ppm

Precise Positioning Management

40xx

GNSS SENSOR SERIE
REFERENZHANDBUCH

Rechtliches

Copyright

Copyright © 2021 ppm GmbH. Alle Rechte vorbehalten.

Kein Teil dieser Veröffentlichung oder der darin beschriebenen Computerprogramme darf ohne die vorherige schriftliche Genehmigung durch ppm GmbH reproduziert, übersetzt, in einem Zugangssystem gespeichert oder in irgendeiner Form oder auf irgendeine Weise übertragen werden, sei es elektronisch, durch mechanisches Fotokopieren, Aufnehmen oder in sonstiger Weise. Ihre Rechte im Zusammenhang mit dieser Veröffentlichung und den Computerprogrammen unterliegen den Einschränkungen und Grenzen der Urhebergesetze der Europäischen Union und/oder der Rechtsordnung, die an Ihrem Aufenthaltsort gilt.

Gedruckt in Deutschland.

Teilnummer: 40xx GNSS Sensor Handbuch deutsch Version B

August 2021

Warenzeichen

Alle Produkte und Markennamen sind Warenzeichen oder eingetragene Warenzeichen ihrer jeweiligen Inhaber.

Software Lizenzvereinbarung

WICHTIG: DURCH INSTALLIEREN DER SOFTWARE ERKLÄREN SIE SICH DAMIT EINVERSTANDEN, SICH DEN BEDINGUNGEN DER LIZENZVEREINBARUNG ("VEREINBARUNG") ZU UNTERWERFEN. DIESE VEREINBARUNG STELLT DAS GESAMTE VERTRAGSWERK ZWISCHEN IHNEN ("LIZENZNEHMER") UND DER PPM GMBH DAR. ("LIZENZGEBER"). LESEN SIE DIE VEREINBARUNG SORGFÄLTIG DURCH; WENN SIE MIT DEN BEDINGUNGEN NICHT EINVERSTANDEN SIND, GEBEN SIE DAS UNGEÖFFNETE CD-PAKET UND DIE MITGELIEFERTEN GEGENSTÄNDE DORT ZURÜCK, WO SIE DIESE ERWORBEN HABEN. DER KAUFPREIS WIRD IHNEN VOLLSTÄNDIG ZURÜCKERSTATTET.

LIZENZ

Der LIZENZGEBER gewährt Ihnen eine eingeschränkte, nicht ausschließliche, nicht übertragbare persönliche Lizenz ("Lizenz") dafür, die in diesem Paket in maschinenlesbarer Form enthaltene Kopie des Computerprogramms ("Programm") auf einem einzelnen Computer (einer zentralen Prozessoreinheit mit dazugehörigem Bildschirm und Tastatur) zu installieren und zu benutzen und eine Sicherungskopie des Programms zur Verwendung mit demselben Computer zu erstellen. Der LIZENZGEBER und seine Zulieferer behalten alle Rechte an dem Programm, die in dieser Vereinbarung nicht ausdrücklich übertragen werden.

EIGENTUM AN PROGRAMMEN UND KOPIEN

Diese Lizenz ist kein Verkauf des Originalprogramms oder irgendwelcher Kopien. Der LIZENZGEBER und seine Zulieferer behalten das Besitztum an dem Programm und alle Urheberrechte und sonstigen Eigentumsrechte daran sowie an allen später von Ihnen erstellten Kopien des Programms, unabhängig von deren Form. Das Programm und die begleitenden Bedienungsanleitungen ("Dokumentation") sind urheberrechtlich geschützte Werke und enthalten wertvolle gewerbliche Geheimnisse und vertrauliche Informationen, die dem LIZENZGEBER und dessen Zulieferern gehören. Sie erklären sich damit einverstanden, alle sinnvollen Maßnahmen zu ergreifen, um die Eigentumsrechte des LIZENZGEBERS und seiner Zulieferer an dem Programm und der Dokumentation zu schützen und sie streng vertraulich zu halten.

NUTZUNGSEINSCHRÄNKUNGEN

Das Programm wird zur Verwendung bei Ihrer internen Geschäftstätigkeit geliefert und muss jederzeit auf einem einzelnen Computer bleiben, der Ihnen gehört oder den Sie mieten. Sie dürfen das Programm physisch von einem Computer auf einen anderen übertragen, vorausgesetzt, das Programm wird immer nur auf einem Computer gleichzeitig benutzt. Sie dürfen das Programm ohne die vorherige schriftliche Genehmigung des LIZENZGEBERS nicht im Rahmen von Time-Sharing oder der Vermietung voll ausgestatteter Büros benutzen oder vermieten, verleasen, weitervermieten, verkaufen, abtreten, verpfänden, übertragen, elektronisch übertragen oder auf sonstige Weise das Programm oder die Dokumentation weder vorübergehend noch dauerhaft veräußern. Sie erklären sich damit einverstanden, das Programm nicht zu übersetzen, ändern, anzupassen, zerlegen, dekompilieren oder zurückzuentwickeln oder abgeleitete Werke von dem Programm oder der Dokumentation oder eines Teiles davon zu erstellen.

Rechtliches

BEENDIGUNG.

Die Lizenz gilt bis zur Beendigung. Die Lizenz endet ohne Kündigung durch den LIZENZGEBER, wenn Sie gegen irgendeine der Bestimmungen dieser Vereinbarung verstoßen. Bei der Beendigung müssen Sie jegliche Nutzung des Programms und der Dokumentation einstellen und diese sowie sämtliche Kopien davon an den LIZENZGEBER zurückgeben.

Garantieausschluss und Haftungsbeschränkung

DER LIZENZGEBER UND SEINE ZULIEFERER GEBEN WEDER AUSDRÜCKLICH NOCH INDIREKT GEWÄHRLEISTUNGEN FÜR DAS PROGRAMM, DIE MEDIEN, DIE DOKUMENTATION, DIE ERGEBNISSE ODER DIE GENAUIGKEIT DER DATEN UND SCHLIESSEN HIERMIT AUSDRÜCKLICH JEGLICHE GARANTIE DER VERMARKTBARKEIT UND DER EIGNUNG FÜR EINEN BESTIMMTEN ZWECK AUS. DER LIZENZGEBER UND SEINE ZULIEFERER GARANTIEREN NICHT, DASS DAS PROGRAMM IHREN ANFORDERUNGEN ENTSPRICHT ODER DASS SEIN BETRIEB OHNE UNTERBRECHUNGEN ODER FEHLERFREI IST.

Der LIZENZGEBER, seine Zulieferer sowie jeder, der an der Erstellung oder Lieferung des Programms oder der Dokumentation an Sie beteiligt war, haftet Ihnen oder einem Dritten gegenüber nicht für besondere, indirekte, Neben- oder Folgeschäden (einschließlich, aber nicht

beschränkt auf entgangene Gewinne oder Einsparungen, Ausfallzeiten, Schäden an oder Ersatz von Ausrüstungs- oder Eigentumsgegenständen, Wiederherstellung oder Ersatz von Programmen oder Daten), die aus Forderungen aus Gewährleistung, Vertrag oder unerlaubter Handlung (einschließlich Fahrlässigkeit), verschuldensunabhängiger Haftung oder auf sonstige Weise entstehen, selbst dann, wenn der LIZENZGEBER oder seine Zulieferer über die Möglichkeit einer solchen Forderung oder solcher Schäden unterrichtet wurden. Die Haftung des LIZENZGEBERS und seiner Zulieferer für direkte Schäden ist begrenzt auf den tatsächlich für diese Programmlizenz bezahlten Betrag. Manche Staaten gestatten den Ausschluss der Begrenzung impliziter Garantien oder Haftung für Neben- oder Folgeschäden nicht, die o. g. Beschränkungen oder Ausschlüsse gelten daher eventuell nicht für Sie.

BESCHRÄNKTE PRODUKTGARANTIE

Alle Empfangsgeräte für das globale Positionsbestimmungssystem (GPS) von ppm sind Navigationshilfen und nicht dazu gedacht, andere Navigationsmethoden zu ersetzen. Dem Käufer wird angeraten, eine sorgfältige Positionsbestimmung durchzuführen und gesunden Menschenverstand walten zu lassen.

LESEN SIE VOR DER BENUTZUNG DES PRODUKTS DIE GEBRAUCHSANLEITUNG SORGFÄLTIG DURCH.

1. GARANTIE DURCH PPM

ppm garantiert, dass unsere GNSS-Empfänger und Hardware-Zubehör keine Material- und Herstellungsfehler aufweisen, und leistet für das Produkt gemäß den veröffentlichten Daten eine Garantie von einem Jahr (oder für einen durch das Gesetz geforderten längeren Zeitraum), gerechnet vom Datum des ursprünglichen Kaufs.

DIESE GARANTIE BEZIEHT SICH NUR AUF DEN URSPRÜNGLICHEN KÄUFER DIESES PRODUKTS.

Im Fall eines Defekts wird ppm das Hardware-Produkt nach eigenem Ermessen entweder reparieren oder ersetzen, ohne dem Käufer Ersatzteile oder Arbeitszeit in Rechnung zu stellen. Für das reparierte oder ersetzte Produkt wird eine Garantie von 90 Tagen ab dem Rücksendedatum, mindestens aber bis zum Ablauf der ursprünglichen Garantie, gewährt. ppm sichert zu, dass die Softwareprodukte oder in Hardwareprodukten enthaltene Software ab dem Versanddatum 30 Tage in den Medien fehlerfrei sind und dass sie im Wesentlichen der dann gültigen Anwenderdokumentation entsprechen, die mit der Software (einschließlich deren Aktualisierungen) geliefert wurde. ppm ist einzig zur Korrektur oder dem Ersatz der Medien oder der Software verpflichtet, so dass sie der dann gültigen Anwenderdokumentation im Wesentlichen entsprechen. ppm sichert nicht zu, dass die Software den Anforderungen des Käufers entspricht, oder dass ihr Betrieb unterbrechungsfrei, fehlerfrei oder frei von Viren bleibt. Der Käufer übernimmt für die Benutzung der Software das volle Risiko.

2. RECHTSMITTEL DES KÄUFERS

DAS AUSSCHLIESSLICHE RECHTSMITTEL DES KÄUFERS UNTER DIESER GARANTIE ODER UNTER EINER IMPLIZITEN GARANTIE IST, JE NACH ENTSCHEIDUNG VON PPM, AUF REPARATUR ODER ERSATZ DES EMPFÄNGERS ODER DER ZUBEHÖRTEILE BESCHRÄNKT, DIE VON DIESER GARANTIE ABGEDECKT SIND. REPARATUREN IM RAHMEN DIESER GARANTIE DÜRFEN NUR IN

EINEM VON PPM AUTORISIERTEN KUNDENDIENSTZENTRUM DURCHGEFÜHRT WERDEN. JEDE REPARATUR DURCH EIN NICHT VON PPM AUTORISIERTES KUNDENDIENSTZENTRUM FÜHRT ZUM ERLÖSCHEN DER GARANTIE.

Rechtliches

3. PFLICHTEN DES KÄUFERS

Um den Service in Anspruch zu nehmen, wenden Sie sich an den Händler, bei dem Sie das Produkt gekauft haben, und geben Sie das Produkt mit einer Kopie der Originalrechnung an ihn zurück. ppm behält sich das Recht vor, kostenlosen Service zu verweigern, wenn der Kaufnachweis nicht vorgelegt wird, oder die in ihm enthaltenen Informationen unvollständig oder unleserlich sind, oder wenn die Seriennummer verändert oder entfernt wurde. ppm haftet nicht für Verluste oder Schäden am Produkt, die während des Lieferwegs des Produkts oder bei seiner Einsendung zur Reparatur auftreten. Der Abschluss einer Transportversicherung wird empfohlen. ppm empfiehlt einen nachvollziehbaren Lieferweg wie UPS oder FedEx für die Rücksendung des Produkts zum Service.

4. EINSCHRÄNKUNG VON IMPLIZITEN GARANTIEN

MIT AUSNAHME DER OBEN IN PUNKT 1 DARGELEGTEN BESCHRÄNKTEN GARANTIE WIRD HIERMIT JEGLICHE DARÜBER HINAUSGEHENDE GEWÄHRLEISTUNG AUSGESCHLOSSEN. DAS GILT SOWOHL FÜR AUSDRÜCKLICHE ALS AUCH IMPLIZITE GARANTIEN, EINSCHLIEßLICH DER ZUSICHERUNG DER EIGNUNG FÜR EINEN BESTIMMTEN ZWECK ODER DER MARKTFÄHIGKEIT UND, SOWEIT ANWENDBAR, FÜR IMPLIZITE GARANTIEN GEMÄSS ARTIKEL 35 DER UN KONVENTION ÜBER DEN INTERNATIONALEN WARENKAUF. Einige nationale, Staats-, oder lokale Gesetze gestatten keinen Ausschluss oder Einschränkungen bei Neben-oder Folgeschäden. In solchen Fällen trifft die obige Einschränkung oder der Ausschluss nicht auf sie zu.

5. AUSSCHLIESSUNGEN

Folgendes ist von der Garantie ausgeschlossen:

(1) regelmäßige Wartung und Reparatur oder Ersatz von Teilen aufgrund normaler Abnutzung;

(2) Batterien und Akkus

(3) Oberflächeneigenschaften

(4) Installationen oder Defekte aufgrund der Installation;

(5) jeder Schaden, durch

(i) den Versand, Zweckentfremdung, Missbrauch, Nachlässigkeit, Eingriffe, oder nicht ordnungsgemäße Anwendung;

(ii) Unglücke wie Feuer, Flut, Wind und Blitzschlag;

(iii) nicht autorisierte Hinzufügungen oder Modifizierungen;

(6) einen von einem nicht durch ppm autorisierten Kundendienstzentrum durchgeführten oder versuchten Service;

(7) Produkte, Komponenten oder Teile, die nicht von ppm hergestellt wurden,

(8) die Zusicherung, dass der Empfänger frei von jedem Anspruch aus der Verletzung eines Patents, einer Handelsmarke, eines Copyrights oder anderen Eigentumsrechts einschließlich von Handelsgeheimnissen ist;

(9) jeder Schaden aufgrund eines Unfalls, der durch ungenaue Satellitenübertragungen entsteht. Ungenaue Übertragungen können durch Veränderungen der Position, des Betriebszustands oder der Geometrie eines Satelliten oder durch Veränderungen an dem Empfänger auftreten, die durch eine Veränderung an dem GPS erforderlich werden können. (Anmerkung: ppm GNSS Empfänger verwenden zum Empfang der Daten über Position, Geschwindigkeit und Zeit GPS oder GPS+GLONASS. GPS wird von der US-Regierung betrieben; GLONASS ist das globale Satelliten-Navigationssystem der Russischen Föderation. Beide sind allein für Fehlerfreiheit und Wartung des jeweiligen Systems zuständig. Bestimmte Bedingungen können Ungenauigkeiten verursachen, welche Modifikationen am Empfänger erforderlich machen. Solche Bedingungen liegen insbesondere bei Veränderungen in der Übertragung von GPS oder GLONASS vor.) Das Öffnen, Zerlegen oder die Reparatur dieses Produkts durch andere als ein von ppm autorisiertes Kundendienstzentrum führt zum Erlöschen der Garantie

6. AUSSCHLUSS VON NEBEN- ODER FOLGESCHÄDEN

PPM IST GEGENÜBER DEM KÄUFER ODER EINER ANDEREN PERSON FÜR KEINE INDIREKTEN, NEBEN- ODER FOLGESCHÄDEN IRGENDEINER ART HAFTBAR, INSBESONDERE PROFITENTGANG, SCHÄDEN DURCH VERZÖGERUNG ODER VERLUST DER NUTZUNGSMÖGLICHKEIT, VERLUST ODER SCHÄDEN DURCH EINEN BRUCH DIESER GARANTIE ODER EINER IMPLIZITEN GARANTIE, UND DAS SELBST IM FALL, DASS DIESER DURCH EINE NACHLÄSSIGKEIT ODER EINEN ANDEREN FEHLER VON PPM ZUSTANDE

Rechtliches

KOMMT, ODER DURCH NACHLÄSSIGE VERWENDUNG DES PRODUKTS VERURSACHT WURDE. PPM HAFTET IN KEINEM FALL FÜR SOLCHE SCHÄDEN, SELBST WENN PPM AUF DIE MÖGLICHKEIT SOLCHER SCHÄDEN HINGEWIESEN WORDEN IST.

Einige nationale, Staats-, oder lokale Gesetze gestatten keinen Ausschluss oder Einschränkungen bei Neben-oder Folgeschäden. In solchen Fällen trifft die obige Einschränkung oder der Ausschluss nicht auf sie zu.

7. GESAMTVERTRAG

Diese schriftliche Garantie ist die vollständige, endgültige und exklusive Vereinbarung zwischen ppm und dem Käufer hinsichtlich der Leistungsqualität der Waren und aller und jeder Garantien und Darstellungen.

DIESE GEWÄHRLEISTUNG UMFASST ALLE VERPFLICHTUNGEN VON PPM FÜR DIESES PRODUKT. DIESE GEWÄHRLEISTUNG GIBT IHNEN BESTIMMTE RECHTE. SIE KÖNNEN ANDERE RECHTE HABEN, DIE VON GEBIET ZU GEBIET VARIIEREN, (einschließlich der Direktive 1999/44/EC in den EU-Mitgliedsstaaten), IN DIESEM FALL GELTEN FÜR SIE BESTIMMTE EINSCHRÄNKUNGEN, DIE DIESE GARANTIE ENTHÄLT, NICHT.

8. WAHL DES RECHTS.

Diese eingeschränkte Garantie unterliegt den Gesetzen von Deutschland ohne Bezugnahme auf Widerspruch zu anderen gesetzlichen Bestimmungen oder zur UN-Konvention über Verträge für den Internationalen Warenhandel, und soll zugunsten von ppm, Nachfolgern und Bevollmächtigten sein.

DIESE GARANTIE BERÜHRT WEDER DIE GESETZLICHEN RECHTE DER VERBRAUCHER UNTER DEN GELTENDEN, ANWENDBAREN GESETZEN AN IHREM WOHNORT, NOCH DIE RECHTE DES KUNDEN GEGENÜBER DEM HÄNDLER, DIE SICH AUS DEM KAUFVERTRAG ERGEBEN.

Für weitere Informationen zu dieser eingeschränkten Garantie rufen Sie uns bitte an oder schreiben Sie uns:

ppm GmbH

Grube 39a

82377 Penzberg

Deutschland

Tel: 0049-8856-8030980

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Tabellenverzeichnis

Tabellenverzeichnis

Bildverzeichnis

Bildverzeichnis

Einleitung

Was ist der 40xx GNSS Empfänger?

Herzlichen Glückwunsch!

Sie haben soeben einen neuen GNSS-Empfänger von PPM erhalten!

Der 40xx-Sensor dient zur Realisierung von Präzisions-GPS-Anwendungen zur Messung von Position oder Zeit. Diese Eigenschaften lassen mit den Möglichkeiten des Gerätes zur Steuerung von Anlagen oder Prozessen kombinieren. Während der Anwendung anfallende GPS-Daten können auf eine interne SD-Karte gespeichert und für ein späteres Post-Processing zur Verfügung gestellt werden. Es können zwei GPS-Boards eingebaut werden. Für Echtzeitanwendungen werden die GPS-Daten auf auswählbaren Schnittstellen ausgegeben. Korrekturdaten können aus verschiedenen Quellen bezogen und zum GPS-Board oder den GPS-Boards übertragen werden. Der Zugriff auf die interne SD-Karte erfolgt über den USB-Anschluss am Gerät. Dabei erscheint der 40xx-Sensor auf dem PC als ein USB-Stick.

Die Vielzahl von Schnittstellen (3 UART, 1 USB, Bluetooth, Ethernet, GSM-Modem, 5 Eventeingänge, 4 digitale Ausgänge, 1 PPS-Ausgang, 1 CAN-Schnittstelle) ermöglichen eine flexible Anpassung an eine Vielzahl von Anwendungen (z.B. Fahrdynamikmessungen, Maschinensteuerung, UAV, Bau- oder Landmaschinensteuerung).

Über 5 mehrfarbige LED kann während der Anwendung der Überblick über den Funktionsstatus behalten werden.

Der Nutzer kann eigene Konfigurationen erstellen. Die Erstellung von Gerätekonfigurationen erfolgt am PC. Die Fileübertragung erfolgt wahlweise über eine serielle Schnittstelle (ComA, ComB, ComC), Bluetooth, GSM-FTP, Ethernet oder im USB-Stick-Mode. Der 40xx-Sensor kann auch interaktiv an einer Terminalschnittstelle programmiert werden. Eventuelle Firmware-Updates erfolgen über eine Fileübertragung auf die interne SD-Karte mit anschließendem Aufruf eines Updatekommandos.

Umfang dieser Anleitung

Diese Anleitung soll Sie schnell mit Ihrem neuen GPS-System vertraut machen. Wir zeigen Ihnen alle notwendigen Schritte, vom Auspacken bis zum Einsatz, sodass Sie das System schnell und erfolgreich einsetzen können.

Sollten Sie bestimmte Informationen vermissen oder andere Anregungen zu dieser Anleitung haben, so würden wir uns über ein Feedback von Ihnen freuen. Senden Sie diese bitte unter dem Vermerk „40xx Handbuch“ an:

info@ppmgmbh.de

Vielen Dank!

Aufbau des 40xx Sensors

Der Aufbau des 40xx Sensors

Der 40xx-Sensor enthält ein GPS-Board von Novatel (andere Hersteller u.U. möglich). Die Steuerung des GPS-Boards übernimmt ein 32-Bit Mikrocontroller (STM32F427 Cortex M4), der sämtliche Schnittstellen nach außen und innen kontrolliert. Weiterhin befindet sich im Gerät ein GSM-Modem mit extern zugänglichem SIM-Kartenslot und die Stromversorgung. Die Stromversorgung, die Schnittstellen ComA, ComB und ComC, sowie die IO-Leitungen sind isoliert aufgebaut.

Als Interface ist ein 25 poliger SUB-D-Steckverbinder mit Stromversorgungs-, Daten- und Signalein-/-Ausgängen vorgesehen. Für weitere digitale Ein- und Ausgaben sind zwei Buchsen mit je zwei isolierten Ein- und Ausgängen vorhanden. Für Ethernet (100 MBit duplex) und CAN sind je ein Anschluss vorhanden. Für eine Bluetooth-Verbindung ist der Antennenanschluss an der Gerätevorderseite (Frontpanel) vorgesehen. Die Verbindung zur internen SD-Karte wird über einen Mini-USB-Steckverbinder im USB-Stickmode hergestellt. Es sind ein oder zwei GPS-Antennenanschlüsse vorhanden. Zur Signalisation von Betriebszuständen sind 4 LED am Frontpanel (LED 1 zweifarbig, LED 2 bis 4 einfarbig) eingebaut. Zur akustischen Signalisation kann ein interner Signalgeber (Buzzer) verwendet werden.

Intern ist das erste GPS-Board über zwei serielle Schnittstellen (gps1 und gps2) mit dem Steuermikrocontroller verbunden. Die eine Schnittstelle (gps1) wird als Kommandoschnittstelle bezeichnet und die zweite Schnittstelle (gps2) ist die Datenschnittstelle. Die Kommandoschnittstelle gps1 ist mit der COM1 bei Novatel-Boards und bei Trimble-Boards mit ComA verbunden. Die Datenschnittstelle gps2 ist mit der COM2 bei Novatel-Boards und bei Trimble-Boards mit ComB verbunden.

Im Gerät befindet sich eine batteriegestützte Echtzeituhr und ein batteriegestützter RAM für Parameter, die sofort nach dem Einschalten zur Verfügung stehen müssen. Die Stützbatterie (CR 2032) ist wechselbar.

Zur Datenspeicherung befindet sich im Gerät eine Mikro-SD-Karte. Sämtliche Geräteeinstellungen werden ebenfalls in Dateien auf der SD-Karte gespeichert. Die SD-Karte ist vom Anwender nicht wechselbar. Der externe Zugriff auf die SD-Karte erfolgt über den USB-Anschluss (USB-Stick-Mode). Der USB-Stick-Mode ist auch ohne externe Stromversorgung, also nur durch Anstecken des USB-Kabels, nutzbar.

Die USB-Schnittstelle ist entweder mit der SD-Karte (USB-Stickmode), mit dem GPS-Board (USB-RS232-Mode) oder überhaupt nicht verbunden.

Das GSM-Modem ist intern mit einer UART-Schnittstellen (gsm) mit dem Steuermikrocontroller verbunden.

An der 25-poligen SUB-D-Geräteschnittstelle sind weitere zwei RS232-Schnittstellen (ComB und ComC) verfügbar. Diese sind über eine Trennisolation mit dem Mikrocontroller verbunden. ComA hat keine Trennisolation. ComA, ComB und ComC sind Kommandoschnittstellen, die auch als Konsolenschnittstellen bezeichnet werden. Hier kann bei Bedarf ein Terminal oder eine anderes Gerät (auch PC) angeschlossen werden. An diesen Schnittstellen können z.B. NTRIP-Daten empfangen oder ausgegeben (dupliziert) werden. Diese Schnittstellen haben V24-Pegel, benötigen also keinen Schnittstellenwandler und können direkt an einen PC oder USB-RS232-Adapter angeschlossen werden.

Hardware

Inhaltsverzeichnis — Hardware

40xx Sensor Front Panel

Bild 1: Front Panel

Tabelle 1: Front Panel-Beschreibung 40xx-Sensor

ON/OFF: Taster zum Ein- und Ausschalten des Geräts.

Nähere Informationen zum Ein-/Ausschalten s. Seite 30

Zum Einschalten des Geräts muss der Taster kurz gedrückt werden. Sobald das Gerät eingeschalten ist, leuchtet die rote Power-Kontroll-LED. Zum Ausschalten des Empfängers muss der Taster 3s bis 5s lang gedrückt werden.

COM A: RS232 Schnittstelle COM A

SIM: Abdeckung des SIM-Kartenleser für das optionale GSM/GPRS-Modem.

Der Schraubverschluss garantiert einen wasser- und staubdichten Verschluss.

SIM-Karte: Die Abbildung der SIM-Karte zeigt an, wie die Karte eingelegt werden muss:
Kontakte nach unten und die abgeschrägte Ecke links. Die SIM-Karte rastet durch Druck in den Kartenleser ein. Erneutes Drücken lässt die S IM-Karte etwas herausspringen. Bitte verwenden Sie zum Herausnehmen der SIM-Karte eine Pinzette.

Bluetooth: SMA-Buchse zum Anschluss einer externen Bluetooth-Antenne.

USB: Mini-USB-Buchse zum Anschluss an einen PC mit Doppelfunktion:

1. USB-Stickmodus zur Datenübertragung.

Dieser Modus ermöglicht den Zugriff auf die integrierte S D-Karte. Diese Funktion steht auch ohne Anschluss der Stromversorgung zur Verfügung.

2. USB-Kommunikationsmodus

Dieser Modus ermöglicht die direkte Kommunikation des internen GNSS-Boards mit dem PC via USB.

4. Status-LEDs: 4 frei programmierbare Status-LEDs.

Je nach Konfiguration des Empfängers signalisieren die LEDs verschiedenste Systemzustände.

40xx Status-LEDs

Steuerung der LEDs:

Kommando:

set led <led>

Nähere Infos zu diesem Kommando finden Sie auf Seite 259.

LED 1:

Rot-grüne Bi-Color-LED

Farbe und interne Bezeichnungen:

- für rot: red
- für grün: green

Default:

- Grüne LED:
  - Blinken signalisiert: Datenfluss zwischen SD-Karte und CPU
- Rote LED:
  - Blinken signalisiert: Datenfluss zwischen GPS-Board und CPU
  - Dauerleuchten signalisiert: kein Datenfluss zwischen GPS-Board und CPU

Hinweis: Bei der Initialisierung bzw. Boot-Prozess blinkt die LED 1 unregelmäßig rot und grün.

LED 2:

Farbe und interne Bezeichnung: yellow

Default: keine Funktion

LED 3:

Farbe und interne Bezeichnung: orange

Default: leuchtet durchgängig wenn ppm40xx über USB mit PC verbunden

LED 4:

Farbe und interne Bezeichnung: green2

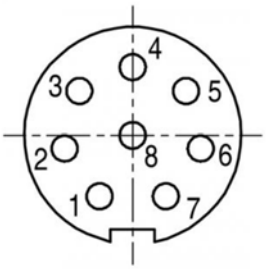
Default: keine Funktion

Pinbelegung COM A

Bild 2: COM A

Bild 3: Polbild COM A

Steckansicht / Draufsicht von außen



Buchsentyp: Binder Serie 702/712
Binder PN: 09 0428 00 08

Tabelle 2: Pinbelegung Port A

Beschreibung	PIN
Chassi GND	1
GPS_LED1	2
LED_gelb	3
VCC3_SD	4
Signal GND iso	5
RxA iso	6
TxA iso	7
GPS_LED1_ISO iso	8

40xx Sensor Back Panel

Bild 4: Back Panel

Tabelle 3: Back Panel-Beschreibung 40xx-Sensor

I/O 1 und I/O 2: Je zwei digitale, schaltbare Ein- und Ausgänge

Power + COM B/C: Systemanschlussstecker auf Basis einer SUB-D25-Buchse.

Das optional erhältliche Anschlusskabel (730124-X) bietet 2 serielle Anschlüsse (SUB-D9) und einen Stromanschluss.

Die Pinbelegung des SUB-D25 wird in der Tabelle 6 auf der Seite 20 dargestellt.

Ethernet: Ethernet-Buchse zum Anschluss eines Ethernet-Kabels PN 1: 400900 oder PN 2: 400901

Über den Ethernet-Anschluss können bis zu 3 ICOM Verbindungen aufgebaut werden.

CAN: CAN-Buchse zum Anschluss des CAN-Bus Kabel PN: 400904.

GPS 1: TNC-Buchse zum Anschluss der Haupt- bzw. Positionierungsantenne

GPS 2: TNC-Buchse zum Anschluss der Heading-Antenne

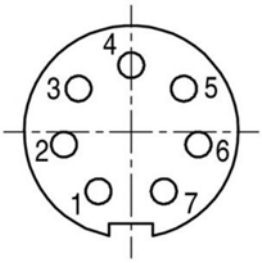
GSM: SMA-Buchse zum Anschluss einer externen GSM/GPRS Antenne.

Pinbelegung I/O 1 und I/O 2

Bild 5: I/O 1 und I/O 2

Bild 6: Polbild I/O 1 und I/O 2

Steckansicht / Draufsicht von außen



Buchsentyp: Binder Serie 702/712
Binder PN: 09 04 24 00 07

Tabelle 4: Pinbelegung I/O 1

Beschreibung	Pin
GND_iso_Event1	1
VCC3_iso_Event1	2
Event1_in2_iso	3
Event1_in1_iso	4
Event1_out2_iso	5
Event1_out1_iso	6
n/c	7

Tabelle 5: Pinbelegung I/O 2

Beschreibung	Pin
GND_iso_Event2	1
VCC3_iso_Event2	2
Event2_in1_iso	3
Event2_in2_iso	4
Event2_out1_iso	5
Event2_out2_iso	6
n/c	7

Pinbelegung Power + COM B/C

Bild 7: Power + COM B/C

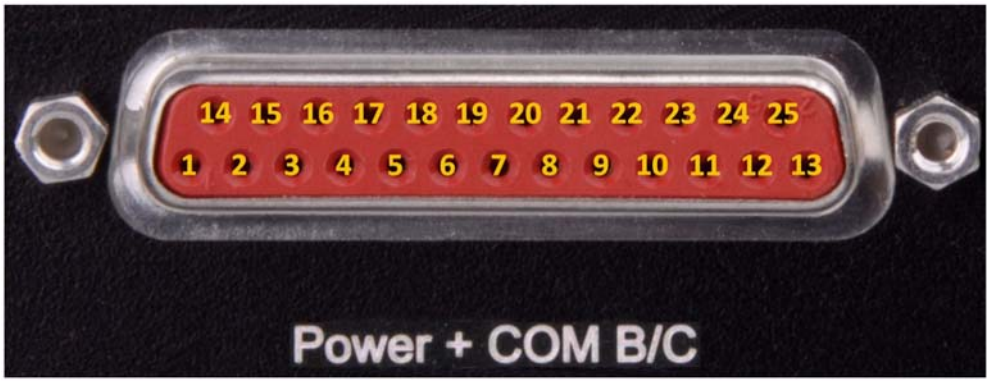


Tabelle 6: Pinbelegung Power + COM B/C

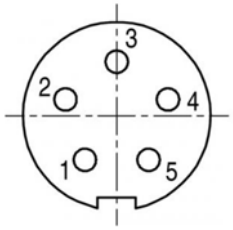
Beschreibung	Pin
n/c	1
Chassi GND	2
GND COM-C iso	3
COM-C RTS iso	4
COM-C TX iso	5
COM-B TX iso	6
GND COM-B iso	7
COM-B RTS iso	8
EXT GPIO 5 iso	9
GND PPS/Event/VARF iso	10
VARF iso	11
EXT_GROUND (GND PWR)	12
EXT_GROUND (GND PWR)	13
n/c	14
PPS out iso	15
COM-C CTS iso	16
COM-C RX iso	17
COM-B RX iso	18
n/c	19
COM-B CTS iso	20
n/c	21
n/c	22
EVENT_in iso	23
PWR in + (9-36 VDC)	24
PWR in + (9-36 VDC)	25

Pinbelegung Ethernet

Bild 8: Ethernet

Bild 9: Polbild Ethernet

Steckansicht / Draufsicht von außen



Buchsentyp: Binder Serie 702/712
Binder PN: 09 0416 00 05

Tabelle 7: Pinbelegung Ethernet

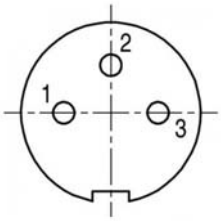
Beschreibung	Pin
Tx_P	1
TX_N	2
COM	3
RX_P	4
RX_N	5

Pinbelegung CAN

Bild 10: CAN

Bild 11: Polbild CAN

Steckansicht / Draufsicht von außen



Buchsentyp: Binder Serie 702/712
Binder PN: 09 0408 00 03

Tabelle 8: Pinbelegung CAN

Beschreibung	**Pin**
CAN_GND	1
CAN_L	2
CAN_H	3

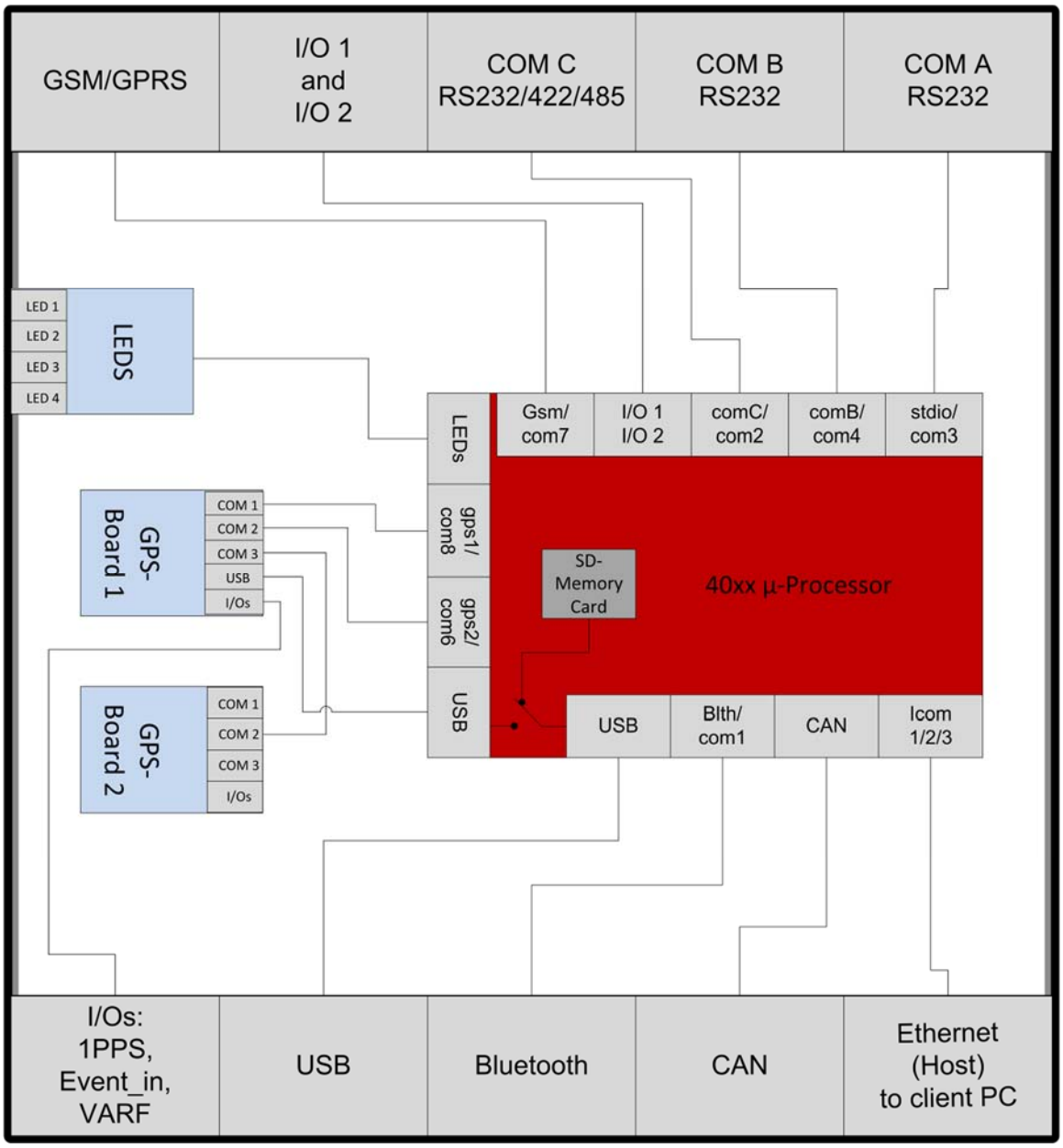
Erläuterungen zum Thema Abschlusswiderstand

Der CAN-Bus besteht aus zwei Datenleitungen und einer Masseleitung, die in einer Linie an allen CAN-BUS-Teilnehmern vorbei geführt wird (Linientopologie). An den Enden muss der Bus mit je einem Abschlusswiderstand von 120 Ohm terminiert werden (zwischen D+ und D-). Die Abstiche vom Bus zum Teilnehmer sollten je nach Bitrate kleiner als 75 cm sein. Der 40xx enthält keinen Abschlusswiderstand.

Von diesen Forderungen kann abgewichen werden, wenn die Bitrate klein, (bis ca.250k) und die Leitungslänge sehr kurz (ca. 1-2 m) ist.

Blockschaltbild interne und externe Schnittstellen

Bild 12: Blockschaltbild interne und externe Schnittstellen

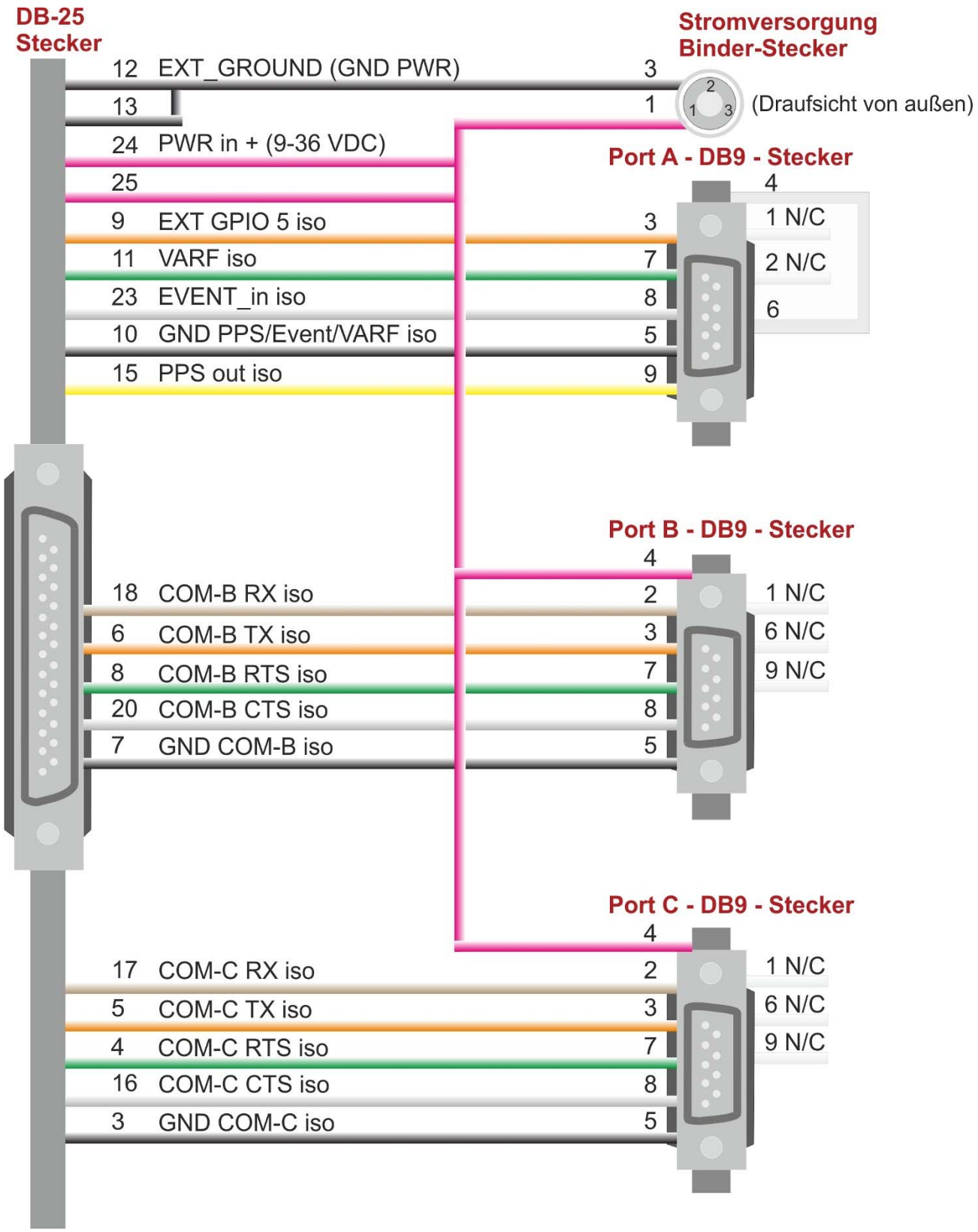


Hinweis zu COM C

Der COM C kann vom Anwender mit dem Kommando ‚set comc' (s. S. 234) auf RS232- oder RS485-Betrieb konfiguriert werden. Default ist RS232.

Optionales Systemanschlusskabel PN:730124-B

Bild 13: Optionales Systemanschlusskabel



40xx serielles Kabel COM A PN:400902

Bild 14: serielles Interface Kabel COM A, mit Pin-Out

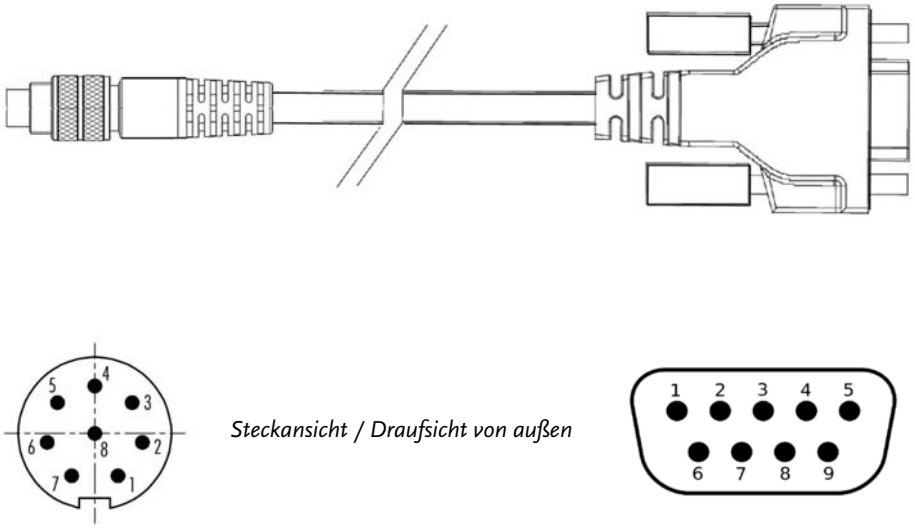


Tabelle 9: Pin-Out serielles Interface Kabel COM A

Signal	Binder 8-polig	SubD-9 female
Chassi GND	1	Buchsengehäuse
GPS_LED1	2	n/c
LED_gelb	3	n/c
VCC3_SD	4	n/c
Signal GND	5	5
RxA	6	3
TxA	7	2
GPS_LED1_ISO	8	n/c

40xx Ethernet-Kabel I PN:400900

Bild 15: Ethernet-Kabel I, mit Pin-Out

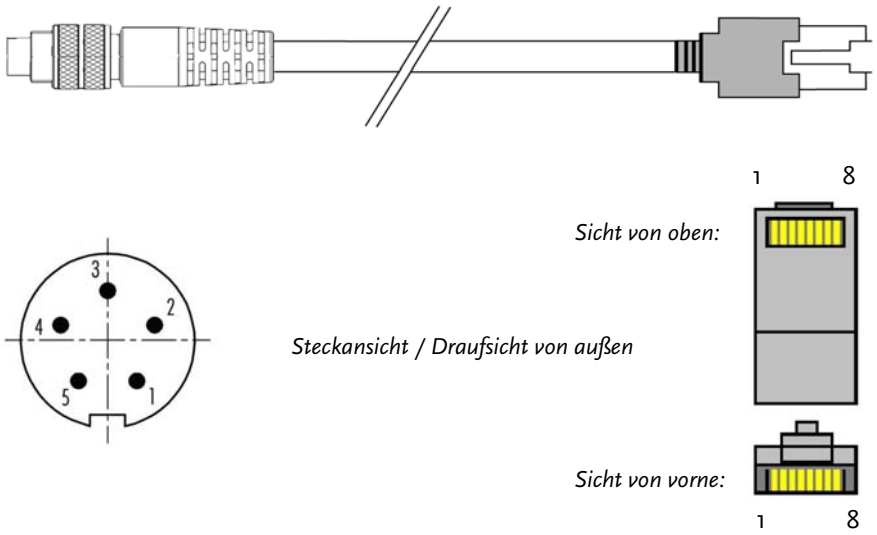


Tabelle 10: Pin-Out Ethernet-Kabel I

Signal	Binder 5-polig	RJ-45 male
Tx_P	1	1
TX_N	2	2
COM	3	n/c
RX_P	4	3
RX_N	5	6

40xx Ethernet-Kabel II PN:400901

Bild 16: Ethernet Kabel II, mit Pin-Out

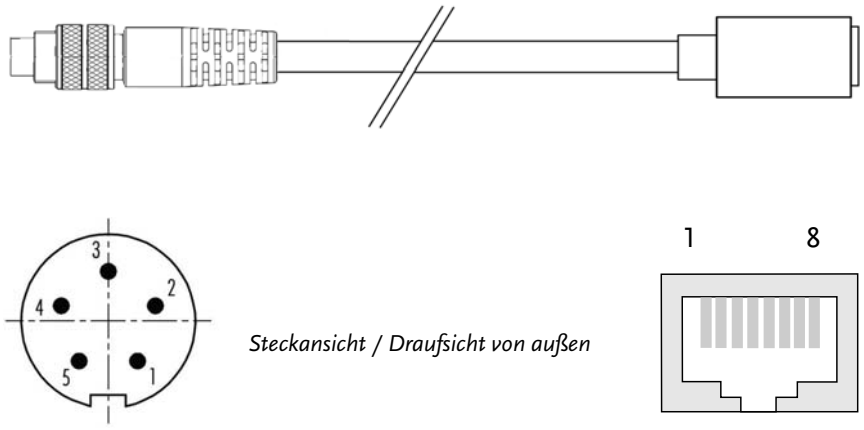


Tabelle 11: Pin-Out Ethernet-Kabel II

Signal	Binder 5-polig	RJ-45 female
Tx_P	1	1
TX_N	2	2
COM	3	n/c
RX_P	4	3
RX_N	5	6

40xx I/O-Kabel PN:400903

Bild 17: I/O Kabel, mit Pin-Out

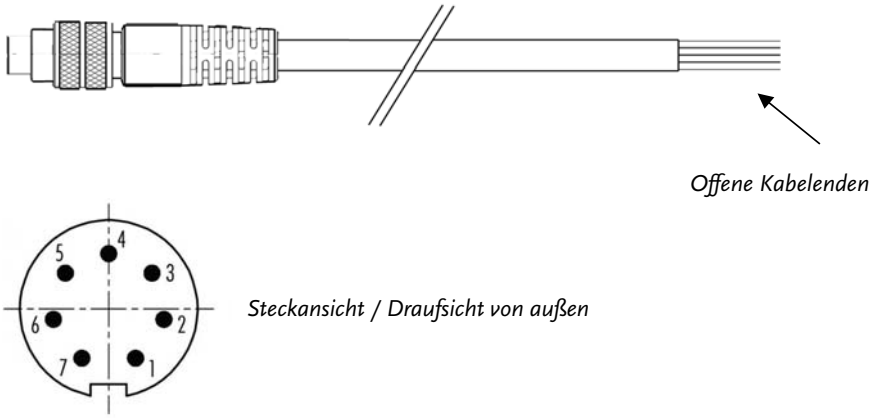


Tabelle 12: Pin-Out I/O-Kabel

Signal	Binder 7-polig	Kabelenden
GND_iso_Event1	1	n/c
VCC3_iso_Event1	2	n/c
Event_in2_iso	3	n/c
Event_in1_iso	4	n/c
Event_out2_iso	5	n/c
Event_out1_iso	6	n/c
n/c	7	n/c

40xx CAN-Kabel PN:400904

Bild 18: CAN Kabel, mit Pin-Out

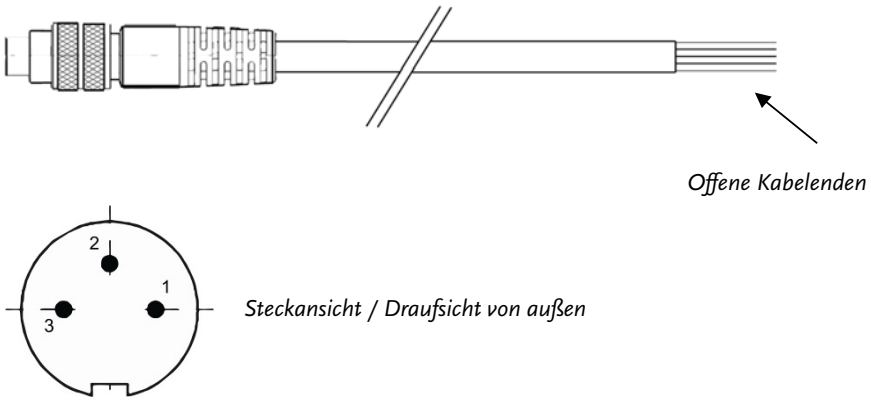


Tabelle 13: Pin-Out CAN-Kabel

Signal	Binder 3-polig	Kabelenden
CAN_GND	1	n/c
CAN_L	2	n/c
CAN_H	3	n/c

Ein– und Ausschalten des 40xx

Ein– und Ausschalten des 40xx

Einschalten:

Der 40xx kann über den ON/OFF-Taster eingeschaltet werden. Dies ist aber nur möglich, wenn der 40xx auch über den ON/OFF-Taster abgeschaltet wurde.

Zum **Einschalten** des 40xx muss der ON/OFF-Taster kurz gedrückt werden. Es erfolgt dabei ein kurzer Pieps-Ton. Sobald der 40xx eingeschalten ist, leuchtet die rote Power-Kontroll-LED am Taster selbst.

Wurde der 40xx durch Trennen der Stromversorgung abgeschaltet, so schaltet sich der 40xx bei erneutem Anschließen der Stromversorgung automatisch ein, und beginnt mit dem Boot-Vorgang.

Wurde der 40xx über den ON/OFF-Taster abgeschaltet, so muss er auch nach Anschließen der Stromversorgung wieder über den ON/OFF-Taster eingeschaltet werden.

Ausschalten:

Der 40xx kann entweder über den **ON/OFF-Taster**, oder durch **Trennen der Stromversorgung** abgeschaltet werden.

Zum **Ausschalten** über den ON/OFF-Taster , muss dieser für 3-5s gedrückt gehalten werden. Es ertönt ein etwas länger anhaltender Pieps-Ton, welcher dem Nutzer signalisiert, dass der ON/OFF-Taster nun wieder freigegeben werden muss.

Wird der ON/OFF-Taster vor dem längeren Pieps-Ton freigegeben, oder bleibt er über den Pieps-Ton hinaus gedrückt gehalten, so bleibt der 40xx eingeschaltet und es passiert nichts weiter.

In beiden Fällen - Ausschalten via ON/OFF-Taster oder durch Trennen der Stromversorgung - fährt das ppmOS kontrolliert herunter. D.h. alle offenen Dateien werden geschlossen und auf der SD-Karte gespeichert.

Hinweis zum USB-Stickmode:

Ist der 40xx eingeschalten, so wird der Stickmode durch Anstecken eines mit einem Computer verbundenen USB-Kabels eingeschaltet. Die USB-Stickmode-Funktion steht nur dann auch im ausgeschalteten Zustand zur Verfügung, wenn der Sensor durch Beenden der Stromversorgung ausgeschaltet wurde. Wurde der Sensor über den ON/OFF-Taster ausgeschaltet bzw. heruntergefahren, so steht dies USB-Stickmode-Funktion nicht zur Verfügung.

Weitere Informationen zum USB-Stickmode finden Sie auf Seite 46.

40xx Spezifikationen

Stromversorgung:

Stromversorgung: 9 - 36 Volt DC

Der 40xx GNSS-Empfänger kann, je nach Integrations– und Anwendungszweck, mit einer direkten Stromversorgung verbunden oder an einen Akku angeschlossen werden. Der Eingangsbereich beträgt 9 –36 Volt.

