTC time) in terminal software view
cron rm all	Deletes all existing CRON Tab entries
gps log on	Opens once a new log file after the first GPS position fix immediate
cron put 0 0 * * * 12-30 * "gps log off ftp put -try 9 -rm ! $(fname); gps log on -dont_cp_cfg; c:/sys/rnx_data.sh"	Define the peride for execution of the cron job, here very full hour: cron put 0 0 * * * 12-30 * Close the current file: gps log off Upload of that closed file (9 times trials) to a FTP drive: ftp put -try 9 If the upload was succesful the file will be erased: -rm ! $(fname) Opens a new file: gps log on -dont_cp_cfg Call a script file to execute GPS status messages for für RINEX files: c:/sys/rnx_data.sh

Commands	Comment
gps <log com2 bestposb once	Call a position data (for RINEX header): BestPos data, interval: once
gps <log com2 versionb once	Call receiver information (type, serial number,..., for RINEX header): BestPos data, interval: once
gps echo off	Switch off GPS echo (switch off output of any GPS data on terminal port)

rnx_data.sh

Commands	Comment
gps <log com2 bestposb once	Call a position data (for RINEX header): BestPos data, interval: once
gps <log com2 versionb once	Call receiver information (type, serial number,..., for RINEX header): BestPos data, interval: once
gps echo off	Switch off GPS echo (switch off output of any GPS data on terminal port)

English

Declaration of Conformity

according to the guideline 2004/108/EG

The manufacture

ppm Precise Positioning Management GmbH
Grube 39a
82377 Penzberg

Declare under our sole responsablitiy that the product

ppm 20xx GNSS sensor

Where appropriate, are in conformity with the following harmonized standards:

EC Directives:

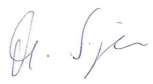
2004/108/EG
2002/96/EG
2002/95/EG

Immunity and Safety:

EN 301 489-V1.8.1
EN 60950-1

Optional enthalten sind Funkmodem Cinterion MC55i (CE 0682)

Penzberg, 17.09.2011



Michael Singer
Managing Director