Die Belegung auf dem Systemstecker:

PIN 12+13: EXT_GROUND (GND PWR)

PIN 24+25: VCC 9-36 VDC

Stromaufnahme:

Die Stromaufnahme variiert je nach Hardware– und Softwarekonfiguration des 40xx. Die Tabelle 14 (*40xx — Stromaufnahme nach Konfiguration*) stellt den Stromverbrauch nach Konfiguration dar:

Tabelle 14: 40xx — Stromaufnahme nach Konfiguration

40xx-Konfiguration	Leistungsaufnahme (W)[1]
40xx-14 (Position)	1,5 — 2,0
40xx-144 (Kurs / Heading)	2,3 — 2,6
40xx-1444 (Raumlage / Attitude)	4,2 — 4,8

[1] Leistungsaufnahme ohne Zusatzoptionen. GSM oder Funk erhöht den Stromverbrauch jeweils um bis zu 2 Watt.

Temperaturbereich (°C):

-25 bis +65

Umwelt:

IP 65

Maße (mm):

126 x 49 x 220

Gewicht (kg):

1,5

Masse (Erdung)

Das Gerät verfügt über Gehäuse-Masse (auch am Gehäuse selbst vorhanden). Eine Erdung, falls erforderlich, muss separat am Gehäuse verbunden werden.

Bitte beachten Sie eventuelle Potentialunterschiede.

Wartung

Wartung

Der 40xx-Sensor enthält eine Lithium-Stützbatterie (CR 2032), die alle 3-5 Jahre gewechselt werden muss.

Darüber hinaus ist der 40xx-Sensor wartungsfrei.

Software - ppmOS

Inhaltsverzeichnis — Software - ppmOS

Software - ppmOS

Einleitung

Funktion und Inbetriebnahme

Zur Inbetriebnahme des 40xx-Sensors sind eine Stromversorgung von 9 V bis 32 V, eine GPS-Antenne und eine Reihe von Konfigurationsdateien erforderlich. Die Konfigurationsdateien werden am PC erstellt und anschließend auf den 40xx-Sensor kopiert. Dazu wird der 40xx-Sensor eingeschaltet und mit einem USB-Kabel mit dem PC verbunden. Der 40xx-Sensor ist auf dem PC als Laufwerk (USB-Stick) verfügbar. Nach der Trennung des USB-Anschlusses hat der Steuermikrocontroller im 40xx-Sensor wieder Zugriff auf die SD-Karte. Unmittelbar nach dem Abziehen des USB-Kabels beginnt die Initialisierung des 40xx-Sensors.

Das ppmOS

Der 40xx GNSS-Empfänger ist ein sehr universeller GNSS–Empfänger, der eine flexible und individuelle Konfiguration für eine große Bandbreite an GNSS-Anwendungen ermöglicht.

Die Einsatzmöglichkeiten des 40xx-Sensors werden einerseits hardwaremäßig durch die Ausstattung des 40xx-Sensors mit GPS-Board und/oder Bluetooth und GSM-Modem festgelegt. Über die Auswahl des GPS-Boards und über zukaufbare GPS-Softwareoptionen wird die Genauigkeit der GPS-Daten festgelegt.

Die weitere Festlegung der Verwendung erfolgt durch die Konfiguration des Betriebssystems **ppmOS**, des steuernden Mikrocontrollers und des GPS-Boards. Diese Einstellungen werden in Dateien gespeichert. Sie lassen sich unterteilen in Konfigurationsdateien und Skripte.

In Konfigurationsdateien werden Einstellungen wie z.B. Baudrate oder Datenrate der Positionsberechnung vermerkt. Skripte sind Kommandodateien oder auch Batchdateien, sie werden vom Betriebssystem **ppmOS** ausgeführt. Der 40xx-Sensor ist also ein vom Anwender frei programmierbares Gerät.

Eine detaillierte Beschreibung des **ppmOS** finden Sie in den folgenden Abschnitten.

Die Version B des 40xx GNSS Sensor Handbuchs bezieht sich auf die Version 1.46 des ppmOS.

Software - ppmOS

Interner SD-Kartenspeicher

SD-Kartenspeicher

Auf der SD-Karte wird ein Dateisystem FAT16 oder FAT32 erwartet. An die Clustergröße werden keine besonderen Anforderungen gestellt. Die interne SD-Karte, mit Laufwerk **c:** bezeichnet, kann im USB-Stickmode an einem PC formatiert werden. Es können lange Dateinamen verwendet werden. Bei Datei- und Verzeichnisnamen wird nicht zwischen Groß- und Kleinschreibung unterschieden. Sie werden aber in Groß- und Kleinschreibung gespeichert. In Pfadangaben ist das Trennzeichen zwischen Verzeichnissen das ‚**/**', alternativ kann auch das ‚****' verwendet werden. Die standardmäßig verwendeten Verzeichnisse im **ppmOS**-Dateisystem sind:

c:/sys

c:/sms

c:/data

Als Anwender können sie beliebig weitere Verzeichnisse anlegen und Daten speichern. Es ist zu beachten, dass das Wurzelverzeichnis beim FAT16-Dateisystem nur eine begrenzte Zahl von Dateieinträgen aufnehmen kann (max. 511 auch bei 32 KB Clustersize). Beim FAT32-Dateisystem gibt es diese Begrenzung für das Wurzelverzeichnis nicht. In einem Unterverzeichnis können bei beiden Dateisystemen beliebig viele Dateieinträge stehen.

Unterverzeichnis DATA:

Positionsdaten werden im Verzeichnis **c:/data** abgelegt. Für die Daten werden zur Laufzeit vom **ppmOS** weitere Unterverzeichnisse angelegt. Sie dienen der besseren Strukturierung der Aufzeichnungen nach Jahr, Monat und Tag (z.B. **c:/data/2017/05/13**).

Unterverzeichnis SYS:

Im Verzeichnis ‚**c:/sys**' werden standardmäßig alle zur Konfiguration und zum Betrieb notwendigen Dateien gespeichert. Bei der Ausführung sucht das **ppmOS** jedoch immer im Pfad, der in der Systemvariable ‚**syspath**' gespeichert ist. Diese hat standardmäßig den Wert ‚**c:/sys/**'. Durch die Zuweisung eines anderen Pfades an die Systemvariable (siehe Kommando ‚**set syspath ...**' S. 269) kann auf einfache Weise eine komplett andere Gerätekonfiguration eingestellt werden.

Unterverzeichnis SMS:

SMS-Nachrichten werden im Verzeichnis c:/sms abgelegt.

Software - ppmOS

Datenspeicherung

Datenspeicherung

Anfallende Daten werden im Dateisystem auf SD-Karten gespeichert. Ebenso werden Konfigurationsdateien oder Skripte aus dem Dateisystem geladen. Im **ppmOS** sind die Dateisysteme FAT16 und FAT32 mit langen und kurzen Dateinamen implementiert.

Ist der 40xx-Sensor im USB-Stickmode mit einem PC verbunden, dann werden Datenraten im Bereich von 3 bis 22MB/s erreicht. Mit dem **ppmOS** werden Transferraten von bis zu 10 MB/s erreicht. Diese Transferrate kann aber auf ca. 200 kByte/s zurückgehen, wenn mehrere Dateien schreibend geöffnet sind.

Software - ppmOS

Dateien zur Steuerung und Konfiguration

Skript– und Konfigurationsdateien

Konfigurations- und Skriptdateien sind einfache ASCII-Dateien, die mit einem Texteditor (z.B. notepad.exe) bearbeitet werden können.

Für diese Dateien gelten folgende Konventionen:

Informationen werden zeilenweise gespeichert. Die Zeilen können mit **LF** (0xa) oder **CR LF** (0xd 0xa) abgeschlossen werden. Zeilen, die leer sind und Zeilen, die mit dem Zeichen **#** beginnen werden ignoriert. Das **#** ist also das Kommentarzeichen. Daraus ergibt sich die Möglichkeit, zusätzliche Informationen in die Dateien aufzunehmen. Oder man kann eine Konfigurationszeile, wenn deren Information gerade nicht benötigt wird, einfach auskommentieren.

Es gibt Konfigurationsdateien, die mit **.cfg** enden, und Skriptdateien, die mit **.sh** enden. Grundsätzlich gibt es im ppmOS **keine** vordefinierte Verwendung von Dateinamensendungen.

Beginnt in einer Konfigurationsdatei eine Zeile mit dem Schlüsselwort **exec**, dann wird alles, was auf der Zeile folgt, als ein Kommando interpretiert und ausgeführt. Es ergibt sich hiermit die Möglichkeit, während der Interpretation der Konfigurationsdatei beliebige im **ppmOS**-Betriebssystem implementierte Kommandos auszuführen.

Im Folgenden werden die Standarddateien beschrieben. Diese sind:

autoexec.sh

gps.cfg

firstfix.sh

bestpos.cfg

upload.cfg

crontab.cfg

ntrip.cfg

network.cfg

Nach der Beschreibung der Standarddateien finden Sie einen Workflow der die Arbeitsweise des ppm40xx bzw. des ppmOS sowie die Funktionen der Standarddateien veranschaulicht.

Software - ppmOS

Dateien zur Steuerung und Konfiguration

autoexec.sh - Skript, das nach Power on, Reset oder Beendigung des USB-Stickmodes ausgeführt wird

In dieser Datei werden alle Kommandos aufgeführt, die für den speziellen Verwendungszweck des 40xx-Sensors erforderlich sind. Dieses Skript wird nach einem Power On, Reset und nach dem Trennen des USB-Stickmodes zu aller erst ausgeführt. In der Regel werden dort mindestens folgende Kommandos notiert:

cron init

gps init

gps.cfg - Konfigurationsdatei für das GPS-Board

In dieser Datei werden alle Initialisierungen aufgeführt, die für den speziellen Verwendungszweck des GPS-Boards im 40xx-Sensor erforderlich sind. Der Inhalt und die Parameter richten sich nach den Angaben aus dem Handbuch des GPS-Boardherstellers. Beispielsweise kann man festlegen, ob die Positionsdaten binär oder im ASCII-NMEA-Format ausgegeben werden. Ebenso kann hier die Positionsberechnungshäufigkeit festgelegt werden. Dies kann je nach GPS-Board bis zu 100 Positionsberechnungen je Sekunde betragen. Hier wird auch die Reaktion auf die Signale am Eventeingang festgelegt. Die Einstellung zur Generierung von periodischen Ausgangssignalen (PPS-Pulse) wird hier ebenfalls vorgenommen. Sollen die Daten nicht nur aufgezeichnet, sondern auch an einer Schnittstelle ausgegeben werden, dann kann die Konfiguration hier erfolgen.

firstfix.sh - Skript, das beim ersten Positionsfix ausgeführt wird

Diese Kommandodatei wird einmalig in dem Moment ausgeführt, wenn das Board eine erste gültige Positionsberechnung ausgibt. Standardmäßig wird unabhängig von der **gps.cfg** das GPS-Board so konfiguriert, dass es auf seiner COM1, die mit der Controllerschnittstelle **gps1** verbunden ist, die NMEA-Datensätze $GPRMC und $GPGGA ausgibt. Diese werden vom **ppmOS** permanent überwacht.

Die Positionsberechnung gilt als gültig, wenn Position, Datum und Uhrzeit ausgegeben werden, das Gültigkeitsflag 'A' (2. Parameter) lautet und die CRC des Datensatzes richtig ist.

Dateien zur Steuerung und Konfiguration

bestpos.cfg - Konfigurationsdatei zur Durchführung einer Positionsberechnung

In diese Datei werden die Konfigurationsanweisungen zur Auslösung einer Positionsberechnung aufgenommen. Die Datei ist immer dann erforderlich, wenn langsamperiodische oder eventgesteuerte Einzelpositionsmessungen durchgeführt werden sollen.

Für Novatel-Boards lautet diese

log com2 bestposb once

und für Trimble-Boards

$PASHQ,POS,B

Darüber hinaus können '**exec**'-Zeilen eingefügt werden. Zeilen, die mit exec beginnen, werden nicht als Konfigurationseinstellung interpretiert, sondern sie werden vom Kommandointerpreter (shell) ausgeführt.

upload.cfg - Datei zum Konfigurieren der Mobilfunk- und FTP-Verbindung

Das eingebaute GSM-Modem stellt eine drahtlose Verbindung zum Internet über das GSM-Netz her. Dazu sind eine GSM-Antenne, eine SIM-Karte und Zugangsdaten erforderlich. Nahezu alle Mobilfunkanbieter vertreiben Daten-SIM-Karten. Die SIM-Karte kann als Prepaid- oder im Rahmen eines normalen Mobilfunkvertrages vom Anwender erworben werden. Sie gehört nicht zum Lieferumfang.

In die '**upload.cfg**'-Datei werden dann die PIN, die Provider-APN, der Nutzername und falls erforderlich das Passwort eingetragen. Für die Datenübertragung (Up- und Download) ist das FTP-Protokoll implementiert. Die erforderlichen Angaben für Nutzer, Passwort und Serveradresse werden hinter dem Schlüsselwort **server** eingetragen. Es wird empfohlen, die Serveradresse als IP-Adresse (vier Zahlen durch Punkt getrennt) anstelle der URL, also die Internetadresse in Textform, anzugeben.

Die Datei hat z.B. folgenden Inhalt (s. nächste Seite):

Dateien zur Steuerung und Konfiguration

Beispiel Inhalt '**upload.cfg**':

```
UploadConfigFile v 0.1

#Sim-Kartenpin
PIN    1234

#Einwahl
apn    web.vodafone.de
user   vodafone
passwd vodafone

#der FTP-Server
server Nutzername:Passwort@111.222.123.321
```

Die FTP-Zeile kann auch folgende Form haben:

```
server Nutzername:Passwort@111.222.123.321/$(fn_ext)
```

Die Shell-Variable **fn_ext** wird zu einem 3-Buchstabencode, der das GPS-Board identifiziert, expandiert. Wenn nun auf dem Server ein Verzeichnis mit diesem Namen angelegt ist, dann wird für alle Up- und Downloads dieses Verzeichnis verwendet. Dies erleichtert die Verwaltung von mehreren 40xx-Sensoren.

crontab.cfg - Konfigurationsdatei für zeitgesteuerte Aktionen

Für erweiterte Anwendungen steht im **ppmOS** ein CRON-Manager zur Verfügung. Es handelt sich dabei um eine Softwarefunktion, die chronologische, also zeitlich festgelegte Aktionen, ähnlich der Aktionen in einem Terminkalender ausführt. Diese Aktionen können sich wiederholen oder einmalig sein. Bedingte Ausführung von Aktionen sind ebenfalls möglich.

Der CRON-Manager wird mit dem Kommando ‚**cron**' gesteuert.

So kann z.B. der Zustand eines Eventsignals periodisch geprüft werden und beim Auftreten eines bestimmten Pegels (high oder low), wird eine vorher konfigurierte Aktion ausgeführt. Aktionen, die ausgeführt werden, sind

Software - ppmOS

Dateien zur Steuerung und Konfiguration

Kommandos des **ppmOS.** Sind es mehrere Kommandos, dann werden sie zweckmäßiger Weise in Skripte zusammengefasst.

Die Auflistung aller zeitlich gesteuerten Aktionen, die ab Power on ausgeführt werden sollen, erfolgt in der Datei **crontab.cfg.** Man kann diese Datei auch als den Terminkalender des 40xx-Sensors bezeichnen. Der Begriff **crontab** und **cron**-Manager oder **cron**-Daemon sind der Unix/Linux Welt entlehnt. Im Gegensatz zu Linux wird die Datei bei der Ausführung des Kommandos '**cron init**', welches zweckmäßiger Weise in der **autoexec.sh** aufgeführt wird, eingelesen und in eine RAM-residente Tabelle eingetragen. Linux überwacht nur diese Datei und hat keine RAM-residente Tabelle.

Alternativ zur **crontab.cfg** können die CRON-Jobs an beliebiger Stelle bzw. zu einem beliebigen Zeitpunkt gestartet werden. Es sind max. 12 gleichzeitige CRON-Jobs möglich (Stand ppmOS v 1.46).

D. h. die Cron-Jobs können z. B. auch in der autoexec.cfg mit dem Kommando ‚**cron put ...**' gestartet werden, eine crontab.cfg ist also nicht zwingend nötig. Nicht mehr benötigte Cron-Jobs können suspendiert oder gelöscht werden. Cron-Tasks können sich auch selbst nach einer bestimmten Wiederholanzahl löschen.

Die Begrifflichkeiten und die Syntax der crontab-Einträge orientieren sich an der crontab in der Linux-Welt. In der **ppmOS** crontab.cfg ist gegenüber der Linux crontab eine Erweiterung auf Sekunden und ein erweitertes Ablaufformat hinzugefügt worden.

Vor der Erläuterung der Syntax für die crontab ist hier ein kurzes Beispiel für einen cron-Task, oder auch cron-Job genannt, angegeben:

Zeile in cron.tab:

```
0,20,40 0 * * * * gps bestpos
```

oder Kommandozeile:

```
cron put 0,20,40 0 * * * * gps bestpos
```

In dem Beispiel werden immer zur vollen Stunde je 3 Positionsberechnungen im Abstand von 20 s ausgeführt.

Dateien zur Steuerung und Konfiguration

Syntax des Konfigurationsfiles für periodische, aperiodische oder einmalige Aktionen:

Zeilen, die mit '**#**' beginnen oder leer sind, werden ignoriert. Zeilen, die mit '**exec**' beginnen werden an die Shell übergeben und dort ausgeführt.

Die crontab.cfg-Zeilen können als 'wenn der Zeitpunkt erreicht ist, dann führe ein Kommando aus' verstanden werden.

Das Format einer crontab.cfg-Zeile lautet wie folgt. Die verwendeten Sonderzeichen werden im Abschnitt 'Beschreibung des **ppmOS** Kommandos **cron**' näher erläutert.

primäre Zeitangabe:

<ss> <mm> <hh> <dd> <MM> <yy> <weekday> [-t|-i] <command>

Darin bedeuten:

<ss>	0..59	Sekunde
<mm>	0..59	Minute
<hh>	0..23	Stunde
<dd>	1..31	Tag
<MM>	1..12	Monat
<yy>	0..99	Jahr
<weekday>	0...6	Wochentag, 0 ist Sonntag, 1 ist Montag
[-t|-i] <command>	jedes implementierte Kommando (resident oder Skript auf SD-Karte). Es können auch mehrere Kommandos, die in **"..."** oder **'...'** eingeschlossen sind, angegeben werden. Die Kommandos werden in die Empfangspipe der Konsolenschnittstelle kopiert, sie erscheinen dem **ppmOS** so, als ob sie über die **coma** empfangen wurden. Bei **-t** werden sie als Task ausgeführt. Dies ist zu empfehlen, wenn die Ausführung etwas länger als ein paar Millisekunden dauert und der Kommandozeileninterpreter von **coma** nicht blockiert werden soll. Bei **-i** werden sie unmittelbar ausgeführt. Dies wird empfohlen, wenn die Ausführung im Mikrosekundenbereich dauert. Der Cronmanager wird solange angehalten, d. h. er kann in dieser Zeit keine Matchpunkte prüfen und keine Cronjobs ausführen.	

Dateien zur Steuerung und Konfiguration

- Das Zeichen ***** bedeutet 'zutreffend für alle Zahlen', es ist also das Joker-Zeichen.
- Es können Aufzählungen z.B. 2,4,6,8,9,11 anstelle einer Zahl stehen.
- Es können Bereichsangaben z.B. 4-10 oder 58-2 anstelle einer Zahl stehen.
- Die Angabe 58-2 ist z. B. gleichwertig mit 58,59,0,1,2.
- Zahlen, Aufzählungen und Bereichsangaben können beliebig aneinander gereiht werden und dürfen keine Leerzeichen oder Tabs enthalten.

Bis hierhin kann man die Zeitangaben als primäre Zeitangabe bezeichnen. Der Sekundenteil kann mit einer sekundären Zeitangabe ergänzt werden, um weiterführende komplexere Zeitangaben zu realisieren. Der Sekundenteil und die sekundäre Zeitangabe ist in Linux nicht zu finden.

sekundäre Zeitangabe:

[[+<Zeitoffset>][/<Zeitabstand>[*<Wiederholungszähler>]][.|s]]

Die Syntax für die sekundäre Zeitangabe hinter der primären Sekundenangabe ist entsprechend BNF (Backus Naur Formalismus) angegeben. Weitere Erläuterungen zur Beschreibung der Kommandosyntax sind im Abschnitt 'Beschreibung des ppmOS-Kommandos **cron**' zu finden.

- Folgt nach der primären Sekundenangabe ein **+<Zeitoffset>**, dann wird die erste Ausführung <Zeitoffset> Sekunden nach dem primären Zeitpunkt erfolgen.
- Folgt hinter **+<Zeitoffset>** ein '.', dann wird das Kommando einmalig ausgeführt und der cron-Job wird gelöscht
- An die Angabe für die Sekunden kann **/<Zeitabstand>** angehangen werden.
- Das bedeutet, dass bei Erreichen des primären Zeitpunktes das Kommando ausgeführt wird und danach wieder alle **<Zeitabstand>** Sekunden. Unter Linux ist dieses Zeichen in allen Feldern zulässig.
- Diese Angabe ersetzt dann alle anderen primären Zeitpunkte.
- Dem **<Zeitabstand>** kann ein ***<Wiederholungszähler>** folgen.
- D. h. die Aktion wird dann alle **<Zeitabstand>** Sekunden insgesamt **<Wiederholungszähler>** mal ausgeführt. Danach beginnt wieder der Zeitvergleich mit der primären Zeitangabe.
- Folgt dem **<Wiederholungszähler>** ein '.', dann endet dieser cron-Job, d.h. der cron-Job wird nach **<Wiederholungszähler>** Wiederholungen entfernt.
- Steht am Ende ein '**s**', wird der cron-Job nach Abarbeitung suspendiert. D.h. er wird nicht mehr ausgeführt aber bleibt in der crontab erhalten. Mit '**cron resume {<ix>|<name>}**' kann er wieder aktiviert werden.

Beispiele siehe nächste Seite.

Dateien zur Steuerung und Konfiguration

Beispiele:

0 0 6,20 * * * * cp c:/sys/gps.cfg c:/bak/gps.cfg

Um 6:00 und 20:00 an jedem Tag wird der Kopierbefehl ausgeführt.

0/20*3 0 * * * * 1-5 c:/sys/cmd_xyz

Zu jeder vollen Stunde wird 3 mal im Zeitabstand von 20 s montags bis freitags das Skript c:/sys/cmd_xyz ausgeführt.

*Weitere Informationen sind bei der Beschreibung des **cron**-Kommandos ab Seite 109 zu finden.*

ntrip.cfg - Konfiguration für die Nutzung einer NTRIP-Serververbindung

Die Konfigurationsdatei **ntrip.cfg** wird im Zusammenhang mit dem **ppmOS**-Kommando **ntrip** benötigt.

Die Datei **ntrip.cfg** hat folgenden Inhalt (Erläuterungen s. nächste Seite):

```
#Konfigurationsfile für das Öffnen der NTRIP-Casterverbindung
#Hinweis: Zeilen, die mit exec beginnen, werden an die shell weitergegeben (ausgeführt)
exec @echo reading from ntrip.cfg

srvtype socket

#address socktcp://www.igs-ip.net:80
address socktcp://78.46.41.8:80

GET /LEIJ1 HTTP/1.0
User-Agent: NTRIP XS/1.14

#user:passwd wird base64 kodiert
Authorization: Basic <username>:<password>

""
```

Die ersten beiden Zeilen sind Kommentare. Die Zeile, die mit **exec** beginnt, ist keine Konfigurationseinstellung, sondern ein Kommando, das an den Kommandointerpreter weitergegeben wird. In diesem Beispiel wird ein Text auf einem eventuell angeschlossenen Terminal angezeigt. Er dient lediglich Testzwecken, um den Ablauf der Applikation verfolgen zu können.

Dateien zur Steuerung und Konfiguration

Für den Anwender ist die Angabe hinter **address socktcp://** wesentlich. Hier wird die IP-Adresse des NTRIP-Caster bzw. Server eingetragen. Die Angabe der IP-Adresse ist der Angabe der URL vorzuziehen, da mit der IP-Adresse die Abfrage eines DNS-Servers entfällt.

Nach dem Aufbau der TCP-Verbindung durch das GSM-Modem werden die Informationen **GET, User-Agent** und **Authorization** zum NTRIP-Server übertragen.

Wichtig:

Der letzte Eintrag muss **""** (zwei Anführungszeichen) sein. Der NTRIP-Server verlangt nach der Authorization-Zeile noch eine leere Zeile, also nur CR+LF. Fehlt diese Information, dann kommt keine Verbindung zustande. Wenn der Server die Angaben akzeptiert, schickt er einen kontinuierlichen Datenstrom, der per Default intern zum GPS-Board weitergeleitet wird.

Das **ppmOS** sendet an den NTRIP-Server periodisch den GPGGA-Datensatz für eine weitere Verbesserung der Korrekturdaten.

Weitere Informationen zu diesem Thema sind in der Beschreibung des Kommandos **ntrip** (s. S. 192)zu finden.

network.cfg - Konfigurationsdatei für das Ethernet-Firmware-Modul

Das optionale File **network.cfg** wird mit dem Kommando ‚ip config' aufgerufen und hat folgenden beispielhaften Inhalt, der den Gegebenheiten bei einer netzwerkbasierten Anwendung angepasst werden muss (mehr s. S. 171):

```
#Netzwerkkonfigurationsfile
#-------------------------
#der Name des Gerätes (darf keine Leerzeichen enthalten)
host_name   ppm40xx-0001

#Methode der Adresszuweisung: dhcp oder static
#static spart Resourcen im 40xx (RAM)
#-----------------------------------------
inet        static

#bei dhcp braucht nichts weiter folgen, es kann dns1 dns2 folgen
#bei static muss address, mask und evtentuell gateway und dns1/2 folgen

address     192.168.2.121
mask        255.255.255.0
gateway     192.168.2.2

#kann auch bei dhcp angegeben werden
dns1        192.168.2.2
#dns2        123.123.123.123
```

Die Zeilen, die mit ‚**#**' beginnen sind Kommentarzeilen. Diese werden ignoriert.

40xx USB-Modi

USB-Modi

Die externe USB-Schnittstelle (Mini-USB-Buchse) kann mit der Mikro-SD-Karte auf dem Controllerboard oder mit der USB-Schnittstelle des Controllers verbunden werden.

Die Verbindung mit der Mikro-SD-Karte wird im weiteren **USB-Stickmode** genannt.

USB–Stickmode

Der USB-Stickmode ist die Standardeinstellung. Dabei verhält sich der 40xx wie ein USB-Stick (USB 2.0 High Speed). Insofern der 40xx-Sensor durch Beenden der Stromversorgung ausgeschaltet wurde, ist dieser Modus auch ohne externe Stromversorgung verwendbar, d.h. es ist nur das USB-Kabel zum PC erforderlich. Ohne externe Stromversorgung bleiben die anderen Funktionselemente des Gerätes — außer die Konsole ‚com a' und dem USB-Stickmode — ausgeschaltet. Dies bedeutet, dass über den COM-A noch bestimmte Statusabfragen möglich sind (z.B. Kommando ‚ver').

Dieser Modus ermöglicht den Zugriff auf die integrierte SD–Karte, auf der u.a. die Konfigurations– und Skriptdateien gespeichert werden.

Hinweis:

Der 40xx–Logger muss immer über das Windows-übliche Auswurfverfahren vom PC getrennt werden, ansonsten kann ein einwandfreies Arbeiten des 40xx-Loggers nicht gewährleistet werden.

Kommando ‚set usb'

Genaueres zum USB–Stickmodus und der Konfiguration der USB–Schnittstelle finden Sie in der Beschreibung des Kommandos ‚set usb' ab Seite 271.

40xx Workflow

Der hier abgebildete Workflow, soll Ihnen dabei helfen zu verstehen, wie das ppmOS und der 40xx arbeiten.

Verwenden Sie die **Zwei-Seiten-Ansicht Ihrer PDF-Software** um den Workflow vollständig anzuzeigen.

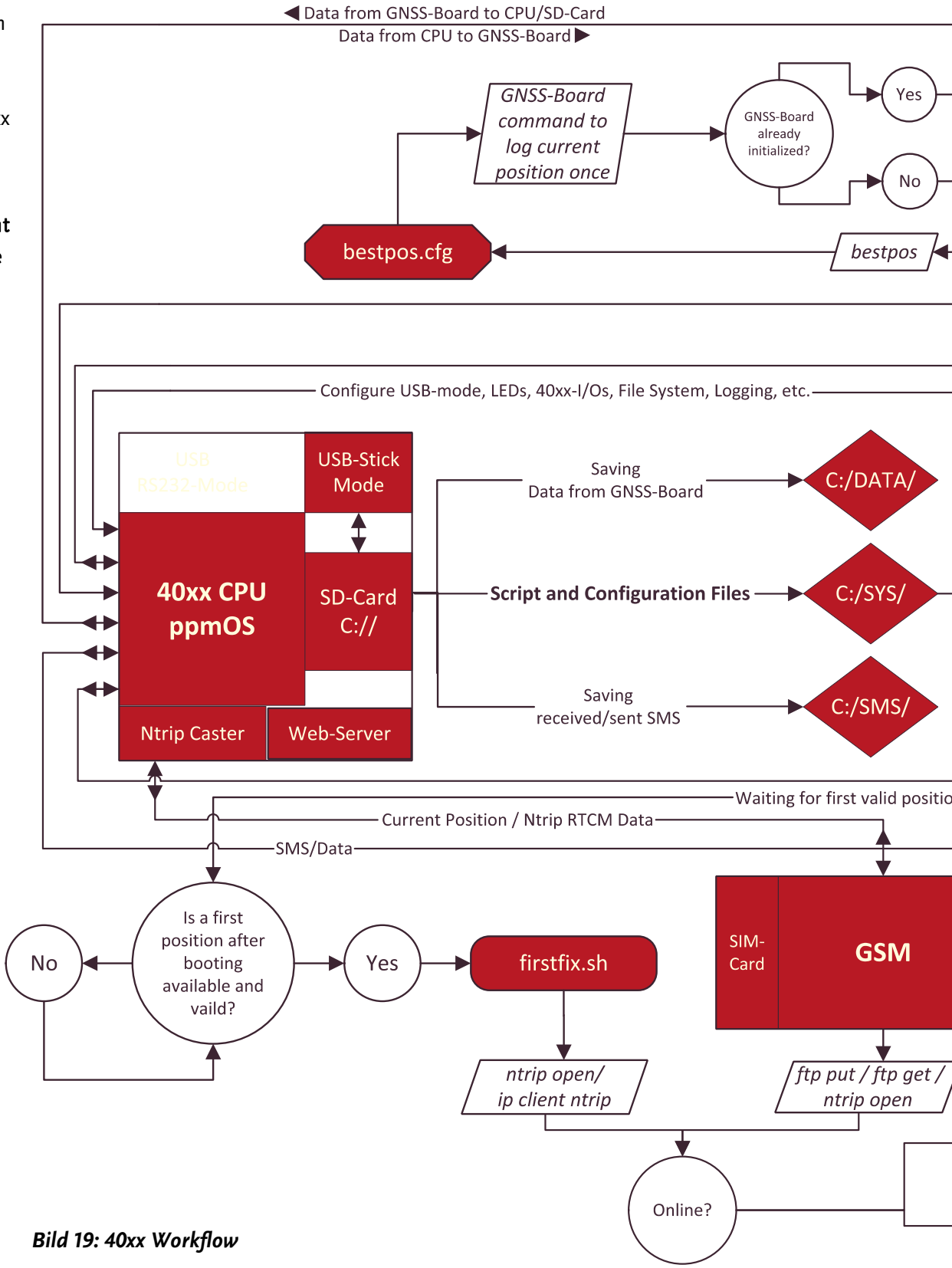
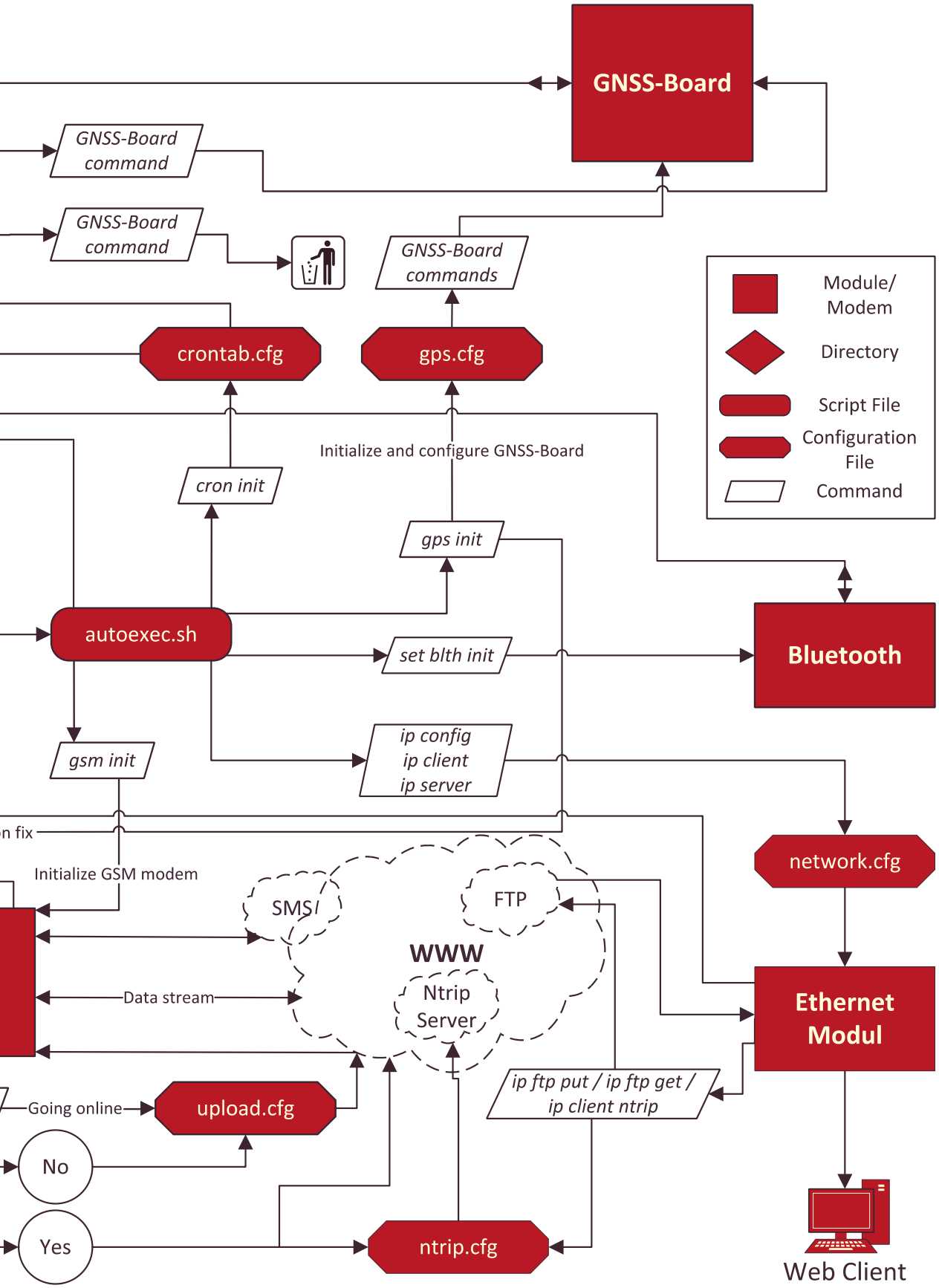


Bild 19: 40xx Workflow

40xx Workflow



Software - ppmOS

Interne Verwendung von GPS-Logs

Interne Verwendung der NMEA-Nachrichten $GPGGA und $GPRMC

Standardmäßig wird - unabhängig von der **gps.cfg** - das GPS-Board so konfiguriert, dass es auf seiner COM1, die mit der Controllerschnittstelle **gps1** verbunden ist, die NMEA-Datensätze $GPRMC und $GPGGA ausgibt. Diese werden vom **ppmOS** permanent überwacht, da sie folgende Bedeutung für den Betrieb des 40xx haben:

- Steuerung der LED 1 (rot-grüne Zweifarb-LED; s. S. 16)
- Stellen der Systemzeit und der Echtzeituhr
- Überwachung des Datums
- Überwachung des Positionsstatus
    - Ist eine Position vorhanden?
    - Welcher Lösungstyp ist vorhanden (RTK, RTK-Float, DGPS, Autonom, ...)
        - Hierzu wird ein Status-Update bei Änderung am COM A ausgegeben
- Überwachung der Anzahl verwendeter Satelliten
- Verwendung für WebSocket
- Sicherung hoher Zuverlässigkeit der Gerätefunktion

Die Überwachung der NMEA-Datensätze $GPRMC und $GPGGA kann alternativ auch an der Schnittstelle **gps2** erfolgen. Damit kann eine Trennung der Daten in rein binär und rein ascii an gps1/gps2 vorgenommen werden. Die Umstellung erfolgt über das Kommando 'gps nmea-scan' (s. S. 159).

Echtzeituhr

- Batteriegestützt (d.h. auch ohne GPS-Zeit/Positionsberechnung ist eine Zeit vorhanden)
- Unabhängig von der Systemzeit
- Wird nach Start des 40xx so lange in Systemzeit übernommen, bis eine GPS-Position/-Zeit verfügbar ist
- Wir d automatisch neu gestellt sobald eine Position verfügbar ist und gibt dann die GPS-Zeit
- Bildet Zeitbasis für Cron Jobs

Systemzeit

- Entspricht der Echtzeituhr bis eine GPS-Position/-Zeit verfügbar ist und wird dann auf GPS-Zeit gestellt
- Periodischer Zeitabgleich mit GPS-Zeit alle 10s

Interne Verwendung von GPS-Logs

  - Ab einer Abweichung von 100µs von der GPS-Zeit erfolgt eine Nachjustierung (s. Kommando 'gps echo' S.154)
- Dient der Datei(namen)erzeugung

Überprüfung der Gültigkeit der verwendeten Position

Die Positionsberechnung gilt als gültig, wenn Position, Datum und Uhrzeit ausgegeben werden, das Gültigkeitsflag im $GPRMC 'A' (2. Parameter) lautet und die CRC des Datensatzes richtig ist.

Sicherung der Gerätefunktion

Zur Sicherung einer hohen Zuverlässigkeit der Gerätefunktion wird die Datenübertragung von GPS-Board zum Controller überwacht. Das Kriterium für diesen Test ist das Empfangen und Verarbeiten von Daten durch den GPS-Task, der die Datenübertragung auf den Schnittstellen **gps1** oder **gps2** kontrolliert. Wenn diese Datenverarbeitung (Empfang, Speichern und/oder Weitersenden) länger als die mit dem Unterkommando 'gps timeout' (s. S. 143) gegebene Zeit ausfällt, dann wird die Kommandozeile, die zur Initialisierung des GPS-Boards ausgeführt wurde, erneut ausgeführt. Das Standardtimeout ist 3 s. Mit der Eingabe von ‚gps timeout 0' wird die Überwachung abgeschaltet.

Event-Verarbeitung

Event-Verarbeitung

Der 40xx-Sensor hat 5 Eventeingänge (Event_in iso und Event 1.1, 1.2, 2.1 und 2.2).

Der Eventeingang Event_in (SUB-D25, Pin 23) ist zusätzlich direkt mit dem GPS-Board verbunden um verzögerungsfrei dort entsprechende Aktionen (z. B. extra Positionsberechnung) auszulösen.

Die Eventeingänge 1.1 bis 2.2 sind in einer separaten Buchse zusammengefasst.

Das **ppmOS** kann an den Eventeingängen lediglich den Zustand abfragen, also 0 oder 1, sowie die Pulse an den Eingängen zählen. Meist ist jedoch nur der Übergang von einem zum anderen Zustand interessant.

Um nun bei einem Übergang eine einmalige Aktion auszuführen ist folgende Programmierung erforderlich:

Beispiel Signal an Input1.1:

```
set io in1.1 irq both
loop -8 -t 100 "@set io -s in1.1 irq clr || @stat -e; @echo Es trat ein Flankenwechsel auf"&
```

Die Abfrage erfolgt zyklisch in einem Task ‚**loop ... &**' alle 100 ms. Mit dieser Programmierung ist gesichert, dass auch Pulse bis zu minimal ca. 40 ns nicht ‚übersehen' werden.

Software - ppmOS

Multitasking im ppmOS

Multitasking im ppmOS

Das **ppmOS** ist ein event- und file-orientiertes preemptives Multitasking-Betriebssystem.

Event-orientiert

Unter event-orientiert ist zu verstehen, das die Ausführung von bestimmten Programmteilen nur dann erfolgt, wenn Ereignisse auftreten (z. B. Zeichen sind in einer UART eingetroffen, ein CAN-Paket ist vom Hardwaremodul empfangen worden oder ein Task übermittelt Daten an einen anderen Task). Eine Reihe von Events werden in Interrupt-Service-Routinen bearbeitet. Das allein macht noch kein Multitasking-Betriebssystem aus. Wenn eine Interrupt-Routine die Daten meist nur vorverarbeitet hat, dann erzeugt sie ebenfalls ein Event, das sie dem dafür zuständigen Task mitteilt. Der zuständige Task wird dann die bestimmungsgemäße Verarbeitung der Daten vornehmen. Wenn keine Events auftreten, dann geht der CPU in einen stromärmeren Stoppzustand.

File-orientiert

Unter file-orientiert ist zu verstehen, das Datenströme mit einer universellen Methode (generisches File) angesprochen und verarbeitet werden können.

Preemptiv

Unter preemptives Multitasking ist zu verstehen, dass zur gleichen Zeit mehrere Aufgaben (Tasks oder Programmteile) quasi gleichzeitig ausgeführt werden können. Da der verwendete Microcontroller nur über einen CPU-Kern verfügt, kann zu einem Zeitpunkt immer nur ein bestimmter Code ausgeführt werden. Dabei bedeutet „quasi gleichzeitig“, dass jeder Task immer nur einen kurzen Moment (Zeitscheibe) arbeitet, also die CPU zur Verfügung hat und nach einer bestimmten Zeit suspendiert wird um den nächsten Task die Gelegenheit zur Ausführung seines Programmcodes zu geben. Diese maximale Zeitscheibe ist im **ppmOS** standardmäßig 1 ms. Sie kann zur Laufzeit für jeden Task separat eingestellt werden.

Das ‚preemptiv‘ sagt, dass ein Task von außen in seiner Laufzeit begrenzt wird und er die Taskumschaltung nicht blockieren kann. In der Regel ist ein Task so programmiert, dass er nach der Bearbeitung seiner Events seine Rechenzeit abgibt. Macht er das nicht, dann sorgt der Taskumschalter (Taskscheduler) dafür, dass dem Task die CPU entzogen wird und ein nächster Task die CPU bekommt.

Task-Handling

Die im **ppmOS** implementierte Taskumschaltmethode ist eine ‚round robin‘-Methode. Die besagt, dass in einer vorbestimmten Reihenfolge allen Tasks nacheinander die CPU zugeteilt wird. Die CPU wird nicht zugeteilt, wenn kein Event für den jeweiligen Task vorliegt. Der Kontextwechsel, also die Zeit von Suspendierung eines laufenden Tasks bis zur Aktivierung des nächsten bereiten Tasks dauert ca. 1-3 µs.

Software - ppmOS

Multitasking im ppmOS

Tasks können in ihrer CPU-Nutzung priorisiert werden. Das geschieht durch Zuweisung einer maximalen Zeitscheibe für jeden einzelnen Task. Eine hohe Priorität eines Tasks bedeutet kürzere maximale Zeitscheiben für die anderen Tasks. Dabei entstehen zwangsläufig Latenzen vom Auftreten eines Events bis zur Bearbeitung des Events. Diese ergeben sich aus der im **ppmOS** implementierten ‚round robin'-Methode. Die Latenzzeiten sind im **ppmOS** berechenbar. Worst-Case-Latenzen liegen im Bereich von wenigen 100 µs bis zu einigen Millisekunden. Die statistisch gesehen durchschnittlichen Latenzen liegen im Bereich von wenigen Mikrosekunden. Für eine Echtzeitverarbeitung unterhalb der Millisekundenschwelle sind aber die Interrupt-Service-Routinen unerlässlich (z. B. Empfang von Zeichen aus der UART oder der Empfang von Ethernet-Paketen), eine Bearbeitung in Shell-Skripten ist in der Regel dann nicht mehr möglich. Interrupt-Service-Routinen bieten Latenzzeiten im Bereich von einigen Mikrosekunden. Die Programmierung zur Behandlung einer Echtzeitanforderung ist in erster Linie von der konkreten Definition der Echtzeit also der längsten erlaubten Reaktions- und Verarbeitungszeit abhängig.

Ein Task ist gekennzeichnet durch eine Taskdatenstruktur, die alle für diesen Task relevanten Daten aufnimmt. Dazu gehört ein eigener Taskstack. Eine zentrale Bedeutung hat das Event-Flag-Wort des Tasks. Es ist eine 32-Bit-Variable. Darüber läuft die Aktivierung des Tasks. Ist das Event-Flag-Wort 0, dann gibt es kein Event für den Task, er ist wartend und verbraucht keine CPU-Zeit. Wird der Wert ungleich 0, dann wird der Taskscheduler dem Task die CPU zuteilen. Von den Bits im Eventf-Flag-Wort hat das Bit 31 eine besondere Bedeutung. Es ist das FREEZED-Bit. Ist es gesetzt, dann ist der Task eingefroren, er erhält keine Rechenzeit unabhängig von eventuell anderen gesetzten Bits. Wenn ein Task in seinem Programmablauf eine bestimmte Zeit warten will, dann sorgt der Taskscheduler für das ‚Einfrieren' und ‚Auftauen' des Tasks sodass der Task in der Wartezeit keine CPU-Zeit verbraucht.

Ressourcen

Schon wenn eine Geräte-Firmware über Interrupts und eine Programmhauptebene verfügt, gibt es das Problem von Unterbrechungen zu Zeitpunkten bei denen Dateninkonsistenz entstehen kann. Dieses Problem muss bei einem Multitasking-Betriebssystem besondere Beachtung erfahren. Es geht dabei um Ressourcen, auf die wegen ihrer Komplexität zu einem Zeitpunkt nur ein Task schreibenden Zugriff haben darf (auch atomarer Zugriff, also nicht unterbrechbarer bzw. teilbarer Zugriff).

Solche Ressourcen können sein: FAT-Dateisystem, GSM-Modem, einzelne Systemstrukturen oder Variablen. Die größte Einheit, auf die ein Task, ohne unterbrochen werden zu können, zugreifen kann ist ein 32-Bit-Wert. Für alle komplexeren Datenstrukturen muss für die Dauer der Änderung eine Zugriffsverriegelung aktiviert werden. Im **ppmOS** gibt es dafür intern mehrere Möglichkeiten, die alle darauf hinauslaufen, dass ein Task, der eine Ressource ändern will, sie vor dem Zugriff in der Weise verriegelt, dass andere Tasks, die diese Ressource ebenfalls nutzen möchten, solange suspendiert werden, bis die Ressource wieder frei ist.

In der Shell-Programmierung brauchen diese Umstände vom Anwender in der Regel nicht berücksichtigt werden, da dies bei der Implementierung der Shell-Kommandos berücksichtigt wurde.

Software - ppmOS

File-Konzept

Das File-Konzept des ppmOS

Das ppmOS ist fileorientiert. D. h., dass Daten, die empfangen oder erzeugt werden, in Files gespeichert oder dorthin bzw. von dort übertragen werden. Dazu gehören in erster Linie die Files oder Dateien auf der internen Mikro-SD-Karte im FAT-Dateisystem (FAT16 oder FAT32). Im Sinne des ppmOS ist z. B. auch eine serielle Schnittstelle (UART) ein File.

Liste der Filetypen im ppmOS:

Tabelle 15: Filetypen im ppmOS

reguläre Dateien auf der internen Micro-SD-Karte
c:/sys/autoexec.sh Files im FAT-Dateisystem auf der internen Mikro-SD-Karte bestehend aus Laufwerksbuchstabe c:, Pfad /sys/ und Dateiname.
serielle Schnittstellen (UART)
coma (com3) Schnittstelle am Frontpanel
comb (com4) Schnittstelle (SUB-D25)
comc (com2) Schnittstelle (SUB-D25)
blth (com1) Schnittstelle über Bluetooth (point to point)
gps1 (com8) interne Schnittstelle zur ersten UART des GPS-Boards
gps2 (com6) interne Schnittstelle zur zweiten UART des GPS-Boards
gsm (com7) interne Schnittstelle zum GSM-Modem
logische Schnittstellen, die auf ein File verweisen
stdout Dieses File ist ein Ausgabefile. Es gibt in jedem Gerät eine physische Schnittstelle, die die Standardausgabe ist. Beim ppm40xx-Gerät ist das coma.
stdin Dieses File ist ein Eingabefile. Es gibt in jedem Gerät eine physische Schnittstelle, die die Standardeingabe ist. Beim ppm40xx-Gerät ist das coma. Standardmäßig werden Shell-Kommandos nach stdin gesendet, die interpretiert und ausgeführt werden.
Siehe nächste Seite.

File-Konzept

Tabelle 15 (f.): Filetypen im ppmOS

pipe1, pipe2, pipe3, ...,pipe20

Temporärer Zwischenspeicher für Daten.

Sie werden z. B. verwendet, um kleinere Datenmengen zu sammeln und von einem anderen Task in größeren Abschnitten verarbeiten zu lassen.

Wenn er voll wird, überschreiben neue Daten die alten Daten (auch zirkularer Puffer genannt), da die Pipe standardmäßig im ‚nonblocking mode' geöffnet wird. D. h. um ein Überlaufen zu verhindern, muss der Abnehmer der Daten diese schneller verarbeiten können bzw. häufig genug die Verarbeitung vornehmen.

Es handelt sich um dynamische Pipes mit einstellbarer Größe und Eigenschaft.

pipe[1...20][-[<size>]][-[b|n][d]]]

b – blocking

Wenn beim Schreiben in eine Pipe der maximale Füllstand erreicht ist, dann warted der schreibende Task solange bis Daten aus der Pipe gelesen wurden. Ohne 'b' wird der schreibende Task nicht angehalten, es gehen aber Daten verloren.

n - no overwrite in non blk

Werden Daten schneller geschrieben als gelesen, dann werden die neueren Daten verworfen. Ohne 'n' werden die ältesten Daten verworfen und die neuen kommen in die Pipe.

d - free mem after last close

Nachdem eine Pipe nicht mehr benötigt wird, dann wird beim letzten File close der reservierte Speicher freigegeben.

Es sind maximal 20 Pipes (pipe1 ... pipe20) möglich.

Alle Pipes haben per Default, aber erst nach der ersten Verwendung, eine Größe von 6000 Byte. Ohne Verwendung wird kein Speicher benötigt.

Beispiel:

set ird comb tee pipe1

cp -t -8 pipe1 -a c:/data/logfile.comb &

File-Konzept

Tabelle 15 (f.): Filetypen im ppmOS

zero **zero_<Länge>** Spezialdatei, aus der Bytes mit dem Wert 0 gelesen werden können. Dieser File-Typ dient dazu eine definierte Menge von Daten, meist zu Testzwecken, zu erzeugen. Aus dem File ‚**zero**' können Bytes (max. 4 GByte) mit dem Wert 0 gelesen werden und aus ‚**zero_<Länge>**' können maximal <Länge> Bytes gelesen werden. Es können zugleich maximal 4 zero-Files geöffnet sein.
null **nul** **dmy** Spezialdatei, in die eine beliebige Anzahl von Bytes geschrieben werden kann. Dieses Bytes werden nirgendwo gespeichert. Dieser Filetyp dient dazu, Ausgaben eines Kommandos zu verwerfen oder Daten oder Files im ‚Nichts' verschwinden zu lassen. Beispiel: **@ls c:/data/file_xyz >null && @echo file existiert** Das List-Kommando wird ausgeführt, dessen Ausgabe wird verworfen und der Rückkehrcode wird in einer bedingten Kommandoverknüpfung verwendet.

File-Konzept

Tabelle 15 (f.): Filetypen im ppmOS

can1 **Ausgabefile:** **can1-<id>[-[e][r][b][p<tmot_pak>|t<tmot_µs>]]** **Eingabefile:** **can1-<id>[-[e][r]][-<buf_size>]** **can1-f<filter_ix>[-[e][r]][-<buf_size>]** Ein- oder Ausgabefile für die CAN-Schnittstelle. Da das CAN-Protokoll paketorientiert ist und jedes Paket eine ID hat, kann es viele CAN-Ein-Ausgabefiles geben. Das ppmOS unterstützt gleichzeitig maximal 10 Empfangs- und 10 Ausgabefiles. Vor der Öffnung und Verwendung eines CAN-Eingabefiles muss ein entsprechender CAN-Filter eingestellt werden. Dieser ermöglicht erst einmal den Empfang von Paketen mit der angegebenen ID. Das Senden und Empfangen ist in Hardware und zusätzlich auch in Software gepuffert. Das ppmOS puffert bei der Ausgabe standardmäßig 100 CAN-Pakete, die jeweils maximal 8 Byte Nutzdaten transportieren können. Bei der Eingabe werden pro Empfangs-FIFO (rx0 und rx1) 100 Pakete mit max. je 8 Byte gepuffert. Für den Empfang kann beim Öffnen für jedes CAN-File ein weiterer Puffer eingestellt werden. Dessen Standardgröße beträgt pro CAN-Empfangsfile 400 Byte. Bei Senden von CAN-Daten muß im Filenamen immer die ID des Paketes angegeben werden. Bei Empfang kann der Filename eine ID enthalten oder er kann sich auf eine Filternummer beziehen. Das ist dann zweckmäßig, wenn der Filter im Maskenmode eingestellt ist und durch ihn Pakete einer ganzen Gruppe von ID's empfangen werden sollen. Weitere Informationen sind beim Kommando ‚can' zu finden.

File-Konzept

Tabelle 15 (f.): Filetypen im ppmOS

icom{1|2|3|4|5} **icom{1|2|3|4|5}-<ip_addr>:<port>[-{udp|tcp}][-ack]** Ein-/Ausgabefile einer TCP- oder UDP-Client-Verbindung. Aus diesem File können Daten gelesen oder es können Daten dorthin geschrieben werden. Weitere Information sind beim Kommando ‚**ip**' zu finden. Beispiel: **ip client start icom1_192.168.1.120:12000** **gps reply gps1 asc icom1** oder **gps reply gps1 asc icom1_192.168.1.120:12000-udp**
ips{a|b|c|d}[*|1|2|3|4] Ein-/Ausgabefile einer TCP-Server-Verbindung. Aus diesem File können Daten gelesen oder es können Daten dorthin geschrieben werden. Weitere Information sind beim Kommando ‚**ip**' zu finden. Beispiel: **ip server start ipsa:10000 -n 4 -e** **set ird gps1 tee ipsa*** Die Daten die über gps1 ankommen, werden an den Server ipsa übertragen und der überträgt die Daten an alle (Zeichen ‚*') oder einen benannten (1, 2, 3 oder 4) verbundenen Client. Für ein hohes Datenaufkommen wird die Variante über einen Pipe-Zwischenpuffer empfohlen. Dabei werden größere TCP-Pakete verschickt, nämlich alles was sich nach im Beispiel 40 ms angesammelt hat: **set ird comc tee pipe2** **loop -8 -t 40 @cp pipe2 ipsa* &**

File-Konzept

Tabelle 15 (f.): Filetypen im ppmOS

mem-{[<name>-][<adr>-]}<len>-[p][d|k][f|e][t][v]

Memoryfile.

Spezial-File für den Zugriff auf einen Speicherbereich mit vorgegebener Länge, den das ppmOS bereitstellt.

Standardmäßig gibt man einen Namen und die Länge an und überlässt dem ppmOS die konkrete Speicherzuweisung.

Für Spezialanwendungen kann auch eine Speicheradresse vorgegeben werden. Das erfordert aber genaue Kenntnisse über das ppmOS. Wird ein Speicherbereich angegeben, der vom ppmOS verwendet wird, dann kommt es zu Programmfehlern oder Neustart der Software.

Ein Memoryfile wird z. B. verwendet um Daten aus einem Stream mehrfach weiterzugeben:

gps reply gps1 asc pipe1

loop -8 -t 5 "@cp pipe1 mem-buf-400-p; @cp mem-buf -a comb; @cp mem-buf -a ipsa1; @cp mem-buf -a icom1"&

[<name>]

Frei wählbarer Bezeichner mit max. 12 Zeichen. Der Name muss mit einem Buchstaben beginnen und er kann Ziffern und das Zeichen ‚_' enthalten. Es wird zwischen Groß- und Kleinschreibung unterschieden.

[<adr>]

Anfangsadresse des Speicherfensters. Ohne genaue Kenntnisse über das ppmOS wird von der Verwendung abgeraten. Wird die Angabe weggelassen, dann wird ein Speicherbereich vom ppmOS im RAM (Heap) an passender Stelle reserviert.

<len>

Länge des Speicherfensters in Byte

p das File ist persistent, es bleibt erhalten bis zum power off bzw. bis es explizit mit ‚rm' gelöscht wird.

Siehe f.

File-Konzept

Tabelle 15 (f.): Filetypen im ppmOS

„**mem-{[<name>-][<adr>-]}<len>-[p][d|k][f|e][t][v]**“ f.

d Das File wird, nachdem es geschlossen wird, automatisch gelöscht (deleted). Ein File kann zur gleichen Zeit mehrfach geöffnet sein. Das Löschen erfolgt erst wenn alle Tasks das File geschlossen haben.

k Das File wird, nachdem es geschlossen wird, zwangsweise gelöscht (killed) unabhängig davon, ob es in einem anderen Task noch geöffnet ist.

f Das File hat nach dem Öffnen die angegebene Länge. Es wird als volles (full) File geöffnet.

e Das File hat nach dem Öffnen die Länge 0. Es wird als leeres (empty) File geöffnet. Das ist der Standardwert wenn weder ‚f‘ noch ‚e‘ angegeben wird.

t Wird verwendet, um eine Zusammenfassung aller Memoryfiles auszugeben (**ls mem--tv** oder nur **mem--tv**).

v wird im Zusammenhang mit ‚t‘ verwendet um eine ausführliche (verbose) Ausgabe aller Memoryfiles zu erhalten (**ls mem--tv** oder nur **mem--tv**).

Es können hunderte Memoryfiles angelegt werden. Die Anzahl ist durch die Summe der Einzellängen begrenzt.

File-Konzept

Tabelle 15 (f.): Filetypen im ppmOS

vcom1 Virtueller COM-Port über USB. (steht nur zur Verfügung, wenn das GPS-Board nicht benötigt wird und ein entsprechender Patch vorgenommen wurde)

Systemvariable

Systemvariable

Im **ppmOS** gibt es Systemvariablen, deren Wert sich entweder von Gerät zu Gerät oder zur Laufzeit ändern können. Mit ihrer Hilfe können in Shellscripten Bezug auf ihre Werte genommen werden ohne den tatsächlichen Wert zu kennen.

Die Systemvariable wird im Moment der Interpretation durch den Shell-Interpreter aufgelöst. D. h. der Variablenname wird durch einen Text, der den Wert der Variablen repräsentiert, ersetzt.

Bei der Anwendung von Systemvariablen ist besonders auf den Zeitpunkt der Variablenauflösung zu achten.

Beispiel

richtig:

```
cron put 0/60 * * * * * * "@echo Time Mark: $(tm)"
```

Die Variable $(tm) wird erst aufgelöst, wenn der Cronmanager den Cronjob ausführt.

falsch:

```
cron put 0/60 * * * * * * @echo Time Mark: $(tm)
```

Die Variable $(tm) würde schon beim Ausführen von ‚cron put ...' aufgelöst werden und später, wenn der Cronmanager den Cronjob ausführt, würde immer die Uhrzeit des ‚cron put ...' ausgegeben werden.

Eine Liste der vorhanden Systemvariablen finden Sie auf den Seiten 63-67.

Systemvariable

In der Version v 1.46 des ppmOS sind folgende Systemvariablen vorhanden:

Tabelle 16: Systemvariablen im ppmOS

Verwendung	Wert setzen	Beschreibung und Beispiel
$(syspath)	**set syspath** **set syspath <path>**	Gibt den aktuell gesetzten Systempfad aus. Speichert den aktuellen Systempfad. Mit dieser Variable können verschiedene Systemkonfigurationen, die durch ein Verzeichnis im Dateisystem repräsentiert sind, ausgewählt werden. Der Wert bleibt auch nach dem Ausschalten des Gerätes erhalten. **@echo $(syspath)** **c:/sys/**
$(sys)	**set sys** **set sys <path>**	Gibt den aktuellen Pfad für Skripts (Kommandos) aus Speichert den Pfad für Skriptaufrufe. Nach der Eingabe eines Kommandos wird zuerst in der Liste der internen Kommandos gesucht. Wird es dort nicht gefunden, dann wird in dem Pfad $(sys) nach einem gleichnamigen Skript gesucht. **set sys c:/sys/** Ein Skriptfile erzeugen **echo @echo hello world! >$(sys)hello** Aufruf ohne Pfadangabe **hello** Ausgabe **hello world!**
	Siehe nächste Seite	

Systemvariable

Tabelle 16 (f.): Systemvariablen im ppmOS

Verwendung	Wert setzen	Beschreibung und Beispiel
$(fname)	nach ‚gps log on' enthält die Variable den aktuellen Logfilenamen, der auch nach ‚gps log off' erhalten bleibt. oder **gps set fname**	**gps log on** **@echo $(fname)** **c:/DATA/2017/06/06/log-170607-195803-83756.9F6**
$(fn_ext)	nur lesbar, wird vom ppmOS aus der Seriennummer des GPS-Boards ermittelt	Es wird ein 3 Buchstaben langer Textstring zurückgegeben, der ein Hashwert der Seriennummer des eingebauten GPS-Boards ist. **@echo $(fn_ext)** **9F6**
$(gpgga)	nur lesbar, der GPS-Task aktualisiert den Inhalt aus dem letzten Empfang an der Schnittstelle **gps1** und/oder **gps2**	Es wird der zuletzt über **gps1** und/oder **gps2** empfangene G*GGA-Datensatz zurückgegeben. Das ‚*' steht für einen Buchstaben (z. B. P, N, L usw.). Es werden die NMEA-Datensätze GPGGA, GNGGA, GLGGA, usw. erkannt. In der Standard-GPS-Initialisierung erfolgt der Refresh alle 1 Sekunde. Siehe auch: **gps nmea-scan** (s. S. 159)
$(gprmc)	nur lesbar, der GPS-Task aktualisiert den Inhalt aus dem letzten Empfang an der Schnittstelle **gps1** und/oder **gps2**	Es wird der zuletzt über **gps1** und/oder **gps2** empfangene G*RMC-Datensatz zurückgegeben. Das ‚*' steht für einen Buchstaben (z. B. P, N, L usw.). Es werden die NMEA-Datensätze GPRMC, GNRMC, GLRMC, usw. erkannt. In der Standard-GPS-Initialisierung erfolgt der Refresh alle 1 Sekunde. Siehe auch: **gps nmea-scan** (s. S. 159)
	Siehe nächste Seite	

Systemvariable

Tabelle 16 (f.): Systemvariablen im ppmOS

Verwendung	Wert setzen	Beschreibung und Beispiel
$(tm)	nur lesbar	Gibt die aktuelle Systemzeit zurück und kann z. B. verwendet werden um einen Dateinamen zu bilden. **gps reply gps2 asc c:/data/gps2-asc-$(tm).log** ergibt **gps reply gps2 asc c:/data/gps2-asc-210331-092723-879925.log** Darin bedeuten: 210331 – 2021 März 31. 092723 - 9:27:23 879925 - Bruchteil der Sekunde, 9:27:23,879925 s
$(tmd)	nur lesbar	Gibt die aktuelle Systemzeit dezimal zurück und kann z. B. verwendet werden um einen Dateinamen zu bilden. **gps reply gps2 asc c:/data/gps2-asc-$(tmd).log** ergibt **gps reply gps2 asc c:/data/gps2-asc-1617182843.880021.log** Darin bedeuten: **1617182843** – Sek. seit 1. 1. 1970, 2021 3. 31. 9:27:23 **.880021** – Bruchteil der Sekunde, 9:27:23,985386 s
	Siehe nächste Seite.	

Systemvariable

Tabelle 16 (f.): Systemvariablen im ppmOS

Verwendung	Wert setzen	Beschreibung und Beispiel
$(tmx)	nur lesbar	Gibt die aktuelle Systemzeit hexadezimal zurück und kann z. B. verwendet werden um einen Dateinamen zu bilden. **gps reply gps2 asc c:/data/gps2-asc-$(tmx).log** ergibt **gps reply gps2 asc c:/data/gps2-asc-6064407b.000.log** Darin bedeuten: **6064407b** – Sek. seit 1. 1. 1970, 2021 3. 31. 9:27:23 **.000** - Zähler (hexadezimal) für $(tmx)-Abfragen in der aktuellen Sekunde zur Unterscheidung von mehreren Abfragen innerhalb einer Sekunde.
$(tma)	nur lesbar	Gibt die aktuelle Systemzeit ASCII-kodiert zurück und kann z. B. verwendet werden um einen Dateinamen zu bilden. **gps reply gps2 asc c:/data/gps2-asc-$(tma).log** ergibt **gps reply gps2 asc c:/data/gps2-asc-B3V92723.000.log** Darin bedeuten: **B3V92723** – 2021 3. 31. 9:27:23 **.000** – Zähler (hexadezimal) für $(tma)-Abfragen in der aktuellen Sekunde zur Unterscheidung von mehreren Abfragen innerhalb einer Sekunde. Minuten und Sekunden werden uncodiert angegeben. Die Stunden werden einstellig von 0, ..., 9, A, B, ..., N angegeben. Der Tag wird von 1, .., 9, A, B, ..., V angegeben. Der Monat wird mit 1, ..., 9, A, B, C angegeben. Das Jahr wird ab 2010 gezählt und mit 0, ..., 9, ..., Z angegeben.
	Siehe nächste Seite.	

Systemvariable

Tabelle 16 (f.): Systemvariablen im ppmOS

Verwendung	Wert setzen	Beschreibung und Beispiel
$(uid)	nur lesbar, für jedes Gerät einmalig und nicht änderbar	Gibt den UID (unique identifier) der im Gerät installierten CPU zurück. Das kann dazu dienen, Logfiles oder andere Steuerungen gerätebezogen zu programmieren. **echo $(uid)** ergibt 3334.3032.3532.470b.001c.002d
$(uid_s)	nur lesbar, für jedes Gerät einmalig und nicht änderbar	Gibt den UID (unique identifier) der im Gerät installierten CPU zurück. Das kann dazu dienen, Logfiles oder andere Steuerungen gerätebezogen zu programmieren. **echo $(uid_s)** ergibt 333430323532470b001c002d, also die Kurzform ohne ‚.'.
$(uid_hash)	nur lesbar, für jedes Gerät einmalig und nicht änderbar	Gibt den UID (unique identifier) der im Gerät installierten CPU zurück. Das kann dazu dienen, Logfiles oder andere Steuerungen gerätebezogen zu programmieren. **echo $(uid_hash)** ergibt 061a7714 Das ist die XOR-Verknüpfung der 3 32-Bit-Werte der UID.
$(dev)	nur lesbar	Gibt den Gerätetyp zurück. **echo $(dev)** ergibt ppm40xx
$(ver)	nur lesbar	Gibt die Version der Firmware zurück. **echo $(ver)** ergibt 1.46

WebSocket und FTP-Server

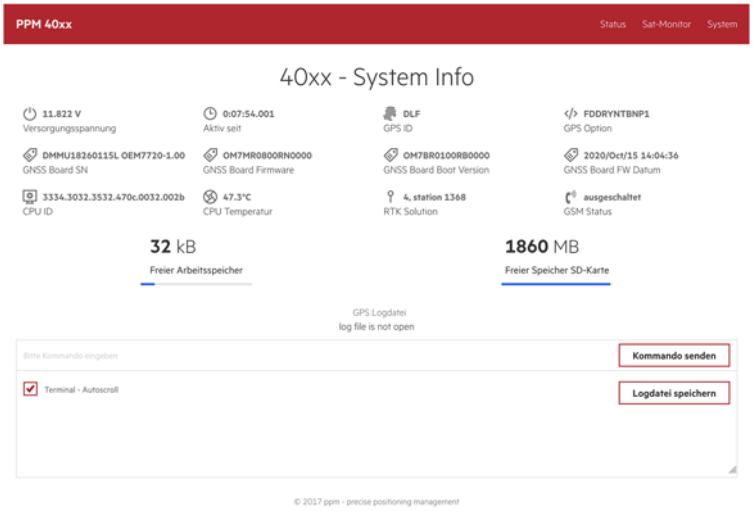
WebSocket und FTP Server

WebSocket

Der 40xx bietet einen Websocket zur Anzeige des aktuellen Gerätestatus über Ethernet. Hier werden permanent aktualisierte Informationen, z.B. zum freien Arbeitsspeicher, des freien SD-Kartenspeichers, der GSM-Empfangsstärke oder zum GPS-Status, angezeigt. Ferner ist ein Kommando-Terminal implementiert.

Bild 20: WebSocket Home

Bild 21: WebSocket System Info



Der 40xx wird mit den notwendigen Dateien und der notwendigen Konfiguration für die WebSocket-Anwendung ausgeliefert. Die Dateien können Sie selbstverständlich auch über support@ppmgmbh.com anfragen.

Im Wurzelverzeichnis der SD-Karte muss ein Verzeichnis z.B. c:/sys/www/ angelegt werden, in welchem die bei Auslieferung vorhandenen WebSocket-Dateien liegen.

Mit den folgenden Kommandos kann dann zum Test im lokalen Netz die 40xx-Status-Website aktiviert und anschließend in einem Browser aufgerufen werden:

```
ip server start ipsb:23655 -r -ws

ip server start http -t 1 c:/www/index.html
```

Es wird das Ethernet-Modul (dchp oder static - Eintrag in c:/sys/network.cfg s. S. 45) initialisiert.

Das Verzeichnis welches die WebSocket-Dateien enthält (hier c:/www/), kann natürlich jeden beliebigen Namen haben (z.B. c:/40xx-Websocket/) und in beliebig vielen Unterordnern liegen. Es muss lediglich im obigen Kommando der korrekte Pfad angegeben werden.

Der Port 23655 ist in den hinterlegten Java Script Anwendungen implementiert, und somit festgelegt. Der http-Port kann beliebig gewählt oder weggelassen werden. Wird er weggelassen, wird der Standard-Port 80 gesetzt.

Software - ppmOS

WebSocket und FTP-Server

Im Web-Browser im lokalen Netz die IP-Adresse des 40xx zusammen mit dem Port 37925 in der URL-Leiste eingegeben werden (<IP-Adresse>:37925).

Zu beachten ist, dass im Browser auf der Webseite erst dann Daten im Web-Browser erscheinen, wenn das GPS-Board initialisiert ist. Also davor oder danach folgendes Kommando ausführen (in der Regel bereits in der Datei autoexec.sh):

```
gps init
```

Es macht Sinn, den Start des WebSockets in ein Skript zu verlegen, und dieses verzögert über einen Cron Job in der autoexec.sh aufzurufen, z.B:

cron put start-ws *+6. * * * * * * c:/sys/start-websocket

Das Skript (hier: 'start-websocket') beinhaltet dann mindestens die obigen Kommandos:

```
ip server start ipsb:23655 -r -ws
ip server start http:37925 -t 1 c:/www/index.html
```

Und kann zusätzlich folgendes enthalten:

#Prüfen ob GPS-Task aktiv, also ob GPS-Board initialisiert ist, falls nicht, dann aktivieren/initialisieren:

```
tsk exist gps || gps init&
```

FTP Server

Der FTP-Server wird gestartet mit:

```
ip server start ftp
```

Die Zugangsdaten lauten:

User: ppm40xx

Passw: 40xx-ftp-pw

Der FTP-Server kann temporär gestartet und auch im laufenden Betrieb beendet werden.

Getestet mit Total-Commander, Midnight-Commander (Linux) und Firefox. In Edge oder IE etwas hakelig.

Download mit Total-Commander ca. 4.3 MB/s

Software - ppmOS

40xx CAN

40xx CAN

Zur Anbindung an bestehende CAN-Datenbusse, verfügt der 40xx über eine entsprechende CAN-Schnittstelle.

Über die CAN-Schnittstelle des 40xx können Daten empfangen und gesendet werden. Hierzu kann entweder ein DBC-File verwendet werden, oder es werden CAN-Files im Sinne des ppmOS-File-Konzepts (s. S. 54) erstellt.

Verwendung von CAN-Files

Bei der Verwendung von CAN-Files, welche z.B. zum Datenaustausch zwischen zwei 40xx verwendet werden können, müssen für den Empfang von Daten Filter gesetzt werden. Die empfangenen Daten werden dann in ein CAN -File geschrieben und können von diesem aus weiterverarbeitet werden (z.B. Weiterleitung in internen Speicher). Für das Senden von Daten reicht es ein CAN-File zu öffnen, und dieses – bzw. dessen Inhalt - anhand dessen ID einem Abnehmer zur Verfügung zu stellen.

Folgend finden Sie ein einfaches Konfigurationsbeispiel in welchem zwei 40xx zum gemeinsamen Datenaustausch verwendet werden. Hierfür wird ein 40xx zur Ausgabe bzw. dem Senden des NovAtel BESTPOS-Datensatzes am CAN-Port, und ein weiterer 40xx zum Empfang dieser Daten konfiguriert. Es handelt sich hier um die minimal notwendige Konfiguration.

Konfiguration zur Ausgabe des BESTPOS-Datensatzes am CAN-Port

Mindestinhalt der Datei 'autoexec.sh':

```
--------------------------------
#GPS initialisieren
gps init
#CAN-Bus initialisieren (hier Bitrate 100k)
can init bps 100k
```

Mindestinhalt der Datei 'gps.cfg':

```
---------------------------------
GPS_Initialisation Novatel 0.1
#Erzeugung des Datensatzes und Einstellung der Ausgaberate
log com2 bestposb ontime 1
```

Mindestinhalt der Datei 'firstfix.sh':

```
---------------------------------
#Ausgabe des BESTPOSB-Datensatzes in das CAN-File „can1-0x21" ab Positionsfix (firstfix.sh).
#Kann auch in anderer Datei stehen (wenn in gps.cfg Kommando 'exec' voranstellen)
gps reply gps2 bin can1-0x21
```

Software - ppmOS

40xx CAN

Konfiguration zum Empfang des BESTPOS-Datensatzes am CAN-Port

can init bps 100k

can filt 0 on rx0 l32 0x21 0x22

Bei CAN-Empfang mit Hexausgabe ist das Standardausgabefile des CAN-Tasks COM-A.

HEX-Ausgabe am COMA bei zeichenweiser Datenübertragung:

```
CAN:   0 0   0         33     17938 1 42                        B
CAN:   0 0   0         33     17995 1 3d                        =
CAN:   0 0   0         33     18053 1 00                        .
CAN:   0 0   0         33     18113 1 00                        .
CAN:   0 0   0         33     18173 1 00                        .
CAN:   0 0   0         33     18233 1 86                        .
CAN:   0 0   0         33     18290 1 c8                        .
```

HEX-Ausgabe am COMA bei datensatzweiser Datenübertragung (hier Schalter p beim Sende-CAN-File):

```
CAN:   0 0   0         33      3037 8 00 00 00 00 00 00 09 09   ........
CAN:   0 0   0         33     35769 8 09 00 00 02 00 01 83 db   ........
CAN:   0 0   0         33     35891 8 9d f0 aa 44 12 1c 2a 00   ...D..*.
CAN:   0 0   0         33     36009 8 00 40 48 00 00 00 ba b4   .@H.....
CAN:   0 0   0         33     36129 8 4d 08 a8 a4 8c 0d 00 00   M.......
CAN:   0 0   0         33     36245 8 00 02 f6 b1 cf 3d 00 00   .....=..
CAN:   0 0   0         33     36363 8 00 00 10 00 00 00 94 57   .......W
```

Binäre Ausgabe der empfangenen BESTPOS Messages:

Ist anstelle der Hexausgabe die binäre Ausgabe der empfangenen (binären) BESTPOS Messages gewünscht, so kann am Empfänger-40xx ein CAN-File mit zum Filter passender ID geöffnet werden. Die eingehenden Daten werden dann in dieses CAN-File geschrieben.

Anschließend kann aus diesem File gelesen werden, sodass die BESTPOS-Datensätze binär z.B. am COM-A ausgegeben bzw. angezeigt werden.

Beispiel:

can open can1-0x22

cat -t -8 can1-0x22 &

Software - ppmOS

40xx CAN

Hinweis zur Nutzung von ID-Gruppen beim Empfang von CAN-Paketen:

Wird per Maskenfilter eine ID-Gruppe definiert, so kann mit nur einer für ein CAN-File verwendeten ID aus eben dieser ID-Gruppe (bzw. Bereich), jedes empfangene CAN-Paket, welches den Maskenfilter passiert, in dieses CAN-File geleitet werden. Beispiel:

can filt 0 on rx0 m32 0x130 0x7f0 #Filter für ID's 0x130 bis 0x13f - 16 ID's: 0x130, 0x131, ..., 0x13f

can open can1-0x133

Es ist lediglich eine beliebige Adresse aus dem Bereich anzugeben, dann werden alle CAN-Pakete, die diesen Maskenfilter passieren, in das CAN-File geleitet (s. hierzu Kommando 'can filt' auf S. 94).

Verwendung von DBC-Files

Als Alternative zum Senden von Daten per CAN-Files, ist im ppmOS bzw. dem 40xx auch das CAN-DBC-Fileformat zur Definition der CAN-Paketkonfiguration implementiert.

Die CAN-DBC-Konfiguration wird über das Unterkommando 'dbc' des Kommandos 'can' realisiert. Eine genaue Beschreibung des (Unter-)Kommandos finden Sie auf Seite 92.

Ein aktuelles DBC-File sowie eine Übersicht bzw. Beschreibung aller im ppmOS verfügbaren CAN-Variablen, senden wir Ihnen auf Anfrage gerne jederzeit zu. Bitte senden Sie hierfür einfach ein E-Mail mit dem Betreff „Übersicht CAN-Variablen ppmOS“ an support@ppmgmbh.com.

Mit Hilfe der Variablenübersicht kann sich der Anwender unter Angabe des Namens beliebige CAN-Pakete aus dieser Variablenmenge zusammensetzen, also im DBC-File definieren. Die so definierten CAN-Pakete werden dann, nach Initialisierung, mit der eingestellten Zykluszeit gesendet.

Folgend finden Sie ein einfaches **Konfigurationsbeispiel zur CAN-DBC-Konfiguration** bzw. Initialisierung.

Die Liste der verfügbaren CAN-Variablen enthält sehr viele Variablen, die eine GPS-Board-Initialisierung voraussetzen. Deshalb muss die Initialisierung immer in folgender Reihenfolge ablaufen:

```
#Initialisierung des GPS-Boards
    gps init
#CAN-Modul initialisieren (hier mit Bitrate 1M bps)
    can init bps 1M
#Initialisierung des CAN-DBC-Moduls
    can dbc init
```

Das CAN-DBC-Modul bzw. der CAN-DBC-Treiber programmiert die erforderlichen GPS-Logs des GPS-Boards mit der gewünschten Zykluszeit (im DBC-File definiert) selbst. Dies geschieht bei der Ausführung von 'can dbc init' nachdem die Syntax geprüft und alle Werte eingelesen wurden.

Software - ppmOS

40xx CAN

Bei Nutzung aller verfügbaren CAN-Variablen werden die folgenden NovAtel-Logs am COM1 des GPS-Boards initialisiert, und somit an den Port gps1 übergeben, an welchem diese geparst werden:

bestposb	bestvelb	dualantennab	gpgga	gpgsv
gprmc	inspvasb	insstdevsb	rawimusxb	timeb

Eine Beschreibung dieser Logs finden Sie im NovAtel-Handbuch: docs.novatel.com/OEM7/Content/Home.htm

Zusätzlich können die für die CAN-Variablen benötigten Logs auch am gps2 geparst werden. Hierfür muss das Kommando 'gps debug set 0x80' (s. S. 155) an den Controller gesendet werden. Es wäre also möglich, nach DBC-Initialisierung, alle Logs am COM1 des GPS-Boards zu deaktivieren und am COM2 zu initialisieren (i.d.R. in der gps.cfg).

Am Ende der CAN-DBC-Initialisierung wird ausgegeben, auf welche GPS-Log-Initialisierungen das GPS-Board mit 'OK', und auf welche es mit 'ERROR' geantwortet hat:

```
DBC: update required   20 ms, set 'log inspvasb       ontime 5.00' error: Message is invalid for this model
DBC: update required   40 ms, set 'log bestvelb       ontime 0.02' error: Requested rate is invalid
DBC: update required   40 ms, set 'log bestvelb       ontime 0.05' ok
GPS: USER rotation 0 0 0 set
DBC: update required   40 ms, set 'asynchinslogging enable'        error: Invalid Message ID
DBC: update required   40 ms, set 'log rawimusxb onnew'            error: Message is invalid for this model
CAN: CAN dbc driver initialized, success!
```

Im oberen Beispiel wurde das CAN-DBC-Modul mit dem Schalter '-r' initialisiert, also 'can dbc init -r'. Wäre der Schalter weggelassen worden, so hätte die Initialisierung aufgrund der Fehlermeldungen „error“ abgebrochen. Dies ist zum Beispiel der Fall, wenn auf dem GPS-Board die für einen Log notwendige Option fehlt (hier: inspvasb und rawimusxb => „Message is invalid for this model“). Mit 'can dbc init -r' kann man den CAN-DBC-Treiber zum Ignorieren der Fehlermeldung(en) überreden (s. S. 92).

Neben den Variablen, die als Quelldatensätze GPS-Logs verwenden, sind auch Variablen integriert, welche die Funktion des ppmOS, RTCM3-Korrekturdatenpakete auf CRC-Richtigkeit zu prüfen, nutzen. Diese Funktion ermöglicht das Erkennen, Auflisten und Filtern von CRC-richtigen bzw- -falschen RTCM3-Korrekturdatenpaketen (s. hierzu Kommando 'cp' S. 106)

Hinweis zu den DBC-Variablen AXB, AYB, AZB, RXB, RYB, RZB - Nyquist-Shannon-Bedingung

Um eine Fehlinterpretation der CAN-Daten der Variablen AXB, AYB, AZB, RXB, RYB, RZB zu vermeiden, ist im ppmOS ein als Tiefpassfilter zu verstehender Filter implementiert. Bei Verwendung dieser Variablen dient der Filter dazu, der Bedingung des Nyquist-Shannon-Abtasttheorems gerecht zu werden. Es soll damit das erhaltene Signal abgerundet werden. Dies ist vor allem dann zu beachten, wenn die Ausgaberate der genannten Variablen geringer ist, als die des RAWIMUSXB Logs, welcher den Quelldatensatz für diese darstellt. Der Filter kann bei Bedarf zur Laufzeit ab- und angeschaltet werden (s. S. 155):

Filter deaktivieren: gps debug set 0x1000

Filter aktivieren: gps debug clr 0x1000

Definieren der Zykluszeiten bzw. Ausgaberaten der CAN-Pakete und Abarbeitung durch das ppmOS

Im Beispiel oben auf S. 73, wäre es ohne den Schalter '-r' auch bei der Fehlermeldung „*Requested rate is invalid*“ zum Abbruch der Initialisierung des CAN-DBC-Moduls gekommen.

Die Fehlermeldung besagt, dass die angeforderte Ausgaberate, hier alle 40ms / 25Hz / 0.04s nicht möglich ist, wobei das ppmOS dabei stets die dem angeforderten Wert (hier 40ms) nächstliegende höhere, vom NovAtel-GNSS-Board ausführbare Ausgaberate anfordert - hier also 0.02s, da 0.04s bei NovAtel nicht möglich sind. (Die bei NovAtel-GNSS-Boards gültigen Werte für das Logging mit hoher Rate sind 0.01, 0.02, 0.05, 0.1, 0.2, 0.25 und 0.5. Für das Logging langsamer als 1 Hz wird jeder ganzzahlige Wert akzeptiert.)

Das ppmOS registriert die Fehlermeldung und fragt dann schrittweise die möglichen Ausgaberaten an, bis das NovAtel-GNSS-Board mit 'OK' antwortet.

Die gewünschte Ausgaberate kann entweder im DBC-File in ms beim Kommentar zur jeweiligen Botschaft mit '-t' festgelegt werden. Hier 40ms:

CM_ BO_ 147 "-t40; GPS speed";

Oder alternativ (z.B. nachträglich und im Betrieb per Terminal-Software oder in einem Skriptfile) mit dem Kommando 'can dbc cycle [set <cycle_ms>] [<id> ...]' (s.S. 93).

Test der CAN-DBC-Funktionalität

Zum Testen der CAN-DBC-Funktionalität des 40xx, kann dieser „stand alone“ - also ohne weiteren (CAN-) Netzwerkteilnehmer - verwendet werden. Dies ist mit Hilfe der Loopback-Funktion möglich. Die Konfiguration herfür lautet:

```
gps init
can init bps 500k
can dbc init -r
tsk fout -f comc can
can loopback silent
can filt 0 on rx0 m32 0 0
```

Die Ausgabedaten (Hex-Werte) des can-Tasks erscheinen standardmäßig am coma. Diese werden hier auf comc umgeleitet. Hier könnte anstelle des COM C auch z. B. ipsc1 o. ä. stehen.

In diesem Beispiel erscheint an der COM C der hexadezimale Dump aller gesendeten CAN-Pakete.

Achtung! *Da hier ca. 28 kByte/s Bandbreite benötigt werden, sollte die Baudrate des COM C auf mindestens 230400 Bd eingestellt sein.*

Um sich ein CAN-Paket allein anzusehen, kann man mit 'can dbc susp' alles ausschalten und dann mit 'can dbc resume 96' genau ein Paket (hier das Paket mit ID 96) aktivieren.

Software - ppmOS

40xx CAN

Hinweis zur Speicherung von CAN-Streams auf SD-Karte

Beim Schreiben bzw. Speichern von Daten auf die interne SD-Karte, sind die nur 512 Byte großen SD-Filepuffer zu beachten, was bei entsprechenden Datenübertragungsraten eines CAN-Streams zum Datenverlust führen kann.

Es wird deshalb empfohlen den Empfang des CAN-Streams in eine Pipe auszulagern, und erst im Anschluss aus dieser in ein SD-File zu schreiben. Folgend ein Konfigurationsbeispiel hierzu:

```
gps init
can init bps 1M
can dbc init –r
mkdir c:/log/
tsk fout -f pipe1 can
cp -t -8 pipe1  c:/log/M4-g1-a-$(tm)&
can loopback silent
can filt 0 on rx0 m32 0 0
cron put flush 0 * * * * * * "@stat -f >null"
```

Hinweis:

Im obigen Beispiel wird mit 'tsk fout -f pipe1 can' alles aus dem 'can'-Task in die pipe1 geschrieben. Dies hat zur Folge, dass der CAN-Treiber die Daten ASCII und Hex gemischt ausgibt (was dem Default entspricht):

```
> CAN:   0 0   0          4     41853 8 24 47 50 47 47 41 2c 31   $GPGGA,1
> CAN:   0 0   0          4     41969 8 30 33 31 34 33 2e 30 30   03143.00
> CAN:   0 0   0          4     42085 8 2c 35 30 35 38 2e 32 30   ,5058.20
> CAN:   0 0   0          4     42201 8 30 35 37 31 35 2c 4e 2c   05715,N,
> CAN:   0 0   0          4     42316 8 30 31 33 35 36 2e 30 36   01356.06
> CAN:   0 0   0          4     42433 8 34 36 37 36 36 2c 45 2c   46766,E,
> CAN:   0 0   0          4     42548 8 39 2c 31 30 2c 30 2e 39   9,10,0.9
> CAN:   0 0   0          4     42664 8 2c 31 33 38 2e 38 32 38   ,138.828
> CAN:   0 0   0          4     42780 8 2c 4d 2c 34 33 2e 37 30   ,M,43.70
> CAN:   0 0   0          4     42894 8 2c 4d 2c 30 36 2c 30 31   ,M,06,01
> CAN:   0 0   0          4     43194 7 33 36 2a 35 43 0d 0a      36*5C..
```

Um eine Ausgabe wie

```
$GPGGA,103143.00,5058.2005715,N,01356.0646766,E,9,10,0.9,138.828,M,43.70,M,06,0136*5C
```

zu erhalten, müsste ein CAN-File via 'can open can1-...' geöffnet werden, aus welchem die Daten über eine pipe auf die SD-Karte geschrieben werden (s. hierzu auch S. 71 „Binäre Ausgabe“).

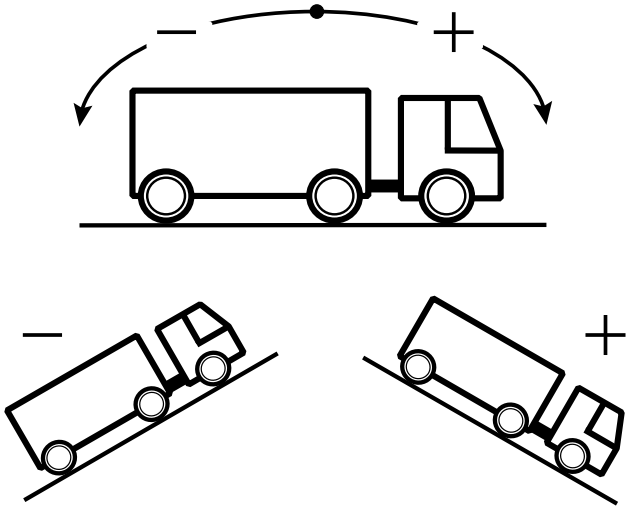
40xx CAN

Orientierungen und Winkelvorzeichen der Pitch-, Roll- und Azimuth-Werte der CAN-DBC-Variablen

Im ppmOS sind die Orientierungen und Winkelvorzeichen der Pitch-, Roll- und Azimuth-Werte der CAN-DBC-Variablen standardmäßig nach der **DIN 70000** definiert. Eine individuelle Anpassung ist möglich (s. S. 77).

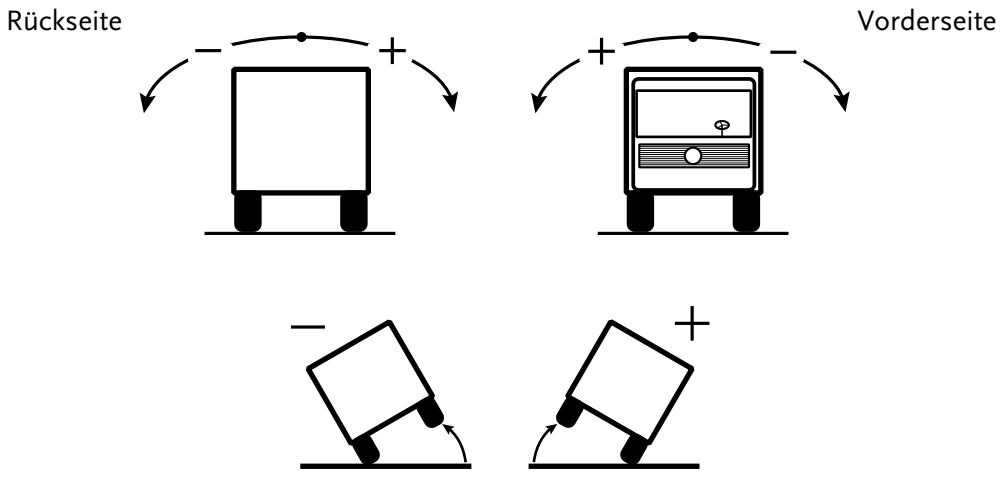
Dies bedeutet, dass sich zum Beispiel bei einer Bergauffahrt ein negativer Pitch-Winkel (Neigungswinkel) ergibt:

Bild 22: DIN 70000 Pitch



Ist ein Fahrzeug nach links gekippt, sodass die linke Seite tiefer liegt als die rechte, ergibt sich ein negativer Rollwinkel:

Bild 23: DIN 70000 Roll



Bei einer Kurvenfahrt ergibt sich bei einer Kurve nach links ein positiver, bei einer Kurve nach rechts ein negativer Gierwinkel (Yaw).

Software - ppmOS

40xx CAN

Hinweis:

Grundsätzlich müssen bei Verwendung einer IMU zusammen mit einem NovAtel GNSS-Empfänger, die Angaben zu Rotationsversätzen zwischen der Ausrichtung der IMU-Achsen und der des vom Anwender definierten Koordinatensystems sowie des von NovAtel definierten „Vehicle Frames“, direkt am NovAtel Empfänger vorgenommen werden, d.h. in der Datei gps.cfg des ppmOS eingetragen werden.

Für die CAN-DBC-Variablen des ppmOS werden die Orientierungen und Winkelvorzeichen der vom NovAtel-Empfänger ausgegebenen Pitch-, Roll– und Azimut-Werte standardmäßig nach **DIN 70000** umgerechnet und ausgegeben. Im Folgenden Kapitel finden Sie eine Beschreibung der Konfigurationsmöglichkeiten zur Anpassung der Orientierungen und Winkelvorzeichen in den CAN-BDC-Variablen des ppmOS.

Individuelle Anpassung der Orientierungen und Winkelvorzeichen

Eine **individuelle Anpassung** der Orientierungen und Winkelvorzeichen der Pitch-, Roll– und Azimut-Werte ist möglich, und kann auf zwei Konfigurationsebenen durchgeführt werden:

1. Im DBC-File

1.1. Richtungsvorzeichen-Änderung:

Hierfür kann am Koeffizienten das Vorzeichen geändert, bzw. das Vorzeichen des Faktors einer CAN-Variable angepasst werden.

Beispiel (Faktor ist fett markiert):

BO_ 106 Sens_RatesBody: 6 ppm

SG_ RZB : 32|16@1- (**0.0001745**,0) [-5|5] "rad/s" Vector__XXX

Wird zu

BO_ 106 Sens_RatesBody: 6 ppm

SG_ RZB : 32|16@1- (**-0.0001745**,0) [-5|5] "rad/s" Vector__XXX

1.2. Achsenzuordnung:

Zyklische Vertauschung der CAN-Variablen innerhalb eines Pakets.

2. In der Konfiguration des GPS-Boards (Eintrag in Datei gps.cfg)

Hierfür stehen im ppmOS die folgenden, vom NovAtel-Kommando SETINSROTATION abgeleiteten Kommandos zur Verfügung: SETINSROTATION IMU und SETINSROTATION VEH.

2.1. INS-Azimuth-Offset und positive/negative Drehrichtung

Die Angabe eines Offsets zu dem über die IMU berechneten Azimuth sowie die Anpassung der Drehrichtung erfolgt über das Kommando SETINSROTATION IMU.

- Syntax:

  'SETINSROTATION IMU <offset_x> <offset_y> <offset_z> [scale_x scale_y scale_z]'

  - Addiert den für <offset_z> eingegebenen Wert auf den Wert 'Azimuth' des NovAtel-Logs 'INSPVASb' welcher als Datenquelle für die CAN-Variable 'INS_ang_Yaw' verwendet wird
  - Die Rotation um den für <offset_z> eingegebenen Wert erfolgt bei positivem Wert gegen den Uhrzeigersinn, bei negativem Wert im Uhrzeigersinn. Es kann also entweder ein negativer oder der entsprechende positive Wert eingetragen werden, z.B: -90 oder (+)270.
  - Default für <offset_xyz> ist jeweils „0“
  - Es wird jeweils nur <offset_z> für die CAN-DBC-Konvertierung übernommen. D.h., dass für <offset_x> und <offset_y> eine 0 eingetragen werden kann bzw. muss, für scale_xyz sind nur „1“ oder „–1“ gültige Werte
  - Default für scale_xyz ist „1“. Mit Angabe von „-1“ für scale_z kann die Drehrichtung von „positiv gegen den Uhrzeigersinn“ (DIN 70000) nach positiv im Uhrzeigersinn geändert werden
  - Es müssen stets beide Kommandos (SETINSROTATION IMU und SETINSROTATION VEH) in die gps.cfg eingetragen werden
  - Beispiel (Groß-/Kleinschreibung und Position in der Datei gps.cfg sind beliebig):

    SETINSROTATION IMU 0 0 -90 1 1 -1

Software - ppmOS

40xx CAN

2.1. GNSS-Heading-Offset und positive/negative Drehrichtung

Das GNSS– bzw. Zwei-Antennen-Heading ist der Winkel zwischen True North und dem Vektor von erster zu zweiter Antenne. Können die Antennen nicht in Fahrzeuglängsachse montiert werden, so würden bei Nordausrichtung nicht 0° ausgegeben. Da dieser Offset **nicht** auf Konfigurationsebene des NovAtel-Empfängers kompensiert werden darf - da dadurch ein Fehler in den IMU-Algorithmus eingebracht werden würde -, bietet das ppmOS mit dem Kommando 'SETINSROTATION VEH x y z' diese Möglichkeit. Dies gilt allerdings ausschließlich für die entsprechende CAN-Variable 'GPS_DualAnt_Heading'.

Die Angabe eines Offsets zu dem über die beiden GNSS-Antennen berechneten Azimuth/Heading sowie die Anpassung der Drehrichtung erfolgt über das Kommando SETINSROTATION VEH.

- Syntax:

  'SETINSROTATION VEH <offset_x> <offset_y> <offset_z> [scale_x scale_y scale_z]'

  - Addiert den eingegebenen z-Wert auf den Wert 'heading' des NovAtel-Logs 'DUALANTENNAHEADINGb', welcher als Datenquelle für die CAN-Variable

    'GPS_DualAnt_Heading' verwendet wird.
  - Die Rotation um den für <offset_z> eingegebenen Wert erfolgt bei positivem Wert gegen den Uhrzeigersinn, bei negativem Wert im Uhrzeigersinn. Es kann also entweder ein negativer oder der entsprechende positive Wert eingetragen werden, z.B: -90 oder (+)270.
  - Default für <offset_xyz> ist jeweils „0“
  - Es wird jeweils nur <offset_z> für die CAN-DBC-Konvertierung übernommen. D.h., dass für <offset_x> und <offset_y> eine 0 eingetragen werden kann bzw. muss, für scale_xyz sind nur „1“ oder „–1“ gültige Werte
  - Default für scale_xyz ist „1“. Mit Angabe von „-1“ für scale_z kann die Drehrichtung von „positiv gegen den Uhrzeigersinn“ (DIN 70000) nach positiv im Uhrzeigersinn geändert werden
  - Es müssen stets beide Kommandos (SETINSROTATION IMU und SETINSROTATION VEH) in die gps.cfg eingetragen werden
  - Beispiel (Groß-/Kleinschreibung und Position in der Datei gps.cfg sind beliebig):

    SETINSROTATION VEH 0 0 -90 1 1 -1 (= SETINSROTATION VEH 0 0 270 1 1 -1)

Hinweis:

Das NovAtel-Kommando HEADINGOFFSET darf in der Konfiguration des GPS-Boards (Datei gps.cfg) bei Verwendung einer IMU nicht verwendet werden! Das Kommando addiert einen Offset auf die „heading“-Werte, welche das GPS-Board zur Orientierung der IMU verwendet. Zusammen mit den Hebelarmen zwischen Antenne(n) und IMU kommt es dann zu Fehlern.

ppmOS-Kommandosyntax

Beschreibung der ppmOS-Kommandosyntax

Das **ppmOS** ist ein kommandozeilenorientiertes Betriebssystem mit dateiorientierten und preemptiven Multitasking.

Der Anwender kann den 40xx-Sensor mit einem Nullmodemkabel (ohne RTS/CTS) mit einem PC, über Bluetooth mit einem Tablett oder über Ethernet mit einem PC verbinden. Die Bedienung kann mit einem beliebigen Terminalprogramm erfolgen. Das Terminal sollte in einem Zeilenmodus (kanonischer Mode) und nicht im Zeichenmodus betrieben werden. Das **ppmOS** generiert ein Kommandozeilenecho nach Empfang des abschließenden CR+LF bzw LF. D.h. das Schreiben und Editieren einer Kommandozeile sollte im Terminalprogramm erfolgen und erst nach Drücken der Entertaste generiert das **ppmOS** das Kommandozeilenecho.

Eine Reihe von Kommandos wurden in ihrer Syntax und ihrem Verhalten nach dem Betriebssystem Linux entlehnt. Linux-Anwender werden sich also mit der Syntax und der Anwendung der im Anhang erläuterten Kommandos schnell zurecht finden. In der Regel wird bei Eingabe eines Kommandos ohne Parameter oder mit dem Parameter **?** eine kurze Kommandozeilenhilfe ausgegeben.

Symbole zur Beschreibung der Kommandos und Regeln bei der Ausführung

Tabelle 17: Symbole zur Beschreibung der Kommandos

<Begriff>	bezeichnet einen Zahlenwert, Zeichen oder eine Textangabe, die durch den Begriff beschrieben wird. Die Klammern < und > werden als ‚dies ist zu ersetzten durch …' gelesen.
[ ]	optionale Angabe, alles was in der Klammer steht, kann angegeben oder weggelassen werden
{ }	eine Liste von auszuwählenden Angaben, eine Angabe aus der Liste muss ausgewählt werden
|	Trennzeichen in einer Liste von optionalen oder Pflichtangaben, wird als ‚oder' gelesen

ppmOS-Kommandosyntax

Sonderzeichen in der Kommandozeile

Tabelle 18: Sonderzeichen in der Kommandozeile

@ oder !	Das Zeichen kann am Anfang eines Kommandos stehen. In dem Fall wird das Echo des Kommandos unterdrückt. (Hinweis: das Kommandoecho kann global mit '**set echo off**' abgeschaltet und mit '**set echo on**' wieder angeschaltet werden.
@@ oder !!	Die Zeichen können am Anfang eines Kommandos stehen. Es wird das Kommandoecho und die erweitere Kommandosyntax unterdrückt.
;	Trennzeichen zwischen zwei Kommandos, d.h. auf einer Zeile können mehrere Kommandos geschrieben werden. Es ist eine unbedingte Aneinanderreihung von Kommandos.
||	Trennzeichen für eine bedingte Kommandoausführung. Es wird erst das links von || stehende Kommando ausgeführt. Hat es den Rückgabewert ungleich 0 (in der Regel ist dies ein Rückgabewert für eine fehlerhafte Ausführung), dann wird das rechts davon stehende Kommando ausgeführt.
&&	Trennzeichen für eine bedingte Kommandoausführung. Es wird erst das links von && stehende Kommando ausgeführt. Hat es den Rückgabewert gleich 0 (in der Regel ist es der Rückgabewert für eine fehlerfreie Ausführung), dann wird das rechts davon stehende Kommando ausgeführt.
"..." '...'	Zeichen und Angaben werden zu einer Angabe zusammengefasst (d.h. geklammert). Es können also Leerzeichen enthalten sein oder mehrere Kommandos zusammengefasst werden. Beispiel: **time "csm c:/file_a >c:/file_a.csm"** Im Beispiel wird eine Zeitmessung der Prüfsummenbildung der Datei c:/file_a durchgeführt. Die Ausgabe des Kommandos **csm** wird in die Datei c:/file_a.csm geschrieben. Ohne "..." werden alle Ausgabe, also auch die des time-Kommandos, in die Datei geschrieben.
>	Zeichen zur Ausgabeumleitung in eine Datei, deren Name dahinter angegeben werden muss. Falls die Datei bereits existiert, wird sie überschrieben. Der Dateiname **null** bedeutet Ausgabe ins Nichts, also keine Ausgabe.
>>	Zeichen zur Ausgabeumleitung in eine Datei, deren Name dahinter angegeben werden muss. Falls die Datei nicht existiert, wird sie angelegt. Und falls die Datei existiert wird die Ausgabe an das Ende angehangen.

ppmOS-Kommandosyntax

Tabelle 18 (f.): Sonderzeichen in der Kommandozeile

&	Das Zeichen kann am Ende einer Kommandozeile stehen. Dadurch wird das Kommando als Task gestartet. Das Kommando wird also unabhängig von anderen Kommandos ausgeführt. Der Task erhält als Namen den Inhalt der Kommandozeile - also das Kommando selbst. Des Weiteren wird dem Task automatisch eine Prozess-ID (PID) zugeordnet (s. Kommando '**tsk**' ab S. 301).

Alle anderen Zeichen oder Wörter müssen so geschrieben werden, wie sie in der Kommandosyntax angegeben sind. Es können beliebig Leerzeichen am Anfang, zwischen und hinter den Argumenten geschrieben werden. Zwischen den Argumenten ist mindestens je ein Leerzeichen oder Tab erforderlich.

Befehle im Detail

Befehlsverzeichnis — ppmOS-Befehle im Detail

Befehle im Detail

Befehlsverzeichnis — ppmOS-Befehle im Detail

Befehle im Detail

Befehlsverzeichnis — ppmOS-Befehle im Detail

Befehle im Detail

bd	Baudrate für serielle Schnittstellen einstellen

Mit diesem Kommando können die Baudraten der seriellen Schnittstellen (UART) eingestellt werden.
Das Gerät verfügt über 7 serielle Schnittstellen. Dem systematischen Namen ist jeweils ein Aliasname zugeordnet.

Name	Alias	Verwendung	Anschluss
`com1`	`blth`	Ansteuerung des Bluetooth-Moduls	nur intern
`com2`	`comc`	universell verwendbar als RS232, RS485 mit und ohne Modemsteuerleitungen RTS und CTS	SUB-D 25, Backpanel
`com3`	`coma`	Terminalschnittstelle zur Geräteadministration, bzw. universell verwendbar	8 pol., Frontpanel
`com4`	`comb`	universell verwendbar als RS232, verfügt über Modemsteuerleitungen RTS und CTS	SUB-D 25, Backpanel
`com5`		nicht verfügbar	
`com6`	`gps2`	interne Verbindung zum GPS-Board, hauptsächlich als Datenschnittstelle verwendet	intern
`com7`	`gsm`	Ansteuerung des GSM-Modems	intern
`com8`	`gps1`	interne Verbindung zum GPS-Board, hauptsächlich als Kommandoschnittstelle verwendet	intern

Die Änderung der Baudrate ist jederzeit möglich. Es muss aber bedacht werden, dass bei dem daran angeschlossenen Gerät genau dieselbe Baudrate eingestellt wird.

Die Defaultbaudraten für coma, comb und comc sind 115200 (einstellbar ‚bd coma -dflt ...').

Wird das Kommando ohne Parameter ausgeführt, dann erscheinen die aktuellen Einstellungen aller Schnittstellen:

```
com1 (blth)  actual:   923077 bd, at reset:        0 not set
com2 (comc)  actual:   913044 bd, at reset:   921600
com3 (coma)  actual:   913044 bd, at reset:   921600
com4 (comb)  actual:   230400 bd, at reset:        0 not set
com6 (gps2)  actual:   461538 bd, at reset:        0 not set
com7 (gsm )  actual:   913044 bd, at reset:        0 not set
com8 (gps1)  actual:   461539 bd, at reset:        0 not set
```

Die Angabe ‚`not set`' bedeutet, dass kein alternativer Initialwert eingegeben wurde.

Parameter:

`{<com_name>|<com_alias>} [[-dflt] [<baud>] | irq [on|off]]`

`<com_name>|<com_alias>`

Angabe der Schnittstelle (Namen siehe Tabelle oben).

Befehle im Detail

bd	Baudrate für serielle Schnittstellen einstellen
dflt Der optionale Parameter erlaubt es, die Baudrate schon ab Power on bzw. Geräte-Reset auf den angegebenen Wert einzustellen. Das hat z. B. Bedeutung bei der Terminalschnittstelle **coma**, bei der die Boot-Meldungen und Initialisierungsmeldungen schon mit der gewünschten Baudrate ausgegeben werden. Diese werden noch vor der Ausführung der ‚**autoexec.sh**', also die frühestmögliche skriptbasierte Kommandoausführung, ausgegeben.	
<baud> Der optionale Parameter gibt die gewünschte Baudrate an. Ohne diesen Parameter wird die aktuell eingestellte Baudrate ausgegeben. Es sind alle Standardbaudraten und auch beliebige Werte möglich. Das Kommando endet mit der Ausgabe der tatsächlich realisierten Baudrate. Die Schnittstelle **comb** kann maximal mit 230400 Bd arbeiten. Die maximale Baudrate der Schnittstellen **gps1** und **gps2** hängt vom jeweilig installierten GPS-Board ab.	
irq [on|off]] Mit dieser Optionalen Angabe kann ein UART-Empfangs-Interrupt gesperrt oder freigegeben werden.	

Befehle im Detail

call	Programm im Arbeits- oder Programmspeicher aufrufen

Das Kommando dient zur Umschalten der Programmausführung (Haupt- oder Bootprogramm) oder zum Aufrufen eines Unterprogramms. Standardmäßig sind zwei Programmversionen im Flash-ROM des Controllers gespeichert. Das dient der Systemsicherheit und der Möglichkeit während des laufenden Betriebs ein Programmupdate ausführen zu können. Die beiden unabhängigen Programmversionen werden mit ‚**boot**' und ‚**main**' bezeichnet. Standardmäßig sind beide binärkompatibel. Nach Power On startet zunächst ‚**boot**' als eine Art Trampolin, um nach erfolgreicher CRC-Prüfung von ‚**main**' dieses Programm auszuführen. Zur Laufzeit kann die jeweils andere Programmversion aufgerufen werden. Dabei kann ein Shell-Kommando übergeben werden. Das Synonym ‚**alternate**' steht für die jeweils andere Programmversion.

Das Kommando kann in 2 Varianten ausgeführt werden. In der Variante 1 ist das Ziel das Boot- oder Hauptprogramm, also der Eintrittspunkt der Firmware.

In der Variante 2 ist das Ziel ein beliebiges über eine Adresse angesprochenes Unterprogramm, das sich im Flash-ROM oder im RAM befinden kann. Es können Funktionsparameter als Integer- oder Gleitkommazahl oder als String übergeben werden.

Diese zweite Variante ist bei der Entwicklermodus von Spezialanwendungen nützlich.

Parameter:

```
  [-v] [-u] [-c] [-s] {boot|main|alternate} [<cmd>]
         -v be verbose
         -u unconditional call
         -c omit crc check
         -d call entry direct, without -d call via 'Reset' cmd
| [-v] [-di] <address> [<arg1> [<arg2> ... [<argv8>]]]
         -di disable interrupts before call
         <address> is a 32 bit number
         <argx> may be:
           decimal or hexadecimal 32 bit number
           floating point number with '.' converted to float
           floating point number with preceding (float) i.e. (float)1234
           floating point number with preceding (double) i.e (double)1234
           <string> enclosed in "..", '..', <..>, [..] or {..}
         -s <string> up to end of line
```

Variante 1:

```
[-v] [-u] [-c] [-d] {boot|main|alternate} [<cmd>]
```

-v

Bei der Ausführung des Kommandos gesprächig sein (verbose).

Befehle im Detail

call	Programm im Arbeits- oder Programmspeicher aufrufen

-u

Ausführung des Kommandos erfolgt ohne implizite Bedingung. D.h. wenn sich das Programm z.B. in ‚**main**' befindet und man würde ‚**call main**' aufrufen, dann würde ohne ‚**-u**' das Kommando mit der Fehlermeldung

```
2nd programm version is already running, no call done
```

abgebrochen werden. Mit **-u** wird es ‚unconditional' ausgeführt.

-c

Vor der Ausführung von **call main**, **call boot** oder **call alternate** wird der standardmässige CRC-Check des auszuführenden Programms weggelassen.

-d

Der Call geht direkt zum Eintrittspunkt. Ohne Schalter **-d** geht der Prozessor durch einen Hardware-Reset bevor der Eintrittspunkt angesprungen wird.

<cmd>

Ein beliebiges optionales Shell-Kommando, das nach dem Neustart ausgeführt wird. Es kann auch ein kombiniertes Kommando, also eine Folge von Kommandos die durch **;** **&&** oder **||** getrennt sind und in **"..."** eingeschlossen sind, sein.

Variante 2:

```
[-v] [-di] <address>} [<arg1> [<arg2> ... [<argv8>]]]
```

-v

Bei der Ausführung des Kommandos gesprächig sein (verbose).

-di

Vor dem Aufruf des Unterprogramms den globalen Interrupt sperren.

<address>

Adresse des Unterprogramms

Befehle im Detail

call	Programm im Arbeits- oder Programmspeicher aufrufen

```
[<arg1> [<arg2> ... [<argv8>]]]
```

Die optionalen Argumente des Unterprogramms. Diese können sein:
- Integerzahlen, als Hex- oder Dezimalzahl geschrieben,
- Gleitkommazahlen , mit Dezimalpunkt ‚.', mit vorangestelltem (float) oder (double) oder
- Textstrings, eingeschlossen in **"..", ‚..', <..>, [..]** oder **{..}**
- bei vorangestelltem Schalter **-s** ein Textstring bis zum Ende der Eingabezeile

Befehle im Detail

can	CAN-Funktionen steuern

Dieses Kommando dient der Steuerung der CAN-Funktionen.

Parameter:

```
      init bps {2.25M|1.8M|...} [txbuf <size>] [rx0buf <size>] [rx1buf <size>]
      bps  [2.25M|1.8M|1.5M|1M|600k|500k|250k|125k|100k|50k|25k|20k|10k|9k|<val>]
      dbc  init [-r] [<dbc_file>]         default is 'c:/sys/ppm40xx-can.dbc', -r relaxed
      dbc  source-VXH-VYH [inspva|bestvel] sets the source for INS_VXH and INS_VYH
      dbc  hdg-VXH-VYH [ins|antenna]       sets the heading source for computation INS_VXH and INS_VYH
      dbc  deinit
      dbc  {susp|resume|tgl}      [<id> ...]             without <id> all ID's are affected
      dbc  cycle [set <cycle_ms>] [<id> ...]             <cycle_ms> becomes rounded to nearest 10 ms
      dbc  stat  [-f <what>]      [<id> ...]             print status information
      dbc  debug [-f <what>]      [<id> ...]             set/display debug mode infos
      dbc  debug fout [-f <file>] [<id> ...]             set/display outfile in debug mode
      dbc  prt   [-f <what>]      [<id> ...]
      dbc  ?
      filt                                               shows initialized filters
      filt       [<ix> [<ix2> ...]] {on|off}             switches filters on or off
      filt        <ix> [on|off] rx{0|1} 132 [e][r] <id1> [e][r] <id2>
      filt        <ix> [on|off] rx{0|1} m32 [e][r] <id>  [e][r] <msk>
      filt        <ix> [on|off] rx{0|1} l16     [r] <id1>    [r] <id2>     [r] <id3>     [r] <id4>
      filt        <ix> [on|off] rx{0|1} m16     [r] <id1>    [r] <msk1>    [r] <id2>     [r] <msk2>
      filt        <ix> [<ix2> ...] rst                   reset filters
      filt ls     <ix> [<ix2> ...] [all] [on|off] [rx0|rx1] [m|l] [32|16]
      open        can1-<id>[-[e][r][b][-<buf_size>]]     using as generic file (read)
      close       can1-<id>                              using default paket handler
      tx-tmot     can1-<id> [pak <pak_cnt>|<tmot_µs>]    to switch off feature -> enter 0
      retransmit  [on|off]
      time_trig   [on|off]
      loopback    [off|silent|norx|notx]
      stat        [tx] [rx] [pak]
file name for writing: can1-<id>[-[e][r][b][p[<tmot_pak>]|t[<tmot_µs>]]]
```

`can`

Der Aufruf ohne Parameter ergibt die Kommandozeilenhilfe (s. o. „Parameter“).

`can init bps {2.25M|1.8M|...}  [txbuf <size>] [rx0buf <size>] [rx1buf <size>]`

CAN-Modul initialisieren, z.B. **`can init bps 100k.`**

Der CAN-Treiber arbeitet zunächst ohne Retransmit.
Der Wert hinter **`bps`** ist die Bitrate (Standard: 10k - 1M). Es können auch andere Werte angegeben werden (`<val>`).

`txbuf <size>`

Mit diesem Parameter wird die Größe des Sendepuffers zwischen <16 – 40000> Byte eingestellt. Der Default-Wert ist 800 Byte. Es sollte beim Einstellen des CAN– Sendepuffers auf den Gesamtarbeitsspeicher des Gerätes geachtet werden. Dieser beträgt 256kb. Freier Arbeitsspeicher kann mit dem Kommando **`stat`** abgefragt werden (Beispiel siehe nächste Seite.).

Befehle im Detail

can	CAN-Funktionen steuern

can init bps… f.:

rx0buf <size>
rx1buf <size>

Einstellung der Puffergrößen für die beiden Empfangspipes zwischen <16 – 25000> Byte . Der Default-Wert ist 800 Byte. Es sollte beim Einstellen des CAN- Empfangspuffers auf den Gesamtarbeitsspeicher des Gerätes geachtet werden. Dieser beträgt 256kb. Freier Arbeitsspeicher kann mit dem Kommando **stat** abgefragt werden:

```
>stat
heap: 0x2000e000 free 142336 min 1960, txt 0x817157c, dat 2536, bss 26940, ctor 60
coma: com3 rxlvl 1 (1999) ovfl rx 0 tx 0 lines 18 0 esc_lines 0 0 unknown 0  intp_ovfl 0 0
RR: POR (0x07) counts [LPW 0, WDG 0, IWD 182 | SWR 539, POR 1013, EXR 1733, BOR 1013] RST 523, JMP 0, WDT 4, CSH 18, CSB 0, PON 3139
open files: 0 max 12
```

can bps [2.25M|1.8M|1.5M|1M|600k|500k|250k|125k|100k|50k|25k|20k|10k|9k|<val>]

Bitrate der CAN-Schnittstelle einstellen. Der Wert hinter **bps** ist die Bitrate (Std: 10k - 1M).
Es können auch "krumme" Werte angegeben werden (**<val>**).
Bei Bitratenänderung bleibt der eingestellte Loopback-Mode erhalten.

can dbc init [-r] [<dbc_file>]

Initialisierung des CAN-DBC-Moduls (vorher 'can init bps ...' ausführen).

Das Default-File ist ppm40xx-can.dbc, es wird in $(syspath) gesucht, also standardmäßig in c:/sys. Durch Angabe von [<dbc_file>] kann ein anderes File verwendet werden.

Wir der Schalter -r (für „relaxed“) angegeben, dann bricht die Initialisierung NICHT ab wenn erforderliche gps-Datensatzinitialisierungen fehlschlagen (was i.d.R. fehlende Optionen des GPS-Boards als Ursache hat).

dbc source-VXH-VYH [inspva|bestvel] sets the source for INS_VXH and INS_VYH

Mit diesem Kommando kann zwischen den NovAtel-Logs INSPVA und BESTVEL als Datenquelle für die CAN-DBC-Variablen INS_VXH und INS_VYH gewählt werden. Default ist 'bestvel'. Wird BESTVEL verwendet, kann in der Konfiguration des NovAtel GNSS-Receivers (also in der gps.cfg) eingestellt werden, ob der Receiver die Geschwindigkeit über dessen Positionierungsfilter oder den Doppler berechnen soll. Bitte sehen Sie hierzu: https://docs.novatel.com/OEM7/Content/Commands/BESTVELTYPE.htm?Highlight=bestvel

Bei INSPVA werden die von der IMU berechneten Werte verwendet.

dbc hdg-VXH-VYH [ins|antenna]

Per Default wird für die Berechnung der CAN-Variablen INS_VXH und INS_VYH die Variable INS_ang_Yaw (bzw. NovAtel INSPVAS) verwendet. Sollen anstelle der IMU-Werte die der Antenne verwendet werden, genauer GPS_DualAnt_Heading (bzw. NovAtel DUALANTENNAHEADING), so kann dies mit ‘dbc hdg-VXH-VYH antenna‘ eingestellt werden.

can dbc deinit

Das Kommando dient zum Deinitialisieren des CAN-DBC-Moduls, wenn man z.B. mit einem anderen DBC-File fortsetzen möchte.

Befehle im Detail

can	CAN-Funktionen steuern

`can dbc {susp|resume|tgl} [<id> ...]`

`susp`
die Ausgabe des CAN-Pakets mit der bzw. den angegebenen CAN-paket-ID wird gestoppt. Ohne Angabe einer ID wird alles gestoppt.

`resume`
eine gestoppte Ausgabe wird fortgesetzt (auch einzelne IDs ansprechbar).

`tgl`
der Zustand resume/susp wird umgeschaltet.

Ohne Angabe von <id> werden alle CAN-Paket beeinflusst.

`can dbc cycle [set <cycle_ms>] [<id> ...] <cycle_ms> becomes rounded to nearst 10 ms`

Anzeige bzw. Änderung der Sendezyklen in ms der CAN-Pakete, wobei optional mit <id> der Sendezyklus eines bestimmten CAN-Pakets angezeigt bzw. geändert werden kann.

`can dbc stat [-f <what>] [<id> ...]`

Ausgabe von Informationen wie z. B. Anzahl der Variablenupdates. Fehlt z. B. ein GPS-Datensatz aus dem bestimmte CAN -Variablen aktualisiert werden, dann fehlen diese in der Liste. Der Parameter -f hat hier z. Zt. keine Bedeutung.

`can dbc debug [-f <what>] [<id> ...] set/display debug mode infos`

Aktivierung/Deaktivierung von Ausgaben des Updatewerts für die CAN-Variablen eines CAN-Paketes, in dem Moment, wenn diese ein Update bekommt (zeitsynchron).

-f 1 Ausgabe anschalten

-f 0 Ausgabe abschalten

Achtung, es können sehr viele Ausgaben erscheinen wenn <id> nicht angegeben wird: Ausgabe für alle CAN-Pakete!

`dbc debug fout [-f <file>] [<id> ...]`

Mit dem Kommando 'can dbc debug ...' kann man sich das Update der CAN-Variablen ausgeben lassen. Diese ASCII-Ausgabe kann nun für jede einzelne Botschaft individuell in ein angegebenes File geschrieben werden.

Die Kommandoerweiterung hat folgende Syntax:

can dbc debug fout [-f <filename>] [<Botschafts_ID> ...]

Als File bieten sich coma, comb, comc sowie bevorzugt die Ethernet-Files ipsa1, ipsa2, ipsb1, ... an.

Bei den seriellen Schnittstellen ist darauf zu achten, dass manche Logs mit hoher Ausgaberate die Bandbreite der seriellen Schnittstellen überschreiten können, weshalb besser Ethernet-Files verwendet werden sollten.

Befehle im Detail

can	CAN-Funktionen steuern

```
can dbc  prt   [-f <what>]      [<id> ...]
```

Ausgabe der geparsten Werte, nachdem sie in das CAN-DBC-Fileformat zurückgewandelt wurden. Es dient zur Kontrolle, ob die Werte korrekt eingelesen wurden.

```
can dbc ?
```

Ausgabe der Kommandozeilenhilfe

```
can filt                                              shows initialized filters
can filt       [<ix> [<ix2> ...]] {on|off}            switches filters on or off
can filt        <ix> [on|off] rx{0|1} l32 [e][r] <id1> [e][r] <id2>
can filt        <ix> [on|off] rx{0|1} m32 [e][r] <id>  [e][r] <msk>
can filt        <ix> [on|off] rx{0|1} l16     [r] <id1>    [r] <id2>     [r] <id3>    [r] <id4>
can filt        <ix> [on|off] rx{0|1} m16     [r] <id1>    [r] <msk1>    [r] <id2>    [r] <msk2>
can filt        <ix> [<ix2> ...] rst                   reset filters
can filt ls     <ix> [<ix2> ...] [all] [on|off] [rx0|rx1] [m|l] [32|16]
```

Zum Empfang von CAN-Paketen muss ein dazugehöriges Empfangsfilter eingestellt werden. Dieses CAN-Modul ‚hört' danach nur auf die eingestellten CAN-Pakete.
Mit diesem Unterkommando können:

- Filter eingestellt und einem, zwei Datenpaketen oder Paketgruppen zugewiesen werden
- eine Liste aller Filter angezeigt werden
- einzelne Filter angesprochen werden

Insgesamt sind 28 Filter möglich (Nr. 0-27). Zum Beispiel könnten mit langer ID 56 verschiedene CAN-Pakete gefiltert empfangen werden.

Beispiel "Liste/Verzeichnis aller Filter anzeigen":

```
>can filt
CAN: filt  0    off rx0.0   1 m32 id              0 0x00000000  msk              0 0x00000000
…
CAN: filt 13    off rx0.13  1 m32 id              0 0x00000000  msk              0 0x00000000
CAN: filt 14    off rx0.14  1 m32 id              0 0x00000000  msk              0 0x00000000
CAN: filt 15 on     rx0.15  2 l32 id1            33 0x00000021  id2           2047 0x000007ff
CAN: filt 16 on     rx0.17  2 l32 id1           123 0x0000007b  id2            456 0x000001c8
CAN: filt 17    off rx0.19  1 m32 id              0 0x00000000  msk              0 0x00000000
CAN: filt 18    off rx0.20  1 m32 id              0 0x00000000  msk              0 0x00000000
…
CAN: filt 26    off rx0.28  1 m32 id              0 0x00000000  msk              0 0x00000000
CAN: filt 27    off rx0.29  1 m32 id              0 0x00000000  msk              0 0x00000000
```

Siehe f.

Befehle im Detail

can	CAN-Funktionen steuern

"**can filt**" f.

Unterparameter
<ix>[<ix2> …]

Filterindex einzeln ansprechen bzw. Liste **<ix>[<ix2> …]**. Es können also ein oder mehrere Filter in einem Kommando angesprochen werden.
Die Filternummer kann als Dezimal- oder als Hexadezimalzahl angegeben werden.
z.B.: 15 **oder 0xf**

{on|off} bzw. **[on|off]**
Filter aktivieren bzw. deaktivieren

rx{0|1}
Dem Filter einen der beiden Empfangskanäle zuordnen: **rx0** oder **rx1**.

l32 [e][r] <id1> [e][r] <id2>

l32	Listenmodus (32 Bit). Es folgen 2 IDs. Kennzeichnet diesen Filter als 'mit exakter Übereinstimmung'
e	es handelt sich um eine "Extended ID" die aus 29 Bits besteht – ansonsten 11 Bit
r	Requests sollen abgefangen werden
<id1>	ID des Datenpakets, das empfangen werden soll, dezimal oder hex.
<id2>	ID eines zweiten Datenpakets, dezimal oder hex.

m32 [e][r] <id> [e][r] <msk>

m32	Maskenmodus (32 Bit) zum Auswählen mehrerer IDs einer Gruppe (Gruppen-ID im Maskenmodus).
e	es handelt sich um eine "Extended ID" die aus 29 Bits besteht – ansonsten 11 Bit
r	Requests sollen abgefangen werden
<id>	Gruppen-ID (32 Bit) zum Auswählen mehrerer IDs einer Gruppe. Angabe dezimal oder hexadezimal.
<msk>	Bitmaske Jede ankommende Paket-ID wird BIT-weise mit der Maske **<msk>** UND-verknüpft und wenn das Ergebnis gleich der Gruppen-ID **<id>** ist, ergibt es einen Filtermatch und das Paket wird empfangen. Angabe dezimal oder hexadezimal.

Einem Filter (**l32**) müssen zwei IDs zugewiesen werden. D. h. es müssen 2 ID's angegeben werden. Falls eine zweite nicht benötigt wird, dann eine nicht benutzte angeben.
Mit l16 bzw. m16 wird alles auf 16Bit gestellt.

Befehle im Detail

can	CAN-Funktionen steuern

```
can filt [ls [on|off| <ix>[ <ix2> ...]] [on|off] [rx0|rx1] [m|l] [32|16]]]
```

Mit diesem Unterkommando werden Filtereinstellungen ausgegeben oder es werden Filtereinstellungen verändert.
Zum Empfang von CAN-Paketen muss ein dazugehöriges Empfangsfilter eingestellt werden. Das CAN-Modul ‚hört' danach nur auf die eingestellten CAN-Pakete.

Beispiele:

```
can filt
```

Die gesamte Filtertabelle wird ausgegeben.

```
can filt ls on
```

Alle eingeschalteten Filter werden ausgegeben.

```
can filt ls l 32
```

Alle Filter, die im Listenmode (exakt Match), mit einer 32 Bit-ID, werden ausgegeben.

```
open        can1-<id>[-[e][r][b][-<buf_size>]]    using as generic file (read)
file name for writing: can1-<id>[-[e][r][b][p[<tmot_pak>]|t[<tmot_µs>]]]
```

Wird ein CAN-File zum Lesen geöffnet, so muss es mit ‚can open' angelegt werden.
Wird ein CAN-File zum Schreiben geöffnet, so muss es nicht vorher per ‚can open' geöffnet werden, da dies durch das Kommando zum Auslesen eines Files in das CAN-File passiert. Beispiel:

```
gps reply gps1 all can1-100-ebp
```

Hinweis:
Die Begriffe „Lesen" und „Schreiben" beziehen sich auf File-Operationen innerhalb des ppmOS.
Sollen z.B. Daten von einem externen, an den 40xx angeschlossenen Teilnehmer empfangen werden, so muss vorher ein CAN-File geöffnet werden, damit z.B. ein Kopier-Task die Daten aus diesem File lesen und z.B. auf SD-Karte schreiben kann.
Öffnen eines CAN-Files.

`e`	wenn angegeben, dann ist <id> eine extended ID, also eine 29-Bit_ID. Ohne Angabe ist es eine Standard-ID mit 11 Bit Länge.
`r`	kennzeichnet die CAN-Pakete in diesem File als Remote-Pakete (rezessiv). Mit Remote-Paketen kann ein Teilnehmer den Empfänger auffordern seine Daten zu senden.
`b`	aktiviert den Blocking-Mode für eine Tx-File (wartet immer bis alles gesendet ist). Ziel ist, keine Daten beim Versenden zu verlieren. Blocking-Mode heißt, dass, falls der Sendepuffer voll wird, weiteres Absenden von Daten den Absender blockiert. Beim Blocking-Mode wartet der Absender immer bis er alle Daten loswird. Der Programmfluss des Absendertask wird solange blockiert, bis wieder Platz im Sendepuffer ist. Die Dauer bis eine Datenmenge zum gepufferten Senden angenommen wird, hängt von der (Erzeuger-)Datenrate, der Übertragungsgeschwindigkeit und beim CAN-Protokoll auch von der Einstellung 'can retransmit {<on>|<off>}' in Bezug zu Buskollisionen und/oder eventuellen Leitungsstörungen ab. Nonblocking heißt dagegen, dass der Absendertask nicht blockiert wird, es gehen aber Daten verloren wenn der Sendepuffer voll wird. Neue Daten überschreiben dann alte Daten. (f. nächste Seite)

Befehle im Detail

can	CAN-Funktionen steuern

„can open ...“ (f.)

b (f.)	Ist die Datenübermittlung immer schneller als die Erzeugung und Erzeugerdatenbursts immer kleiner als der Sendepuffer, dann kann 'b' entfallen. Ist das nicht der Fall und es dürfen keine Daten verloren gehen, dann muss 'b' angewendet werden. !! Hierbei muss beachtet werden ob der Datensender warten kann !! Der GPS-Task des 40xx kann nicht warten. Das GPS-Board sendet konstant mit seiner Ausgaberate. Muss der GPS-Task beim Weitersenden warten, dann läuft er nach 4 s in seinen Timeouthandler und initialisiert sich neu.
-<bufsize>	Stellt die Puffergröße für ein CAN-Empfangsfile ein. Ohne Angabe wird die Standardgröße (**ppmOS v 1.46**) von 400 Byte eingestellt. Dieser Empfangspuffer existiert je geöffnetes CAN-Empfangsfile. Darüber hinaus existieren im CAN-Treiber separate Puffer für das Senden und je einer für die Empfangspipe 0 und 1 für alle CAN-Pakete. Die Puffergrößen für den CAN-Treiber in Empfangs und Senderichtung werden mit dem Kommando ‚**can init ...**‘ eingestellt.
P[<tmot_pak>]	Es werden immer 8 Byte gesammelt, bevor sie gesendet werden. Es werden also volle CAN-Pakete gesendet. Sind nach dem angegebenen Timeout keine 8 Byte da, dann wird der aktuelle Rest gesendet (1...7). Ohne <tmot_pak> werden 3 Paketzeiten gewartet.
t[<tmot_µs>]	Es werden immer 8 Byte-Pakete gesendet mit angegebenen Timeout (in µs). Ohne <tmot_µs> werden 432 µs (bei 1 MBit/s) gewartet, ansonsten die angegebene Zeit bis ein eventueller Rest (also 1...7 Byte) gesendet wird.

Die Schalter 'p' oder 't' werden verwendet, wenn ein CAN-File zum Schreiben geöffnet wird, und sollten nur verwendet werden, wenn:

- die Datenquelle die Daten nur zeichenweise zur Verfügung stellt

- eine hohe Datenrate (Bandbreitenausnutzung) zu erwarten ist. (s. hierzu Kommando 'gps reply' „GPS-Zeichenweitergabe vom GPS-Task“ - s. S. 157)

Beispiele für GPS-Datenübertragung:

Zeichenweise Datenübergabe

```
gps reply gps1 all can1-100-p
gps reply gps2 asc can1-101-p
gps reply gps2 bin can1-102-p
```

Datensatzweise Datenübergabe (Timeoutsteuerung nicht erforderlich):

```
gps reply gps1 asc can1-100
gps reply gps1 bin can1-101
gps reply gps2 all can1-102
gps reply gps1 ppmpos can1-103 (zu ‚ppmpos‘ s.S. 158 Kommando ‘gps reply [gps1|gps2] ppmpos‘ und S. 320)
gps reply gps2 ppmpos can1-104
```

Nach dem Öffnen eines CAN-Files kann dieses in anderen Shell-Kommandos verwendet werden.

(Beispiele s. nächste Seite)

Befehle im Detail

can	CAN-Funktionen steuern

Beispiele:

Daten an externen Teilnehmer senden:

(hier: 29-Bit-ID, Blocking-Mode, mit Standard-Timeout)

```
can init bps 500k
gps reply gps1 all can1-100-ebp
```

Daten von externem Teilnehmer empfangen (vorher muss ein Filter definiert werden; hier: auf SD-Karte speichern deshalb Auslagerung in eine Pipe—s. hierzu S.75)

```
can init bps 500k
can filt 0 on rx0 l32 320 321
can open can1-320
cp -t -8 -s can1-320 pipe1&
cp -t -8 -s pipe1 c:/data/canlog.txt&
```

Wenn man ohne CAN-Netzwerk, also ohne einen Teilnehmer testen möchte, dann kann folgendes Skript hilfreich sein:

```
can init bps 1M

#jede Sekunde die Uhrzeit in ein CAN-File mit Paket-ID 200 schreiben
loop -8 -t 1000 @date >can1-200&

#ein Empfangsfilter für die Paket-ID 200 setzen
can filt 0 on rx0 l32 200 201

#ein Empfangsfile für die Paket-ID 200 öffnen
can open can1-200

#das Empfangene ausgeben
cat -t -8 can1-200&

#den Ausgang mit dem Eingang brücken, also das Gesendete selbst empfangen
can loopback silent
```

```
close       can1-<id>                              using default paket handler
```

Geöffnetes CAN-File schließen.

```
tx-tmot     can1-<id> [pak <pak_cnt>|<tmot_µs>]    to switch off feature -> enter 0
```

Mit diesem Kommando kann ein Sende-Timeout für ein CAN-File angegeben werden. Der Timeout kann entweder über eine Anzahl an Paketen (pak <pak_cnt>) oder über eine Zeitangabe in Mikrosekunden (<tmot_µs>) definiert werden.

```
can retransmit [on|off]
```

Das CAN-Protokoll sieht eine Paketwiederholung vor falls ein gesendetes Paket nicht bestätigt (acknowledged) wird, also jeder Teilenehmer auf dem Bus das Paket CRC-richtig empfangen hat. Wenn das Retransmit eingeschaltet ist, wiederholt das CAN-Modul das Paket ohne Mitwirkung von Software. Das Retransmit muss überlegt eingesetzt werden, da ein fehlerhafter CAN-Teilnehmer am CAN-Bus wegen der automatischen Paketwiederholung der anderen Teilnehmer der gesamte Datenverkehr stark begrenzt oder sogar vollständig blockiert werden kann. In der Standardeinstellung ist das Retransmit abgeschaltet. Es kann jederzeit an- oder abgeschaltet werden.

Befehle im Detail

can	CAN-Funktionen steuern

```
can time_trig  [on|off]
```

Die Hardware des CAN-Moduls des Microcontrollers ist in der Lage, beim Empfang von CAN-Paketen diesen einen Zeitstempel zuzuordnen. In der Standardeinstellung ist der Zeitstempel aktiviert. Das Erzeugen des Zeitstempels geschieht auf Hardware-Ebene und bedeutet keine Softwarebelastung.

```
can loopback [off|silent|norx|notx]
```

Dieses Kommando dient dazu, die Funktion des Datenverkehrs zu überprüfen.

`off`	Normalfunktion des CAN-Moduls, kein Loopback. (Default)
`silent`	Das CAN-Modul ist vollständig vom Bus getrennt und Tx und Rx sind verbunden. Das ist ein vollständig isolierter Testmodus, der zum Erlernen und Testen der CAN-Funktionen geeignet ist.
`norx`	Das CAN-Modul kann Daten auf dem CAN-Bus ausgeben, es kann aber keine Pakete empfangen. Tx und Rx sind miteinander verbunden.
`notx`	Das CAN-Modul kann keine Daten auf dem CAN-Bus ausgeben, es kann aber Pakete empfangen. Tx und Rx sind miteinander verbunden. Dieser Modus ist geeignet passiv am Bus mitzuhören.

```
can stat        [tx] [rx] [pak]
```

Es wird der Zustand des CAN-Treibers ausgegeben.
Beispiel:

CAN: this 0x200233e8, bit rate 1000000 bps, t_q 111 ns (9000000 Hz), T_16bit 7281778 ns (137 Hz)

```
CAN: txbuf       0x200235d0,  1600 byte (  800 payload), IRQ cnt      4094, ovfl 0
CAN: rx0buf      0x20023568,  1600 byte (  800 payload), IRQ cnt      2720, ovfl 0 (0)
CAN: rx1buf      0x2002359c,  1600 byte (  800 payload), IRQ cnt         0, ovfl 0 (0)
CAN: retransmit  off
CAN: time_trig   on
CAN: loopback    silent
CAN: IRQ cnt sce 8          status change errors
CAN: IRQ src err 8           wuk 0          slk 0          bof 0          passive 0          warn 0
CAN: lerr: stuff 0          form 0          ack 8          rec 0          domin.  0          crc  0
CAN: reactivations after stall   0
```

Es sind folgende Informationen (nach Beispiel von ‚can open …') ersichtlich:

Bitrate	1 Mbit/s
Sendepuffer	kann 800 Byte Nutzdaten puffern, 4094 Pakete gesendet, kein Überlauf Die Puffergröße kann in ‚**can init …**' eingestellt werden.
Empfangspuffer 0 und 1	kann 800 Byte Nutzdaten puffern, 2720 Pakete empfangen, kein Überlauf Die Puffergröße kann getrennt für rx0 und rx1 in ‚**can init …**' eingestellt werden.
Retransmit	ist aus
Loopback	ist an
Interrupt- und Fehlerinfos	keine Fehler

Befehle im Detail

cat	Dateiausgabe

Mit diesem Kommando kann eine Datei oder die Daten aus einer Schnittstelle angezeigt werden. Die Ausgabe erfolgt an der jeweiligen Terminalschnittstelle oder mit Hilfe der Ausgabeumleitung ‚>' in eine andere Datei oder Schnittstelle. Das Kommando arbeitet in seiner Grundfunktion wie das gleichnamige GNU/Linux-Kommando.

Parameter:

```
[-n] [-exec] [-fgets] [-o <offs>] [-l <len>] [-t {-8|<tmot_ms>} [-s]] <file>
                                  -n     put line number as prefix (implies -fgets)
                                  -exec  check any line beginning with key word 'exec'
                                         and execute rest of line as shell command
                                         lines beginning with whitespaces an '#' treat as ignored comments
                                  -fgets read infile synchronized on LF boundaries (-exec implies -fgets)
                                  -o     seek infile to offset <offs>
                                  -l     output max <len> byte
                                  -t     wait timeout on end of file
                                         -8         forever
                                         <tmout_ms> milli seconds
                                  -s     don't put message in case of timeout
| -imr                 [-o <offs>] [-l <len>] [-t {-8|<tmot_ms>} [-s]] <file>
| -litef <imu_type> <msg_nr> {imr [-rt] |asc-scl|euler-asc|asc|hex} [-o <offs>] [-l <len>] [-t {-8|<tmot_ms>} [-s]] <file>
                           <imu_type>   one of (LCI100C, LCI100N), only LCI100C is implemented
                           <msg_nr>     message that is expected, implemented is 129
                           imr          convert data to a binary IMR file (Novatel), -rt -> read RTC on rx every record
                           asc-scl      convert data to ASCII to scaled increment values (rad, m/s)
                           euler-asc    convert data to Euler angles roll, pitch and yaw (deg)
                           asc          convert data to ASCII
                           hex          convert data to hexadecimal ASCII
| -m <src> <len>
```

[-n]

Dieser optionale Parameter wird sinnvoller Weise bei Textdateien angegeben. Damit wird vor jeder Zeile eine Zeilennummer ausgegeben.

[-exec]

Dieser optionale Parameter wird bei Textdateien angegeben. Beginnt eine Zeile mit dem Schlüsselwort **exec**, dann wird das dahinter angegebene Kommando dem Kommandointerpreter übergeben und von ihm ausgeführt. Davon wird in Konfigurationsdateien Gebrauch gemacht. Diese werden beim einlesen ebenfalls auf das Schlüsselwort ‚**exec**' abgesucht.

[-fgets]

Dieser optionale Parameter wird bei Textdateien angegeben. Die Datei wird zeilenweise gelesen und zeilenweise ausgegeben. D. h. es wird immer bis zu einem CR, LF, CR+LF oder LF+CR gelesen. Ohne diesen Parameter wird die Datei in Blöcken zu 512 Byte gelesen und ausgegeben.

Befehle im Detail

cat	Dateiausgabe

`[-o <offs>]`

Mit diesem optionalen Parameter wird angegeben, dass die Ausgabe ab dem Dateioffset **<offs>** beginnt. Es werden also **<offs>** Byte überlesen. Die Zahlenangabe kann dezimal oder hexadezimal mit vorangestelltem 0x erfolgen. Ist der Schalter **-n** oder **-fgets** gesetzt, dann ist die Angabe ein Zeilenoffset. Die Zeilenzählung beginnt bei 1 im Gegensatz zur Byteoffsetzählung, die bei 0 beginnt.

Wenn der Wert für **<offs>** negativ ist, dann wird ab dem Offset, der sich aus Filelänge plus **<offs>** ergibt, ausgegeben. Ist der sich so ergebende Fileoffset kleiner als 0, dann wird das File vom Anfang an ausgegeben.

`[-l <len>]`

Mit diesem optionalen Parameter wird angegeben, dass **<len>** Byte ausgegeben werden sollen. Die Zahlenangabe kann dezimal oder hexadezimal mit vorangestelltem 0x erfolgen. Ist der Schalter **-n** oder **-fgets** gesetzt, dann ist die Angabe ein Zeilenanzahl.

`[-t {-8|<tmot_ms>} [-s]]`

Die Ausgabe von **<file>** endet am Dateiende. Mit diesem Schalter kann eine Zeit eingestellt werden, die das Kommando nach dem Einlesen des letzten Zeichens wartet, bis ein Dateiende angenommen wird. Der Schalter hat besondere Bedeutung bei Schnittstellen. Bei dieser Sonderform einer Datei kommen die Zeichen asynchron und oft mit Pausen zwischen den Zeichen bzw. Zeilen an. Die Angabe erfolgt in Millisekunden. Die Angabe **-8** bedeutet eine unendliche Wartezeit.

Beispiel:

```
cat -t -8 gps2 >coma&
```

Es wird ein Task gestartet, der sämtliche über **gps2** einkommende Daten an **coma** wieder ausgibt.

Wird der Schalter „**-s**“ (silent) angegeben, so wird bei einem Timeout keine Message dazu ausgegeben.

`<file>`

Name des auszugebenden Files oder Schnittstelle.

Beispiele für ein File:

```
c:/daten/logfile.xyz, coma, comb, comc, blth, gps1, gps2, icom1, icom2, …, ipsa1, ipsa*, can, …, null
```

Befehle im Detail

cat	Dateiausgabe

```
-imr [-o <offs>] [-l <len>] [-t {-8|<tmot_ms>} [-s]] <file>
```

```
-litef <imu_type> <msg_nr> {imr [-rt] |asc-scl|euler-asc|asc|hex} [-o <offs>] [-l <len>] [-t {-8|<tmot_ms>} [-s]] <file>
```

Im ppmOS ist ein IMR-Datenfilter implementiert, der mit den Unterkommandos '**-imr**' und '**-litef**' genutzt werden kann.

Bei dem IMR-Dateiformat handelt es sich um ein proprietäres Dateiformat von NovAtel, welches in der Waypoint Post Processing Software „Inertial Explorer" zur Integration von IMU-Daten verwendet wird.

'**-imr**' dient der Konvertierung eines binären IMR-Files in ein ASCII IMR-File.

'**-litef**' dient der Konvertierung der LITEF Message „129" in das IMR-Datenformat bzw. in eine binäre IMR-Datei. Aktuell (ppmOS v1.46) ist dies die einzig unterstützte Message. Ebenso wird derzeit nur die LITEF LCI100C IMU unterstützt.

Da die Implementierung des IMR-Datenfilters auf dem Kommando ‚cat' aufsetzt, sind alle im ppmOS möglichen Filetypen (s.S. 54) für die Ein– und Ausgabe möglich. Hierbei können die umgewandelten Daten - also entweder die Msg 129 oder ein IMR-File - mit einer Ausgabeumleitung („>") an ein File im Sinne des ppmmOS (also auch in eine Datei) gesendet werden (siehe Beispiele).

`<imu_type>`
Angabe der verwendeten IMU (aktuell (ppmOS v1.46) wird nur die LITEF LCI100C unterstützt)

`<msg_nr>`
Nummer der erwarteten Message (aktuell (ppmOS v1.46 nur Msg 129)

`imr`
Daten in das binäre IMR-File-Format konvertieren

`-rt`
Bei jeder Datenaufzeichnung die Echtzeituhr auslesen

`asc-scl`
Daten nach ASCII als skalierte Inkrementalwerte umwandeln (rad, m/s)

`euler-asc`
Die Eulerwinkel Roll, Pitch und Yaw (deg) als ASCII-Daten ausgeben

`asc`
Daten nach ASCII konvertieren

`hex`
Daten in hexadezimales ASCII konvertieren

Beispiele siehe nächste Seite.

Befehle im Detail

cat	Dateiausgabe

```
„-imr ...
 -litef ...“ ff.
```

Beispiel 1:

```
cat -imr c:/data/imr-file.imr >c:/data/ascii-imr-file.log
```

Beispiel 2:

```
cat -litef LCI100C 129 imr -t -8 comc >c:/data/imr-file.imr &
```

Das ‚&‘ bedeutet, dass dieses Kommando als Task ausgeführt wird. „-t -8“ wird dabei zweckmäßig angewendet, um einen Daten-Stream durchgehend zu lesen.

Beispiel 3 - LCI100C verbunden mit COM C des 40xx und Ausgabe der Eulerwinkel über Ethernet (ipsa1):

```
set intp comc off
set comc rs485
bd comc 460800
cat -litef LCI100C 129 euler-asc -t -8 comc >ipsa1 &
```

```
-m <src> <len>
```

Einen Speicherbereich ausgeben. Das ist im Zusammenhang mit einer Ausgabeumleitung sinnvoll.

Beispiel:

```
cat -m 0x20000000 2000 >comb
```

oder

```
cat -m 0x20000000 2000 >c:/tmp/dump-speicher.bin
```

Befehle im Detail

cmds	Anzeige der verfügbaren Kommandozeilenbefehle

Hiermit werden alle verfügbaren Kommandozeilenbefehle alphabetisch sortiert oder unsortiert aufgelistet. In der unsortierten Liste wird für jedes Kommando die Anzahl der seit Power on bzw. Reset erfolgten Aufrufe ausgegeben. Die Liste kann in einer variablen Breite, also Anzahl der Kommandos pro Ausgabezeile, ausgegeben werden. Am Ende der Liste wird die maximal mögliche Anzahl der Kommandos ausgegeben. D.h. ist sind nicht immer alle Kommandos aufrufbar (gelinkt).

Parameter:

`[-u] [<n_colums>] | -l`

`-u`

Mit dem optionalen Schalter wird die Liste unsortiert ausgegeben. Die Ausgabe erfolgt in der Reihenfolge, in der die einzelnen Kommandos beim Programmstart gelinkt wurden. Anderenfalls wird die Liste alphabetisch sortiert.

`<n_colums>`

Mit der optionalen Zahlenangabe kann die Anzahl der Kommandos, die in einer Zeile ausgegeben werden sollen, angegeben werden. Der Default-Wert ist 3 und der Maximalwert ist 8. Die Zahl vor dem Kommandonamen ist die Anzahl der Kommandoaufrufe seit Power on bzw. Reset.

Befehle im Detail

cmds	Anzeige der verfügbaren Kommandozeilenbefehle

```
-l
```

In dieser Aufrufvariante wird eine Tabelle mit weiteren Informationen zu jedem Kommando ausgegeben. Beispiel:

```
 0: ver      0x200074c0  3 0x8169594 0x811e579         0x0 0x81622e4        1
 1: cmds     0x200074d8  4 0x815e48c 0x811ed0d 0x20003468 0x81622e4       11
 2: find     0x200074f0  4 0x816286c 0x812137d         0x0 0x81622e4        0
 3: ulink    0x20007508  5 0x8162874 0x811f46d 0x20003468 0x81622e4        0
 4: echo     0x20007520  4 0x815e6e4 0x811ea6d         0x0 0x81622e4        3
 5: pe       0x20007538  2 0x816287c 0x8121761         0x0 0x8162880        0
 6: po       0x20007550  2 0x8162884 0x8120e19         0x0 0x8162880        0
 7: time     0x20007568  4 0x8162888 0x811f9e9 0x20003468 0x81622e4        0
.
.
```

Darin bedeuten:
1. Spalte
Kommandoindex

2. Spalte
Kommandoname

3. Spalte
Adresse der Kommandostruktur, also die Adresse an der diese Daten stehen

4. Spalte
Länge des Kommandonamens

5. Spalte
Adresse des Kommandonamens

6. Spalte
Adresse des Kommandohandlers

7. Spalte
Falls es sich bei dem Kommandohandler (6. Spalte) um eine Elementfunktion einer Klasse handelt, dann ist in der 7. Spalte die Adresse des Objekts angegeben.

8. Spalte
Adresse von alternativen Kommandoendezeichen (Delimiter - Begrenzungszeichen). In der Regel folgt in einer Kommandozeile hinter dem Kommandonamen ein Leerzeichen (0x20 - SPC), ein Tabulatorzeichen (0x9 - TAB), das Wagenrücklaufzeichen (0xd – CR), das Zeilenvorschubzeichen (0xa – LF) oder das Nullzeichen (0x0 – EOS). Bei einigen Kommandos können zusätzliche andere Zeichen den Kommandonamen begrenzen. Das ist der Fall bei den Kommandos ‚**pe**' und ‚**po**'. Diese beiden Kommandos werden durch einen der Buchstaben ‚**bsw**' begrenzt. D. h. es gibt die Kommandos ‚**peb**', ‚**pes**', ‚**pew**', ‚**pob**', ‚**pos**' und ‚**pow**'. Sie bezeichnen den Datentyp, auf den das Kommando angewendet wird (8 bit - byte, 16 bit - short oder 32 bit – word).

9. Spalte
Anzahl der Kommandoaufrufe

Befehle im Detail

cp	Daten kopieren

Der Befehl dient zum Kopieren von Dateien oder Datenströmen in andere Dateien oder Datenströme.
Dateien im **ppmOS** sind:

Reguläre Dateien auf der internen SD-Karte im Gerät (FAT-Dateisystem)
z. B. **c:/data/2017/01/18/log-170118-155233-938354.9F6**

Die seriellen Schnittstellen
com1, com2, com3, com4, com6, com7,
coma, comc, comb, gps1, gps2, gsm, stdin, stdout

Ethernetstreams
z. B. **icom1, ipsa2, ipsb* oder icom3_mein-datenserver.de:20000**

CAN-Files
z. B. **can1-0x1f8**

Der Bluetooth-Datenlink
blth

Daten-Pipes
pipe1, pipe2, pipe3,...,pipe20

Speicherbereiche
z. B. **mem-0x20003468-0x200-r**

Spezialdateien
null, zero, dmy, mem

Parameter:

```
[-v]                        be verbose
[-rtcm3 [-no-filter] [-c]]
[-t {-8|<tmot_ms>} [-s]]
[-bs <size>] <infile> [-a] <outfile> [[-a] <rtcm_err_file>]
          -bs <size>        writes data in chunks of <size> bytes
          -a                append on <file>
          -rtcm3            RTCM V3 transport layer check
          -no-filter        copy unfiltered and check for statistics
          -c                print copyright information
```

-v

Beim Kopiervorgang gesprächig sein (verbose): Anzeige der kopierten Datenbytes ‚**copied 1000000 byte, o.k.**'.

Befehle im Detail

cp	Daten kopieren

```
cp [-rtcm3 [-no-filter] [-c]]
```

Mit diesen Schaltern kann geprüft werden ob gültige RTCM3-Datensätze von einem file (coma, comb, gps1, gps2,...) an ein anderes übertragen werden. Zudem können damit aus Datenströmen gültige RTCM3-Messages herausgefiltert werden, sodass nur diese weitergeleitet werden.

Bsp. 1: Weiterleitung mit Ausgaben ('-v'), Prüfung und Filterung ('-rtcm3'):

```
cp -v -rtcm3 -t -8 comc gps2&
```

Bsp. 2: Weiterleitung mit Prüfung und Filterung ('-rtcm3'), ohne Ausgaben (kein '-v'):

```
cp -rtcm3 -t -8 comc gps2&
```

Ein Kopiervorgang zur Datenweiterleitung sollte stets als Task ('&') zusammen mit '–t –8' gestartet werden, damit die Datenübertragung durchgehend aufrecht erhalten bleibt (s. dazu Beschreibung zu ‚cp [-t {-8|<tmot_ms>} [-s]]' S. 108).

Sobald der Kopier-Task mit 'tsk term {<index> | <name>} [...| <name>]' beendet wird, wird bei gesetztem Schalter '-rtcm3' eine Auswertung hinsichtlich der RTCM3-Prüfung ausgegeben:

#Starten des Kopiertasks mit Prüfung und Filterung:

```
cp -v -rtcm3 -t -8 comc gps2&
```

#Beenden des Kopiertasks:

```
tsk term "-rtcm3"
```

#Auswertung:

```
tsk    19 (0x2001ad10) ev 0x10000001 term bit set
cp terminated
-----------------------------------------------------------------
file length:     22130 byte checked
errors/gaps:      7760 byte outside of crc ok packets
errors/gaps:     35.07 % of file data outside packets
errors/gaps:        86 gaps with erroneous or unknown data
crc ok     :     14370 byte of packets with crc ok
crc ok     :     64.93 % of file data byte with crc ok
crc ok     :        86 packets with crc ok      (d3: 210      4)
=================================================================
flen      1500, read     22130, written     14370, diff -7760
```

-no-filter

Wird der Schalter ‚-no-filter' zusätzlich angegeben, so erfolgt eine Prüfung der Daten aber keine Filterung. D.h. die Daten werden überprüft und unabhängig vom Prüfergebnis werden alle Daten kopiert. Am Ende wird die Statistik bzw. Auswertung ausgegeben.

'-c'

Mit dem Schalter **'-c'** können Copyright-Information der verwendeten Quelle www.rtklib.com ausgegeben werden.

Befehle im Detail

cp	Daten kopieren

```
[-t {-8|<tmot_ms>} [-s]]
```

Der Kopiervorgang endet erst nach einem angegebenen Timeout.

Üblicher Weise dauert ein Kopiervorgang solange an, wie Daten aus dem **<infile>** gelesen werden. D. h. wenn bei einem Leseversuch aus **<infile>** keine Daten gelesen werden, dann werden beide Files geschlossen und der Kopiervorgang ist beendet.

Bei bestimmten Filetypen (com1, ..., icom1, ..., can1-..., usw.) kommen Daten aperiodisch oder mit Pausen zwischen den Daten. Um den Kopiervorgang aufrecht zu erhalten, wird ein Timeout gesetzt. Der Wert für **<tmot_ms>** wird in Millisekunden angegeben oder es wird der Spezialwert **-8** angegeben, der eine unendliche Wartezeit bedeutet.

Mit dem optionalen Schalter **-s** (silent) wird beim Auftreten eines Timeouts die Mitteilung ‚**timeout while waiting for data**' unterdrückt.

Beispiel:

```
cp -t -8 -s comc c:/tmp/comc_log.txt&
```

Der Schalter ‚**-t -8**' wird zweckmäßiger Weise angewendet, wenn der Kopiervorgang als Task gestartet wird (Zeichen ‚**&**' am Ende der Kommandozeile).

```
[-bs <size>] <infile> [-a] <outfile> [[-a] <rtcm_err_file>]
```

Mit diesem Kommando wird angegeben von welchem File (File im Sinne des ppmOS; z.B. COM B, ipsa1, pipe1, ...) an welches File die Daten kopiert werden.

```
-bs
```

Mit diesem Schalter werden die Daten als Blöcke in der über <size> (in Byte) angegebenen Größe geschrieben.

```
-a
```

Mit diesem Schalter werden die an das **<outfile>** kopierten Daten am Ende von **<outfile>** angehängt (append).

Das Selbe gilt für **<rtcm_err_file>**, in welches der Prüfbericht bzw. die Auswertung aus der RTCM3-Prüfung geschrieben wird. Für **<rtcm_err_file>** kann eine beliebige Datei (z.B. c:/DATA/RTCM3-Check.log) angelegt werden.

Befehle im Detail

cron	zeitgesteuerte Aktionen programmieren und verwalten

Mit dem Cron-Manager ist das **ppmOS** in der Lage, zeitgesteuert vom Anwender programmierte Aktionen zu starten.

Die Funktionalität entspricht der GNU/Linux crontab. Eine der Erweiterungen gegenüber der GNU/Linux-Implementierung ist die Möglichkeit, bis auf Sekundenebene Zeiten einstellen zu können.

Eine zeitgesteuerte Aktion wird Cron-Job genannt. Die Cron-Jobs werden zur Laufzeit in einer RAM-residenten Tabelle gespeichert (GNU/Linux überwacht die Datei /etc/crontab). Die Tabelle speichert max. 12 Einträge. Ein Cron-Job hat eine Zeiteinstellung und ihm ist eine Aktion zugeordnet.

Aktionen sind Kommandos des ppmOS. Eine Aktion kann ein Einzelkommando, eine bedingte Kommandoausführung oder eine mit ‚;' verkette Kommandofolge sein. Ein Kommando ist ein interner Befehl des ppmOS oder eine Skriptdatei (auch Shell-Skript oder Batch-Datei genannt).

Die Aktion kann sich wiederholen oder einmalig sein. Die Zeitbasis für die Ablaufsteuerung ist die Echtzeit, also der reale Kalender. Der Cron Manager ist auf den Sekundennulldurchgang der Echtzeit synchronisiert. In der Regel wird die vom GNSS gelieferte UTC oder GPS-Zeit zugrunde gelegt. Es ist auch eine reine sekundenbasierte relative Zeitsteuerung möglich.

Parameter:

```
  ls
| init                        looks for $(syspath)crontab.cfg and puts it
| put [-s] {@|!}<filen>       -s suspended, i. e. only stored
| put [-s] [ix <ix>] [[-m] <name>] <time_elements> [-i|-t] <cmd>
                                -s suspended, i. e. only stored
                                -m multiple jobs with the same name are possible
                                optional suffix for seconds:
                                [+<offs>][/<step>[*<repeats>]][.|s]   . remove, s suspend
                                -i exec immediate
                                -t as task else message to stdin
| replace {<ix> | <name>} [-i|-t] <cmd>
| exec    {<ix> | <name>} [{<ix> | <name>} ...]
| susp    {<ix> | <name> | all}
| resume  {<ix> | <name> | all}
| rm      {<ix> | <name> | all}
| mv      {<ix_src> | <name_src>}  <ix_dest>
| cp      {<ix_src> | <name_src>}  <ix_dest>
| swap    {<ix1     | <name1>   } {<ix2> | <name2>}
| echo    [on|off]
| stat
```

Befehle im Detail

cron	zeitgesteuerte Aktionen programmieren und verwalten

ls

Ausgabe der Cron-Job-Liste.

Beispiel:

```
>cron ls
CRON: job[ 0] sms-ls      : s *+30/7 * * * * * * @sms -ls
CRON: job[ 1] fixchk      :   * * * * * * * -i @gps stat >null && @set led yellow tgl >null
CRON: job[ 2] fixoff      :   * * * * * * * -i @gps stat >null || @set led yellow off >null
```

Jeder Cron-Job hat einen Index in der Tabelle (Angabe in **[...]**). Ein Cron-Job kann mit einem 12 Zeichen langen Namen benannt werden (hier im Beispiel **sms-ls**, **fixchk** und **fixoff**). Die Zuteilung eines Namens wird empfohlen. Sie macht die Programmierung von Cron-Jobs deutlich einfacher.

Hinter dem Doppelpunkt erscheint ein ‚s', wenn der Job suspendiert wurde. D. h. ein suspendierter Cron-Job bleibt in der Cron-Tabelle gespeichert, er hat aber keine Wirkung.

Es folgen die Angaben zu Zeitsteuerung. Die Zeitsteuerung umfasst 7 Felder. Wird ein Feld mit ‚*****' markiert, dann bedeutet das ‚trifft für alle Zahlen zu'.

Die Felder sind Sekunden, Minuten, Stunden, Tag, Monat, Wochentag und das Jahr.

Danach folgt die Ausführungsart der Aktion. Ohne Angabe wird das angegebene Kommando über eine interne Pipe an die coma-Shell gesendet. Diese Shell führt das/die Kommandos ‚in order', also in der Reihenfolge einschließlich Wartezeit, bis das vorherige Kommando sich beendet hat, aus.

Die Angabe ‚**-i**' bedeutet der Cronmanager (Cron-Task ‚**cron**') führt das Kommando unmittelbar (immediate) aus. Ein Kommando auf diese Weise zu starten hat nur Sinn, wenn das Kommando schnell (also innerhalb weniger ms) ausgeführt werden kann.

Die Angabe ‚**-t**' bedeutet der Cronmanager startet einen Task, der das Kommando ausführt. Das Kommando wird also unabhängig von anderen Kommandos ausgeführt.

cron init

Der Cronmanger (auch Crontask) löscht die aktuelle Cronjobliste, sucht die Datei ‚**c:/sys/crontab.cfg**', liest daraus die Einträge und speichert sie in der RAM-residenten Tabelle – dem internen ‚Terminkalender'.

cron put [-s] {@|!}<filen>

Der Cronmanager lädt eine Cron-Tabelle aus der angegebenen Datei. Einträge mit schon vorhanden Jobnamen oder Jobindex werden ersetzt. Alle anderen Einträge bleiben erhalten. Der optionale Schalter ‚**-s**' bewirkt ein ‚suspended put', also nur Speichern in der crontab.

Befehle im Detail

cron	zeitgesteuerte Aktionen programmieren und verwalten

```
cron put [-s] [ix <ix>] [[-m] <name>] <time_elements> [-i|-t] <cmd>
```

Einen Cron-Job speichern. Der optionale Schalter ‚**-s**' bewirkt ein ‚suspended put', also nur Speichern in der crontab. Der Cron-Job kann optional an einem bestimmten Platz in der Tabelle ‚**ix <ix>**', wobei **<ix>** von 0 bis 11 zählt, gespeichert werden. Ein eventuell vorhandener Eintrag wird überschrieben. Ohne diesen Parameter wird der erste freie Speicherplatz belegt. Cron-Jobs, die gleiche Ausführungszeitpunkte haben, werden in der Tabellenreihenfolge, also von Index 0 bis Tabellenende, ausgeführt.

Eine Cron-Job-Definition kann max. 200 Zeichen lang sein. Dabei wird vom Anfang der **<time_elements>** bis zum Ende von **<cmd>** gezählt.

Mit **[-m] <name>** wird der Cron-Job benannt. Der Name kann max. 12 Zeichen lang sein. Der Name darf nicht mit einer Ziffer beginnen und er darf keine Leerzeichen oder Sonderzeichen (& " ; usw.) enthalten. Gleichnamige Cron-Jobs werden ersetzt. Mit dem Zusatz ‚**-m**' (multiple) können mehrere gleichnamige Cron-Jobs gespeichert werden. Das ist dann sinnvoll, wenn man eine Gruppe von Cron-Jobs anlegen will, die dann mit den anderen Unterkommandos (**replace, exec, susp, resume**, ...) zusammen in einem Befehl angesprochen werden sollen.

<time_elements> sind immer 7 Teilangaben. Sie beziehen sich in der Reihenfolge auf:

Sekunde, Minute, Stunde, Tag, Monat, Jahr und Wochentag.

Die Teilangaben bilden einen oder mehrere Zeitpunkte. Ausgehend von diesem primären Zeitpunkt bzw. Zeitpunkten kann eine sekundäre Ablaufsteuerung auf Basis von relativen Sekundenangaben gemacht werden. Die Angaben können einzelne Zahlen, Aufzählungen von Zahlen getrennt durch Komma oder Zahlenbereiche, die aus einer Anfangszahl gefolgt von einem ‚-' und einer Endzahl bestehen, sein. Die Angabenvarianten können kombiniert werden. Eine Angabe darf kein Leerzeichen enthalten.

Beispiele für Zahlenangaben:

```
13
7-8
30-59,1,4
59-30
```

Siehe f.

Befehle im Detail

cron	zeitgesteuerte Aktionen programmieren und verwalten

"`cron put [-s] ...`" f.:

Die Teilangaben:

Element	Bereich	Hinweis
Sekunde	0 bis 59 oder *	**‚58-3'** ist gleichbedeutend mit ‚**58,59,0,1,2,3'** und genau das Gegenteil von ‚**4-57**'
Minute	0 bis 59 oder *	
Stunde	0 bis 23 oder *	
Tag	1 bis 31 oder *	Es gibt keine Prüfung, ob der betreffende Monat bzw. Jahr, falls angegeben, den angegebenen Tag hat.
Monat	1 bis 12 oder *	
Jahr	0 bis 99 oder *	Es wird der Bereich von 2000 bis 2099 überstrichen.
Wochentag	0 bis 6 oder *	Die Zählung beginnt sonntags mit 0

Die denkbar einfachste Angabe ist ‚*** * * * * * ***'. Sie bedeutet Zeitpunktmatch jede Sekunde.

Befehle im Detail

cron	zeitgesteuerte Aktionen programmieren und verwalten

```
optional suffix for seconds:
[+<offs>][/<step>[*<repeats>]][.|s]
```

Hinter der Sekundenangabe kann eine sekundäre Wiederholzählung programmiert werden.
Diese wird ausgeführt, wenn die primäre Zeitangabe mit dem aktuellen Zeitpunkt übereinstimmt. Ab dem Moment zählt nur noch die sekundäre Zeitangabe. Die sekundäre Zeitangabe ist also eine Art Unterprogramm, dass bei Erreichen des primären Zeitpunkts ausgeführt wird.

Darin bedeuten **+<offs>** ein optionaler Zeitversatz in Sekunden, ab dem die Ablaufsteuerung beginnt.

Die optionale Angabe **/<step>** gibt einen Zeitabstand der Ausführung in Sekunden an.

Die optionale Angabe ***<repeats>** gibt einen Zähler für die Ausführung des Kommandos **<cmd>** an.

Die sekundäre Zeitangabe (Zeitunterprogramm) kann mit einem Dezimalpunkt ‚.', mit einem ‚**s**' oder ohne spezielles Zeichen abgeschlossen werden.

Wird mit dem Dezimalpunkt abgeschlossen, dann wird der Cron-Job beim Ablauf des Zeitunterprogramms gelöscht. Mit der Angabe von ‚**s**' wird der Cron-Job lediglich suspendiert. Er kann also zu einem anderen Zeitpunkt mit ‚**cron resume ...**' wieder aktiviert werden. Ohne spezielles Endezeichen geht die Zeitkontrolle wieder auf die primäre Zeitangabe über, nachdem das Kommando <cmd> <repeats>-mal ausgeführt wurde.

Beispiel:

```
cron put test1 *+10. * * * * * * c:/sys/start-something
```

Zehn Sekunden nach dem der Cron-Job aufgesetzt wurde, wird einmalig das Skript **start-something** ausgeführt.

```
cron put test1 0/4*3 0 * * * * * -t gps bestpos
```

Ein Cron-Job mit dem Namen ‚**test1**' wird gespeichert, der immer zur vollen Stunde im Zeitabstand von 4 Sekunden dreimal das Kommando ‚**gps bestpos**' als separaten Task ausgeführt.

```
cron replace {<ix> | <name>} [-i|-t] <cmd>
```

Das Kommando eines vorhandenen Cron-Jobs wird durch das neue Kommando **<cmd>** ersetzt. Dabei kann die Ausführungsart immediate ‚**-i**', als Task ‚**-t**' oder pipelined nach stdin gewählt werden. Die zuvor gespeichert Ausführungsart wir in jedem Fall überschrieben.

```
cron exec    {<ix> | <name>} [{<ix> | <name>} ...]
```

Ein oder mehrere Cron-Jobs werden explizit, ohne Test des Ausführungszeitpunkts, also ohne Bedingung sofort in der gespeicherten Ausführungsart ausgeführt. Das Suspend-Flag wird ebenfalls ignoriert.

Befehle im Detail

cron	zeitgesteuerte Aktionen programmieren und verwalten

cron susp {<ix> | <name> | all}

Ein Cron-Job wird suspendiert, die Zeitpunkttests werden ausgesetzt. Die Angabe ‚**all**' bedeutet, dass alle Cron-Jobs suspendiert werden.

cron resume {<ix> | <name> | all}

Ein suspendierter Cron-Job wird wieder aktiviert. Der Cron-Task testet ab dem nächsten Sekundentakt wieder die Ausführung. Die Angabe ‚**all**' bedeutet, dass alle suspendierten Cron-Jobs aktiviert werden.

cron rm {<ix> | <name> | all}

Ein Cron-Job wird aus der Cron-Tabelle entfernt. Die Angabe ‚**all**' bedeutet, dass alle Cron-Jobs gelöscht werden.

cron mv {<ix_src> | <name_src>} <ix_dest>

Mit diesem Unterkommando kann einem bestehenden Cron Job ein neuer Platz in der Cron-Job-Tabelle zugeordnet werden.

Besteht bereits ein Cron Job an der Stelle die dem zu bewegenden Cron Job zugeordnet werden soll, so wird dieser mit dem zu bewegenden überschrieben und ist somit gelöscht.

<ix_src>
Aktueller Platz in der Tabelle des zu bewegenden Cron Job

<name_src>
Name des Cron Jobs

<ix_dest>
Neuer Platz in der Tabelle für den zu bewegenden Cron Jobs

cron cp {<ix_src> | <name_src>} <ix_dest>

Mit diesem Kommando kann ein bestehender Cron Job an eine anzugebende Stelle in der Cron-Job-Tabelle kopiert werden.

Wird die Kopie an eine Stelle in der Tabelle gesetzt („<ix_dest>"), an der bereits ein Cron Job besteht, so wird dieser überschrieben und somit gelöscht.

cron swap {<ix1 | <name1> } {<ix2> | <name2>}

Mit diesem Kommando können die Stellen zweier Crob Jobs in der Cron Job Tabelle getauscht werden.

Befehle im Detail

cron	zeitgesteuerte Aktionen programmieren und verwalten

```
cron echo    [on|off]
```

Die Arbeitsweise des Cron-Tasks kann durch Eingabe von ‚**cron echo on**' verfolgt werden. Ist für einen Cron-Job ein Zeitpunktmatch erreicht, dann gibt der Cronmanager eine Informationszeile aus:

```
CRON: executing job[3] led-toggle      '@set led green tgl'
```

Mit ‚**cron echo off**' wird die Kontrollausgabe abgeschaltet und mit ‚**cron echo**' die aktuelle Einstellung angezeigt.

```
cron stat
```

Der Status des Cron-Task wird angezeigt.

Beispiel:

```
CRON: entries  4 (max. 12), executed  11612, check cycle errors 0, echo is off
```

Darin bedeuten 4 der 12 Einträge sind belegt. Es gab insgesamt 11612 Zeitmatchpunkte, d. h. es wurden 11612 Kommandos ausgeführt.

Befehle im Detail

csm	Prüfsummen und CRC berechnen

Mit diesem Kommando kann von der Firmware, von Speicherbereichen oder von Dateien verschiedene Arten von Prüfsummen berechnet werden. Es können vier verschiedene Prüfwerte berechnet werden. Das Kommando dient auch zur Prüfung der Firmware. Der Rückgabewert kann für eine bedingte Ausführung von Skriptkommandos genutzt werden. Es gibt verschiedene CRC-Algorithmen bzw. Hashalgorithmen. Über ein und denselben Daten sind die ermittelten CRC folglich unterschiedlich. Ohne weitere Parameter ermittelt das Kommando 'csm' eine einfache 32 Bit Prüfsumme (Addition aller 32 Bit Werte modulo 32 Bit) und die CRC identisch zur Linuxversion 'cksum'. Mit den zur Verfügung stehenden Schaltern kann man einzelne CRC-Berechnungen anschalten (+...) oder abschalten (-...). Die Schalter müssen in der unter „Parameter“ angegebenen Reihenfolge angegeben/weggelassen werden.

Parameter:

```
[+crc32] [+crc-ccitt] [+eth] [-sum] [-crc] [+upload-sum] [+novatel] {<file>| -m [<src> [<len>]|boot|main]}
```

`+crc32`

Es wird eine 32 Bit CRC mit Hilfe des Onchip-CRC-Moduls, also in Hardware, ermittelt.

Das Prüfpolynom lautet 0x4C11DB7 (Ethernet-Polynom).

`+crc-ccitt`

LITEF's Datensatzprüfsumme

`+eth`

Mit diesem Schalter wird derselbe CRC-Algorithmus, wie beider Datenübertragung via Ethernet verwendet.

`-sum`

Wenn dieser Schalter angegeben wird, dann wird die default berechnete 32 Bit Prüfsumme nicht ermittelt. Bei dieser Prüfsummenberechnung werden alle 32 Bit Worte addiert. Ein Überlauf wird nicht berücksichtigt.

`-crc`

Wenn dieser Schalter angegeben wird, dann wird die default berechnete 32 Bit Prüfsumme entsprechend dem GNU/Linux-Programm cksum nicht ermittelt.

`+upload-sum`

verwendet im Kommando 'dnld', 32 Bit Summe (Add. aller 8 Bit Werte mod. 32 Bit)

Befehle im Detail

csm	Prüfsummen und CRC berechnen

+novatel

Mit diesem Schalter wird zusätzlich eine 32 Bit CRC mit dem Prüfpolynom `0xEDB88320` ermittelt. Dieses Prüfpolynom wird von der Fa. Novatel bei der Datenübertragung verwendet.

<file>

Die Daten werden aus dem angegebenen File gelesen. Das File kann eine Datei aus dem FAT-Dateisystem oder z.B. auch eine Pipe sein.

csm -m <src> [<len>]

Diese alternative Angabe der Datenquelle bezieht sich auf den Speicher des Microcontrollers. Es ist eine Quelladresse und eine Länge in Byte anzugeben.

Beispiele:

```
csm -m 0x8000000 0x40000
```

Ergebnisausgabe:

```
csm 0x7d2bc6af  crc 0x00526655  len     262144  addr 0x8000000
```

```
csm +crc32 +novatel -m 0x8000000 0x40000
```

Ergebnisausgabe:

```
crc32 0x01c26b1d  csm 0x7d2bc6af  crc 0x00526655  novatel 0x8722454a  len     262144  addr 0x8000000
```

Bei einer anderen Firmware-Version erscheinen andere Werte.

csm -m [boot|main]

Die Aufrufvariante ‚**csm -m**' ermittelt und prüft die Prüfsummen der beiden Firmware-Programme.

Die Aufrufvariante ‚**csm -m boot**' ermittelt und prüft die Prüfsumme des Bootprogramms.

Die Aufrufvariante ‚**csm -m main**' ermittelt und prüft die Prüfsumme des Hauptprogramms.

Anmerkung: Beide Programmversionen sind funktional identisch. Sie unterscheiden sich lediglich in der Adresse, für die sie erzeugt (gelinkt) wurden.

Stimmen die ermittelten Prüfsumme mit den intern gespeicherten überein, dann wird folgende Meldung ausgegeben:

```
crc is o.k.  boot loader
crc is o.k.  2nd programm version
```

Wird ein Fehler erkannt, dann erscheint z. B.:

```
crc is o.k.  boot loader
crc is wrong 2nd programm version
```

Befehle im Detail

date	Datum und Zeit einstellen oder ausgeben

Mit diesem Befehl wird die batteriegestützte Echtzeituhr des Gerätes gestellt oder angezeigt oder die Systemzeit angezeigt. Weiterhin können mit diesem Befehl Zeitumrechnungen von der Unix-Zeitangabe in die normale kalendarische und umgekehrt durchgeführt werden. Die Unix-Zeit zählt vom 1. 1. 1970 die Sekunden in einer 32 Bit Zahl. Die Systemzeit kann auf lokale Zeit, UTC, GPS-Zeit oder die eines NTP-Servers gestellt werden. Wenn das GPS-Board initialisiert ist und der erste GNSS-Fix ermittelt wird, dann werden auch die Systemzeit und die batteriegepufferte Echtzeituhr auf UTC gestellt.
In einem Unterkommando stehen Mittel zur Verfügung, die Systemzeit zur Laufzeit zu trimmen, um eine hoch genaue Zeitmessung zu erzielen.

Parameter:

```
  [-hh] [-r|-d|-jd|-g]     -hh print uptime as 'hh:mm:ss.ssssss' else as float
                                    -r  RTC
                                    -d  print diff
                                    -jd print Julian day
                                    -g  omit gps week time
| [-hh] [-c] {<val> | <[YY]YY.MM.DD> <HH:MM:SS>| gps-week <week> <week_s>}
                                    -c  compute only, else set RTC and system time
| -ntp [-v] [-dry | -m <trim_max_s>] <ntp_server1> [<ntp_server2> ...]
                           set RTC via NTP-Server, -v verbose, -dry don't set anything, -m change max # s
| -trim [off | on | [-abs] <trim>]
| -auto-set [off | on]     set cyclic sys time and RTC with UTC (gps) if differ
| -r2s                     read rtc and set sys time
| -s2r                     read sys time and set rtc
| -t                       run time test for rd32() and rd64()
| -uf                      test float printf
```

```
date
```

Ausgabe:

```
DATE: utc 17.02.22 10:44:58 1487760298.259649 0 494.849496
```

Es wird die Systemzeit ausgegeben.

Die Kennzeichnung ‚**utc**' bedeutet, dass die Systemzeit auf UTC eingestellt ist. Es gibt noch die Einstellung ‚**gps**', was bedeutet, dass die Systemzeit auf die GPS-Zeit eingestellt ist. Das Anzeigeformat ist systematisches Datum (YY.MM.DD HH:MM:SS), UNIX-Zeitdarstellung und Julianischer Tag Zeitdarstellung.

Die Ausgabe erfolgt im systematischen Datumsformat Jahr. Monat. Tag. Die Ausgabe 17.02.22 ist also der 22. Februar 2017. Darauf folgt die Uhrzeit im 24-Stundenformat. Dahinter folgt die Zeit im Unix-Format. Dahinter folgt der Tag des Jahres.

Danach folgt die Angabe des Julianischen Tages. Siehe dazu auch:

https://de.wikipedia.org/wiki/Julianisches_Datum#Berechnung_aus_dem_Kalenderdatum

Es folgt die Zahl des aktuell verwendeten GPS-UTC-Zeitversatzes, der hier im Beispiel 0 ist, weil die Systemzeit auf UTC eingestellt ist. Ist sie auf GPS-Zeit eingestellt (Kennung ‚**gps**'), dann repräsentiert diese Zahl die aktuelle Zeitdifferenz GPS-Zeit minus UTC-Zeit. Ab dem 1.1.2017 beträgt sie 18 s und in der Regel wird sie alle 1-6 Jahre um eine Sekunde erhöht.

Am Ende der Ausgabezeile steht die ‚Uptime' in Sekunden. Es ist die Zeit, die seit dem Einschalten bzw. Reset des Geräts vergangen ist.

Befehle im Detail

date	Datum und Zeit einstellen oder ausgeben

date –hh

Ausgabe:
```
DATE: utc 21.06.15 15:21:02 1623770462.473955 18 2162 228080.473955 5:26:55.831234
```

Mit dem Schalter '–hh' wird die Zeit im Format hh:mm:ss.ssssss ausgegeben.

date -r

Ausgabe:
```
DATE: rtc 17.02.22 10:45:15 1487760315.757812 0x58ad6bbb
```

Es wird die Zeit der batteriegestützten Echtzeituhr (rtc = Real-Time Clock) angezeigt. Diese läuft unabhängig von der Systemzeit.

date -d

Ausgabe:
```
diff sys_time-rtc_time: -0.018865 s
```

Es wird die Differenz aus Systemzeit und batteriegestützter Echtzeituhr angezeigt. Da die Zeit der batteriegestützten Echtzeituhr nur beim Systemstart in die Systemzeit übernommen wird, wird es zwischen beiden Zeiten über die Betriebsdauer eine sich verändernde Differenz ergeben.

date -jd

Ausgabe:
```
DATE: utc 17.02.22 10:45:08 1487760308.978795 0x58ad6bb4,  53th day, Julian Day 2457806.948009, up 505.568
```

Die Ausgabe erfolgt im systematischen Datumsformat Jahr. Monat. Tag. Die Ausgabe 17.02.22 ist also der 22. Februar 2017. Darauf folgt die Uhrzeit im 24-Stundenformat. Dahinter folgt die Zeit im Unix-Format in dezimaler und als hexadezimaler Zahlendarstellung. Dahinter folgt der Tag des Jahres.

Danach folgt die Angabe des Julianischen Tages. Siehe dazu auch:

https://de.wikipedia.org/wiki/Julianisches_Datum#Berechnung_aus_dem_Kalenderdatum

Am Ende folgt die vergangene Zeit (Laufzeit oder Uptime) seit Power on bzw. dem letztem Reset des Gerätes.

date –g

Ausgabe:
```
DATE: utc 17.02.22 10:45:08 1487760308.978795 0x58ad6bb4,  53th day, Julian Day 2457806.948009, up 505.568
```

Wird dieser Schalter angegeben, so wird die GPS-Wochenzeit nicht mit ausgegeben.

Befehle im Detail

date	Datum und Zeit einstellen oder ausgeben

date {<val> | <[YY]YY.MM.DD> <HH:MM:SS>| gps-week <week> <week_s>}

Zum Stellen der batteriegestützten Echtzeituhr und gleichzeitig der Systemzeit wird dieses Kommando verwendet. Wenn das GPS-Board initialisiert ist und die erste Positionsberechnung erfolgt (first fix), dann werden die batteriegestützte Echtzeituhr und die Systemzeit automatisch erneut gestellt. Die Eingabe kann im systematischen Datumsformat erfolgen, es wird dann die UTC berechnet, oder eine GPS-Woche mit Wochensekunden angegeben werden.

Die Zeit kann durch das Kommando

```
set rtc gps 18
```

auf die GPS-Zeit mit dem ab dem 1.1.2017 gültigen Zeitversatz von 18 s umgestellt werden und mit

```
set rtc utc
```

wieder auf die UTC zurückgestellt werden. Die batteriegestützte Uhr bleibt immer auf UTC eingestellt. Soll nach einem Power on mit der GPS-Zeit gearbeitet werden, dann muss in der **autoexec.sh** oben genanntes Kommando aufgeführt werden.

date -hh -c {<val> | <[YY]YY.MM.DD> <HH:MM:SS>| gps-week <week> <week_s>}

Mit dem optionalen Schalter kann eine Datums- und Zeitumrechnung vorgenommen werden, **ohne** die Systemzeit zu verstellen. Bei der Eingabe im systematischen Datumsformat werden die UTC, der Tag des Jahres und der Julianische Tag berechnet. Bei der Eingabe einer Zahl mit oder ohne Dezimalpunkt wird das systematische Datum und der dazugehörige Julianische Tag berechnet. Alternativ kann eine GPS-Woche mit Wochensekunden angegeben werden.

Beispiel 1:

```
date -c 2000.1.1 12.0.0
```

Ausgabe:

```
DATE: utc 00.01.01 12:00:00 946728000.000000 0x386dec40,   1st day, Julian Day 2451545.000000
```

Beispiel 2:

```
date -c 1500000000
```

Ausgabe:

```
DATE: utc 17.07.14 02:40:00 1500000000.000000 0x59682f00, 195th day, Julian Day 2457948.611111
```

Am 14. Juli 2017 wird bzw. wurde eine ‚runde' UTC-Zeit erreicht.

Beispiel 3:

```
date -c gps-week 1465 80000
```

Ausgabe:

```
DATE: utc 08.02.03 22:13:02 1202076782.000000 0x47a63c6e, 18, 1465, 80000.000000,  34th day, Julian Day 2454500.425718
```

-hh
Ausgabe der Zeit im Format hh:mm:ss.ssssss

Befehle im Detail

date	Datum und Zeit einstellen oder ausgeben

```
date -ntp [-v] [-dry | -m <trim_max_s>] <ntp_server1> [<ntp_server2> ...]
```

Mit dem Unterkommando ‚**-ntp**' können die Systemzeit und die Echtzeituhr über einen NTP-Server gestellt werden. Ferner kann die Systemzeit lediglich mit der Zeit eines NTP-Servers verglichen werden.

```
-v
```

Wird der Schalter ‚**-v**' („be verbose") angegeben, werden zusätzliche Informationen ausgegeben. Unter anderem:

Zu Server und Client:

Datum im systematischen Datumsformat YY.MM.DD, Uhrzeit im 24-Stundenformat, Zeit im Unix-Format in dezimaler Zahlendarstellung, IP-Adresse und URL des NTP-Servers, die RTT (Round Trip Time), Zeit-Offset (‚toffs') zwischen Server und Client (bei negativem Wert liegt Client hinter Server)

```
-dry
```

„dry" steht hier für „Trockenlauf". Wird der Schalter ‚-dry' angegeben, wird nur der Offset zwischen Client- und Server-Zeit ausgegeben und es findet kein Trimmen der Systemzeit und Echtzeituhr (bzw. Client-Zeit) auf Server-Zeit statt.

```
-m <trim_max_s>
```

Mit dem Schalter ‚**-m**' kann angegeben werden um wieviel Sekunden die Systemzeit und Echtzeituhr maximal zur NTP-Server-Zeit hin getrimmt werden sollen.

In den folgenden Beispielen wird ein Zeitserver der PTB verwendet (https://www.ptb.de/cms/ptb/fachabteilungen/abtq/gruppe-q4/ref-q42/zeitsynchronisation-von-rechnern-mit-hilfe-des-network-time-protocol-ntp.html):

Beispiel 1 (Trimmen:

```
date -ntp ptbtime1.ptb.de
```

Ausgabe:

```
ntp: toffs  0.000080 s, sys time and RTC set, trimmed backward
```

Beispiel 2:

```
date -ntp -dry ptbtime1.ptb.de
```

Ausgabe:

```
ntp: toffs -0.001163 s
```

Siehe f.

Befehle im Detail

date	Datum und Zeit einstellen oder ausgeben

„date -ntp [-v] [-dry | -m <trim_max_s>] <ntp_server1> [<ntp_server2> ...]“ ff.

Beispiel 3:

```
date -ntp -v ptbtime1.ptb.de
```

Ausgabe:

```
ntp: Server 21.06.16 09:53:25  1623837205.208943 s  IPAddr 192.53.103.108   ptbtime1.ptb.de
ntp: Client 21.06.16 09:53:25  1623837205.208021 s  RTT  0.022579 s  toffs -0.000921 s
ntp: Mode 4  VN 3  LI 0  Stratum 1   Poll 3   Prec 2^-25  RevId PTB
ntp: toffs -0.000921 s, sys time and RTC set, trimmed forward
```

Beispiel 4:

```
date -ntp -v -dry ptbtime1.ptb.de
```

Ausgabe:

```
ntp: Server 21.06.16 09:53:20  1623837200.834886 s  IPAddr 192.53.103.108   ptbtime1.ptb.de
ntp: Client 21.06.16 09:53:20  1623837200.834866 s  RTT  0.020750 s  toffs -0.000019 s
ntp: Mode 4  VN 3  LI 0  Stratum 1   Poll 3   Prec 2^-25  RevId PTB
ntp: toffs -0.000019 s
```

```
date -trim [off | on | [-abs] <trim>]
```

Die Systemzeit kann mit diesem Kommando getrimmt werden. Wenn das GPS-Board initialisiert ist und eine gültige Position berechnet wird, dann wird die Ganggenauigkeit der Systemzeit durch einen Softwarealgorithmus automatisch getrimmt. Default ist also „**date –trim on**“. Die Systemzeit läuft nach dem Trimmen mit einem Jitter von ca. +/- 15 µs synchron zur Zeit, die das GNSS zur Verfügung stellt. Ohne GNSS-Zeit kann ein Trimmwert spezifisch für ein jeweiliges Gerät bestimmt werden und mit diesem Befehl z. B. in der autoexec.sh eine verbesserte Ganggenauigkeit eingestellt werden. Die Ganggenauigkeit, die sich aus der GNSS-Zeit und dem Trimmalgorithmus ergibt, ist deutlich besser, als eine manuell eingestellte Trimmung.

```
-auto-set [off | on]
```

Mit diesem Unterkommando kann das automatische Nachziehen der Systemzeit auf GPS-Zeit (was ab einem Offset von 100 µs passiert) ab– bzw. eingeschaltet werden. Default ist „**on**“.

```
-r2s
```
Echtzeituhr lesen und Systemzeit stellen

```
-s2r
```
Systemzeit lesen und Echtzeituhr stellen

```
-t | -uf
```

Beide Kommandos sind Testkommandos. Sie haben für den Gerätebetrieb keine Bedeutung.

Befehle im Detail

df	Laufwerksverwaltung des FAT-Dateisystems (Micro-SD-Karte)

Mit diesem Kommando können Informationen über das FAT-Dateisystem ausgegeben werden. Das Dateisystem befindet sich auf einer internen Micro-SD-Karte. Als Dateisystem wird die erste Partition, falls die Micro-SD-Karte partitioniert ist, verwendet. Die Micro-SD-Karte muss mit dem Dateisystem FAT16 oder FAT32 formatiert sein.
Das Einlegen und Entfernen der Micro-SD-Karte wird automatisch erkannt. Beim Einlegen wird das Dateisystem gemountet. Das Mounten, also Initialisierung der SD-Karte und des SD-Kartentreibers, kann auch explizit mit diesem Kommando erfolgen bzw. wiederholt werden. Dabei kann ein ‚protection level' eingestellt werden, der z. B. das Dateisystem ‚read only' mountet. Diese Einstellungen können auch temporär sein.

Mit dem Kommando können Informationen zum Dateisystem und zu den kartenspezifischen Daten (CSD-Informationen der Micro-SD-Karte) ausgegeben werden.

Parameter:

```
[-s|-c|-e|[-v <lvl>] -i[ro|nof|csd] | -pl [<lvl>]] <drv_letter>:
```

```
df c:
```

Anzeige des benutzten und freien Speichers der Micro-SD-Karte. Als Information wird der Dateisystemtyp (FAT32 oder FAT16) der SD-Kartentyp, die formatierte Clustergröße und interne Fehlerzähler (crc, try, si) angezeigt.

Ausgabe:

```
drv c: FAT32, cl 32 KB, sect 7774208, size 3878912 KB, ro rw, SDHC, crc 0, try 0, si 0, used 3023776 KB, free 855136 KB
```

Sollten gerade eine oder mehrere Dateien geöffnet sein, dann kann die freie Kapazität nicht ermittelt werden und es erscheint folgende Ausgabe:

```
drv c: FAT32, cl 32 KB, sect 7774208, size 3878912 KB, ro rw, SDHC, crc 0, try 0, si 0, files are open, can't calc free space at this moment
```

```
df -s c:
```

(-s short – Kurzform)

Anzeige der Speicherkapazität der Micro-SD-Karte. Als Information wird der Dateisystemtyp (FAT32 oder FAT16) der SD-Kartentyp, die formatierte Clustergröße und Fehlerzähler (crc, try, si) angezeigt.

Ausgabe:

```
drv c: FAT32, cl 32 KB, sect 7774208, size 3878912 KB, ro rw, SDHC, crc 0, try 0, si 0
```

Befehle im Detail

df	Laufwerksverwaltung des FAT-Dateisystems (Micro-SD-Karte)

df -c c:

Anzeige des benutzten und freien Speichers. Hierbei wird der Wert aber durch erneutes Abzählen der freien Cluster bestimmt, und nicht wie bei 'df c:' durch Angabe des Wertes, den der FAT-Treiber zur Laufzeit aktuell hält.

df -e c:

Mit diesem Unterkommando werden Fehlerabbrüche beim SD-Kartenzugriff unterdrückt. Damit kann die SD-Karte, falls sie unformatiert bzw. fehlerhaft ist, mit Hilfe des Befehls ‚**sector**' ausgelesen bzw. bearbeitet werden. Das Kommando kann zur Datenrettung verwendet werden.

Ausgabe:

```
drive error reset, being able to use cmd 'sector' if no FAT file system is found
```

df -i c:

Dieses Unterkommando initialisiert die SD-Karte und den SD-Kartentreiber. Im Normalbetrieb ist das Kommando nicht erforderlich, da bei Power on, Geräte-Reset und dem Einlegen der SD-Karte dieses Kommando intern automatisch ausgeführt wird.

Ausgabe:

```
drv c: FAT32, cl 32 KB, sect 7774208, size 3878912 KB, ro rw, SDHC, crc 0, try 0, si 0
```

df -iro c:

(ro – read only)

Dieses Unterkommando initialisiert die SD-Karte und den SD-Kartentreiber wie im Kommando ‚**df -i c:**' mit der Zusatzeinstellung ‚**read only**'. Es können keine Daten auf die SD-Karte geschrieben werden. Zu erkennen ist die Einstellung an den beiden ‚**ro**'.

Ausgabe:

```
drv c: FAT32, cl 32 KB, sect 7774208, size 3878912 KB, ro, SDHC, crc 0, try 0, si 0
```

df -inof c:

(nof – no flush)

Dieses Unterkommando dient der Reinitialisierung der SD-Karte und den SD-Kartentreiber wie im Kommando ‚**df -i c:**' mit der Zusatzeinstellung dass keine eventuell geöffneten Dateien ‚geflusht' werden.
Ausgabe:

```
drv c: FAT32, cl 32 KB, sect 7774208, size 3878912 KB, ro rw, SDHC, crc 0, try 0, si 0
```

Befehle im Detail

df	Laufwerksverwaltung des FAT-Dateisystems (Micro-SD-Karte)

```
df -v 1 -i c:
```

(v – verbose)

Bei der Initialisierung der SD-Karte und des SD-Kartentreibers gesprächig sein.

Ausgabe:

```
FAT type          FAT32  Sectors          7774208
BootSecOffset      8192  Reserved Sectors    6296
FirstFATSector    14488  FAT Sectors          948
Num. of FAT's         2
FirstRootSector   16384  RootDirSectors         0
FirstDataSector   16384  MaxCluster        121218
SecPerCluster        64  BytesPerCluster    32768
drv c: FAT32, cl 32 KB, sect 7774208, size 3878912 KB, ro rw, SDHC, crc 0, try 0, si 0
```

Befehle im Detail

df	Laufwerksverwaltung des FAT-Dateisystems (Micro-SD-Karte)

```
df -icsd c:
```

Die SD-Karten-Daten (CSD - card specific data) werden angezeigt. Daraus kann z. B. der Hersteller, das Produktionsjahr und verschiedene anderen Daten entnommen werden. Das Kommando dient lediglich der Information.

Es wird empfohlen, nach diesem Kommando die SD-Karte mit ‚**df -i c:**' neu zu initialisieren.

Ausgabe:

```
card specific data:
----------------------
CSD.CSDStruct VerId    1
CSD.SysSpecVersion     0
CSD.Reserved1          0
CSD.TAAC               14
CSD.NSAC               0
CSD.MaxBusClkFrec      25000 kbit/s
CSD.CardComdClasses    0x5b5
CSD.RdBlockLen         9
CSD.PartBlockRead      0
CSD.WrBlockMisalign    0
CSD.RdBlockMisalign    0
CSD.DSRimplemented     0
CSD.Reserved2          0
CSD.DeviceSize         7591
CSD.MaxRdCurrentVDDMin 0
CSD.MaxRdCurrentVDDMax 0
CSD.MaxWrCurrentVDDMin 0
CSD.MaxWrCurrentVDDMax 0
CSD.DeviceSizeMul      0
CSD.EraseGrSize        1
CSD.EraseGrMul         127
CSD.WrProtectGrSize    0
CSD.WrProtectGrEnable  0
CSD.ManDeflECC         0
CSD.WrSpeedFact        2
CSD.MaxWrBlockLen      9
CSD.WriteBlockPartial  0
CSD.Reserved3          0
CSD.ContentProtectApp  0
CSD.FileFormatGroup    0
CSD.CopyFlag           0
CSD.PermWrProtect      0
CSD.TempWrProtect      0
CSD.FileFormat         0
CSD.ECC                0
CSD.CSD_CRC            22
CSD.Reserved4          1

CID.ManufacturerID     0x01
CID.OEM_AppliID        0x5041
CID.ProdName           'R04GS'
CID.ProdRev            2.2
CID.ProdSN             4020300363
CID.Reserved1          0
CID.ManufactDate       5/2015
CID.CID_CRC            78
CID.Reserved2          1

CardCapacity           3980394496
CardBlockSize          512
RCA                    0x266e
Card_Type              2
cmd R1: CMD13, 0 (0x0), timeout, stat 0x20a004, propably SD 1.1
```

Befehle im Detail

df	Laufwerksverwaltung des FAT-Dateisystems (Micro-SD-Karte)

df -pl c:

Ausgabe des momentan eingestellten ‚protection level' für die SD-Karte.

```
drv: c: protection level actual 1 -> sector 0 ro, other sectors rw, write attempts return an error code
```

Es ist ein Schreibschutz für den Bootsektor eingestellt und Schreibversuche werden intern als Fehler gemeldet. In alle anderen Sektoren kann geschrieben werden, was für die die normale Verwendung der SD-Karte als Datenspeicher notwendig ist.

df -pl <protect_level> c:

Dieses Unterkommando dient zum Einstellen eines Schreibschutzes für die SD-Karte. Standardmäßig ist lediglich der Bootsektor schreibgeschützt. Der Schreibschutz kann jederzeit mit diesem Kommando geändert werden.

In der Angabe <protect_level> haben die untersten 3 Bit folgende Bedeutung:

Bit 0 = 1 ⟶ der Bootsektor ist schreibgeschützt, 0 ⟶ beschreibbar (1 ist die Standardeinstellung)
Bit 1 = 1 ⟶ alle weiteren Sektoren sind schreibgeschützt, 0 ⟶ sie sind beschreibbar (0 ist die Standardeinstellung)
Bit 2 = 1 ⟶ Schreibversuche geben intern einen Fehler zurück, 0 ⟶ Schreibvorgang wird fehlerfrei gemeldet (0 ist Standard)

Beispiele:

```
df -pl 0 c:
```

Ausgabe:

```
drv: c: protection level old 1, actual 0 -> sector 0 rw, other sectors rw, write attempts return an error code
```

```
df -pl 1 c:
```

Ausgabe:

```
drv: c: protection level old 0, actual 1 -> sector 0 ro, other sectors rw, write attempts return an error code
```

```
df -pl 2 c:
```

Ausgabe:

```
drv: c: protection level old 1, actual 2 -> sector 0 rw, other sectors ro, write attempts return an error code
```

```
df -pl 3 c:
```

Ausgabe:

```
drv: c: protection level old 2, actual 3 -> sector 0 ro, other sectors ro, write attempts return an error code
```

```
df -pl 7 c:
```

Ausgabe:

```
drv: c: protection level old 3, actual 7 -> sector 0 ro, other sectors ro, write attempts do not return an error code
```

Befehle im Detail

dhry	Testprogramm Dhrystone, CoreMark und Benchmarking

Dieses Kommando mit seinen drei Teilkommandos ist ein reines Testprogramm. Es dient der Ermittlung der Rechenleistung des verwendeten Microcontrollers insbesondere die verbleibende Rechenleistung unter verschiedenen realen Anwendungsbedingungen des Gerätes.

Es stehen folgende Benchmark-Unterprogramme zur Verfügung:

Dhrystone V 2.1
CoreMark V 1.0
simple Benchmark

Wenn Benchmark-Werte ohne und mit spezieller Gerätekonfiguration ermittelt werden, dann erlaubt das einen Rückschluss auf die Leistungsreserven in einer speziellen Anwendung. Damit ist eine Optimierungshilfe für spezielle Anwendungen gegeben.

Parameter:

```
   [-v] <no_of_loops>   do dhrystone
 | -sb                  do simple benchmark
 | -cm [<arg1> ...]     do coremark V1.0
                        1. - 1st init value of data 0
                        2. - 2nd init value of data 0
                        3. - 3rd init value of data 0x66
                        4. - number of iterations (0 for auto : default value)
                        5. - reserved for internal use
                        6. - reserved for internal use
                        7. - for malloc users only, ovreride the size of the input data buffer
```

```
dhry [-v] <no_of_loops>
```

Der Dhrystone-Benchmark existiert schon ca. 30 Jahre. Er bezieht sich auf elementare Programmfunktionen eines Prozessors ohne mathematische, grafische oder dateiorientierte Teiltests. Er hat heute an Bedeutung etwas an Bedeutung verloren, da die Realitätsnähe und Vergleichbarkeit nicht mehr gegeben sind.

Beispiel:

```
dhry 3000000&
```

Ausgabe:

```
dhrystone: loops 3000000, time 4.745174615 s, dhrystones per sec 632221
```

Das entspricht einem Dhrystone MIPS-Wert von 360 DMIPS.

Dieser Wert wird erreicht, wenn das Gerät lediglich eingeschaltet wird und keine weitere Konfiguration erfolgt (‚leere' **autoexec.sh**).

Befehle im Detail

dhry	Testprogramm Dhrystone, CoreMark und Benchmarking

```
dhry -cm
```

Es wird der zur Zeit weit verbreitete CoreMark-Test durchgeführt. Für weitere Informationen und Vergleiche mit anderen Microcontrollern oder PC-Prozessoren wird auf die folgende Internetseite verwiesen:

http://www.eembc.org/coremark/index.php

Ausgabe:

```
2K performance run parameters for coremark.
CoreMark Size    : 666
Total time       : 14.456346 s
Iterations/Sec   : 415.0 CoreMark
CoreMark/MHz     : 2.470
Iterations       : 6000
Compiler version : GCC6.2.1 20161205 (release) [ARM/embedded-6-branch revision 243739]
Compiler flags   : -O2
Memory location  : STACK
seedcrc          : 0xe9f5
[0]crclist       : 0xe714
[0]crcmatrix     : 0x1fd7
[0]crcstate      : 0x8e3a
[0]crcfinal      : 0xa14c
Correct operation validated. See readme.txt for run and reporting rules.
CoreMark 1.0 : 415.0 / GCC6.2.1 20161205 (release) [ARM/embedded-6-branch revision 243739] -O2 / STACK
```

Befehle im Detail

dhry	Testprogramm Dhrystone, CoreMark und Benchmarking

```
dhry -sb
```

Es werden elementare Softwareroutinen getestet. Bei den mathematischen Funktionen wird deutlich, dass die arithmetischen Grundoperationen (+-*/) beim Zahlenformat **float** aufgrund des mathematischen Coprozessors (FPU mit einfacher Genauigkeit) ebenso schnell ist, wie die Ganzzahlarithmetik.

Ausgabe:

```
simple benchmark Rev. 2 08/2016, on STM32F427xx @ 168000000 Hz, GCC 6.2.1
---------------------------
offset       empty_call                  107.77 ns    18 Cyc    216 ms    2000000
---------------------------
short        a*b                          11.97 ns     2 Cyc    239 ms    2000000
short        a/b                          35.94 ns     6 Cyc    144 ms    1000000
int          a*b                           5.96 ns     1 Cyc    341 ms    3000000
int          a/b                          47.92 ns     8 Cyc    117 ms     750000
---------------------------
ushort       a*b                          11.97 ns     2 Cyc    239 ms    2000000
ushort       a/b                          29.94 ns     5 Cyc    138 ms    1000000
unsigned     a*b                           5.96 ns     1 Cyc    341 ms    3000000
unsigned     a/b                          47.92 ns     8 Cyc    117 ms     750000
---------------------------
long long    a+b                          11.96 ns     2 Cyc    299 ms    2500000
long long    a*b                          23.95 ns     4 Cyc    165 ms    1250000
long long    a/b                         420.82 ns    71 Cyc    165 ms     312500
ulong long   a+b                          11.95 ns     2 Cyc    299 ms    2500000
ulong long   a*b                          23.95 ns     4 Cyc    165 ms    1250000
ulong long   a/b                         387.67 ns    65 Cyc    155 ms     312500
---------------------------
float        a+b                           5.95 ns     1 Cyc    682 ms    6000000
float        a*b                           5.95 ns     1 Cyc    682 ms    6000000
float        a/b                          47.85 ns     8 Cyc    467 ms    3000000
float        log(a)                      892.53 ns   150 Cyc    750 ms     750000
float        sin(a)                      353.25 ns    59 Cyc    277 ms     600000
float        cos(a)                      478.95 ns    80 Cyc    352 ms     600000
float        atan(a)                     502.89 ns    84 Cyc    366 ms     600000
float        sqrt(a)                      47.85 ns     8 Cyc    467 ms    3000000
---------------------------
double       a+b                         563.23 ns    95 Cyc    403 ms     600000
double       a*b                         472.87 ns    79 Cyc    348 ms     600000
double       a/b                        3791.28 ns   637 Cyc   1170 ms     300000
double       log(a)                    18585.95 ns  3122 Cyc   1402 ms      75000
double       sin(a)                     7150.33 ns  1201 Cyc    435 ms      60000
double       cos(a)                    10178.25 ns  1710 Cyc    617 ms      60000
double       atan(a)                   12088.53 ns  2031 Cyc    732 ms      60000
double       sqrt(a)                    6333.86 ns  1064 Cyc   1932 ms     300000
---------------------------
char         200  a1[i]=a2[i]           1524.58 ns   256 Cyc     33 ms      20000
short        200  a1[i]=a2[i]           3031.67 ns   509 Cyc     63 ms      20000
int          200  a1[i]=a2[i]           5999.99 ns  1008 Cyc    122 ms      20000
---------------------------
char         200  *p1++=*p2++           1524.62 ns   256 Cyc     33 ms      20000
short        200  *p1++=*p2++           3019.77 ns   507 Cyc     63 ms      20000
int          200  *p1++=*p2++           7185.10 ns  1207 Cyc    146 ms      20000
int          200  *--p1=*--p2           6000.00 ns  1008 Cyc    122 ms      20000
char         200*4 memcpy()             3855.94 ns   648 Cyc    159 ms      40000
---------------------------
float        sprintf(s, "%14.10f", a)   44447.33 ns  7467 Cyc    223 ms       5000 123.0123443604
double       sprintf(s, "%14.10f", a)   45895.16 ns  7710 Cyc    230 ms       5000 123.0123456789
float        sprintf(s, "%10.6f", a)    39861.71 ns  6697 Cyc    200 ms       5000 123.012344
double       sprintf(s, "%10.6f", a)    38440.68 ns  6458 Cyc    193 ms       5000 123.012346
time64_t     sprintf(s, "%lu.%06lu", a) 19931.86 ns  3349 Cyc    100 ms       5000 123.012345
time64_t     sprintf(s, "%.6f", a) flt  39894.74 ns  6702 Cyc    200 ms       5000 123.012344
time64_t     sprintf(s, "%.6f", a) dbl  40563.63 ns  6815 Cyc    203 ms       5000 123.012346
```

Befehle im Detail

dnld	Kommando zum Download von Dateien insbesondere zum Firmware-Update

Das Kommando **dnld** ermöglicht den Dateidownload in ein beliebiges File des **ppmOS**. Im speziellen Fall wird es zum Firmware-Update über eine serielle Schnittstelle (**coma**, **comb** oder **comc**) oder über den Bluetooth-Link (**blth**) benötigt. Es ist nicht für eine Konsolensitzung via Tastatur bestimmt, da die Abfolge und insbesondere die CRC-Absicherung der Datenübertragung einem Protokoll unterliegt, welches ‚von Hand' nicht zu realisieren ist. Ein Firmware-Update dauert vom Start des Updates bis zum Neustart der Firmware ca. 15 s, wovon die serielle Datenübertragung (921,6 kBd) für ein ca. 400 kB File 6.6 s dauert.

Die Datenübertragung erfolgt blockorientiert und sie ist mit CRC gesichert. Eine implementierte Fehlertoleranz durch Blockwiederholung erhöht die Sicherheit der Datenübertragung wesentlich. D. h., falls während der Datenübertragung eine Störung oder sogar eine Leitungsunterbrechung auftritt, dann kann sie auch nach Unterbrechung von mehreren Sekunden wieder aufgenommen werden.

Am Ende der Dateiübertragung kann vom Absender (optional) ein Shell-Kommando übermittelt werden, dass unmittelbar nach Abschluss der Datenübertragung auf dem Gerät ausgeführt wird. Auf der PC-Seite ist dazu das Kommandozeilenprogramm ‚**upload.exe**' erforderlich, das das Protokoll realisiert.

Eine Dateiübertragung läuft am Beispiel eines Firmware-Updates wie folgt ab (PC und Gerät sind über eine Terminalschnittstelle verbunden):

1. Auf dem PC wird das Kommandozeilenprogramm ‚**upload.exe**' mit Parametern gestartet. Das geschieht zweckmäßiger Weise mit einer Batchdatei.

```
upload.exe 40xx_firmware_xyz.bin com9:921600:rts-cts -stm32 -safe c:/sys/40xx_2nd.bin "'echo; prog -u c:/sys/40xx_2nd.bin&'"
```

Der 1. Parameter
`40xx_firmware_xyz.bin`
benennt, das zu übertragende File. Es muss im Dateisystem des PC liegen (Festplatte lokal oder auch clientseitig von einem Server abrufbar).

Der 2. Parameter
`com9:921600:rts-cts`
gibt die serielle PC-Schnittstelle (auch USB-Seriell-Adapter), die Baudrate (40xx **coma** und **comc** max. 921600 Bd, **comb** max. 230400 Bd) und das Handshakeprotokoll RTS/CTS an. Die Baudrate muss mit der an der Terminalschnittstelle des 40xx übereinstimmen.

Tip: Eine verbesserte Ausnutzung der Übertragungsbandbreite kann erreicht werden, wenn auf dem (Windows-)PC im Gerätemanager bei den Eigenschaften des betreffenden COM-Ports auf der Registerkarte Anschlusseinstellungen/ Erweitert/BM-Einstellungen/Wartezeit (ms): der Wert 1 eingestellt wird (standardmäßig steht dort 16). Dann wird bei 921 kBaud eine Bandbreitenausnutzung inkl. DÜ-Protokoll (CRC, Blockprotokoll) von 66 % erreicht.

Der 3. und 4. Parameter
`-stm32 -safe`
benennt die Protokollart und ‚**-safe**' stellt das Protokoll auf Blockwiederholung im Fehlerfall ein.

Siehe f.

Befehle im Detail

dnld	Kommando zum Download von Dateien insbesondere zum Firmware-Update

Der 5. Parameter
`c:/sys/40xx_2nd.bin`

benennt das Zielfile im 40xx-Gerät. Beim Firmware-Update ist es eine Datei im FAT-Dateisystem. Für eine andere Dateiübertragung kann es auch ein anderes File sein (**pipe..., coma, comc, comb, blth, can1-**..., Ethernet-Files **ips**..., **icom**...).

Der 6. Parameter (optional)
`"'echo; prog -u c:/sys/40xx_2nd.bin&'"`
gibt hier eine Folge von 2 Shell-Kommandos (unbedingte Kommandoverkettung mit **;**) an. Die äußere Klammerung "..." ist auf der PC-Seite nötig, um die Wörter zu einem Parameter zusammenzufassen. Die innere Klammerung mit ‚...' benötigt das ppmOS, um die beiden Kommandos **echo** und **prog** zu einem Kommando zusammenzufassen.

Tip:

Mit dem folgenden Shellscript lässt sich ein Filetransfer (über **coma**, beide Datenrichtungen) über einen Ethernetport protokollieren:

```
ip server start ipsa:10000 -i
set ird coma tee ipsa1
set ord coma tee ipsa1
```

Wenn nun auf einem PC (kann natürlich auch der Quell-PC sein) eine TCP-Client-Verbindung (ASCII raw mode) zum 40xx-Server hergestellt wird, dann kann der Protokollablauf sehr einfach nachvollzogen werden.

Parameter:

```
  <csm> param <flen> <fcsm> <dest_fname>   <flen> as decimal, <csm> and <fcsm> as hex without leading 0x
| <csm> blk <fileoffs> <data_bytes>        first 2 values are hex without leading 0x, first 3 <data_bytes>
                                           are coding tab for '\r', '\n' an 0
| <csm> check <fid> [<cmd>]                after download execute optional command <cmd>
```

```
<csm> param <flen> <fcsm> <dest_fname>

@@dnld a6041da5 param 396676 34be3a92 c:/sys/40xx_2nd.bin
```

Hiermit kündigt der PC die Übertragung eines Files an. Dies ist das erste Kommando einer Dateiübertragung. Die vorangestellten ‚@@' sind Steuerzeichen für den Shell-Interpreter. Das erst ‚**@**' bedeutet ‚kein Kommandoecho' und das zweite ‚**@**' bedeutet ‚den Rest der Zeile nicht nach erweiterter Syntax absuchen , (siehe auch Kommando ‚**set ext-syntax**' S.240). Dieser Request ist mit **<csm>** versehen, um die Integrität prüfen zu können. **<flen>** und **<fcsm>** beziehen sich auf das zu übertragende File.

Der Controller antwortet mit

dnld_ack a33f04b

Der Wert **a33f04b** ist das Ergebnis der Operation

<flen> xor <fcsm> xor 0x3e8bc75d

Befehle im Detail

dnld	Kommando zum Download von Dateien insbesondere zum Firmware-Update

```
<csm> blk <fileoffs> <data_bytes>
```

Der PC sendet die binären Daten vom **<fileoffs>** mit diesem Kommando. In den binären Daten **<data_bytes>** werden lediglich die Zeichen CR, LF und EOS (0xd CR, 0xa LF, 0x0 EOS) umkodiert, da diese sonst das vorgegebene Zeilenformat aufheben würden. Das erste Byte in **<data_bytes>** enthält den Code für oxd, das 2. für 0xa und das 3. für 0x0. Nach dem 3. kommen dann die Daten (payload). Die Codetabelle kann bei jeder Zeile anders sein, weshalb sie in jeder Zeile mitgesendet werden muss. Die **<csm>** ermöglicht die Erkennung eines Übertragungsfehlers. Es werden max. 253 Byte in einer Zeile übertragen. Nur dadurch ist gewährleistet, dass immer mindestens 3 Codes zur Verfügung stehen, die Sonderzeichen CR, LF und EOS umzukodieren.

Beispiel:

PC:

```
@@dnld fce3fe3b blk 1fa !$%Ž<½½O8pÛ#LÇÆÐH#à"E©ýRA¬-#_#°Ô[Â-–Vu<VRÈ6#j +Øn|!›c##Zg#@#y£]Ü}z{ŸpÍf^tà¶#˜W«âœŽ ¡'9 `•<À'‹‹Ýæ Rû¥,åæd†X
[+¼ïFê°6˜©· }h³"-/-30î©ê#-¤]#l  m2Ô'póÐþV°ÝIKqÙL#6Çû#÷Ã" ´Î•=uÊ(€:ðŸ ûöF»¸ûñ|yÿôö›áCëÿåšÍ¼ê-Ð}ìwp†4ÀmG0#K#=©VÅ9«#,'##C#Å=%.r Á*Ì Ž#x€O#¡|
##»Í##ëš#¤öK
```

Controller:

```
dnld_blk a33f14c
```

Wenn das PC-Programm **upload.exe** nach einer vorgegeben Zeit (3 s) keine Antwort auf einen Datenblock bekommt oder die Checksumme falsch ist, dann wird er max. siebenmal wiederholt.

```
<csm> check <fid> [<cmd>]
```

Mit diesem Unterkommando wird die Fileübertragung abgeschlossen.

PC:

```
@@dnld 2596b34f check a33f04b 'echo; echo prog -u c:/sys/40xx_2nd.bin&'
```

Controller:

```
dnld_chk 396676 bytes, csm 34be3a92 success, cmd: 'echo; echo prog -u c:/sys/40xx_2nd.bin&'
```

Mit der Antwort gibt der Controller dem PC-Programm bekannt, ob die Übertragung erfolgreich war. Das PC-Programm hat dann die Möglichkeit dies anzuzeigen.

Befehle im Detail

echo	Ausgabe von Text

Das Kommando dient zum Ausgeben von Text während der Ausführung von Shell-Skripten.

Mit Hilfe der Ausgabeumleitung kann der Text in ein beliebiges File oder Schnittstelle geschrieben werden.

Der Ausgabetext kann in **"..."** oder **'...'** eingeschlossen werden. Der Shell-Interpreter wird keine Ersetzung von Shell-Funktionen oder Shell-Variablen vornehmen.

Zweckmäßiger Weise wird, wenn keine Ausgabeumleitung verwendet wird, ein '**@**' vorangestellt. Das unterdrückt die Ausgabe der Kommandozeile (Kommandoecho - siehe auch **'set echo'** S. 239).

Beispiele:

```
@echo hello world!

@echo Der letzte GPRMC-Datensatz lautet: $(gprmc)
```

Mit Umleitung in ein File lässt sich 'zur Not' ein Skript erstellen (z. B. via SMS)

Beispiel:

```
echo gps init >c:/sys/gps-print-to-comb
echo "loop -8 -t 1000 '@echo $(gpgga) >comb; @echo $(gprmc) >comb' &" >>c:/sys/gps-print-to-comb
```

Man beachte in der 2. Zeile die '>>' zur Ausgabeumleitung, die ein Anhängen an das in der 1. Zeile angelegte File bewirken.

Parameter:

[-n] [<text_to_print_out> | "<text_to_print_out>" | '<text_to_print_out>']

Für dieses Kommando gibt es keine Kommandozeilenhilfe.

-n

Der standardmäßige Zeilenvorschub am Ende der Ausgabe wird weggelassen.

Befehle im Detail

find	Daten im Speicher des Microcontrollers suchen

Mit diesem Kommando können Daten im RAM, FlashROM bzw. im gesamten Adressraum gesucht werden. Die Suche kann auch negiert werden, also Ausgabe aller Fundstellen, die nicht den Suchdaten entsprechen. Es wird neben Fundadressen auch ein Zähler ausgegeben.

Das Kommando hat für die normale Anwendung des Gerätes eine untergeordnete Bedeutung.

Die Ausgabe kann begrenzt werden, um ein 'Fluten' des Terminals zu vermeiden. Da nur geringe Teile des gesamtem32 Bit -Adressraumes durchsuchbar sind, sind Kenntnisse über den verwendeten Microcontroller (STM32F429) unbedingt erforderlich. Beim Zugriff auf nicht vorhandene Speicherbereiche kann das ppmOS abstürzen. Es läuft danach wieder wie nach einem Power on bzw. Reset an.

Die Suchdaten können maximal 256 Byte lang sein.

Beispiel:

```
>find 0x8000000 0x200000 "copyright"
searching from 0x8000000 to 0x8200000 (0x200000), step 1, pattern len 9 (0x9)
pattern: 00000000  63 6f 70 79 72 69 67 68 74  copyright
found: 0x805125c
found: 0x816125c
find: pattern[9] == found      2 times
```

Hinter '**pattern:**' werden noch einmal die Suchdaten zur Kontrolle ausgegeben und hinter '**found:**' erscheinen jeweils die Anfangsadressen der Speicherbereiche mit den Suchdaten.

Es gibt zwei Fundstellen im FlashROM, die mit:

```
peb -p 0x805125c 23 23
```

untersucht werden können. Es erscheint die Ausgabe, dass die Rechte dieser Firmware bei der ppm GmbH liegen:

```
0805125c  63 6f 70 79 72 69 67 68 74 20 70 70 6d 20 47 6d 62 48 20 32 30 31 37  copyright ppm GmbH 2017
```

Parameter:

```
<src> <len> [-f{8|16|32}] <val> [<val>] [not] [-p <max_err_prt>|all]     <val> can be a string "..."
```

```
<src> <len>
```

Angabe des Speicherbereichs mit Anfang und Länge. Die Zahlen können dezimal oder hexadezimal (mit vorangestelltem 0x...) angegeben werden.

Befehle im Detail

find	Daten im Speicher des Microcontrollers suchen

`-f{8|16|32}`

Dieser optionale Parameter gibt an, welches Datenformat die Daten **<val>** haben. Ohne **-f**... wird Byte angenommen.

Es bedeuten:
-f8 8 Bit Daten
-f16 16 Bit Daten
-f32 32 Bit Daten

<val> [<val>]

Die Suchdaten sind eine Folge von numerischen oder Stringwerten. die, mit Leerzeichen getrennt, eingegeben werden. Ein Stringwert ist eine Anzahl von Zeichen, die in **"..."** eingeschlossen sind. Die Arten der Suchwerte können beliebig gemischt werden. Es ist zu beachten, dass die Zeichenkette (Stringwerte) als 8-Bit-Werte übernommen werden und die Nichtstring-werte die Länge des Datenformats haben.

Beispiel:

```
"hello world!" 0xd 10
```

not

Es werden die Adressen der Daten ausgegeben, die nicht den Suchwerten entsprechen. Das ist z. B. sinnvoll, um nach dem Löschen eines FlashROM-Bereiches, festzustellen, ob der gesamte Bereich 0xff enthält bzw. ob es Daten gibt die **nicht** 0xff sind..

-p <max_err_prt>|all

Gibt die maximale Zahl von Ausgabezeilen an. Die Zeile mit der Anzahl der Fundstellen wird immer ausgegeben.
Mit **‚all'** werden alle Fundstellen und mit **‚0'** nur die Zusammenfassung (Anzahl der Fundstellen) ausgegeben.

Befehle im Detail

ftp	FTP-Filetransfer über das GSM-Modem

Das ‚**ftp**'-Kommando startet einen GSM-FTP-Client. Es dient der Fileübertragung über das GSM-Modem vom Gerät zu einem Server oder von einem Server zum Gerät. Es steht erst nach der Initialisierung des GSM-Modems zur Verfügung.

Für die Übertragung über Ethernet wird das Kommando ‚**ip ftp {put|get} ...**' verwendet.

Parameter:

```
  [-v|-s] [-w] put [-try <n>] [-rm[-anyway]] [-t[e] <tmot_s>] {{@|!}[<infofile>]|<usr>:<pwd>{@|!}<server>[/[path/][<file>]]} [<local_file>]
| [-v|-s] [-w] get [-a|-i|-d]                                 {{@|!}[<infofile>]|<usr>:<pwd>{@|!}<server>[/[path/][<file>]]} [<local_file>]
```

-v|-s

Mit den optionalen Schaltern werden zusätzliche Statusausgaben während der Fileübertragung aktiviert (**-v** verbose) oder es werden alle Statusausgaben unterdrückt (**-s** silent).
Achtung: mit ‚**-v**' wird der GSM-Modem-Protokollverlauf mit ausgegeben. Es entstehen viele Ausgaben, deshalb ist dieser Schalter nur zu Diagnosezwecken sinnvoll.
Ohne einen dieser Schalter werden folgende Informationen ausgegeben:

```
>ftp put ! c:/tmp/200k
reading from upload.cfg
at^moni
mrx0:
mrx0: Serving Cell                                          I Dedicated channel
mrx0: chann rs  dBm MCC MNC  LAC cell NCC BCC PWR RXLev  C1 I chann TS timAdv PWR  dBm Q ChMod
mrx0: 16 46  -64 262  02 0CB2 1E53   4   1  33  -107  42 I    No connection
mrx0:
mrx1: OK
ftp: put connected
-------------------------------------
cpu time 0.217765754 s
<      1 s  1500      1500
<      2 s  4500      6000
<      3 s  9000     15000
<      4 s 13500     28500
<      5 s 12000     40500
<      6 s  9000     49500
<      7 s  7500     57000
<      8 s  9000     66000
<      9 s  9000     75000
<     10 s 10500     85500
<     11 s 12000     97500
<     12 s 13500    111000
<     13 s 12000    123000
<     14 s 12000    135000
<     15 s 13500    148500
<     16 s 12000    160500
<     17 s 12000    172500
<     18 s 13500    186000
<     19 s 12000    198000
-------------------------------------
 >    20 s     0         0      0 bps
<     20 s  6800    204800  10240 bps
ftp: put success!
```

Die Ausagbe kann mit ‚**gsm echo tcp off**' abgeschaltet und mit ‚**gsm echo tcp on**' wieder angeschaltet werden.

Befehle im Detail

ftp	FTP-Filetransfer über das GSM-Modem

-w

Mit dem optionalen Schalter wird eingestellt, dass das Kommando erst zurückkehrt, wenn die Übertragung vollständig erledigt ist, also bis zum erneuten Freigeben des GSM-Modems. Anderenfalls kehrt das Kommando nach dem Starten des ftp-Tasks zurück.

Der Schalter wird empfohlen bei z. B. cron-job-gesteuerten Fileübertragungen. Insbesondere bei denen, die mit

ls `-l <ftp_shell_script>` **-f "ftp -w put ! ..."**

erzeugt werden.

put | **get**

Mit ‚**put**' wird die Fileübertragung vom Gerät zum FTP-Server eingestellt und mit ‚**get**' vom FTP-Server zum Gerät.

-try <n>

Mit dem optionalen Schalter ‚**-try**' kann eine Anzahl der Übertragungsversuche eingestellt werden, um die Datei vollständig zu übertragen. Der Defaultwert, also ohne ‚**-try**', ist 1.

Eine Wiederholung wird durch das Aufsetzen eines Cron-Jobs mit den Parametern

cron put *+3. * * * * * * -t <ftp_put_cmd>

um 3 Sekunden verzögert eingeleitet. Darin ist **<ftp_put_cmd>** die ursprüngliche ‚**ftp put ...**'-Kommandozeile. Sie wird als separater Task (**-t**) ausgeführt. Das verhindert ein eventuelles Blockieren des Cron-Managertasks, falls ‚**-w**' in ‚**ftp -w put ...**' angegeben wurde.

-rm | **-rm-anyway**

Mit diesen optionalen Schaltern wird das Löschen der Quelldatei bei ‚**ftp put ...**' eingestellt. Ohne diesen Parameter wird die Datei nach der Fileübertragung **nicht** gelöscht.

Mit ‚**-rm**' wird sie gelöscht, wenn die Übertragung erfolgreich war. Dieser Schalter hat besondere Bedeutung, wenn eine FTP-Fileübertragung zu einem Server (put) über das Kommando ‚`ls -l <ftp_shell_script> -f "ftp put -rm ..."`' eingeleitet wird, da dann bei wiederholtem Aufruf von ‚ls -f `"ftp put -rm ..."`' die betreffende Datei nicht wiederholt übertragen wird.

Mit ‚**-rm-anyway**' wird sie auf jeden Fall nach Ausführung des Kommandos gelöscht.

Befehle im Detail

ftp	FTP-Filetransfer über das GSM-Modem

`-t[e] <tmot_s>`

Mit ‚**ftp put ...**' ein Timeout eingestellt werden, nachdem nach Erreichen des Fileendes des Eingabefiles, falls es sich um eines mit verzögertem Datenaufkommen handelt, dieses geschlossen wird und damit die FTP-Übertragung beendet wird. Der Wert für **<tmot_s>** ist auf maximal 1 Jahr begrenzt, was eher ein theoretischer Wert ist.

Mit ‚**-te**' wird die Übertragung am aktuellen Dateiende begonnen, d. h. es wird auf ‚neue' Daten ab Start der FTP-Verbindung gewartet.

Damit ist es z. B. möglich, eine Echtzeit-FTP-Fileübertragung über GSM zu realisieren.

Besondere Bedeutung hat dies z. B. bei dem Eingabefile **gps2** oder anderen Schnittstellen bzw. Streams, bei denen die Daten ohne Zwischenspeicherung in ein reguläres File im FAT-Dateisystem sofort zu einem entfernten FTP-Server übertragen werden soll.

Beachtet werden muss hierbei das Timeout des FTP-Servers und die Maximalzeit eines Transfers. Das hier eingestellte Timeout sollte also kleiner oder gleich dem Servertimeout sein.

Beispiel 1:

```
gps reply pipe1
ftp put -t 10 PPM20xx:xyzabc@217.160.231.191/9F6/realtime.log pipe1
```

Die über gps2 ankommenden Daten werden zusätzlich in die Pipe ‚**pipe1**' geschrieben. Es wird ein Echtzeit-FTP-Transfer mit angegebener IP-Adresse, User und Passwort gestartet. Auf dem Server wird das File `realtime.log` im Verzeichnis `9F6` des Users angelegt.

Beispiel 2:

```
set ird gps2 tee pipe1
ftp put -t 10 ! pipe1
```

Die UART-Schnittstelle gps2 wird mit einem T-Stück (Verzweigung, Duplizierung) zusätzlich nach pipe1 umgeleitet. Es wird ein Echtzeit-FTP-Transfer zum Default-Server (siehe upload.cfg - s. S. 39) gestartet. Auf dem Server wird das File ‚**pipe1**' angelegt'. Es ist zu beachten, dass in der Kurzform kein Zielfilename spezifiziert werden kann und je nach FTP-Serverkonfiguration das alte File entweder überschrieben, der Transfer vom Server abgelehnt oder die Daten vom Server unter einem generischen Filenamen (evtl. pipe1.001) gespeichert wird.

Die Alternative zu dieser Methode ist natürlich eine dedizierte TCP-Socketverbindung. Siehe hierzu das Kommando ‚**tcp ...**' S. 296.

Befehle im Detail

ftp	FTP-Filetransfer über das GSM-Modem

```
{{@|!}[<infofile>]|<usr>:<pwd>{@|!}<server>[/[path/][<file>]]} [<local_file>]
```

Die Angaben zu FTP-Serveradresse, User und Passwort werden per Default im File **$(syspath)upload.cfg** (**$(syspath)** enthält per Default **c:/sys/**) gespeichert. Wird diese Variante gewählt, dann reicht es, dies mit

! oder **@**

in der Kommandozeile anzugeben.

Beispiel:

```
ftp put ! c:/data/File_xyz
```

folgt unmittelbar hinter ‚**!**' oder ‚**@**' ein Dateiname, dann werden die Angaben aus diesem File gelesen. Das ermöglicht die Verwendung von alternativen FTP-Servern bzw. Benutzerlogins.

Die Langform der Server- und Login-Daten

```
<usr>:<pwd>{@|!}<server>[/[path/][<file>]]
```

ermöglicht die Angaben auf der Kommandozeile insbesondere wenn ein spezielles Verzeichnis auf dem Server angesprochen werden soll oder das Zielfile einen anderen Namen erhalten soll.

Diese Angabe treffen sinngemäß auch auf ‚**ftp get ...**' zu , nur dass hier **<local_file>** das Zielfile ist.

```
-a|-i|-d
```

Einer dieser optionalen Schalter kann angegeben werden, um dem FTP-Server die beabsichtigte Übertragung anzuzeigen.

Dabei steht **-a** für ASCII-Transfer, **-i** für einen beliebigen binären Transfer und **-d** für die Übermittlung von Verzeichnisinformationen.

Der Defaultwert für alle Übertragungen ist **-i**.

Befehle im Detail

gps	Steuerung der GPS-Funktionen

Mit dem gps-Kommando wird das GPS-Board oder, wenn installiert, zwei GPS-Boards gesteuert. Es können GPS-Boards der Hersteller Novatel und Trimble installiert werden. Mit diesem Kommando wird nach dem Einschalten die Datenaufzeichnung bzw. Datenweiterleitung eingestellt. Dazu wird nach dem Power on bzw. Reset, falls in der **autoexec.sh** das Kommando ‚**gps init**' aufgeführt ist, der gps-Task gestartet. Dieser schaltet die Stromversorgung für das GPS-Board ein und er sucht im Systempfad nach der Datei **gps.cfg**. Diese enthält die anwenderspezifischen Initialisierungen des GPS-Boards (Ausgabe welcher Datensätze, mit welcher Datenrate und an welchem Port).
Nach der Initialisierung wird je nach GPS-Board und Empfangsbedingung im Bereich von einigen Sekunden die erste Positionsberechnung (first fix) erfolgen. Ab diesem Moment wird das Skript **firstfix.sh** im Systempfad gesucht und ausgeführt. Es erfolgt das Stellen der Systemzeit und der batteriegestützten Echtzeituhr.

Parameter:

```
 put    [-d  <delay_ms>] {<data_to_gps_board> | < <file_to_gps_board>}  -d delay after transmitted lines
| put     -d [<delay_ms>]                    -d set or display global delay after every transmitted line
| init [-d [-f] <delay_ms> ] [-gps1-rx <buf_size>] [-gps2-rx <buf_size>] [-nogpspwr] [-noantpwr] [-i] [-l|-satvis]
         -d          set delay between lines (init. strings) to be transmitted
         -f          all values for <delay_ms> are possible, else at least 200ms or 1 s, 2 s, 3 s, 4 s, 5 s
         -gps1-rx    set rx buffer size for gps1 to <buf_size>, default 3000, min. 1000
         -gps2-rx    set rx buffer size for gps2 to <buf_size>, default 10000, min. 1000
         -nogpspwr   don't switch on power for gps board, additional implicit 'gps timeout 0'
         -noantpwr   don't switch on antenna power, additional implicit 'gps timeout 0'
         -i          omit gps boards interface initialisation
         -l          omit all other implicit logs
         - satvis    activate satellite visible log up to first fix
| timeout   [<tmot_ms>]                  reinit gps after # ms task timeout, 0 -> off
| bd        [{gps1|gps2} [<baud>]]
| set       filename-pattern [{UTC_LONG|UTC_ASC|UTC_HEX|ASH_ATM|2LETTER_UTC [<Letter1><Letter2>]}]
| set       fname                        creates a new filename according time and stores it in $(fname)
| bestpos
| log       [on [-nc] [-hdr[s]] [-ftr] [-append] [-container [<size>]] [<file>] [-dont-cp-cfg]]
| log       [off [<cmd_line>]]       in <cmd_line> the substring $(fname) will be replaced by the filename
| gpgga-tx  {eth|gsm} [[tx-flg] [on|off] | file [[-a] <file>|close] | [cycle <sec> | next <sec> | once]]
| echo      [from|to] [on|off|esc]
| debug     {clr|set} <msk>   bit  0 (     1) print bestposb converted to ascii comming from gps1
                         bit  1 (     2) print bestposb converted to hex   comming from gps1
                         bit  2 (     4) print bestposb converted to ascii comming from gps2
                         bit  3 (     8) print bestposb converted to hex   comming from gps2
                         bit  4 (  0x10) print satvis2 when initialized
                         bit  5 (  0x20) print bestsats when initialized
                         bit  6 (  0x40) print json data when connected to WebSocket
                         bit  7 (  0x80) parse all bin messages from gps2 too (CAN DBC)
                         bit  8 ( 0x100) print header of all bin messages on gps1
                         bit  9 ( 0x200) print header of all bin messages on gps2, bit 7 must be set too!
                         bit 10 ( 0x400) print header of all unhandled bin messages on gps1
                         bit 11 ( 0x800) print header of all unhandled bin messages on gps2
                         bit 12 (0x1000) don't filter CAN DBC variables AXB, AYB, AZB, RXB, RYB and RZB
| flush     [cycle <flush_cyc_sec> | next <time_sec> | on | off | once]
| stat      [gps1|gps2|both]
| blocksize [<size>|LF|CR]
| reply     [gps1|gps2] [{bin|asc|nmea|all} [[-c] <file>|off]]   -c create else append (nmea only gps2)
| reply     [gps1|gps2] ppmpos              [[-c] <file>|off]]   bestpos to ppmpos message
| nmea-scan [gps1||gps2|both|off]                                 set where NMEA sentences are scaned
| nmea-scan  gps2 {1st-letter|2nd-letter} [<letters>|*]    set/check 1st/2nd letter to be scaned, * -> all
| setrtc                                 set RTC and sys time with UTC retrieved from $GPRMC
| adapter   gps2-switch [COM3|COM2]      setting COM3 means also connect COM2 to extern comc interface
          | pps-switch   [comc-tx|off]   comc-tx means also disconnect COM2 tx from comc tx
          | COM2-invert [on|off]         COM2 tx switch to inverted or noninverted
| -c        <cmd>
```

Befehle im Detail

gps	Steuerung der GPS-Funktionen

```
gps put    [-d  <delay_ms>] {<data_to_gps_board> | < <file_to_gps_board>}
```

Ein ASCII-Kommando zum GPS-Board senden. Davor wird implizit für 2 Sekunden ‚**gps echo on**' geschaltet, um die Antwort des GPS-Boards sehen zu können.
Das Kommando dient hauptsächlich zum manuellen Testen von Konfigurationen des GPS-Boards. Wenn die passenden gefunden sind, dann können diese in die ‚**gps.cfg**' aufgenommen werden. Es werden alle Zeichen vom ersten Nichtleerzeichen nach dem ‚**put**' bis zum Ende der Eingabezeile, also auch eventuelle Leerzeichen, und CR+LF übertragen.

Der Schalter -d <delay_ms> bedeutet dass ein Delay von <delay_ms> nach jeder Zeile gewartet wird.

Wird anstelle von ‚**<data_to_gps_board>**' ‚**< <file_to_gps_board>**' angegeben, so wird der Inhalt eines Text-Konfigurationsfiles zeilenweise an das GPS-Board übertragen. Die Zeilen werden einschließlich eventuell führender und nachfolgender Leerzeichen übertragen. Davor wird implizit ‚**gps echo on**' geschaltet, um die Antworten des GPS-Boards sehen zu können.
Zeilen in dem Text-Konfigurationsfile, die mit dem Schlüsselwort ‚**exec**' beginnen (anführende Leerzeichen und TAB werden überlesen) , werden nicht übertragen. Sie werden dem Shellinterpreter zur Ausführung übergeben. Damit ist es möglich, beim Einlesen und Übertragen von GPS-Einstellungen gleichzeitig Shellkommandos auszuführen. Das könnten z. B. Hinweise ‚@echo Konfigurationsfile xyz gelesen' sein.

```
gps put -d    [<delay_ms>]
```

Mit diesem Unterkommando kann ein genereller Delay beim Senden von Zeilen mit ‚**gps put**' eingestellt werden. Wird ‚**<delay_ms>**' weggelassen, also die Angabe über die Dauer des Delays in ms, so wird der momentan eingestellte Delay angezeigt.

```
gps init [-d [-f] <delay_ms> ] [-gps1-rx <buf_size>] [-gps2-rx <buf_size>] [-nogpspwr] [-noantpwr][-i][-l|-satvis]
```

Mit diesem Unterkommando wird das GPS-Board initialisiert. Es wird in der Regl in der autoexec.sh aufgeführt. Die Initialisierung beginnt mit dem Einschalten der Stromversorgung für das GPS-Board und dem Einschalten der Antennenspeisung. Danach wird die ‚gps.cfg' eingelesen und an das GPS-Board übertragen.
Das Kommando wartet max. 45 s auf die Beendigung der Initialisierung bevor es zurückkehrt und folgende Kommandos ausgeführt werden können. Es wird eine GPS-Timeoutüberwachung aktiviert. Diese überwacht die Datenausgabe des GPS-Boards. Wenn länger als 3 s (einstellbar mit ‚**gps timeout <tmot_ms>**') keine Daten vom GPS-Board (gps1 und gps2) empfangen werden, dann wird das GPS-Board rückgesetzt und die Initialisierung erneut durchgeführt. Dies sichert das Wiederanlaufen im Fehlerfall.

Mit ‚**-d <delay_ms>**' kann ein Delay bei der Initialisierung eingestellt werden.

Das GPS-Board wird in der Regel so initialisiert, dass die relevanten Daten an der gps2-Schnittselle ausgegeben werden. Diese werden vom Controller empfangen, weitergeleitet und bei Bedarf gespeichert (Logger-Funktion).
Mit ‚**-gps2-rx <buf_size>**' bzw. ‚**-gps1-rx <buf_size>**' ist dabei die Größe des Empfangspuffers einstellbar. Sie richtet sich nach der Anzahl der verschiedenen Datensätze und deren Ausgaberaten. Insbesondere beim Logging der Daten auf SD-Karte muss bei höheren Datenrate (20 Hz und mehr) und mehreren Datensätzen der Standardpuffer (gps1: 3000 Byte, gps2 10000 Byte) vergrößert werden. Bei Datenraten von 50 Hz und darüber wird eine Puffergröße von 16 kByte oder mehr empfohlen.
Der optionale Schalter ‚**-nogpspwr**' gibt an, dass die Stromversorgung zum GPS-Board nicht eingeschaltet wird, ‚**-noantpwr**' dass die Stromversorgung zur Antennenspeisung nicht eingeschaltet wird, sie werden in der aktuellen Einstellung belassen. Siehe f.

Befehle im Detail

gps	Steuerung der GPS-Funktionen

"`gps init`" f.:

Der optionale Schalter ‚**-i**' gibt an, dass die implizite Initialisierung nicht durchgeführt wird. Bei Normalanwendungen darf er nicht angegeben werden.

Der optionale Schalter ‚**-l**' gibt an, dass alle ansonsten implizit aktivierten Logs vom GPS-Board ausgelassen werden. Dies betrifft vor allem die Datensätze $GPGGA und $GPRMC, welche für die Überwachung des Positionsstatus benötigt werden (Steuerung der LEDs, Zeit, Datum).

Die grundlegende Initialisierung der COM-Ports für Novatel-GPS-Boards lautet:

```
serialconfig com1 460800 n 8 1 n on
serialconfig com2 460800 n 8 1 n on
```

Mit dem optionalen Schalter '**-satvis**' kann der NovAtel SATVIS Log aktiviert werden. Dieser enthält Informationen zu den sichtbaren Satelliten.

```
gps timeout [<tmot_ms>]              reinit gps after # ms task timout, 0 -> off
```

Zur Sicherung einer hohen Zuverlässigkeit der Gerätefunktion wird die Datenübertragung von GPS-Board zum Controller überwacht. Das Kriterium für diesen Test ist das Empfangen und Verarbeiten von Daten durch den GPS-Task, der die Datenübertragung auf den Schnittstellen gps1 oder gps2 kontrolliert. Wenn diese Datenverarbeitung (Empfang, Speichern und/oder Weitersenden) länger als die mit diesem Unterkommando gegebene Zeit ausfällt, dann wird die Kommandozeile, die zur Initialisierung des GPS-Boards ausgeführt wurde, erneut ausgeführt.

Das Standardtimeout ist 3 s.

Mit der Eingabe von ‚**gps timeout 0**' wird die Überwachung abgeschaltet.

```
gps bd [{gps1|gps2} <baud>]
```

Mit diesem Unterkommando kann die Baudrate der Controllerschnittstellen **gps1** und **gps2** eingestellt werden. Es muss beachtet werden, dass die jeweilige UART des GPS-Boards **vor** der Veränderung der Baudrate von **gps1** und/oder **gps2** umgestellt werden muss. Insbesondere ist hierbei die Übertragung des GPS-spezifischen Baudratenbefehls abzuwarten (z. B. 100 ms mit Kommando ‚**@wait -s 0.1**') bevor ‚**gps bd ...**' aufgerufen wird.

Ohne Schnittstellen- und Baudratenangabe wird die aktuelle Einstellung ausgegeben.

Die Baudrateneinstellung wird intern an das Kommando ‚**bd gps1 ...**' oder ‚**bd gps2 ...**' weitergeleitet. Weitere Optionen sind beim Kommando ‚**bd**' erklärt.

Befehle im Detail

gps	Steuerung der GPS-Funktionen

gps set filename-pattern [{UTC_LONG|UTC_ASC|UTC_HEX|ASH_ATM|2LETTER_UTC [<Letter1><Letter2>]}]

Wird eine GPS-Datenspeicherung standardmäßig gestartet, dann wird ein File im FAT-Dateisystem angelegt. Der Name des Datenfiles wird nach dem mit diesem Kommando einstellbaren Muster gebildet.

In ihm ist die aktuelle Systemzeit in unterschiedlicher Präzision enthalten. Es ist die Systemzeit im Moment der Erzeugung des Dateinamens.

Ohne Musterparameter wird die aktuelle Einstellung ausgegeben.

Mit ‚**@echo $(fname)**' kann der Name des aktuellen GPS-Datenfiles mit Speicherpfad ausgegeben werden.

Die Filenamenerweiterung ist bei den Mustern **UTC_LONG**, **UTC_ASC** und **UTC_HEX** gleich. In den Beispielen ist sie ‚.**9F6**'. Diese 3 Zeichen (Buchstaben und Ziffern) werden aus der Seriennummer des GPS-Boards abgeleitet. Sie sind also eine unverwechselbare Kennung des jeweiligen GPS-Boards.

Bei den Mustern **ASH_ATM** und **2LETTER_UTC** sind Teile der Systemzeit darin kodiert.

Es folgen Beispiele für die einzelnen Muster.

UTC_LONG (Standard)

log-170228-132610-107346.9F6

Das ist das Standardfilenamenmuster. Die ersten 4 Zeichen sind immer ‚**log-**'. Sie stehen für ‚Logfile'. Danach folgt das Datum im Format **yymmdd**. Es folgt ein Trennzeichen ‚**-**' und darauf folgt die Uhrzeit im Format **hhmmss**. Es folgt wieder ein Trennzeichen ‚**-**'. Die folgenden 6 Ziffern sind der gebrochene Anteil der Sekunden, also Mikrosekunden.

UTC_ASC

72SD2624.9F6

Das erste Zeichen ist der Einer der Jahreszahl. Nach 2019 wird mit ‚A', ‚B', ‚C' usw. weitergezählt.
Das zweite Zeichen ist der Einer des Monats. Nach dem September wird mit ‚A', ‚B', bis ‚C' weitergezählt.
Das dritte Zeichen ist der Einer des Tages. Nach dem 9. wird mit ‚A', ‚B', bis ‚V' weitergezählt.
Das vierte Zeichen ist der Einer der Stunde. Nach dem 9 wird mit ‚A', ‚B', bis ‚N' weitergezählt.
Es folgen zwei Stellen für die Minute und zwei Stellen für die Sekunde.

Siehe f.

Befehle im Detail

gps	Steuerung der GPS-Funktionen

"`gps set filename-pattern`" f.:

```
UTC_HEX

58b57a8a.9F6
```

Die acht Stellen sind die hexadezimale Darstellung der Systemzeit in Sekunden im UNIX-Zeitformat. Siehe dazu auch Zeitumrechnung mit ‚**date -c ...**' S. 120.

```
ASH_ATM

G1326O17.059
```

Das Filenamensformat passend für den Ashtech Atom-Konverter to Rinex.

```
G<let_num_4><sngl_letter><year_2>.<day_of_year>
<let_num_4>  := 4 Stellen 0...9 od. A...Z   hier: Zeit hhmm
<sngl_letter>:= A...Z                       hier: Sekunden im Raster von 3 s
<day_of_year>:=                             siehe ‚date -jd'
```

```
2LETTER_UTC [<Letter1><Letter2>]

FP281330.72d      (ohne Letterangabe)
rt281327.72d      (mit Letterangabe ‚rt')
```

Die ersten beiden Zeichen sind wählbar (Default ‚**FP**').

Das dritte und vierte Zeichen ist der Monatstag.

Das fünfte und sechste Zeichen ist die Stunde (24 h Zählung)

Das sechste Zeichen ist der Einer des Jahres modulo 10, d. h. 2017 → 7 und 2020 → 0

Das achte Zeichen ist der Einer des Monats. Nach dem September wird mit ‚A', ‚B', bis ‚C' weitergezählt.

Befehle im Detail

gps	Steuerung der GPS-Funktionen

```
gps set fname
```

Dieses Unterkommando erzeugt einen neuen Dateinamen und speichert ihn in **$(fname)**. Die darin gespeicherte Systemzeit ist die im Moment der Erzeugung des Dateinamens. Mit **$(fname)** kann der Dateiname dann beliebig weiterverwendet werden. Z. B. Öffnen eines Files im Jahres-, Monats- und Tagesverzeichnis des Unterverzeichnis ‚**c:/data**'.

Beispiel:

```
gps log on
gps set fname
set ird gps1 tee $(fname)
```

Mit ‚**gps log on**' wird ein Logfile angelegt und alle über **gps2** ankommenden Daten werden dort hin gespeichert.
Mit ‚**gps set fname**' wird eine neuer Name erzeugt (es wird noch **kein** File angelegt oder geöffnet).
Mit ‚**set ird gps1 tee $(fname)**' werden die über **gps1** ankommenden Daten in ein zweites Logfile gespeichert, das im selben Pfad liegt wie das von ‚**gps log on**'.

Im Gegensatz zu **$(tm)**, das nur einen Textstring ohne Pfad und Extension liefert, enthält **$(fname)** eine vollständige Dateiangabe, also mit Pfad und Name und Extension. Die Extension, also der 3-Buchstabencode des GPS-Boards kann mit **$(fn_ext)** ermittelt bzw. verwendet werden.

```
gps bestpos
```

Mit diesem Kommando kann eine Aktion des GPS-Board getriggert werden. In der Regel ist es die Extraberechnung einer Position genau zum Zeitpunkt des Aufrufes von ‚**gps bestpos**'. Es wird die Datei ‚c:/sys/bestpos.cfg' gesucht und deren Inhalt wird an das GPS-Board übertragen. Dabei werden Zeilen, die mit ‚**exec**' beginnen nicht übertragen, sondern vom Shell-Interpreter ausgeführt.

Befehle im Detail

gps	Steuerung der GPS-Funktionen

```
gps log [on [-nc] [-hdr[s]] [-ftr] [-append] [-container [<size>]] [<file>] [-dont-cp-cfg]]
```

Mit diesem Kommando wird die Speicherung von Daten, die über **gps2** hereinkommen, gestartet.

Falls bereits ein Logfile geöffnet ist, wird es geschlossen bevor das neue geöffnet wird. Wird **<file>** nicht angegeben, was der Standardfall ist, dann erzeugt das ppmOS einen Speicherpfad in c:/data. Es werden Verzeichnisse für das Jahr, den Monat und den Tag angelegt, falls diese noch nicht existieren.

Mit Angabe des optionalen Schalters ‚**-nc**' (not clear) ist gesichert, dass keine Daten beim Übergang vom alten zum neuen Logfile verloren gehen. Ohne ‚**-nc**' wird beim ‚**gps log on**' der Empfangspuffer gelöscht.

Nur ‚**gps log**' gibt den aktuellen Zustand aus:

```
GPS:  1802430 bytes written to data file c:/DATA/2017/02/28/log-170228-142332-835653.9F6
```

Das Kommando kehrt mit dem Wert 0 zurück, wenn ein Logfile offen ist und mit einem Wert ungleich 0, wenn kein Logfile offen ist. Der Rückkehrwert kann in der Shell-Programmierung genutzt werden, um bedingte Shell-Kommandos auszuführen:

```
gps log && @echo log file ist open
```

oder

```
gps log || gps log on
```

Letzteres Kommando öffnet ein Logfile nur, falls es noch nicht offen ist. Einfach nur ‚**gps log on**' würde auch ein Logfile öffnen, aber falls schon eins offen ist, würde dies zuerst geschlossen und dann würde das neue angelegt werden. Man hätte also 2 Logfiles.

```
-hdr
```

Der optionale Schalter ‚**-hdr**' erzeugt den PPM-Statusheader Typ 2. Es ist ein ASCII-Dateiheader im NMEA-Format.

```
$PPMS2,DLF,DMMU18260115L,OM7MR0800RN0000,1.46,utc,21.08.16,10:26:49,2171, 124027.134015,63,4,33,,,,11.709,RST,0x0a,0,0,445,610,3545,4501,3545,227,0,0,430,280,2386*0F
```

Dieser Datensatz gibt in komprimierter Form Auskunft über das verbaute GPS-Board, Softwareversionen und den ‚Gesundheits-' und Betriebszustand des Gerätes.

Darin bedeuten: Siehe f.

Befehle im Detail

gps	Steuerung der GPS-Funktionen

"`gps log`" f.:

Feld	Bedeutung
`$PPMS2`	Kennung, PPM-Statusdatensatz Typ 2
`DLF`	Aus der Seriennummer abgeleitete 3-Zeichen-Signatur, die auch als Dateinamensextension verwendet wird.
`DMMU18260115L`	Seriennummer des GPS-Boards
`OM7MR0800RN0000`	Firmwarecode des GPS-Boards
`1.46`	Version der 40xx-Firmware
`utc`	Zeitbasis der folgenden Zeitangabe (gps oder utc)
`21.08.16`	Datum der Erzeugung dieses Datensatzes im Format yy.mm.dd
`10:26:49`	Uhrzeit der Erzeugung dieses Datensatzes im Format hh:mm:ss
`2171`	GPS-Woche
`124027.134015`	GPS-Wochensekunden
`63`	Uptime in s (Zeit seit Power on oder Reset)
`4`	Positionsfixindikator aus dem GPGGA-Datensatz
`33`	Zahl der genutzten Satelliten aus dem GPGGA
`,,,,`	Powerstatusflags (bis einschließlich V 1.46 nicht implementiert)
`11.709`	Stromversorgungseingangsspannung in V
`RST`	Resetursache als 3-Buchstabenkürzel, hier ‚power on reset'
`0x0a`	Das oberste Byte (Bit 31...24) im ‚Reset Control & Status'-Register der RCC (Reset und Clock Control-Unit) des Microcontrollers, welches die Reset-Ursache anzeigt. Hier sind die Bits ‚software reset' und ‚reset pin reset' gesetzt, was bedeutet, dass ein ‚Softreset' mit dem Shellkommando ‚**Reset**' ausgelöst wurde, der chipintern das NRST-Pin aktiviert. Dies ist also ein softwaregetriggerter Hardwarereset.
	Es folgen die Zähler für die verschiedenen Resetursachen. Sie werden im batteriegestützten RAM nichtflüchtig, also ohne externe Stromversorgung, gespeichert.
`0`	LPW, Low-power Reset
`0`	WDG, Window watchdog Reset (nicht aktiviert)
`445`	IWD, Independent watchdog Reset (aktiviert und einstellbar über ‚**stat -wdt ...**')
`610`	SWR, Software Reset (in dem Fall wird auch immer EXR hochgezählt)
`3545`	POR, Power on / Power down Reset (bei Einschaltung, nachdem das Gerät nach einer Stromunterbrechung ca. länger als 90 s aus war)

Befehle im Detail

gps	Steuerung der GPS-Funktionen

"`gps log`" - Tabelle zu PPM-Statusheader Typ 2 - f.:

`4501`	EXR, Reset Pin Reset (wird auch beim Software-Reset SWR hochgezählt)
`3545`	BOR, Brown out Reset (Unterspannungs-Reset), nach Spannungsunterbrechung erfolgt Neustart über PON
`227`	RST, Software Reset, ausgelöst durch Kommando ‚**Reset**'
`0`	JMP, Software Reset, ausgelöst durch Kommando ‚**Reset -j**' (jump)
`0`	WDT, Software Reset, ausgelöst durch Kommando ‚**Reset -w**' (wait for wdt-Reset)
`430`	CSH, Software Crash nach der Bootphase abgefangen und Neustart durch den ‚Hard Fault Handler'
`280`	CSB, Software Crash in der Bootphase abgefangen und Neustart durch den ‚Hard Fault Handler'
`2386`	PON, Power on Reset, Aufwachen aus ‚Unterspannungsschlafzustand' oder bei Einschaltung nach einem Ausschalten via On/Off-Taste.
`*0F`	Prüfsumme nach NMEA-Standard (XOR-Verknüpfung aller Zeichen nach dem ‚$' bis vor dem ‚*')

Befehle im Detail

gps	Steuerung der GPS-Funktionen

`-hdrs`

Der optionale Schalter ‚**-hdrs**' erzeugt den PPM-Statusheader Typ 0. Es ist einen ASCII-Dateiheader im NMEA-Format.

```
$PPMS0,utc,21.08.19,15:20:53,1629386453.344950 s,18,2171, 400871.344950,3039.875038 s,POR,0x07,0,0,445,619,3561,4526,3561,227,0,0,440,286,2394*5B
```

Feld	Bedeutung
`$PPMS0`	Kennung, PPM-Statusdatensatz Typ 0
`utc`	nachfolgendes Datum und Uhrzeit sind UTC. Alternativ könnte dort stehen **gps**, falls die Systemzeit auf GPS-Zeit eingestellt wurde.
`21.08.19`	Datum der Erzeugung dieses Datensatzes im Format yy.mm.dd
`15:20:53`	Uhrzeit der Erzeugung dieses Datensatzes im Format hh:mm:ss
`1629386453.344950 s`	Zeitangabe im UNIX-Format (Sekunden seit dem 1.1.1970 0:0:0) hier zusätzlich mit gebrochenem Anteil der Sekunden.
`18`	falls Die Zeitangabe GPS-Zeit ist, dann wird hier der verwendete GPS-UTC-Zeitversatzes angegeben. Er repräsentiert die aktuelle Zeitdifferenz GPS-Zeit minus UTC-Zeit.
`2171`	GPS-Woche
`400871.344950`	GPS-Wochensekunden
`3039.875038 s`	Uptime in s (Zeit seit Power on oder Reset) mit gebrochenem Anteil der Sekunden.
`POR`	Reset-Ursache als 3-Buchstabenkürzel, hier ‚power on reset'
`0x07`	Das oberste Byte (Bit 31...24) im ‚Reset Control & Status'-Register der RCC (Reset und Clock Control-Unit) des Microcontrollers, welches die Resetursache anzeigt. Hier sind die Bits ‚software reset' und ‚reset pin reset' gesetzt, was bedeutet, dass ein ‚Softreset' mit dem Shellkommando ‚**Reset**' ausgelöst wurde, der chipintern das NRST-Pin aktiviert. Dies ist also ein softwaregetriggerter Hardware-Reset.
	Es folgen die Zähler für die verschiedenen Resetursachen. Sie werden im batteriegestützten RAM nichtflüchtig, also ohne externe Stromversorgung, gespeichert.
`0`	LPW, Low-power Reset
`0`	WDG, Window watchdog Reset (nicht aktiviert)
`445`	IWD, Independent watchdog Reset (aktiviert und einstellbar über ‚**stat -wdt ...**')
`619`	SWR, Software Reset (in dem fall wird auch immer EXR hochgezählt)
`3561`	POR, Power on / Power down Reset (bei Einschaltung, nachdem das Gerät nach einer Stromunterbrechung ca. länger als 90 s aus war)
`4526`	EXR, Reset Pin Reset (wird auch beim Softwarereset SWR hochgezählt)
`3561`	BOR, Brown out Reset (Unterspannungsreset), nach Spannungsunterbrechung erfolgt Neustart über PON

Siehe f.

Befehle im Detail

gps	Steuerung der GPS-Funktionen

"`-hdrs`" — PPM-Statusheader Typ 0 — f.:

Feld	Bedeutung
`227`	RST, Software Reset, ausgelöst durch Kommando ‚**Reset**'
`0`	JMP, Software Reset, ausgelöst durch Kommando ‚**Reset -j**' (jump)
`0`	WDT, Software Reset, ausgelöst durch Kommando ‚**Reset -w**' (wait for wdt-Reset)
`440`	CSH, Software Crash nach der Bootphase abgefangen und Neustart durch den ‚Hard Fault Handler'
`286`	CSB, Software Crash in der Bootphase abgefangen und Neustart durch den ‚Hard Fault Handler'
`2394`	PON, Power on Reset, Aufwachen aus ‚Unterspannungsschlafzustand' oder bei Einschaltung nach einem Ausschalten via On/Off-Taste.
`*5B`	Prüfsumme nach NMEA-Standard (XOR-Verknüpfung aller Zeichen nach dem ‚$' bis vor dem ‚*')

`-ftr`

Der optionale Schalter ‚**-ftr**' erzeugt den Datensatz PPMS1 / PPM-Statusfooter (Footer - Gegenteil von Header) vom Typ 1. Es ist einen ASCII-Dateiheader im NMEA-Format.

Er steht als letzter Datensatz im Logfile.

Beispiel:

```
$PPMS1,      3227*07
```

Er enthält die Länge des Logfiles als dezimale Zahl (hier **3227** Byte) so, wie sie auch im Directory-Eintrag angegeben ist. Der Datensatz wird abgeschlossen mit der NMEA-Prüfsumme ‚***07**'.

Der Datensatz wird beim regulären Schließen des Logfiles als letztes angehängt. Aus der Anwesenheit und bei Übereinstimmung mit der im Directory angezeigten Filelänge kann auf eine ordnungsgemäße Speicherung des Logfiles geschlossen werden.

```
gps log on ...[-append] ... <file>
```

Mit diesem Unterkommando werden die Daten in ein explizit vorgegebenes File **<file>** gespeichert. Es können alle Filetypen des **ppmOS** angegeben werden. Mit ‚**-append**' werden die neuen Daten am Ende an das File **<file>** angehängt.

Befehle im Detail

gps	Steuerung der GPS-Funktionen

```
gps log on ... [-append] [-container [<size>]] [<file>] [-dont-cp-cfg]]
```

Mit dem Zusatz **-container [<size>]** wird ein spezielles Containerfile erzeugt, wenn es im FAT-Dateisystem gespeichert wird (Standardpfad **c:/data/<Jahr>/<Monat>/<Tag>** ohne **<file>**). Wird **<size>** nicht angegeben, dann werden hierfür 16 MByte angenommen.

Das Containerfile hat die Eigenschaft, dass das File nicht größer als die Containergröße wird. Zu Beginn hat das Containerfile eine Größe von 512 Byte. Erst beim Speichern wächst die Größe auf die vorgegebene Maximalgröße an. Danach bleibt die Größe konstant. Wird das Ende des Containerfiles erreicht, dann wird die Speicherung am Anfang fortgesetzt. Es handelt sich bei diesem File also um einen zirkulares File (auch Ringspeicher), bei dem neue Daten die ältesten überschreiben.

Die Angaben über den aktuellen Schreib- bzw. Lesezeiger (logischer Fileanfang), die maximale Containergröße und den Loopcounter (zählt wie oft das Containerfile schon vollgeschrieben wurde) werden in einem speziellen Dateiheader (512 Byte), der vor den eigentlichen Daten liegt (am physischen Fileanfang), zusammen mit einer spezieller Filekennung gespeichert. Das **ppmOS** erkennt beim Öffnen diesen Filetyp und realisiert die Filezugriffe entsprechend. D. h. der spezielle Dateiheader ist für den Nutzer transparent.

Wird das Containerfile per USB-Stickmode auf den PC übertragen, dann hat Windows, Linux, bzw. das MacOS zunächst keine Kenntnis von diesem Filetyp. Der Zugriff muss dann über eine Software erfolgen, die in der Lage ist, den Fileheader zu lesen.

-dont-cp-cfg
Wird mit **gps log on** ein GPS-Logfile gestartet wird zusätzlich zu dieser Datei stets eine Kopie der gps.cfg im Verzeichnis des GPS-Logfiles abgelegt. Es hat den selben Namen wie das GPS-Logfile und das Suffix *.cfg. Damit kann grob nachvollzogen werden mit welcher Konfiguration des GPS-Boards die Daten im GPS-Logfile entstanden sind.
Wird der Schalter **-dont-cp-cfg** angegeben, also **gps log on -dont-cp-cfg**, so wird keine Kopie der gps.cfg erstellt.

```
gps log [off [<cmd_line>]]    //in <cmd_line> the substring $(fname) will be replaced by the filename
```

Mit diesem Unterkommando wird der aktuelle Zustand der Datenspeicherung angezeigt, wenn hinter ‚**log**' nichts folgt.

Beispiel:

```
GPS:  1824676 bytes written to data file c:/DATA/2017/02/28/log-170228-142332-835653.9F6
```

Mit ‚**gps log off**' wird noch 400 ms gewartet und dann wird ein eventuelles Logfile geschlossen.

Das Warten hat den Grund, dass, falls z. B. mit einem Cron-Job ‚**log off**' gegeben wird und Datensätze gespeichert werden, die im Sekundentakt kommen, diese entsprechend der gewählten Baudrate immer erst ein paar Millisekunden nach dem Sekundentick ankommen, diese noch im Datenfile gespeichert werden.

Optional kann ein Kommando mit übergeben werden, dass nach dem Schließen des Logfiles via Systemmessage an den Shell-Interpreter coma übergeben wird. Der Unterschied zu ‚**gps log off; <cmd_line>**, also mit ‚;' dazwischen besteht darin, dass mit ‚;' ‚**log off**' und **<cmd_line>** von der Shell ausgeführt werden, von der aus ‚gps init' erfolgte. Ohne ‚;' wird das Kommando **<cmd_line>** an den Shell-Interpreter von coma gesendet.

Befehle im Detail

gps	Steuerung der GPS-Funktionen

```
gps gpgga-tx {eth|gsm} [[tx-flg] [on|off] | file [[-a] <file>|close] | [cycle <sec> | next <sec> | once]]
```

Wenn das GPS-Board Korrekturdaten von einem NTRIP-Server erhält, dann ist es möglich die Korrekturdaten zu verbessern, wenn dem NTRIP-Server die aktuelle Position mitgeteilt wird. Dieser Service ist nicht bei jedem NTRIP-Server (Mountpoint) verfügbar. Die aktuelle Position wird durch periodische Übermittlung des $GPGGA-Datensatzes mitgeteilt.

Im ppmOS können diese Korrekturdaten über eine GSM-Verbindung oder über Ethernet bezogen werden.

In der Defaulteinstellung wird der $GPGGA-Datensatz alle 20 s gesendet.

Einstellungen für eine NTRIP-GSM-Verbindung werden mit dem Schalter ‚**gsm**' und für eine NTRIP-Ethernet-Verbindung mit dem Schalter ‚**eth**' vorgenommen.

Mit ‚**gps gpgga-tx {eth|gsm} tx-flg off**' kann das Senden des Datensatzes abgeschaltet werden.

Mit ‚**gps gpgga-tx {eth|gsm} cycle 10**' wird das Sendeintervall auf z. B. 10 s eingestellt. Der neue Zyklus wird erst nach dem nächsten Sendezeitpunkt wirksam. Deshalb sollte auch der ‚**next**'-Parameter mit ‚**gps gpgga-tx {eth|gsm} next 0**' eingestellt werden, falls vorher ein langer Zyklus eingestellt war.

Mit ‚**gps gpgga-tx {eth|gsm} next 10**' wird der eingestellte Sendezyklus in z. B. 10 s gestartet. Mit dem Wert 0 wird er sofort gestartet.

Mit ‚**gps gpgga-tx {eth|gsm} once**' wird der $GPGGA-Datensatz einmalig gesendet und gleichzeitig das periodische Senden abgeschaltet.

Soll bei einer GSM-Verbindung das File, aus dem die GPGGA-Datensätze gelesen werden, geändert werden, dann ist die Einstellung mit ‚**ntrip -tx ...**' vorzunehmen.

Mit ‚**gps gpgga-tx eth file [-a] <file>**' wird bei einer NTRIP-Verbindung über Ethernet das betreffende IP-Client-File **icom {1|2|3|4|5}** eingestellt (siehe dazu ‚**ip client ntrip ...**' S. 180). Damit kann der $GPGGA-Datensatz auch in ein beliebiges anderes File/Schnittstelle gesendet werden.

Mit dem Schalter ‚**-a**' wird angegeben, dass die Daten am Ende eines eventuell bestehenden Files angehängt werden. Es wird zusätzlich implizit ‚**gps gpgga-tx eth tx-flg on**' ausgeführt. Dieses Unterkommando ist geeignet, eine Konfiguration zu testen oder Fehler zu finden.

Mit ‚**gps gpgga-tx eth file close**' kann dieses Ausgabefile geschlossen werden. Es wird ebenfalls geschlossen, wenn ein neues File geöffnet wird.

Befehle im Detail

gps	Steuerung der GPS-Funktionen

```
gps echo [from|to] [on|off|esc]
```

Über die Schnittstelle **gps1** bezieht der Controller des 40xx periodisch alle 1 s den $GPRMC- und $GPGGA-Datensatz. Das dient dazu, ständig den Positionsstatus (Zeit, Anzahl Satelliten, Positionslösungstyp) unabhängig von der Datenkonfiguration an der **gps2**-Schnittstelle verfügbar zu haben (Steuerung der roten und grünen Status-LED).

Mit diesem Unterkommando kann die Kommunikation des Controllers mit dem GPS-Board über die Schnittstelle **gps1** auf der Schnittstelle **coma** angezeigt werden. Wird ‚**from**' oder ‚**to**' weggelassen, dann werden die Daten beider Datenrichtungen angezeigt.

Unter der Datenrichtung ‚**from**' sind alle Daten, die an der Schnittstelle **gps1** empfangen werden gemeint.

Unter der Datenrichtung ‚**to**' sind alle Daten gemeint, die über **gps1** an das GPS-Board gesendet werden. Das ist z. B. bei der Initialisierung der Fall.

Während der Initialisierung des GPS-Boards ist das Echo für beide Datenrichtungen eingeschaltet. Es wird am Ende der Initialisierung wieder abgeschaltet.

Zusätzlich wird bei der Aktivierung von ‚**to**' der periodische Zeitabgleich zwischen GPS-Zeit und Systemzeit angezeigt:

```
time sys utc 17.03.03 11:36:55 1488541015.041269 0 2943.336562
time gps utc 17.03.03 11:36:55 1488541015.041244 0 2943.336537
diff  tm  0.000025138 s  delta_diff  0.000000849 s  -trim  -1605
```

Alle 10 s erfolgt ein Zeitvergleich. Im Beispiel ist zu erkennen, dass die Systemzeit zur GPS-Zeit aktuell eine Differenz von ca. 25 µs hat und die Veränderung (Jitter) dieses Wertes 849 ns beträgt. Hinter ‚**-trim**' ist der aktuelle Trimmwert der Systemzeit angezeigt.

Das Trimmen wird vom GPS-Task erledigt. Ist die Abweichung größer als 100 µs, dann wird die Systemzeit nachgestellt. einige Minuten nach dem ersten Fix ist ein Zustand, wie oben im Beispiel gezeigt, erreicht.

gps echo esc

Mit **esc** („escape") erscheinen die Datensätze vom GNSS-Board, die an der Schnittstelle gps1 des Mikrocontrollers eingehen, auf einem Terminal, welches Escape-Sequenzen versteht, positionsfest. Damit soll jeder NMEA-Datensatztyp immer auf einer konstanten Zeile, beginnend ab Zeil 5, angezeigt werden.

Befehle im Detail

gps	Steuerung der GPS-Funktionen

```
gps debug {clr|set} <msk>
bit 0 RMC, 1 GGA, 2 bepo bi, 3 bepo hx, 4 satvis2, 5 bestsats, 6 json, 7 clr
```

An der Schnittstelle **gps2** werden die vom Anwender konfigurierten Datensätze des GPS-Boards empfangen und weiterverarbeitet. Zusätzlich werden an der Schnittstelle **gps1** für systeminterne Zwecke weitere Datensätze konfiguriert.

Man kann diese Daten grob in ASCII-Datensätze (dazu zählen die NMEA-Datensätze) und binäre Daten einteilen. Zur Kontrolle des Empfangs kann mit diesem Unterkommando eine Ausgabe aktiviert werden.

Der Wert hinter 'gps debug ...' ist ein 16-bittiges Flag-Wort, in dem je 1 Bit für eine spezielle Debug-Ausgabe zuständig ist.

Die Ausgabe erfolgt an der Schnittstelle, an der das ‚**gps init**' eingegeben wurde (Standard ist **coma**).

Z.B.: gps debug set 0x100

Bit 0 =1 (1)	Ausgabe BESTPOSB von gps1 kommend nach ASCII konvertiert
Bit 1 =1 (2)	Ausgabe BESTPOSB von gps1 kommend nach HEX konvertiert
Bit 2 =1 (4)	Ausgabe BESTPOSB von gps2 kommend nach ASCII konvertiert
Bit 3 =1 (8)	Ausgabe BESTPOSB von gps2 kommend nach HEX konvertiert
Bit 4 =1 (0x10)	Kontrollausgabe des ‚satvis2'-Datensatzes.
Bit 5 =1 (0x20)	Kontrollausgabe des ‚bestsats'-Datensatzes.
Bit 6 =1 (0x40)	Ausgabe von json-Daten bei Verbindung mit WebSocket
Bit 7 =1 (0x80)	Auch alle binär-messages von gps2 parsen (bei CAN DBC-Verwendung)
Bit 8 =1 (0x100)	Ausgabe des Headers aller NovAtel binär-Messages von gps1
Bit 9 =1 (0x200)	Ausgabe des Headers aller NovAtel binär-Messages von gps2
Bit 10 =1 (0x400)	Ausgabe des Headers aller unbehandelten NovAtel binär-Messages von gps1
Bit 11 =1 (0x800)	Ausgabe des Headers aller unbehandelten NovAtel binär-Messages von gps2
Bit 12 =1 (0x1000)	Die CAN DBC-Variablen AXB, AYB, AZB, RXB, RYB, RZB nicht filtern (s. S. 73)

```
gps flush [cycle <flush_cyc_sec> | next <time_sec> | on | off | once]
```

Daten, die an der Schnittstelle gps2 ankommen, werden in einem Empfangspuffer zwischengespeichert. Sie werden dort für einen effektiven Blocktransfer gesammelt. Mit diesem Unterkommando wird die Zykluszeit eingestellt, nach der auch der Verzeichniseintrag aktualisiert wird.
Das gilt insbesondere bei Logfiles, die im FAT-Dateisystem (interne SD-Karte) angelegt werden. Standardmäßig sind hier 60 s eingestellt. Das bedeutet auch, dass z. B. ein kontinuierliches Wachsen eines Logfiles, dies immer erst nach standardmäßig 60 s sichtbar wird (z. B. ‚**ls $(fname)**').
Wenn der Flush-Cycle abgeschaltet wird (‚**gps flush cycle -1**' oder ‚**gps flush next -1**'), dann wird der Verzeichniseintrag erst beim Schließen des Files (‚**gps log off**', ‚**gps log on**', ‚**Reset**' oder ‚**call {boot|main|alternate}**') aktualisiert. Bei plötzlichem Wegfall der Stromversorgung geht das ppmOS in ein ‚Notaus', bei dem auch alle Dateien geschlossen werden. Insbesondere falls mehrere Dateien (im FAT-Dateisystem) geöffnet sind, kann das erfolgreiche Schreiben aller Puffer nicht immer garantiert werden. Deshalb ist bei Veränderung dieses Parameters ein Kompromiss aus Schreibeffizienz, Belastung/Abnutzung der SD-Karte und Datensicherheit bei unvorhersehbarem Stromausfall zu suchen.

Befehle im Detail

gps	Steuerung der GPS-Funktionen

```
gps stat [gps1|gps2|both]
```

Schreibt man nur `gps stat` wird der aktuelle Status des GNSS-Empfangs ausgegeben. Beispiel:

```
GPS: 9F6 up 9559.476, qi 1, sat 13, no fix 96 s, gps 9453 s, dgps 10 s, no dgps, pps 4043, GPRMC delay 0.055799644 s
```

darin bedeuten:

`9F6`	3-Buchstabenkennung des GPS-Boards (Gerätekennung abgeleitet aus der Seriennummer des GPS-Boards)
`up 9559.476`	Uptime, Zeit seit Power on bzw. letztem Reset.
`qi 1`	Quality Indicator aus dem letzten GPGGA-Datensatz
`sat 13`	Anzal der sichtbaren Satelliten aus dem letzten GPGGA
`no fix 96 s`	Dauer in der kein gültiger Positionsfix vorlag.
`gps 9453 s`	Dauer mit gültigem Positionsfix mit ‚**qi 1**'
`dgps 10 s`	Dauer mit gültigem Positionsfix mit ‚**qi**' größer 1

`no dgps`
oder
`station 0136`
	wenn qi 1 oder nicht angegeben werden kann
	
	wenn qi größer 1 ist, dann ist das die DGPS-Stationsnummer (letzter Wert im GPGGA-Datensatz)

`pps 4043`	Zähler dem vom Controller registrierten PPS-Pulse (puls per second) vom GPS-Board.
`GPRMC delay 0.055799644 s`	Zeitdauer vom letzten PPS-Puls bis zum dazugehörigen empfangenen GPGGA-Datensatz mit dem Sekundenbruchteil .00.

Das Kommando ‚**gps stat**' hat einen Rückgabewert, der sich für eine bedingte Shell-Kommandoausführung verwenden lässt. Hier ein Beispiel für die LED-Signalisation des Empfangsstatus (gelbe LED blinkt bei Positionsfix, gelbe LED ist aus ohne Positionsfix):

```
cron put fixchk * * * * * * * -i "@gps stat >null && @set led yellow tgl >null"
cron put fixoff  * * * * * * * -i "@gps stat >null || @set led yellow off >null"
```

Bei Angabe der Schnittstellen gps1 oder gps2 (oder beide mit `both`) wird der Status der Schnittstelle(n) ausgegeben, z.B.: Mit `gps stat gps1` wird beispielsweise folgendes ausgegeben:

```
GPS: gps1 rx size 3000   fill lvl 0      overflow 0
```

Darin bedeuten:

`rx size`	Größe des Empfangspuffers in Byte (Empfangspuffer einstellbar mit Kommando ‚gps init')
`fill lvl`	Füllstand des Empfangspuffers in Byte
`overflow`	Überlauf des Empfangspuffers in Byte

Befehle im Detail

gps	Steuerung der GPS-Funktionen

```
gps blocksize [<size>|LF|CR]
```

Daten, die an der Schnittstelle gps2 ankommen, werden in einem Empfangspuffer zwischengespeichert. Sie werden dort für einen effektiven Blocktransfer gesammelt und erst bei Erreichen der **<size>** in einer Speicheraktion weggeschrieben.

Dies hat besondere Bedeutung für Schreibaktionen auf der internen Micro-SD-Karte. Das Schreiben ist dann besonders effektiv, wenn es in Vielfachen eines Sektors (512 byte) erfolgt. Diese Angabe bezieht sich auf ‚**gps log on ...**' und ‚**gps reply ...**'.

Der Standardwert ist 64. Ein Vielfaches davon ergibt wieder 512 und der Wert ist kleiner als die Länge eines NMEA-Datensatzes. Er ist ein Kompromiss aus verzögerungsfreies Weiterleiten/Speichern und Effektivität der Datenverarbeitung.

Der kleinste einstellbare Wert ist 1 und der größte ist 512.

Als Parameter kann auch ‚CR' (Wagenrücklauf) oder ‚LF' (Line feed) angegeben werden. Der Parameter ‚**LF**' wird bei Aktivierung von NMEA-Datensätzen empfohlen, wenn sie über andere Schnittstellen (**com..., blth, can1-..., icom..., ipsa..., ipsb...**, usw.) weitergeleitet werden sollen.

```
reply     [gps1|gps2] [{bin|asc|nmea|all} [[-c] <file>|off]]
```

Die über die Schnittstelle **gps2** ankommenden Daten werden mit ‚**gps log on ...**' in der Regel in einem File gespeichert.

Wird der Datenstrom zusätzlich - oder der Datenstrom der **gps1** - noch an anderen Schnittstellen oder Datenfiles benötigt, dann können mit diesem Unterkommando weitere zusätzliche Weiterleitungen aktiviert werden.

Es kann dabei zwischen ASCII-Datensätzen, explizit NMEA-Datensätzen (also unter Ausschluss anderer ASCII-Datensätze) oder binären Datensätzen unterschieden werden, der Datenstrom kann also in Teildatenströme aufgespalten werden.

Wird weder **gps1** noch **gps2** angegeben, dann wird **gps2** angenommen.

Mit ‚**bin**' werden nur binäre Novatel-Daten in das angegebene File ausgegeben.
Mit ‚**asc**' werden alle nichtbinären Daten, i.d.R. also NMEA oder andere ASCII-Daten, in das angegebene File ausgegeben.
Mit ‚**all**' oder ohne Zusatz werden **alle** Daten der angegebenen Schnittstelle weitergeleitet.
Mit ‚**nmea**' werden nur NMEA-Daten von der gps2 weitergeleitet (NMEA-Filter nur an gps2 möglich). Hierfür muss allerdings vorher das Kommando 'nmea-scan [gps1||gps2|both|off]'(s. S.159) für **gps1** oder **gp2** gesendet werden, und es dürfen über das Kommando 'nmea-scan gps2 {1st-letter|2nd-letter} [<letters>|*]' (s. S.159) nur die Anfangszeichen aus NMEA-Datensätzen zugelassen werden. Es wäre also auch möglich mit ‚**nmea**' andere ASCII-Datensätze als NMEAs zuzulassen.

Aufgrund der für gps1 und gps2 unterschiedlichen, implementierten Filter, ist bezüglich der GPS-Zeichenweitergabe vom GPS-Task an die zugewiesenen reply-Files folgendes zu beachten:

gps reply **gps1 bin** -> datensatzweise
gps reply **gps1 asc** -> datensatzweise
gps reply **gps1 all** -> zeichenweise
gps reply **gps2 bin** -> zeichenweise
gps reply **gps2 asc** -> zeichenweise
gps reply **gps2 all** -> datensatzweise
gps reply **gps2 nmea** -> zeichenweise
gps reply **gps1 ppmpos** -> datensatzweise (zu ‚ppmpos' s. S. 158 und S. 320)
gps reply **gps2 ppmpos** -> datensatzweise

(s. nächste Seite)

Befehle im Detail

gps	Steuerung der GPS-Funktionen

```
reply  (f.)
```

Handelt es sich bei den reply-Files um CAN-Files, so ergeben sich folgende CAN-Paketgrößen:

Datensatzweise -> 8 Byte-CAN-Pakete (Beispiel:

Zeichenweise -> 1 Byte-CAN-Pakete

Um 1 Byte-CAN-Pakete zu vermeiden, ist im Kommando zum Öffnen eines CAN-Files eine Timeout-Steuerung implementiert (‚can open' s. S.96).

Bei 'gps2 all' wirkt immer die Einstellung 'gps blocksize' (s. S. 157)

Für **<file>** sind alle im Dateisystem verfügbaren Filenamen erlaubt (**coma**, **comb**, **comc**, **blth**, **icom1**, **icom2**, ..., **pipe1**, ..., **ipsa***, **ipsa1**, ..., **c:/log/filexyz**, **null**, **mem-<adr>-<len>-w**, **can1-...**, usw.).

Mit dem Schalter ‚**-c**' wird angegeben, dass ein neues File angelegt wird bzw. gelöscht wird bevor Daten geschrieben werden. Ohne diesen Schalter werden die Daten am Dateiende angehangen, es wird also nichts vorher gelöscht. Der Schalter ‚**-c**' hat nur Bedeutung bei Files im FAT-Dateisystem auf der internen Micro-SD-Karte.

Mit ‚**gps reply ... off**' wird das jeweilige Ausgabefile geschlossen. Ein eventuell offenes Ausgabefile wird auch vor dem Start einer (neuen) Ausgabeweiterleitung geschlossen.

gps reply kann in 8 verschiedene Files umgeleitet werden.

Mit ‚**gps reply**' werden die aktuellen Einstellungen angezeigt.

```
gps reply [gps1|gps2] ppmpos [[-c] <file>|off]]
```

Die über die Schnittstelle **gps1** oder **gps2** ankommenden Daten einer Novatel-Bestpos-Message werden mit diesem Kommando in eine kompakte, allgemeine Positionsmessage ('ppmpos') konvertiert und in das angegebene File geschrieben.

Wird weder **gps1** noch **gps2** angegeben, dann wird **gps2** angenommen.

Es ist zu beachten, dass der Novatel-Bestpos-Datensatz in der GPS-Konfiguration (im File gps.cfg) für die gewünschte Schnittstelle (**com1** alias gps1 oder **com2** alias gps2) aufgenommen wird.

Beispiel:

```
log com1 bestposb ontime 0.2
```

Der erzeugte binäre Datensatz 'ppmpos' mit einer Länge von 40 Byte hat folgende Struktur (Hinweise für Programmierer s. Anhang A S. 320):

```
uint16_t sync_cnt;       //Synchronwort 0x4f50 mit frei laufenden Zähler in den unteren 3 Bit
uint16_t gps_week;
uint32_t week_ms;
double   lat;            //Breitengrad
double   lon;            //Längengrad
double   hgt_wgs84;      //Höhe über dem WGS84 Ellipsoid
uint8_t  solution_state; //Referenz: 'OEM6 Firmware Reference Guide_Rev11_om-20000129_2017-03.pdf'
uint8_t  position_type;
uint8_t  nSat_solution_L1E1B1;
uint8_t  nSat_solution_multifs;
uint8_t  ext_solution_status;
uint8_t  reserved;
uint16_t paket_crc;
```

Befehle im Detail

gps	Steuerung der GPS-Funktionen

`nmea-scan [gps1||gps2|both|off]`

Mit diesem Unterkommando können die intern benötigten GPS-Datensätze für die Realisierung der LED-Ansteuerung, Positionsfixinfo und Websocket (GPGGA + GPRMC) optional auch über die Schnittstelle **gps2** bezogen werden (anstelle der **gps1** - was der Default ist). Damit kann eine Trennung der Daten in rein binär und rein ascii an **gps1**/**gps2** vorgenommen werden.
Default ist ‚**gps nmea-scan gps1**'.

`nmea-scan  gps2 {1st-letter|2nd-letter} [<letters>|*]`

Dieses Unterkommando spezifiziert den ASCII-Filter an gps2. Hiermit werden die ersten beiden Zeichen eines ASCII-Datensatzes definiert, die für die interne Verwendung herausgefiltert werden.
Einstellungen können abgefragt werden mit:

gps nmea-scan gps2 1st-letter
GPS: expecting on gps2 nmea scan for 1st letter: '$[#<'

gps nmea-scan gps2 2nd-letter
GPS: expecting on gps2 nmea scan for 2nd letter: 'GICV ' (s. nächste Seite)

Für eine ausschließliche NMEA-Filterung wären erforderlich:

gps nmea-scan gps2 1st-letter $
gps nmea-scan gps2 2nd-letter GI

Das 'I' deshalb, weil auch '$INRMC' für die Positionsfixinfo erforderlich ist.

Wenn eine Leerzeichenerkennung nötig ist, wie z.B. bei der Ausgabe 'log com2 loglist', dann muss die Buchstabenliste in '...' oder "..." geklammert werden, also:

gps nmea-scan gps2 2nd-letter 'GICV '

Mit der Einstellung **'*'** werden alle Zeichen als 1. bzw. 2. Zeichen akzeptiert.

`gps setrtc`

Mit diesem Unterkommando wird die batteriegestützte Echtzeituhr auf die aktuelle via GPS bezogene Uhrzeit gestellt. Das Stellen erfolgt immer in UTC, selbst wenn gerade die Systemzeit auf GPS-Zeit eingestellt ist. D.h. nach dem Einschalten des Gerätes wird die Systemzeit zunächst aus der batteriegestützten Echtzeituhr geladen und später, nach dem ersten Positionsfix, präzise auf die vom GNSS übermittelte Zeit eingestellt.

In der Regel ist dies nicht erforderlich, da das Stellen automatisch beim ersten Fix erfolgt und später erneut, falls die Systemzeit um mehr als 100 µs von der GNSS-Zeit abweicht. Dies wird in einem Prüfintervall von 10 Sekunden geprüft.

Befehle im Detail

gps	Steuerung der GPS-Funktionen

```
adapter   gps2-switch [COM3|COM2]
        | pps-switch  [comc-tx|off]
        | COM2-invert [on|off]
```

Mit diesem Unterkommando kann die per Default eingestellte Verbindung von **gps2** mit **COM2** des GPS-Boards getrennt, und alternativ mit dem **COM3** des GPS-Boards hergestellt werden. Ist der **gps2** mit dem **COM3** verbunden, so ist der **COM2** mit dem **comc** des Controllers verbunden (s. Blockschaltbild auf S. 23).

Verbindung von **gps2** mit **COM2** und sonst nichts:

gps adapter gps2-switch COM2

Verbindung von **gps2** mit **COM3** und **COM2** mit **comc**:

gps adapter gps2-switch COM3

Der Controller selbst kann dann keine Daten mehr an **comc** ausgeben. Aber er 'hört' alles mit, was **COM2** ausgibt. Deshalb sollte also noch der Interpreter von comc abgeschaltet werden:

set intp comc off

gps adapter gps2-switch COM3

Dieses Kommando kann auch zum Firmware-Update des GPS-Boards verwendet werden.

```
adapter pps-switch  [comc-tx|off]
```

Mit diesem Unterkommando kann das PPS-Signal des GPS-Boards am Tx-Pin des comc (25 pol. SUB-D) ausgegeben werden. Bei ‚off' wird das PPS-Singal nicht ausgegeben (Default).

```
adapter COM2-invert [on|off]
```

Mit diesem Unterkommando kann eine externe IMU am mit dem COM2 verbundenen externen Port angeschlossen werden (empfohlen: comc).

set intp comc off

gps adapter gps2-switch COM3

gps adapter COM2-invert on

Befehle im Detail

gps	Steuerung der GPS-Funktionen

```
gps -c <cmd>
```

Mit diesem Unterkommando können Diagnoseinformationen bezüglich der Schnittstelle **gps1** ausgegeben werden.
Mit dem Stand der Firmware v 1.21 können folgende Parameter abgefragt werden:

```
gps -c cmds
```

Ausgabe:

```
$                    <ERROR              <OK
VERSIONA
```

Die Schnittstelle **gps1** ‚hört' auf diese 4 ‚Kommandos/Schlüsselwörter'
An der Schnittstelle **gps1** wird — bezüglich der NMEA-Daten — auf '$' und dann im Kommando-Handler weiter mit 'G' für die Datensätze G..RMC, G..GGA, G..GSV, G..GST und G..HDT geparst. Der Buchstabe hinter dem ersten 'G' wird nicht ausgewertet. Es können also Datensätze aller satellitensysteme (GP, GN, GL, GA, GB, G..) ausgewertet werden.

```
gps -c cmds -u
```

Ausgabe:

```
    1 VERSIONA              24 <OK                   1 <ERROR
 4084 $
   85 (default)
```

Es wird die Anzahl des Empfangs des jeweiligen Schlüsselwortes ausgegeben. Das Schlüsselwort ‚`VERSIONA`' wurde einmal bei ‚`gps init`' empfangen.
Die beiden anderen Schlüsselwörter werden sekündlich empfangen. Es sind die Kennungen für die betreffenden NMEA-Datensätze.

Befehle im Detail

gsm	Steuerung der GSM-Modemfunktionen

Das ‚**gsm**'-Kommando wird zur Initialisierung und Steuerung des GSM-Modems benötigt. Mit dem Initialisieren werden noch die Kommandos ‚**sms**', ‚**ftp**', ‚**tcp**' und ‚**ntrip**' gelinkt. D. h. sie stehen danach im Shell-Kommandointerpreter zur Verfügung. Das Modem wird über AT-Kommandos konfiguriert und gesteuert. Die Verbindung erfolgt über die serielle Schnittstelle **gsm**, welche ebenfalls vollständig umleitbar ist (s. ‚**set ird**' S. 256 und ‚**set ord**' S. 261). Bei der Initialisierung wird der Cron-Job ‚**sms-ls**' gestartet. Mit ihm wird das GSM-Modem periodisch abgefragt, ob SMS eingetroffen sind. SMS werden dem Shell-Kommandointerpreter von **coma** zur Ausführung übergeben. Über diesen Weg ist eine Fernsteuerung über SMS gegeben. Die Ausführung von Kommandos erfolgt aber nur, wenn die Absendertelefonnummer im internen ‚Telefonbuch' gespeichert ist. Das Telefonbuch kann mit dem Kommando ‚**set phone ...**' vom Anwender bearbeitet werden (Einträge hinzufügen oder entfernen).

Parameter:

```
  put <single_command_to_modem>
| put < <file_to_modem>
| stat [-dbm|-f]                -dbm check -f incl. lock status
| echo [tcp|lock] [on|off]      tcp -> echo while ftp up/download or tcp connect
| -c <cmd>
| init [-pwm]
```

```
gsm put <single_command_to_modem>
```

Mit diesem Unterkommando kann ein AT-Befehl direkt zum Modem gesendet werden.

Beispiel Abfrage der Modemversion:

```
gsm put ati1
```

Ausgabe:

```
ati1
mrx0:
mrx0: Cinterion
mrx0: EES3
mrx0: REVISION 02.004
mrx0: A-REVISION 01.000.15
mrx0:
mrx1: OK
```

Nächstes Beispiel s. nächste Seite.

Befehle im Detail

gsm	Steuerung der GSM-Modemfunktionen

`gsm put` (f.)

Beispiel Abfrage des Empfangsstatus (Feldstärke, Zelleninformation, usw.) - s. nächste Seite.

```
gsm put at^moni; gsm echo off
```

Ausgabe:

```
at^moni
mrx0:
mrx0: Serving Cell                                          I Dedicated channel
mrx0: chann rs  dBm MCC MNC  LAC cell NCC BCC PWR RXLev  C1 I chann TS timAdv PWR  dBm Q ChMod
mrx0: 43 42  -68 262  02 0CB2 8819   4   4  33  -107  38 I    No connection
mrx0:
mrx1: OK
```

Die Zahl **-68** zeigt die Empfangssignalstärke an, wobei die Zahl hier im Beispiel eine gute Empfangsbedingung bedeutet. Die Angabe ‚8819' gibt die Zelle an, in der das Modem gerade eingebucht ist.

Mit diesem Unterkommando wird implizit ‚**gsm echo on**' ausgeführt, um die empfangenen Ausgaben zu sehen. Danach sollte explizit wieder das Kommando ‚**gsm echo off**' gegeben werden, da sonst der Datenaustausch zwischen GSM-Modem und Controller die Schnittstelle **coma** „flutet".

```
gsm put < <file_to_modem>
```

Mit diesem Unterkommando können eine Serie von AT-Befehlen, die in einem File stehen, direkt zum Modem gesendet werden. Es ist für den Normalbetrieb in der Regel nicht erforderlich.

Mit diesem Unterkommando wird implizit ‚**gsm echo on**' ausgeführt, um die empfangenen Ausgaben zu sehen. Danach sollte explizit wieder das Kommando ‚**gsm echo off**' gegeben werden.

Für den Normalbetrieb stehen die ‚höheren' Kommandos ‚**ftp**', ‚**ntrip**', ‚**tcp**' und ‚**sms**' zur Verfügung.

Das Zeichen '<' leitet eine Eingabeumleitung von einem File ein (Angleichung an übliche Kommandozeilensyntax).

Befehle im Detail

gsm	Steuerung der GSM-Modemfunktionen

```
stat [-dbm|-f]                -dbm check -f incl. lock status
```

Es werden Informationen zur ‚echo'-Einstellung und dem momentanen SMS-Abfragestatus ausgegeben.

```
GSM: reentrance lock 0, lock echo 1, echo 0, echo tcp 1
sms: in progress 0
```

Im Normalbetrieb ist dieses Kommando nicht erforderlich.

```
gsm echo [tcp|lock] [on|off]
```

Ein- oder Ausschaltung des Echos (Protokollausgabe) des Datenverkehrs zwischen GSM-Modem und Controller. Im Normalbetrieb ist nur das Kommando ‚**gsm echo off**' von Bedeutung, wenn z. B. mit ‚**gsm put at^moni**' die Empfangsfeldstärke abgefragt wurde.

```
gsm -c <cmd>
```

Mit diesem Unterkommando können Diagnoseinformationen bezüglich der Schnittstelle **gps1** ausgegeben werden.

Mit dem Stand der Firmware v1.40 können folgende Parameter abgefragt werden:

Kommando:

```
gsm -c cmds
```

Ausgabe:

```
 +CGATT:        +CMGR:        +CPIN:
CONNECT               ERROR                 OK
^SIS:                 ^SISR:                ^SISW:
^SLMS:
 10+1 commands, max. 10
```

Für die Schnittstelle gsm sind 10 Kommandos ‚gelinkt', d. h. der Controller ‚hört' auf diese Schlüsselwörter beim Datenverkehr mit dem GSM-Modem.

Kommando:

```
gsm -c cmds -u
```

Ausgabe:

```
 2285 OK                       3 ERROR                     0 CONNECT
  457 +CPIN:                   0 ^SISR:                    0 ^SISW:
    0 ^SIS:                    0 +CGATT:                1359 ^SLMS:
    0 +CMGR:
 3221 (default)
 10+1 commands, max. 10
```

Es werden die Zähler für die Aufrufe ausgegeben. Z. B. wurde die SIM-PIN 457 mal abgefragt (**+CPIN**). Das ist bei der SMS-Abfrage und bei anderem Datenverkehr mit dem Netz bzw. Provider der Fall. Diese Angaben haben diagnostischen Charakter und werden im Normalbetrieb nicht benötigt.

Befehle im Detail

gsm	Steuerung der GSM-Modemfunktionen

```
gsm init [-pwm]
```

Das Modem wird eingeschaltet und initialisiert. Das Kommando ist einmal vor der Verwendung erforderlich. Es wird zweckmäßiger Weise in der **autoexec.sh** aufgeführt. Die Betriebsspannung für das GSM-Modem wird pulsbreitenmoduliert mit definierter Anstiegsgeschwindigkeit eingeschaltet. Mit dem Schalter **-pwm**, der im Normalbetrieb nicht erforderlich ist, wird die Spannung ohne Verzögerung geschaltet.

Es muss beachtet werden, dass nach dem Einschalten ca. mindestens 20 bis 30 s vergehen, bis sich das Modem im Netz eingebucht hat. Bei grenzwertigem Empfang kann diese Zeit länger sein. Vorzeitige Verwendung (**sms**, **ftp**, usw.) endet dann mit einer Fehlermeldung.

Gleichzeitig wird ein Cron-Job mit dem Namen ‚**sms-ls**' zum Auslesen der SMS aus dem GSM-Modem gestartet.

```
cron put sms-ls *+30/7 * * * * * * @sms -ls
```

Die Abfrage ‚**@sms -ls**' kann auch jederzeit durch eine andere Programmierung erledigt werden.

Sollen von dem Gerät keine SMS empfangen werden, dann wird der Cron-Job mit ‚**cron rm sms-ls**' gelöscht oder mit ‚**cron susp sms-ls**' lediglich suspendiert werden. Eintreffende SMS (max. 25) werden aber vom GSM-Modem empfangen und zwischengespeichert. Eine Reaktivierung des Auslesens geschieht dann mit ‚**cron resume sms-ls**'. Für weitere Informationen siehe auch das Kommando ‚**sms**' S. 278.

Die Modemfunktion kann abgeschaltet werden durch Ausschalten mit ‚**set pwr gsm off; cron susp sms-ls**'. Eine erneute Aktivierung erfolgt dann wieder mit ‚**gsm init**'.

Befehle im Detail

hexdump	Ausgabe von Daten im Hexdumpformat

Das Kommando **hexdump** gibt den Inhalt von Dateien oder Daten-Streams (com- oder TCP-Schnittstellen) in hexadezimaler Form aus. Die Ausgabe kann durch verschiedene Schalter vielfältig angepasst werden. Es dient der Diagnose und zur allgemeinen Optimierung von Gerätekonfigurationen. Es ist dann hilfreich, wenn es darum geht binäre Daten auf einfachen Weg zu diagnostizieren.

Beispiel 1:
Kontrolle des Inhalts von binären Datensätzen (Novatel **bestposb**)

```
gps reply bin pipe1

GPS: reply bin to pipe1

hexdump -t 2000 pipe1

00000000  51 4f 93 07 20 a1 3c 0c f4 1c 25 23 29 7c 49 40  QO.. .<...%#)|I@
00000010  de a7 20 7b 6c de 2b 40 00 80 2f 78 84 4c 66 40  .. {l.+@../x.Lf@
00000020  00 10 0c 0c 12 00 d7 76                          .......v
00000028  52 4f 93 07 08 a5 3c 0c f3 1c 25 23 29 7c 49 40  RO....<...%#)|I@
00000038  e8 a7 20 7b 6c de 2b 40 00 00 2f 78 84 4c 66 40  .. {l.+@../x.Lf@
00000048  00 10 0c 0c 12 00 f7 29                          .......)
00000050  53 4f 93 07 f0 a8 3c 0c 85 16 19 23 29 7c 49 40  SO....<....#)|I@
00000060  ab 89 72 7b 6c de 2b 40 00 00 f4 b9 7e 4c 66 40  ..r{l.+@....~Lf@
00000070  00 10 0c 0c 12 00 f3 3d                          .......=
```

Parameter:

```
[-v] [-f{8|16|32}] [-o <offset>] [-l <len>] [-b <obj_per_line>] [-pm0has] [-t {-8|<tmot_ms>} [-s]] <file>
```

-v

Der optionale Parameter ‚**-v**' (verbose) erzeugt die Ausgabe einer Kontrollzeile mit allen Parametern.

```
>hexdump -v -f32 pipe1
hexdump pipe1 from 0x0 (0) to EOF words
```

Befehle im Detail

hexdump	Ausgabe von Daten im Hexdumpformat

`-f{8|16|32}`

mit diesem optionalen Parameter wird das Ausgabeformat (der Datentyp) festgelegt. Ohne Angabe wird ‚**-f8**' angenommen.

-f8 Ausgabe im Byteformat (8 Bit)
-f16 Ausgabe im Format short (16 Bit)
-f32 Ausgabe im Format word (32 Bit)

Der Hexdump von einer festen Adresse (hier Anfang des FlashROM) hat in den 3 Formaten folgendes Aussehen:

Beispiel 2:

```
hexdump -f8 mem-0x8000000-32-r
00000000  00 ff 00 10 c1 05 00 08 41 06 00 08 41 06 00 08  ........A...A...
00000010  41 06 00 08 41 06 00 08 41 06 00 08 00 00 00 00  A...A...A.......

hexdump -f16 mem-0x8000000-32-r
00000000  ff00 1000 05c1 0800 0641 0800 0641 0800  ........A...A...
00000010  0641 0800 0641 0800 0641 0800 0000 0000  A...A...A.......

hexdump -f32 mem-0x8000000-32-r
00000000  1000ff00 080005c1 08000641 08000641  ........A...A...
00000010  08000641 08000641 08000641 00000000  A...A...A.......
```

Es wird deutlich, dass der Controller Daten im Format **little endian** speichert. D. h. das niedrigstwertige Byte eines Shorts oder Words wird auf den niedrigen Adresse abgelegt und das höherwertige auf der höheren Adresse.

`[-o <offset>] [-l <len>]`

Wenn das File eine größere Länge hat und ein bestimmter Ausschnitt ausgegeben werden soll, dann wird mit ‚**-o**' der Offset, ab dem ausgegeben werden soll und mit ‚**-l**' die Länge der Ausgabe festgelegt. Das erste Byte, Short bzw. Word eines Files hat den Offset 0. Die Längenangabe wird immer in Byteeinheiten unabhängig vom Ausgabeformat angegeben. Die Angaben können als Hexadezimalzahl (beginnt mit 0x...) oder Dezimalzahl gemacht werden.

Befehle im Detail

hexdump	Ausgabe von Daten im Hexdumpformat

`-b <obj_per_line>`

Dieser optionale Schalter legt die Anzahl der Bytes/Shorts/Words, die auf einer Zeile ausgegeben werden, fest. Ohne Angabe dieses Parameters werden 16 Bytes, 8 Shorts bzw. 4 Words pro Ausgabezeile angenommen. Der Wert `<obj_per_line>` kann zwischen 1 und 64 Bytes, 32 Shorts bzw. 16 Words liegen.

Aus Kenntnis, daß der binäre Bestpos-Datensatz 40 Bytes lang ist, ergibt sich mit folgendem Kommando eine deutlich besser lesbare Ausgabe.

Beispiel 3:

```
hexdump -b 40 -pma -t 1100 pipe1
```

Ergibt eine Ausgabe mit einen Bestpos-Datensatz pro Zeile:

```
50 4f 93 07 f8 9f 60 0c b8 72 5c 4b 29 7c 49 40 20 de d4 e6 6c de 2b 40 00 80 96 c3 df 6f 66 40 00 10 0c 0c 12 00 16 02
51 4f 93 07 e0 a3 60 0c 97 52 5b 4b 29 7c 49 40 d9 ff d5 e6 6c de 2b 40 00 80 bf 8c e5 6f 66 40 00 10 0c 0c 12 00 58 d8
52 4f 93 07 c8 a7 60 0c 97 52 5b 4b 29 7c 49 40 dd ff d5 e6 6c de 2b 40 00 80 c0 8c e5 6f 66 40 00 10 0c 0c 12 00 46 13
53 4f 93 07 b0 ab 60 0c a7 4a 5a 4b 29 7c 49 40 0a 00 d8 e6 6c de 2b 40 00 00 d5 7c eb 6f 66 40 00 10 0d 0d 12 00 ef c1
```

`-pm0has`

Mit diesen optionalen Kombinationsschalter lassen sich einzelne Ausgabefelder unterdrücken.

Jeder Buchstabe steht für ein Ausgabefeld

`p`	Ein eventueller Prefix wird weggelassen. Das Kommando **hexdump** erzeugt keinen Prefix im Gegensatz zu z. B. ‚**pew**‘
`m`	Die Speicheradresse bzw. der Bytezähler wird weggelassen.
`0`	Die angezeigte Speicheradresse beginnt bei 0 und nicht bei der tatsächlichen
`h`	Die Hexadezimale Form der Ausgabe wird weggelassen.
`a`	Der ASCII-Spalte wird weggelassen.
`s`	Die Leerzeichen zwischen den Hex-Werten wird weggelassen. Leerzeichen (je 2) zwischen Adresse bzw. Bytezähler und Hex-Spalte sowie zwischen Hex-Spalte und ASCII-Spalte werden immer ausgegeben.

Beispiel 4:

```
hexdump -b 40 -h -t 1100 pipe1

00000000  VO....l....J)|I@[Y..l.+@....qvf@........
00000028  WO....l.FK.J)|I@....l.+@....nvf@......t.
00000050  PO....l.HK.J)|I@....l.+@....nvf@.......A
00000078  QO....l.s..J)|I@....l.+@....kvf@......'.
```

Befehle im Detail

hexdump	Ausgabe von Daten im Hexdumpformat

```
-t {-8|<tmot_ms>} [-s]
```

Ohne Längenangabe bzw. wenn diese größer als die Filelänge ist, wird bis zum Dateiende gelesen. Das ist z. B. für ein File im FAT-Dateisystem auf der Micro-SD-Karte in der Regel richtig und sinnvoll. Handelt es sich bei dem File z. B. um eine Schnittstelle (**coma, comb, ipsa1**, ...), dann kommen die Datenbytes in der Regel Datenbytes periodisch mit Pausen dazwischen. Wenn also bis zum aktuellen Dateiende gelesen wird, dann würde bei einer Pause die Ausgabe beendet sein. Um der Ausgabe eine gewisse Wartezeit zu geben (timeout) kann der Schalter **-t** angegeben werden. Ein Dateiende und damit das Kommandoende wird erst dann erkannt/angenommen, wenn länger als das angegebene Timeout `<tmot_ms>` keine Daten kommen.

Die Angabe ‚**-t -8**' bedeutet unendliche Wartezeit. Im Kombination mit dem Zeichen ‚**&**' (führe Kommando als Task aus) am Ende der Kommandozeile, kann eine permanente Datenausgabe konfiguriert werden.

```
<file>
```

Das File, dessen Inhalt als Hexdump ausgegeben werden soll. Hier sind alle im ppmOS gültigen Filetypen (für Lesen) möglich:

```
coma, comb, comc, blth, gps1, gps2, gsm,
ipsa1, …, ipsa4, …, ipsd1, …, ipsd4,
icom1, …, icom5,
c:/…
pipe1, pipe2, pipe3,…,pipe20
zero-...,
mem-<adr>-<len>-r,
can1-<id>-…,
vcom1
sector-c-<start_sector>-<number_of_sector>-<number_of_sectors>
```

Befehle im Detail

ip	Steuerung der TCP/IP-Datenübertragung über den Ethernetanschluss

Mit ‚**ip**' werden alle Einstellungen für den Datenaustausch über das kabelgebundene Ethernet vorgenommen. Darunter ist die Konfiguration der Ethernet-Adresse, DNS, Hostname und Gateway zu verstehen.

Es können Server gestartet werden, die als Remoteshell arbeiten oder einfach im Raw-Mode den beliebigen Datenaustausch ermöglichen. Mit dem Kommando kann ein Web-Server (**http**-Protokoll) gestartet werden.

Ein **Echo**-Server kann zu Testzwecken gestartet werden. Ebenso können Clienttasks gestartet werden, die Verbindungen zu einem entfernten Server aufbauen. Als dedizierte Clients können FTP-Clients und/oder NTRIP-Clients gestartet werden. Das Kommando ‚**ping**' ermöglicht die Überprüfung, ob ein entfernter Rechner erreichbar ist.

Bei der gesamten Vielfalt der Anwendungsmöglichkeiten muss der zur Verfügung stehende Arbeitsspeicher des Gerätes berücksichtigt werden. Dieser begrenzt die maximale Zahl von gleichzeitigen Verbindungen (Raw-Clients, Shell-Clients, Web-Clients, Server).

Parameter:

```
  [-s]                                                  -s silent, suppress printing text
  config [-w <wait_ms>] [<cfg_file>| -d addr <adr> ...]  default is network.cfg
| config show [server [-x]]]                            -x hex output
| -v                                                    show version and exit
| ftp {put|get} <addr>[:<port>] <usr> <passwd> [<path>/]<remote_file> <local_file>
| server start ips{a|b|c|d}:<port>[-{udp|tcp}] [-n <max_clients>|-r] [-e] [-i|-ws|-ntp] [-l] [-t <tmot_ms>] [-ack]
| server start ftp
| server start http[:<port>] [-t <no_of_service_tasks>] [<index_file>]
| server echo  [udp|tcp] [-w <max_wait_ms>] [start|stop]    initializes an echo service on port 7
| client start           [-w <max_wait_ms>] icom{1|2|3|4|5}{_|-}<addr>:<port>[-{udp|tcp}][-ack]
| client ntrip           [-w <max_wait_ms>] icom{1|2|3|4|5} [{!|@}<ntrip_cfg_file>] [[-a] <rx_wr_to_file>]
| ips{a|b|c|d}[1|2|3|4] [stop | stat [clr] | tmot [<tmot_ms>]]
| icom{1|2|3|4|5}       [stop | stat [clr] | rxtmot [<tmot_ms>]]
| remove [server [ftp|ips{a|b|c|d}] [<tmot_ms>]]          default 1000 ms
```

`-s`

Der optionale Parameter unterdrückt Ausgaben bei den jeweiligen Kommandos (silent).
Das ist dann sinnvoll, wenn nur der Rückgabewert eines Kommandos für eine bedingte Kommandoausführung benötigt wird.

Beispiel:

```
@ip -s icom1 stat || @ip -s client start icom1-192.168.2.102:10000
```

Es bedeutet ‚**stille**' Abfrage des Verbindungsstatus von **icom1**. Besteht keine Verbindung zu dem Server, dann ‚**stiller**' Aufbau der Verbindung.

Befehle im Detail

ip	Steuerung der TCP/IP-Datenübertragung über den Ethernetanschluss

```
ip config [-w <wait_ms>] [<cfg_file>| -d addr <adr> ...]   default is network.cfg
```

Das Ethernet-Firmware-Modul wird gestartet. Die Konfiguration wird, falls **<cfg_file>** nicht angegeben wurde, aus dem File **$(syspath)network.cfg** gelesen. Der Shell-Funktionsaufruf **$(syspath)** liefert den aktuell eingestellten Systempfad, der per Default **c:/sys/** ist.

Dieses Kommando wird implizit ausgeführt, wenn ‚**ip ftp ...**', ‚**ip server start ...**', ‚**ping ...**' usw. ausgeführt werden. Es braucht also explizit nicht angegeben werden. Es sollte explizit ausgeführt werden, wenn z. B. der eigentliche Start einer Verbindung/Server schnell ablaufen soll.

Ausgabe:

```
IP: host name   ppm40xx-0001
IP: mac addr    1e:35:2e:30:3f:74
IP: ip4 addr    192.168.2.121
IP: mask        255.255.255.0
IP: gateway     192.168.2.2
IP: dns1        192.168.2.2
IP: interface   eth0 link is up, 100base-TX full duplex
```

Ist das File **network.cfg** nicht vorhanden oder das angegebene File kann nicht geöffnet werden oder im Konfigurationsfile steht die Zeile ‚`inet dhcp`', dann erfolgt eine Default-Konfiguration via DHCP-Server. Dieser muss in diesem Fall im lokalen Netz natürlich vorhanden sein. Im ppmOS wird dafür ein Task ‚**DHCP-Client**' gestartet, der die Kommunikation mit dem DHCP-Server realisiert. Wenn das Gerät erfolgreich konfiguriert wurde (IP-Adresse, Gateway und DNS), dann kann der Task ‚**DHCP-Client**' mit dem Kommando ‚`tsk term DHCP-Client`' terminiert werden, um Arbeitsspeicher freizugeben.

Ausgabe:

```
IP: interface   eth0
IP: can't open c:/sys/network.cfg
IP: using DHCP instead
IP: interface   eth0 link is up, 100base-TX full duplex
IP: host name   ppm40xx-xxxx
IP: mac addr    1e:35:2e:30:3f:74
IP: ip4 addr    192.168.2.118
IP: mask        255.255.255.0
IP: gateway     192.168.2.2
IP: dns1        192.168.2.2
```

Siehe f.

Befehle im Detail

ip	Steuerung der TCP/IP-Datenübertragung über den Ethernetanschluss

"`ip config`" f.:

Das File **network.cfg** hat folgenden beispielhaften Inhalt, der den Gegebenheiten bei einer netzwerkbasierten Anwendung angepasst werden sollte:

```
#Netzwerkkonfigurationsfile
#--------------------------

#der Name des Gerätes (darf keine Leerzeichen enthalten)
host_name   ppm40xx-0001

#Methode der Adresszuweisung: dhcp oder static
#static spart Resourcen im 40xx (RAM)
#-------------------------------------------
inet        static

#bei dhcp braucht nichts weiter folgen, es kann dns1 dns2 folgen
#bei static muss address, mask und evtentuell gateway und dns1/2 folgen

address     192.168.2.121
mask        255.255.255.0
gateway     192.168.2.2

#kann auch bei dhcp angegeben werden
dns1        192.168.2.2
#dns2        123.123.123.123
```

Die Zeilen, die mit ‚**#**' beginnen sind Kommentarzeilen. Diese werden ignoriert.

Wird der Parameter `-w <wait_ms>` angegeben, dann wartet das Kommando maximal die angegeben Zeit (in ms) bevor es zurückkehrt. Ohne ‚**-w**' wartet das Kommando nicht, es kehrt sofort mit einem Ergebniscode zurück. Eine Wartezeit bzw. ein Nichterfolg ergibt sich bei fehlendem DHCP-Server oder bei fehlendem Ethernet-Anschluss.

-d

Hinter -d können alle Begriffspaare, die in der network.cfg zeilenweise stehen, hintereinander aufgeführt werden.

Beispiel 1:

```
ip config -d addr 192.168.2.133
```

Hierbei wird die network.cfg ignoriert, die IP-Adresse wird gesetzt und die Netmask wird implizit auf 255.255.255.0 gesetzt. Mehr braucht man nicht, um einen 40xx im Netz verwenden zu können.

Beispiel 2:

```
ip config -d addr 192.168.2.133 gw 192.168.2.1
```

Zusätzlich den Gateway (hier 'gw' oder 'gateway' möglich) und implizit den DNS1 auf die Gateway-Adresse setzen, falls man mit URL's arbeiten möchte.

Beispiel 3:

```
ip config -d inet dhcp
```

Alles vom DHCP-Server im Netzwerk beziehen.

Befehle im Detail

ip	Steuerung der TCP/IP-Datenübertragung über den Ethernetanschluss

```
ip config show [server [-x]]]
```

Mit diesem Unterkommando wird die aktuelle Konfiguration angezeigt.

Beispiel statische Adressvergabe, ohne Ethernet-Kabel:

```
IP: host name   ppm40xx-0001
IP: mac addr    1e:35:2e:30:3f:74
IP: ip4 addr    192.168.2.121
IP: mask        255.255.255.0
IP: gateway     192.168.2.2
IP: dns1        192.168.2.2
IP: interface   eth0 link is down
```

Wenn das Ethernet-Kabel angeschlossen wird bzw. ist, dann erscheint in der letzten Zeile:

```
IP: interface   eth0 link is up, 100base-TX full duplex
```

Mit dem optionalen Zusatz ‚**server**' wird eine Statustabelle mit Informationen zu allen gestarteten Servern ausgegeben.

```
ip server start ipsa:10000 -n 2
ip server start ipsb:11000 -n 2 -i

ip config show server
```

Ausgabe kein Client verbunden:

```
IP:  name this       sockp      sflg olvl fout                fin                owner tsk        rx'd       tx'd   up s remote
-------------------------------------------------------------------------------------------------------------------------------------------
-
IP: ipsa1 0x20006a54 0x0        0000    0 dmy                 dmy                0x0
IP: ipsa2 0x20006a70 0x0        0000    0 dmy                 dmy                0x0
IP: ipsb1 0x20006ad8 0x0        0000    0 dmy                 dmy                0x0
IP: ipsb2 0x20006af4 0x0        0000    0 dmy                 dmy                0x0
```

Ausgabe drei Clients verbunden:

```
IP:  name this       sockp      sflg olvl fout                fin                owner tsk        rx'd       tx'd   up s remote
-------------------------------------------------------------------------------------------------------------------------------------------
-
IP: ipsa1 0x20006a54 0x20010b28 0000    0 cbm_0x200184f9_1501 dmy                0x20018b60         33   51201961    217
192.168.2.102:59252
IP: ipsa2 0x20006a70 0x20010e28 0000    0 cbm_0x2001e3f9_1501 dmy                0x2001ea60         45     893213     61
192.168.2.102:59254
IP: ipsb1 0x20006ad8 0x20010ca8 0000    0 cbm_0x2001bbe1_1501 cbm_0x2001b604_1501  0x2001c248       4733       7625    183
192.168.2.102:59253
IP: ipsb2 0x20006af4 0x0        0000    0 dmy                 dmy                0x0
```

Siehe f.

Befehle im Detail

ip	Steuerung der TCP/IP-Datenübertragung über den Ethernetanschluss

"`ip config show`" f.:

Derselbe Status mit dem optionalen Zusatz ‚**-x**':

```
ip config show server -x
```

Ausgabe:

```
IP: ipsa 0x20006a40  pid    10  tmot     1000 ms  port 10000  max clients 2
IP:   client 20006a54  20010b28 08164988 20018b08 08164988 200001bc 20017d30 20006a40  (.. .I..... .I..... 0}. @j.
IP:   client 20006a70  20010e28 08164988 2001ea08 08164988 200001bc 2001dc30 20006a40  (.. .I..... .I..... 0.. @j.
IP:   client 20006a8c  00000000 08164988 200001bc 08164988 200001bc 00000000 20006a40  .....I..... .I..... ....@j.
IP:   client 20006aa8  00000000 08164988 200001bc 08164988 200001bc 00000000 20006a40  .....I..... .I..... ....@j.
IP: ipsb 0x20006ac4  pid    11  tmot     1000 ms  port 11000  max clients 2
IP:   client 20006ad8  20010ca8 08164988 2001c1f0 08164988 2001c1c0 2001b418 20006ac4  ... .I..... .I..... ... .j.
IP:   client 20006af4  00000000 08164988 200001bc 08164988 200001bc 00000000 20006ac4  .....I..... .I..... .....j.
IP:   client 20006b10  00000000 08164988 200001bc 08164988 200001bc 00000000 20006ac4  .....I..... .I..... .....j.
IP:   client 20006b2c  00000000 08164988 200001bc 08164988 200001bc 00000000 20006ac4  .....I..... .I..... .....j.
IP: ipsc 0x20006b48  pid     0  tmot     1000 ms  port     0  max clients 0
IP:   client 20006b5c  00000000 08164988 200001bc 08164988 200001bc 00000000 00000000  .....I..... .I..... ........
IP:   client 20006b78  00000000 08164988 200001bc 08164988 200001bc 00000000 00000000  .....I..... .I..... ........
IP:   client 20006b94  00000000 08164988 200001bc 08164988 200001bc 00000000 00000000  .....I..... .I..... ........
IP:   client 20006bb0  00000000 08164988 200001bc 08164988 200001bc 00000000 00000000  .....I..... .I..... ........
IP: ipsd 0x20006bcc  pid     0  tmot     1000 ms  port     0  max clients 0
IP:   client 20006be0  00000000 08164988 200001bc 08164988 200001bc 00000000 00000000  .....I..... .I..... ........
IP:   client 20006bfc  00000000 08164988 200001bc 08164988 200001bc 00000000 00000000  .....I..... .I..... ........
IP:   client 20006c18  00000000 08164988 200001bc 08164988 200001bc 00000000 00000000  .....I..... .I..... ........
IP:   client 20006c34  00000000 08164988 200001bc 08164988 200001bc 00000000 00000000  .....I..... .I..... ........
```

Es wird die gesamte Verbindungstabelle für alle ‚**ips...**' ausgegeben.
Die beiden letzten Unterkommandovarianten haben informatorischen Charakter und werden hauptsächlich zu Debugging-Zwecken benutzt.

```
ip -v
```

Abfrage der Version des Oryx Embedded Ethernet-Moduls mit 'ip -v' (aktuell v.1.8.2.)

```
ip ftp {put|get} <addr>[:<port>] <usr> <passwd> [<path>/]<remote_file> <local_file>
```

Dieses Unterkommando ermöglicht den Filetransfer von und zu einem FTP-Server. Es kann immer nur ein Transfer gestartet werden. Die Portnummer kann für einen Standard-FTP-Transfer weggelassen werden. Username **<usr>** und Passwort **<passwd>** dürfen keine Leerzeichen enthalten.
Tip:
Als Pfad kann auch **$(fn_ext)** angegeben werden. Dieser Shell-Funktionsaufruf ergibt den 3-Buchstabencode des GPS-Boards. Es ist der die File-Extension für GPS-Logfiles auf dem 40xx. Damit ist es möglich Up- und Downloads gerätespezifisch durchzuführen. Das Unterverzeichnis mit dem 3-Buchstabencode muss auf dem Server bereits existieren.

Befehle im Detail

ip	Steuerung der TCP/IP-Datenübertragung über den Ethernetanschluss

```
ip server start ips{a|b|c|d}:<port>[-{udp|tcp}] [-n <max_clients>|-r] [-e] [-i|-ws|-ntp] [-l] [-t <tmot_ms>] [-ack]
```

Mit diesem Unterkommando wird ein Server gestartet. Die Serververbindungen sind im ppmOS generische Files. Sie sind für den bidirektionalen Datenaustausch gedacht. Die Server haben die Stammnamen ‚**ipsa**', ‚**ipsb**', ‚**ipsc**' und ‚**ipsb**'. Jeder Server ‚**hört**' an einer Portnummer. Die jeweiligen Clientverbindungen der Server werden mit einer Ziffer 1 ... 4 gekennzeichnet. Es sind also theoretisch 16 Verbindungen möglich.

Hier gilt es aber, die Speicherressourcen des Controllers zu berücksichtigen. In der Praxis können je nach weiterer Konfiguration maximal ca. 4-7 TCP-Verbindungen gleichzeitig bestehen.

`-udp|tcp`

Mit den optionalen Schaltern ‚**-udp**' und ‚**-tcp**' kann angegeben werden, ob es sich um einen UDP- oder TCP-Server handelt. Wird nichts angegeben, wird ein TCP-Server gestartet.

Ein Sondername ist der Servername (auch Filename) mit angehängtem ‚*****'. Er hat die Bedeutung, dass bei einer Schreiboperation in dieses ‚**File**' die Daten an alle Clients des betreffenden Servers gesendet werden.

Beispiel:

```
gps reply ipsa*
ip server start ipsa:10000 -n 3 -i
```

Alle Clients (3 in dem Beispiel) erhalten den Datenstrom, der über **gps2** vom GPS-Board empfangen wird. Empfangene Daten werden nicht interpretiert. Wenn ein Server Daten empfängt, diese jedoch von keinem Task abgeholt werden, dann erscheint, nachdem der Empfangspuffer voll ist, die folgende Fehlermeldung bei jedem Daten-'drop':

```
IP: couldn't store received data to cbm_0x2001828c_1501, rx'd 48  written 0
```

Wenn der Datenstrom nicht anderweitig gestoppt werden kann, dann können z. B. mit einem Cron-Job die empfangenen Daten ‚entsorgt' werden.

```
cron put * * * * * * "@cp ipsa1 null; @cp ipsa2 null; @cp ipsa3 null"
```

Zum Thema Empfangsüberlauf (blocking/nonblocking write/read) siehe auch das Kommando (S. 242)

```
set fctl [rd|wr] <filename> [{clr|set} <val>]
```

IP-Server-Streams werden standardmäßig im nonblocking Mode geöffnet.

`-n <max_clients>`

Die optionale Angabe stellt die maximal möglich gleichzeitige Anzahl von Clientverbindungen zu dem betreffenden Server ein. Ohne Angabe wird **-n 1** eingestellt.

`-r`

Die Angabe kann anstelle **-n** gemacht werden. Sie bewirkt, dass **-n 1** eingestellt wird und der Server ein ‚reopen' durchführt, wenn ein Client ‚anklopft', also eine Verbindung aufbauen will und eine Verbindung schon besteht. In dem Fall wird die eventuell bestehende Verbinfung erst geschlossen und dann neu geöffnet. Ohne ‚**-r**' würde ein neuer Verbindungsrequest mit einem Fehler ‚**IP: ... connection request ... rejected, out of resources...**' abgelehnt werden.

Siehe f.

Befehle im Detail

ip	Steuerung der TCP/IP-Datenübertragung über den Ethernetanschluss

"`ip server start ips`" f.:

Der Schalter ‚**-r**' hat Bedeutung bei einer WebSocket-Anwendung, bei der der 40xx eine App zur Verfügung stellt, die in einem Browser dargestellt wird. Um bei einem ‚**Seite neu laden**' vom Browser aus, die Verbindung zu behalten muss der Schalter ‚**-r**' angegeben werden.

Standardmäßig ist der Server eine Shellserver im Raw-ASCII-Modus. D. h. der einkommende Datenstrom wird dem Shellinterpreter übergeben und von ihm interpretiert. Es werden Kommandozeilen erwartet, die mit **CR**, **LF** oder **CR+LF** terminiert sind.

`-e`
Wenn der Shell-Interpreter für diesen Server aktiv ist, dann kann mit ‚**-e**' das standardmäßige Kommandoecho initial abgeschaltet werden. Dies kann jedoch auch separat mit ‚**set echo ips{a|b|c|d} [on|off]**' jederzeit angezeigt und eingestellt werden. In der Client-Serververbindung reicht dafür das Kommando ‚**set echo [on|off]**'.

`-i`
Wenn die empfangenen Daten nicht interpretiert, sondern nur weitergeleitet werden sollen, dann ist der Schalter ‚**-i**' anzugeben. Nur bei initial aktiviertem Interpreter kann dieser durch ‚**set intp ips{a|b|c|d} [on|off]**' ein- oder ausgeschaltet werden. In der Client-Serververbindung ist das Kommando ‚**set intp [off]**' möglich, denn wenn der Interpreter abgeschaltet ist, dann diese Verbindung den Interpreter nicht wieder selbst aktivieren. Dies müsste dann ein anderer Task übernehmen.

`-ws`
Dieser Schalter startet im Zusammenhang mit der „Start-WebSocket"-Anwendung den WebSocket-Server (s. S. 68).

`-ntp`
Mit diesem Schalter wird ein **NTP-Server** gestartet. Bei wiederholten Anfragen weniger als 100ms kommt es zum Restart des Servers. Das minimale Abfrageintervall kann bis auf 70ms herunter gehen.

Ein **NTP-Client** kann über das Kommando ‚date' konfiguriert werden (s. S. 118)

`-l`
Zu Testzwecken kann der Server mit dem Schalter ‚**-l**' gestartet werden. Falls der Interpreter für diesen Server deaktiviert ist, dann gibt es ein Loopback für alle empfangenen Daten. D. h. alles was empfangen wird, wird unverändert gesendet. Dies kann jedoch auch separat mit ‚**set loop ips{a|b|c|d} [on|off]**' jederzeit angezeigt und eingestellt werden. Das Loopback wird erst wirksam, wenn der Interpreter abgeschaltet ist.

`-t <tmot_ms>`
Der optionale Schalter gibt das Timeout an, nach dem der Server einen Verbindungsfehler registriert.
Ohne Angabe werden ‚**-t 1000**' eingestellt.

`-ack`
Nach dem TCP-Protokoll werden alle Pakete, wenn sie richtig empfangen wurden, durch Acknowledge-Pakete, die an den Absender zurückgeschickt werden, quittiert. Daraus ergibt sich die Verbindungssicherheit und Fehlertoleranz einer TCP-Verbindung. Der Absender wiederholt nach einem Timeout Pakete mehrmals, wenn sie nicht quittiert werden. Dieses Warten auf das Acknowledge kann mit ‚**-ack**' abgeschaltet werden. Das kann bei einer schlechten Verbindung zwar Paketverluste zur Folge haben, aber es kommt nicht so schnell zum Stall einer Verbindung. Siehe dazu auch das Kommando ‚**set fctl ...**' (S. 242)zum Thema blocking/nonblocking write.

Befehle im Detail

ip	Steuerung der TCP/IP-Datenübertragung über den Ethernetanschluss

ip server start ftp

Mit diesem Unterkommando kann im 40xx ein FTP-Server gestartet werden.

Default-Einwahldaten:

User: ppm40xx
Password: 40xx-ftp-pw

Der FTP-Server kann temporär gestartet und auch im laufenden Betrieb beendet werden.

(Download mit Total Commander ca. 4.3 MB/s)

ip server start http[:<port>] [-t <no_of_service_tasks>] [<index_file>]

Mit diesem Unterkommando wird ein Web-Server gestartet. Wenn er am anderen als den Standard-Port 80 (http-Port) hören soll, dann kann eine andere Portnummer angegeben werden.

-t <no_of_service_tasks>

Dieser Schalter gibt an, wie viele Tasks der Web-Server maximal starten kann, um quasi gleichzeitig die Anfragen eines Webbrowsers zu bearbeiten. Wird der Parameter weggelassen, dann wird ‚**-t 2**' eingestellt. Maximal kann ‚**-t 8**' eingestellt werden.

Mit z. B. ‚**-t 7**' ist ein flüssiges Browsen in einer auf dem 40xx gehosteten komplexen Intranetseite möglich. Dies kann am Beispiel der Internetseite der ppm GmbH **ppmgmbh.com**, die komplett mit allen Seiten, Bildern, Downloaddateien und Videos auf die Micro-SD-Karte des 40xx kopiert wurde (ca. 2 GB), demonstriert werden. Ein Unterschied zum Browsen (via 16 MBit DSL) auf der originalen Internetseite ist dann kaum feststellbar.

Es muss auch hier der Gesamtspeicherbedarf von anderen gestarteten Tasks berücksichtigt werden.

<index_file>

Der optionale Parameter gibt den Pfad und Namen der Startseite an. Per Default wird ‚**c:/www/index.html**' angenommen. Dort sind alle erforderlichen Unterverzeichnisse anzulegen und die Daten zu speichern. Den Web-Server mit verschiedenen Startseiten starten zu können, ermöglicht das Hosten verschiedener Web-Seiten. Das kann zum Testen oder für verschiedene Konfigurationsszenarien sehr nützlich sein.

Befehle im Detail

ip	Steuerung der TCP/IP-Datenübertragung über den Ethernetanschluss

```
ip server echo  [udp|tcp] [-w <max_wait_ms>] [start|stop]          initializes an echo service on port 7
```

Der Echoserver dient der Überprüfung und Test der Performance einer UDP- oder TCP-Verbindung. Daten, die er empfängt, werden unverändert wieder zurückgesendet. Zu diesem Zweck kann auch das Kommando ‚**ping**' verwendet werden.

`udp|tcp`
Mit den optionalen Schaltern ‚**-udp**' und ‚**-tcp**' kann angegeben werden, ob eine UDP- oder TCP-Verbindung überprüft bzw. getestet werden soll. Wird nichts angegeben, dann wird von TCP ausgegangen.

Ein Client verbindet sich:

```
Echo service: connection established with client 192.168.2.102 port 59661
```

Der Client trennt die Verbindung:

```
Echo service: 1187 bytes received, 1187 bytes sent in 18486 ms
```

`-w <max_wait_ms>`

Stellt die maximale Zeit ein, die der Server auf den Abschluss der impliziten ‚**ip config**' wartet.

`Start|stop`

Der Echoserver wird gestartet bzw. gestoppt. Bei ‚stop' beendet sich der Task ‚**tcp_echo_server**'.

Ausgabe:

```
IP: server echo is stopped
```

Befehle im Detail

ip	Steuerung der TCP/IP-Datenübertragung über den Ethernetanschluss

```
ip client start [-w <max_wait_ms>] icom{1|2|3|4|5}{_|-}<addr>:<port>[-{udp|tcp}][-ack]
```

Dieses Unterkommando startet einen Clienttask, der Verbindung zu einem Server mit angegebener Portnummer aufnimmt. Der Datenaustausch kann auch hier bidirektional sein.

```
udp|tcp
```

Mit den optionalen Schaltern ‚**-udp**' und ‚**-tcp**' kann angegeben werden, ob die Clients mit UDP oder TCP gestartet werden.

```
-w <max_wait_ms>
```

Stellt die maximale Zeit ein, die der Client auf den Abschluss der impliziten ‚**ip config**' wartet.

Nachdem der Clienttask gestartet ist, kann mit dem Filenamen ‚**icom**`{1|2|3|4|5}`' so operiert werden, wie es mit einem regulären File möglich ist.

Beispiel Logging eines Ethernet-Streams auf Micro-SD-Karte:

```
ip client start icom1-192.168.2.122:12000
cp -t -8 icom1 c:/log/example.log&
```

Der Kopiertask müsste bei Bedarf mit

```
tsk term "c:/log/example.log"
```

beendet werden.

Die Verwendung von ‚**icom**`{1|2|3|4|5}`' ist die Kurzform des Filetyps Ethernetclient. Die Langform dieses Filetyps

```
icom{1|2|3|4|5}-<ip_address>:<port>[-{udp|tcp}][-ack]
```

kann z. B. für einmalige Kopieraktionen oder ähnlich verwendet werden. Voraussetzung dafür ist das vorher ausgeführte ‚**ip config**'.

Beispiel Kopieren eines 51.2 MB großen ‚Dummyfiles' und Zeitmessung:

```
time cp c:/tmp/51.2mb icom1-192.168.2.102:12000
```

Ausgabe:

```
cpu time 16.358602095 s
```

Was eine Datenrate von ca. 3.1 MB/s ergibt.

Die Langform wird nicht empfohlen in z. B. ‚**loop ...**' Kommandos oder in anderen sich häufig wiederholenden Situationen, weil dabei das ‚**icom...**' in jedem Schleifendurchlauf geschlossen und neu geöffnet wird, was die maximale Datenübertragungsrate stark senkt.

Befehle im Detail

ip	Steuerung der TCP/IP-Datenübertragung über den Ethernetanschluss

```
ip client ntrip [-w <max_wait_ms>] icom{1|2|3|4|5} [{!|@}<ntrip_cfg_file>] [[-a] <rx_wr_to_file>]
```

Mit diesem Kommando wird eine Clientverbindung zu einem NTRIP-Server aufgebaut, um RTK-Daten zu beziehen, die zum GPS-Board oder zu einer anderen Schnittstelle geschickt werden.
In der kurzen Aufrufform:

```
ip client ntrip  icom{1|2|3|4|5}
```

wird die Verbindung hergestellt und die RTK-Daten werden nach **gps2** kopiert. Zusätzlich wird der $GPGGA-Datensatz alle 20 s zum NTRIP-Server übertragen, damit dieser, falls der Mountpoint diesen Service bietet, bessere Korrekturdaten zur Verfügung stellen kann. Die Verbindungsinformationen (NTRIP-Serveradresse und Authentifikation) werden aus dem File

$(syspath)ntrip.cfg

gelesen. Standardmäßig ist der Systempfad **c:/sys/.**

Aus dieser Datei werden alle nichtleeren Zeilen, außer Kommentarzeilen, die mit ‚#' beginnen, zum NTRIP-Server übertragen.
Beispiel für eine ‚**ntrip.cfg**' mit Kommentaren:

```
#Konfigurationsfile für das Öffnen der NTRIP-Casterverbindung
#Hinweis: Zeilen, die mit exec beginnen, werden an die shell weitergegeben (ausgeführt)

exec @echo reading from ntrip.cfg

#alle nichtleeren Zeilen, die hinter der 'address'-Zeile folgen, werden übertragen
#hinter ‚Authorization: Basic , muß ‚user:passwd' base64-kodiert angegeben werden

address socktcp://78.46.41.8:80
GET /LEIJ1 HTTP/1.0
User-Agent: NTRIP XS/1.14
Authorization: Basic VXNlcm5hbWU6UGFzc3dvcnQ=

exec @echo ntrip.cfg scan completed

""
```

Wichtig ist die leere Zeichenkette "" als letzte zu übertragende Zeile. Das Protokoll der NTRIP-Authentifizierung erwartet nach der ‚Authorization:'-Zeile eine Leerzeile (CR+LF).

Optionale Angaben:

```
-w <max_wait_ms>
```

Stellt die maximale Zeit ein, die der Client auf den Abschluss der impliziten ‚**ip config**' wartet.

```
{!|@}<ntrip_cfg_file>
```

Wenn ein anderes als das Default-Konfigurationsfile ‚c:/sys/ntrip.cfg' verwendet werden soll.

```
[-a] <rx_wr_to_file>
```

Ausgabe des RTK-Datenstromes an das anzugebende File, wenn es nicht ‚**gps2**' sein soll. Hier sind wieder alle Dateitypen des ppmOS möglich. Der optionale Schalter ‚**-a**' bedeutet, dass die Daten am Ende des Files angehangen werden. Anderenfalls wird das File neu angelegt bzw. falls es existiert, wird es vorher gelöscht.

Befehle im Detail

ip	Steuerung der TCP/IP-Datenübertragung über den Ethernetanschluss

```
ip ips{a|b|c|d}[1|2|3|4] [stop | stat [clr] | tmot [<tmot_ms>]]
```

```
stop
```

Mit diesem Unterkommando werden alle oder eine Clientverbindung eines Servers gestoppt, sie werden getrennt. Der Server bleibt aktiv. Soll der gesamte Server beendet werden, dann ist ‚**ip remove server ips...**' zu verwenden.

```
ip ipsb stop
```

Ausgabe (es bestanden 2 Clientverbindungen):

```
IP: ipsb1 break in shell client task, eventFlags 0x4, error 0
IP: ipsb1 connection to 192.168.2.102:50506 closed
IP: ipsb1 shell client task terminated
IP: ipsb2 break in shell client task, eventFlags 0x4, error 0
IP: ipsb2 connection to 192.168.2.102:50507 closed
IP: ipsb2 shell client task terminated
```

Nach der erneuten Verbindungsanfrage der Clients wird ausgegeben:

```
IP: ipsb1 connection established: local 192.168.2.121:11000 remote 192.168.2.102:50592
IP: ipsb2 connection established: local 192.168.2.121:11000 remote 192.168.2.102:50593
```

```
ip ipsb stat
```

Ausgabe:

```
IP: server is not started
```

oder

```
IP: ipsb1 0x20006ad8 0x200109a8 0000    0 cbm_0x200171f1_1501  dmy               0x20017858          0          0    1603
192.168.2.102:50592
IP: ipsb2 0x20006af4 0x20010b28 0000    0 cbm_0x2001a619_1501  dmy               0x2001ac80        677   10240798    1603
192.168.2.102:50593
```

Der aktuelle Status wird abgefragt. Mit dem Zusatz ‚-s' (‚**ip -s ips{a|b|c|d}[1|2|3|4] stat**') wird die Ausgabe unterdrückt. In jedem Fall gibt der Returncode Aufschluss auf die Verbindung. Ein Rückgabewert 0 bedeutet ‚Verbindung besteht' und ungleich 0 ‚keine Verbindung'. Das kann in einem Cron-Job genutzt werden, um eine dauerhafte Serverbereitstellung zu sichern.
Beispiel:

```
cron put strt-ipsa */30 * * * * * * -t "@ip -s ipsa stat || @ip -s server start ipsa:10000 -n 2"
```

```
stat clr
```

Mit dem Zusatz ‚**clr**' wird die Statistik gelöscht.

```
tmot [<tmot_ms>]
```

Mit dem Parameter ‚**tmot**' wird die Zeit ausgegeben bzw. eingestellt, die bei der TCP-Übertragung bei einem fehlenden Acknowledge gewartet wird, bis eine Retransmission des TCP-Paketes eingeleitet wird. Der Default-Wert ist 1000 ms.

Befehle im Detail

ip	Steuerung der TCP/IP-Datenübertragung über den Ethernetanschluss

```
ip icom{1|2|3|4|5}       [stop | stat [clr] | rxtmot [<tmot_ms>]]
```

stop

Mit diesem Unterkommando kann eine Clientverbindung getrennt werden.

```
--------------------------------------
 >      237 s          0          0        0 bps
<       237 s    2343000    8574912    36181 bps
```

Beim Schließen wird ausgegeben wie Verbindung bestand (237 s), wieviel Byte gesendet (8574912), und empfangen (0) wurden. Die letzte Zahl in den Zeilen ist die Datenrate in Byte/s. Die zweite Zahl in den beiden Zeilen ist die Datenmenge seit der letzten Abfrage.

stat

Der aktuelle Status wird abgefragt. Mit dem Zusatz ‚-s' (‚**ip -s icom2 stat**') wird die Ausgabe unterdrückt. In jedem Fall gibt der Returncode Aufschluss auf die Verbindung. Ein Rückgabewert 0 bedeutet ‚Verbindung besteht' und ungleich 0 ‚keine Verbindung'.

Ausgabe bei einer NTRIP-Verbindung:

```
 >       29 s       7714      12885      444 bps
<        29 s         74        237        8 bps
icom2: ip data become written to com6
icom2: socket flags 0x0, putchar_fifo 0
```

Zusätzlich zu den Informationen, wie bei ‚**stop**' erläutert, wird hier noch das Ausgabefile (com6 alias gps2) mitgeteilt.

stat clr

Mit dem Zusatz ‚**clr**' wird die Statistik gelöscht.

rxtmot [<tmot_ms>]

Mit dem Parameter ‚**rxtmot**' wird die Zeit ausgegeben bzw. eingestellt, die bei Abfrage der File-Länge oder beim Lesen aus dem Stream, falls 0 Byte im Puffer sind, gewartet wird. Der Default-Wert ist 0.

Befehle im Detail

ip	Steuerung der TCP/IP-Datenübertragung über den Ethernetanschluss

ip remove [server [ftp|ips{a|b|c|d}] [<tmot_ms>]] **default 1000 ms**

Mit diesem Unterkommando können einzelne Server, alle Server oder das gesamte IP-Modul entfernt werden. Der dafür allozierte Arbeitsspeicher wird freigegeben. Das hat Bedeutung, wenn in verschiedenen Anwendungssituationen temporär eine Verbindung benötigt wird und zu einem anderen Zeitpunkt der gesamte Arbeitsspeicher für eine andere Funktion benötigt wird.

ip remove

Das gesamte IP-Modul wird deaktiviert und der Arbeitsspeicher wird freigegeben.

ip remove server

Alle Server ‚**ips...**' werden gestoppt und entfernt.

ip remove server ftp|ips{a|b|c|d}

Einer der Server **ftp, ipsa, ipsb, ipsc** oder **ipsd** wird gestoppt und entfernt.

Mit diesen Kommandos werden den betreffenden Tasks Signale gesetzt. Danach schließen sie Files, geben allozierten Speicher zurück und beenden sich dann selbst. Auf das Beenden wird per Default 5 s gewartet. Diese Wartezeit kann mit dem letzten Parameter in ms eingestellt werden.

jump	Verzweigung der Skriptausführung in ein anderes Skript

Dieses Kommando ist nur in Skripten anwendbar.

Das Kommando realisiert einen unbedingten Sprung in ein neues Skriptfile

Das Kommando muss am Anfang einer Skriptzeile stehen. Dahinter muss ein Filename folgen.

Das aktuelle Skriptfile wird geschlossen und die nächsten Kommandos werden aus dem neuen File gelesen.

Parameter:

<skript_file>

<skript_file>

Name des Skriptfiles, in dem die Ausführung fortgesetzt wird.

Befehle im Detail

loop	Shellkommando zur wiederholten Ausführung von Kommandos

Das Kommando ‚**loop**' wird zur wiederholten Ausführung eines Kommandos bzw. einer in "..." eingeschlossenen Gruppe von Kommandos verwendet. Die Wiederholung kann mit einer vorgegebenen Zykluszeit oder ohne Pause erfolgen. Bei der Ausführung kann der Returncode des ausgeführten Kommandos ausgewertet werden, um den Loopbefehl vorzeitig zu beenden. Das Zeitraster der Ausführung ist dem Multitasking untergeordnet. D. h. die Ausführung erfolgt in der Reihenfolge der Taskliste, sie ist nicht timer-gesteuert. Der Ausführungszeitpunkt jittert typisch um einige Mikrosekunden. Im worst case, also wenn mehrere andere Tasks max. Rechenzeit beanspruchen, ist mit einem Jitter des Ausführungszeitpunktes von mehreren Millisekunden zu rechnen.

Parameter:

```
{<nloops>|-8} [-t <ms> [-f[f]]] [{-eq|-ne} <val>] <cmd>
                        -8  infinity loop
                        -f no sleep, -ff fastest
                        -eq or -ne means loop while …
```

`<nloops>`

Der Parameter **<nloops>** ist eine Zahl. Sie legt die Anzahl der Schleifendurchläufe fest. Mit vorangestelltem ‚**0x**' wird die Zahl als Hexadezimalzahl interpretiert.

`-8`

Die alternative Angabe ‚**-8**' symbolisiert unendlich, d. h. die Schleife endet nicht. Die Schleife kann dennoch verlassen werden, wenn der Parameter ‚**-eq**' oder ‚**-ne**' angegeben wird und die entsprechende Bedingung erfüllt ist.

`-t <ms>`

Mit diesem optionalen Parameter wird eine Zykluszeit in Millisekunden eingestellt. Die Zykluszeit kann nur eingehalten werden, wenn die Ausführungszeit der Kommandos bzw. der in **"..."** eingeschlossenen Kommandofolge kleiner ist als die Zykluszeit. Die Pause nach dem Kommando ergibt sich aus Zykluszeit minus Ausführungszeit.

`-f oder -ff`

Die Zykluszeit wird in Millisekunden agegeben. Der Test des nächsten Ausführungszeitpunkts erfolgt standardmäßig in 1 ms Intervallen, wobei der betreffende Task in der 1-ms-Zwischenzeit suspendiert wird. Soll der Ausführungszeitpunkt mit kleinerem Jitter erreicht werden, dann kann ‚**-f**' angegeben werden. Das Loopkommando ermittelt den nächsten Ausführungszeitpunkt mit wesentlich kleinerem Jitter.

Mit ‚**-ff**' wird der jeweils nächste Ausführungszeitpunkt durch permanentes Polling ermittelt. Das ergibt ein Jitter im µs-Bereich.

Zu beachten ist jedoch, daß je nach CPU-Last durch andere Taskaktivitäten wieder ein größeres Jitter entstehen kann, welches im worst case sogar einige Millisekunden ergeben kann.

Befehle im Detail

loop	Shellkommando zur wiederholten Ausführung von Kommandos

`-eq <val>`

Mit diesem Parameter wird eingestellt, dass das Loop-Kommando solange läuft, wie der Return-Wert des ausgeführten Kommandos gleich dem angegebenen Wert ist. Wird hier der Wert 0 angegeben, dann bedeutet dies bei den meisten Kommandos Ausführung solange wie fehlerfrei.

`-ne <val>`

Mit diesem Parameter wird eingestellt, dass das Loop-Kommando solange läuft, wie der Return-Wert des ausgeführten Kommandos ungleich dem angegebenen Wert ist. Wird hier der Wert 0 angegeben, dann bedeutet dies bei den meisten Kommandos Ausführung solange Ausführung mit Fehler.

Beispiel 1:

```
loop 80 -t 100 "@set led orange on >null; @wait -s 0.05; @set led orange off >null"&
```

Die orange LED blinkt 80 mal im 10 Hz-Takt für jeweils 50 ms.

Beispiel 2:

```
loop -8 -t 3000 '@echo $(gpgga) >comb; @echo $(gprmc) >comb' &
```

Ausgabe der NMEA-Datensätze $GPGGA und $GPRMC alle 3 s an der Schnittstelle comb.

Diese Ausgabe wieder beenden (Task terminieren) mit:

```
tsk term "loop -8 -t 3000"
```

Beispiel 3:

```
loop 1 "@loop -8 -t 10 -eq 0 @pob -s pc.13; @echo key pressed, loop exit; loop -8 -t 10 -ne 0 @pob -s pc.13; @echo fertig!"&
```

Es wird einmalig (‚**loop 1**‘) unendlich gewartet, solange die On/Off-Taste nicht gedrückt wird. Wird sie gedrückt, dann wird ‚**key pressed, loop exit**‘ ausgegeben. Danach wird unendlich lange auf das Loslassen der Taste gewartet. Wird sie losgelassen, dann wird das mit ‚**fertig!**‘ quittiert.

Bei Beispiel 3 ist zu beachten, dass nach 3 s Drücken der Taste und Loslassen während des Pieptons, sich das Gerät ausschaltet.

Für dieses Beispiel also nur kurz drücken (antippen), was für die Praxis durchaus genutzt werden kann, wenn irgend eine Aktion beim kurzen Tastendruck ausgeführt werden soll. Wenn diese Aktion immer wieder ausgeführt werden soll, dann müsste am Anfang dieses komplexen Kommandos nicht ‚**loop 1 ...**‘ sonder ‚**loop -8 ...**‘ stehen.

Befehle im Detail

ls	Datei- und Verzeichnisinformationen listen

Mit dem Kommando ‚**ls**' wird der Inhalt von Verzeichnissen oder Informationen von einzelnen Dateien ausgegeben. Verzeichnisse sind im FAT-Dateisystem auf der internen Mikro-SD-Karte vorhanden bzw. sie können dort angelegt werden. Dateien im sinne des ppmOS sind welche im FAT-Dateisystem aber auch Pipes, Schnittstellen, Bluetooth-, CAN- oder TCP-Verbindungen. Die ausgegebenen Informationen hängen vom jeweiligen Dateityp ab.

Das Kommando wird ebenfalls verwendet, wenn automatisiert Shell-Skripte oder Reportdateien erstellt werden, die sich auf Dateien in einem oder mehreren Verzeichnissen befinden. Das Kommando kann rekursiv arbeiten und das Ausgabeformat ist vom Anwender frei wählbar. Es wird in dieser Variante nur der Dateiname verwendet.

Reguläre Ausdrücke bzw. Jokerzeichen sind nicht erlaubt (Stand Firmware v 1.22).

Parameter:

```
[-{l|L[a]} <listfile>] [-r [-f "format string with %s as place holder"]] [-s[s]] [8.3] <path>
```

```
ls <path>
```

Ausgabe des Verzeichnis oder einer Dateiinformation.

Beispiel 1:

```
ls c:/sys/autoexec.sh
AUTOEXEC.SH          155  16.12.27 10:08:12  16.02.26 17:02:40.140  autoexec.sh
```

Ausgabespalten:
1. Dateiname im 8.3 Format
2. Dateilänge in Byte
3. Datum der Dateierzeugung
4. Uhrzeit der Dateierzeugung
5. Datum des letzten Schreibzugriffs
6. Uhrzeit des letzten Schreibzugriffs mit Millisekunden
7. langer Dateiname

Beispiel 2:

```
ls c:/data
2016               <dir>  16.02.28 10:03:14  16.02.28 10:03:14.000
2017               <dir>  17.01.18 15:52:32  17.01.18 15:52:33.930
            0 bytes in    0 files,    2 directories, sect   3447552, cluster     53614
```

Bei Verzeichnissen wird zusätzlich eine Zusammenfassung ausgegeben.

Siehe f.

Befehle im Detail

ls	Datei- und Verzeichnisinformationen listen

„`ls <path>`„ f.:

Ausgabespalten:
1. Summe der Filelänge aller Files in dem Verzeichnis
2. Anzahl der Files in dem Verzeichnis
3. Anzahl der Verzeichnisse in dem Verzeichnis
4. Sektornummer des ersten Sektors des Verzeichnisses
5. Clusternummer des ersten Clusters des Verzeichnisses

Beispiel 3:

```
echo hello >pipe1
ls pipe1
pipe1           7          0
```

Ausgabespalten:
1. Name
2. Anzahl der Zeichen, die noch nicht ausgelesen sind
3. aktueller Dateizugriffsoffset

Hinweis: als Pfadtrennzeichen kann der Slash ‚**/**‘ oder Backslash ‚****‘ verwendet werden.

`8.3`

Mit diesem optionalen Parameter werden beim Listing die langen Dateinamen weggelassen.

`-s` oder `-ss`

Mit dem optionalen Parameter ‚**-s**‘ werden nur die einzelnen Zusammenfassungen der Verzeichnisse ausgegeben. Mit dem Schalter ‚**-ss**‘ wird nur eine Gesamtzusammenfassung über alle Dateien und Verzeichnisse angezeigt. Der Schalter ‚**-ss**‘ unterscheidet sich vom Schalter ‚**-s**‘ nur im Zusammenhang mit dem Schalter ‚**-r**‘.

Beispiel:

```
ls -r -ss c:/

summary: c:/
   3063916455 bytes in 4500 files,  221 directories
```

`-r`

Der Schalter ‚**-r**‘ bewirkt ein rekursives Listen der Verzeichnisse und deren Unterverzeichnisse ab dem angegebenen.

Befehle im Detail

ls	Datei- und Verzeichnisinformationen listen

```
-f "format string with %s as place holder"
```

Mit dem Schalter ‚**-f**' werden nur Files ausgegeben und die Ausgabe kann variabel formatiert werden. Der Text innerhalb der „...“ ist beliebig. Das Sonderzeichen ‚**%s**' wird bei der Ausgabe durch den vollständigen Dateinamen mit vorangestellter Pfadangabe ersetzt.

Wenn der Schalter ‚**8.3**' angegeben wird, dann wird nur der kurze (**8.3**) Dateiname ausgegeben, anderenfalls der lange Dateiname.

Der Schalter ‚**-f**' ist besonders geeignet, Skripte zu erzeugen, die mit den gelisteten Dateien operieren.

Beispiel:

```
ls -f "@csm %s >>c:/tmp/config-checksums" c:/sys >c:/tmp/do-sys-csm
```

Das File c:/tmp/do-sys-csm hat dann beispielhaft folgenden Inhalt:

```
@csm c:/sys/AUTOEXEC.SH >>c:/tmp/config-checksums
@csm c:/sys/BESTPOS.CFG >>c:/tmp/config-checksums
@csm c:/sys/BLTH_CFG.SH >>c:/tmp/config-checksums
@csm c:/sys/FIRSTFIX.SH >>c:/tmp/config-checksums
@csm c:/sys/GPS.CFG >>c:/tmp/config-checksums
@csm c:/sys/NTRIP.CFG >>c:/tmp/config-checksums
@csm c:/sys/UPLOAD.CFG >>c:/tmp/config-checksums
@csm c:/sys/CRONTAB.CFG >>c:/tmp/config-checksums
@csm c:/sys/bluetooth-cfg.sh >>c:/tmp/config-checksums
@csm c:/sys/NETWORK.CFG >>c:/tmp/config-checksums
```

Nach dem das Skript c:/tmp/do-sys-csm gestartet wurde, hat die erzeugt Datei c:/tmp/config-checksums folgenden beispielhaften Inhalt:

```
>cat c:/tmp/config-checksums
csm 0x820df6c9  crc 0x29f00921  len        155  file c:/sys/AUTOEXEC.SH
csm 0x6f2a60ea  crc 0xb799ce50  len         75  file c:/sys/BESTPOS.CFG
csm 0xf57010c1  crc 0x16cb2af6  len        177  file c:/sys/BLTH_CFG.SH
csm 0xdd69804c  crc 0x15e09671  len        350  file c:/sys/FIRSTFIX.SH
csm 0xdd364bb5  crc 0x510a9197  len       1493  file c:/sys/GPS.CFG
csm 0x210bad1f  crc 0x6403b52b  len        806  file c:/sys/NTRIP.CFG
csm 0xea537c12  crc 0x8717c131  len        369  file c:/sys/UPLOAD.CFG
csm 0xa21d2729  crc 0x3fedaf5a  len         53  file c:/sys/CRONTAB.CFG
csm 0x7f14c5fc  crc 0xbd0da64a  len        527  file c:/sys/bluetooth-cfg.sh
csm 0xf835f5fd  crc 0xd6450fcc  len        620  file c:/sys/NETWORK.CFG
```

Vom Ergebnisfile c:/tmp/config-checksums kann natürlich ebenso eine CRC bzw. Checksumme berechnet werden. Auf diese Weise könnte die Konsistenz bzw. Unverändertheit von Files bzw. Verzeichnissen hinreichend geprüft werden.

Befehle im Detail

ls	Datei- und Verzeichnisinformationen listen

```
-{l|L[a]} <listfile>
```

Die Schalter ‚**-l**', ‚**-L**' und ‚**-La**' schreiben das Listing entsprechend der weiteren Parametrisierung in das angegebene Listfile.

Wird der Schalter ‚**-l**' verwendet, dann wird das Listfile nur erzeugt, wenn es noch **nicht** existiert. Bei dem Schalter ‚**-L**' wird ein eventuell vorhandenes gleichnamiges File überschrieben. Mit ‚**-La**' werden die Ausgaben an ein eventuell vorhandenes gleichnamiges File angehängt oder es wird erzeugt, wenn es noch nicht existiert.

Der Schalter ‚**-l**' kann z. B. zum effektiven FTP-Upload von Datenfiles über das GSM-Modem verwendet werden.

Beispiel:

```
ls -l c:/tmp/upload-all -r -f "@ftp -s -w put  -rm ! %s" c:/data/2017/03 && c:/tmp/upload-all; rm c:/tmp/upload-all
```

Es wird das File upload-all erzeugt, wenn es noch nicht existiert. Das File enthält Kommandozeilen mit je einem ftp-Kommando pro rekursiv in **c:/data/2017/03** gefundene Datei. Zu beachten ist der Schalter ‚**-w**' im ftp-Kommando, da er das Kommando auf die Beendigung des Uploads warten lässt. Wurde das Kommandofile erfolgreich erzeugt, dann wird es ausgeführt und anschließend gelöscht. In der Zeit der Ausführung könnte dieses kombinierte ls-Kommando wieder aufgerufen werden, da aber der Schalter -l angegeben wurde, bricht es ab, solange upload-all existiert. Um das Kommando für alle Fälle ausführbar zu machen, sollte

`‚rm c:/tmp/upload-all'` in die autoexec.sh aufgenommen werden.

Befehle im Detail

mkdir	Verzeichnis im FAT-Dateisystem erstellen

Ein Verzeichnis im FAT-Dateisystem anlegen.

Das angegebene Verzeichnis wird im angegebenen Pfad angelegt. Im Standardaufruf werden Verzeichnisse, die im angegebenen Pfad nicht existieren ebenfalls mit angelegt. Der Verzeichnis- und/oder Pfadname kann lange Namen, die auch Leerzeichen enthalten können, enthalten. Sind Leerzeichen im Pfadnamen enthalten, dann muss der Name in **"..."** eingeschlossen werden. Konnte das Verzeichnis nicht angelegt werden, dann wird ein Fehlerwert ungleich 0 zurückgegeben. Wurde es angelegt bzw. wenn es schon existiert, dann wird der Wert 0 zurückgegeben. Diese Eigenschaft lässt sich in einer bedingten Kommandoausführung anwenden.

Parameter:

```
[-e] <path>/<dir>   // -e create <dir> in an existing <path> only, else create full <path> and <dir>
```

`-e`

Das Verzeichnis **<dir>** wird nur angelegt, wenn alle Verzeichnisse in **<path>** existieren.

Befehle im Detail

mv	File im FAT-Dateisystem umbenennen oder verschieben

Das Kommando ‚**mv**' dient zum Umbenennen oder Verschieben von Dateien oder zum Umbenennen oder Verschieben von Verzeichnissen.

Beim Umbenennen sind alle Namenskonventionen erlaubt (langer Name, Leerzeichen im Namen, kurzer 8.3 Name). Sind Leerzeichen im Namen oder Pfad enthalten, dann muss der gesamte Namne in **"..."** eingeschlossen werden.

Beim Verschieben von Verzeichnissen ist darauf zu achten, dass der Zielpfadname mit ‚/' abgeschlossen wird um zu unterscheiden , dass das Verzeichnis dorthin verschoben werden soll. Anderenfalls wird es umbenannt.

Der Unterschied zwischen Verzeichnis umbenennen und verschieben wird durch das abschliessende ‚**/**' am Zielpfad angezeigt.

Das Kommando hat einen Rückkehrwert von 0 bei fehlerfreier Ausführungen, anderenfalls wir ein Wert ungleich 0 zurückgegeben. diese Eigenschaft lässt sich in einer bedingten Kommandoausführung verwenden.

Anstelle des ‚**/**' als Trennzeichen kann auch ‚****' verwendet werden.

Parameter:

```
<old_path>/<oldname> [<new_path>/][<newname>]     / at end means remain <oldname>
```

Beispiel 1:

```
mv c:/data data.bak
```

Das Verzeichnis data wird in **‚data.bak'** umbenannt.

Beispiel 2:

```
mv c:/data/2016 c:/data/2016.bak
```

Das Verzeichnis ‚**2016'** wird in ‚**2016.bak**' umbenannt.

Beispiel 3:

```
mv c:/data/2016.bak c:/tmp/
```

Das Verzeichnis ‚**2016.bak'**, das in **‚c:/data'** liegt, wird nach ‚**c:/tmp/2016.bak**' verschoben.

Beim Verschieben ist der Zielpfad mit ‚**/**' abzuschließen.

Befehle im Detail

ntrip	Korrekturdatenempfang via GSM-Modem steuern

Mit diesem Kommando werden Korrekturdaten über das GSM-Modem empfangen. Die Korrekturdaten werden von einem NTRIP-Server empfangen. Das Kommando wird zum Öffnen, Schließen und zum Steuern der Verbindung verwendet.

Eine NTRIP-Verbindung besteht in der Regel aus zwei Datenströmen. Der erste ist der Korrekturdatenstrom zum Gerät, der zum GPS-Board weitergeleitet wird. Der zweite ist ein Datenstrom aus NMEA-GPGGA-Datensätzen, der zum NTRIP-Server geht. Mit diesem wird, wenn der NTRIP-Server (bzw. dessen Mountpoint) über diese Fähigkeiten verfügt, der Korrekturdatenstrom auf die aktuelle Position, die im GPGGA steht, verbessert berechnet werden.

Die notwendigen Angaben zur Einwahl in das GSM-Netz stehen in **$(syspath)upload.cfg** (in der Regel hat **$(syspath)** den Inhalt ‚**c:/sys/**'). Die Angaben zum verwendeten NTRIP-Server (IP-Adresse, Mountpoint, Login und Passwort) stehen in **$(syspath)ntrip.cfg**.

Parameter:

```
  [-v] [-s] open [{@|!}<cfg_file> | <ip_adr>:<port>] [-rx[a] <file>] [-tx {<file>|gpgga-tx}]
| [-v] close [wait [<wait_s>]]   ohne <wait_s> -> forever
| stat
| echo [on|off]
| -rx[a] {<file>|close}      received data   write to   ..., a -> append
| -tx    {<file>|close}      data to be tx'd read  from ... (default: Stdio tee pipe1, pipe1 becomes tx'd)
```

-v

Bei der Ausführung werden zusätzliche Informationen während der Verbindung ausgegeben (verbose). Diese können später wieder mit ‚**gsm echo off**' und ‚**ntrip echo off**' abgeschaltet werden. Der Parameter ist nur für Diagnosezwecke erforderlich.

-s

Während der gesamten Verbindungszeit werden keine Verbindungs- bzw. Diagnoseinformationen ausgegeben (silent). Informationen zur übertragenen Datenmenge in beide Richtungen können jederzeit mit dem Kommando ‚**ntrip stat**' ausgegeben werden.

Befehle im Detail

ntrip	Korrekturdatenempfang via GSM-Modem steuern

ntrip [-s] open

Das ist das Standardkommando zur Herstellung einer NTRIP-Verbindung.

Hierbei werden implizit die RTK-Daten nach **gps2** übertragen und der GPGGA-Datensatz wird alle 20 s zum NTRIP-Server übertragen.

Die Verbindungs- und Einwahlinformationen werden aus ‚**upload.cfg**' und ‚**ntrip.cfg**' gelesen. Wenn Daten übertragen werden, werden zusätzlich Diagnoseinformationen (übertragene Datenmengen) ausgegeben.

Beispiel (Ausschnitt):

```
 >   140 s   400     59969
 >   141 s   400     60369
<    141 s    74       681
 >   143 s   857     61226
 >   144 s   400     61626
 >   145 s   400     62026
 >   146 s   400     62426
```

Dabei bedeuten ‚**>**' eingehende Daten und ‚**<**' ausgehende Daten.

Es folgen die Dauer der Verbindung in Sekunden, Datenmenge seit letzter Ausgabe und gesamte Datenmenge in der jeweiligen Richtung.

Mit dem Schalter ‚**-s**' (silent) wird die Diagnoseausgabe von Beginn an unterdrückt. Die Diagnoseausgabe kann jederzeit mit

‚**ntrip echo on**'
‚**ntrip echo off**'

an- bzw. abgeschaltet werden.

---

ntrip open {@|!}<cfg_file>

Anstelle des Files ‚c:/sys/ntrip.cfg' wird <cfg_file> für die Spezifizierung der NTRIP-Verbindung verwendet.

---

ntrip open <ip_adr>:<port> -tx <file>

Anstelle der im File ‚c:/sys/ntrip.cfg' angegebenen IP-Adresse und Portnummer werden die hier angegebenen verwendet. Die Verbindungsspezifikationen (Mountpoint, Protokoll, User und Passwort) müssen dann separat über das im Parameter **-tx <file>** angegebene File übertragen werden. Nach der Herstellung der Verbindung kann wieder auf die GPGGA-Übermittlung umgeschaltet werden mit ‚**ntrip -tx gpgga-tx**'.

Befehle im Detail

ntrip	Korrekturdatenempfang via GSM-Modem steuern

```
ntrip -rx[a] <file>
```

Anstelle der impliziten Übertragung der RTK-Daten nach **gps2** werden die Daten in das File **<file>** geschrieben. Das kann eine COM-Schnittstelle (**coma, comb, comc , gps1, gps2** oder **blth**), ein TCP-Stream (**ips..., icom...**), die CAN-Schnittstelle (**can1...**), eine Pipe (**pipe1, pipe2** oder **pipe3**), ein Memoryfile (**mem-...-w**) oder ein reguläres File im FAT-Dateisystem sein.

Mit

```
ntrip -rx close
```

wird das Ausgabefile geschlossen. Bei ‚**ntrip -rx <file>**' wird das alte File implizit geschlossen.

```
[-tx {<file>|close|gpgga-tx}]
```

Der optionale Parameter ermöglicht die Einstellung eines Files, aus dem periodisch Daten zum NTRIP-Server übertragen werden. Per Default (ohne **-tx**) sind das GPGGA-Datensätze, die im Zyklus von 20 s übertragen werden. Wenn eine abweichende Einstellung ‚**ntrip -tx file_xyz**' gemacht wurde, dann kann mit

```
ntrip -tx gpgga-tx
```

wieder auf die als Default eingestellte GPGGA-Übertragung zurückgeschaltet werden.

Es ist zu beachten, dass, wenn der Schalter bei ‚**ntrip open -tx ...**' angegeben wird, auch die Login-Daten im angegebenen File erwartet werden.

Sollen keine GPGGA-Datensätze und nichts weiter als die Login-Daten übertragen werden, dann ist folgende Kommandosequenz auszuführen:

```
ntrip open; ntrip -tx close
```

Mit

```
ntrip -tx close
```

wird das File geschlossen und es werden keine Daten zum NTRIP-Server übertragen und wenn vorher ‚**ntrip echo on**' eingeschaltet war, dann erscheint folgende Zusammenfassung:

```
TCPio terminate signal received
 ------------------------------------
 >   102 s     0     61659    605 bps
<    102 s     0        89      1 bps
ntrip: success!
```

Befehle im Detail

ntrip	Korrekturdatenempfang via GSM-Modem steuern

ntrip stat

Es wird eine Zwischeninformation über das Ein- und Ausgabefile, die Datenrichtung, die Verbindungsdauer, die übertragene Datenmenge seit letzter Abfrage, die gesamte übertragene Datenmenge und die Datenrate in Byte/s ausgegeben:

```
ntrip: received       129721 bytes,    129721 written to com6
ntrip: transmitted      1273 bytes,      1273 read  from GPGGA transmitter
 >    306 s  2237    129721    424 bps
<     306 s    74      1273      4 bps
```

Der Status ist auch noch nach dem Schließen der NTRIP-Verbindung abrufbar.

ntrip close [wait [<wait_s>]]

Mit diesem Kommando wird eine NTRIP-Verbindung geschlossen. Ohne den optionalen Schalter ‚**wait**' kehrt das Kommando sofort zurück. Es wartet also nicht auf die Beendigung des GSM-Verbindungstasks. Mit ‚**wait**' ohne Sekundenangabe wird gewartet bis das Kommando beendet ist und mit Sekundenangabe wird maximal die angegebene Zeit gewartet.

ntrip echo [on|off]

Mit diesem Kommando wird die periodische Ausgabe von Zwischeninformationen an- oder ausgeschaltet bzw. der on/off-Zustand mitgeteilt.
Die Ausgabe erfolgt immer, wenn sich die empfangene oder gesendete Datenmenge ändert, jedoch nicht häufiger als einmal pro Sekunde.

Befehle im Detail

peb, pes, pew	Auslesen von Speicheradressen (peek byte, peek short, peek word)

Mit diesem Kommando werden Daten von beleibiegen Speicheradressen ausgelesen und hexadezimal ausgegeben. Da bei dem verwendeten Cortex M4 Prozessor auch sämtliche Peripherie memory mapped ist, also im Speicheradressraum ansprechbar ist, kann das Kommando auch zum Auslesen von Peripherieregistern verwendet werden.

Mit ‚peb' werden die Werte als 8-Bit-Werte ausgelesen und ausgegeben.
Mit ‚pes' werden die Werte als 16-Bit-Werte ausgelesen und ausgegeben.
Mit ‚pew' werden die Werte als 32-Bit-Werte ausgelesen und ausgegeben.

Es dient dazu, die Verwendungsmöglichkeit des ppmOS auf Fälle zu erweitern, für die keine vordefinierten Kommandos bzw. Unterprogramme implementiert sind.

Besondere Bedeutung hat das Auslesen von allgemeinen Ein-/Ausgabeports (GPIO) im Zusammenhang mit einer bedingten Kommandoausführung.

Beispiel 1:

```
peb 0x8000000 20

memb: 08000000  00 ff 00 10 c1 05 00 08 41 06 00 08 41 06 00 08  ........A...A...
memb: 08000010  41 06 00 08                                      A...


pes 0x8000000 20

mems: 08000000  ff00 1000 05c1 0800 0641 0800 0641 0800  ........A...A...
mems: 08000010  0641 0800 0641 0800 0641 0800 0000 0000  A...A...A.......
mems: 08000020  0000 0000 0000 0000                      ........


pew 0x8000000 20

memw: 08000000  1000ff00 080005c1 08000641 08000641  ........A...A...
memw: 08000010  08000641 08000641 08000641 00000000  A...A...A.......
memw: 08000020  00000000 00000000 00000000 08033be1  ............;..
memw: 08000030  08000641 00000000 08000641 0800adf9  A.......A.......
memw: 08000040  08000641 080056dd 08000641 0800ae21  A....V..A...!...
```

Parameter:

```
  [-pmhas] <adr> [ <cnt> [<obj_per_line>]]  -pm0has omit: p->prefix, m->memadr, h->hex, a->ascii, s->no spaces
| [-s] p{a|...|k} [-m <mask>]
| [-s] p{a|...|k}.{0|1|...|15}
```

Befehle im Detail

peb, pes, pew	Auslesen von Speicheradressen (peek byte, peek short, peek word)

-pm0has

Mit diesen optionalen Kombinationsschalter lassen sich einzelne Ausgabefelder unterdrücken.

Jeder Buchstabe steht für ein Ausgabefeld

p	Ein eventueller Prefix (‚**memb:**', ‚**mems:**' bzw. ‚**memw:**' wird weggelassen.
m	Die Speicheradresse bzw. der Bytezähler wird weggelassen.
0	Die angezeigte Speicheradresse beginnt bei 0 und nicht bei der tatsächlichen.
h	Die Hexadezimale Form der Ausgabe wird weggelassen.
a	Der ASCII-Spalte wird weggelassen.
s	Die Leerzeichen zwischen den Hex-Werten wird weggelassen. Leerzeichen (je 2) zwischen Adresse bzw. Bytezähler und Hex-Spalte sowie zwischen Hex-Spalte und ASCII-Spalte werden immer ausgegeben.

Beispiel 2:

```
pew -pas 0x8000000 20

08000000  1000ff00080005c10800064108000641
08000010  0800064108000641080006410000000
08000020  00000000000000000000000008033be1
08000030  0800064100000000080006410800adf9
08000040  08000641080056dd080006410800ae21
```

<adr>

Mindestens dieser Parameter ist anzugeben. Er kann als Dezimalzahl oder mit vorangestelltem ‚**0x**' als Hexadezimalzahl angegeben werden.

<cnt>

Dieser optionale Parameter gibt die Anzahl der auszugebenden Werte an. Er kann als Dezimalzahl oder mit vorangestelltem ‚**0x**' als Hexadezimalzahl angegeben werden.

<obj_per_line>

Dieser optionale Wert legt die Anzahl der Bytes/Shorts/Words, die auf einer Zeile ausgegeben werden, fest. Ohne Angabe dieses Wertes werden 16 Bytes, 8 Shorts bzw. 4 Words pro Ausgabezeile angenommen. Der Wert `<obj_per_line>` kann zwischen 1 und 64 Bytes, 32 Shorts bzw. 16 Words liegen.

Damit kann aus Kenntnis der Länge einer Datenstruktur eine deutlich besser lesbare Ausgabe erzeugt werden.

Befehle im Detail

peb, pes, pew	Auslesen von Speicheradressen (peek byte, peek short, peek word)

```
peb [-s] p{a|...|k} [-m <mask>]
```

Mit dieser Kommandovariante können die 16 Bit breiten Ein-/Ausgabeports des Microcontrollers gelesen werden. Dabei steht der Buchstabe hinter p für den Port (a bis k). Wird der optionale Parameter ‚**-m <mask>**' angegeben, dann wird der gelesene Wert mit **<mask>** bitweise UND-verknüpft bevor er ausgegeben wird. Damit lassen sich andere nicht relevante Portleitungen für die Ausgabe ausblenden.

Der Rückgabewert des Kommandos ist der Portwert bzw. der mit **<mask>** bitweise UND-verknüpfte Portwert. Auf diese Weise lässt sich eine Gruppe von IO-Leitungen auswerten. Der Rückgabewert kann in einer bedingten Kommandoausführung verwendet werden.

Mit dem optionalen Schalter ‚**-s**' wird die gesamte Ausgabe unterdrückt (silent). Dieser Schalter wird bevorzugt bei der bedingten Kommandoausführung angewendet.

Beispiel 3:

```
@peb -s pb -m 0x3000 && @echo beide Eingänge der Event-IO-Buchse 1 sind low
```

Beispiel 4:

```
@peb -s pe -m 0xa00 && @echo beide Ausgänge der Event-IO-Buchse 1 sind low
```

```
peb [-s] p{a|...|k}.{0|1|...|15}
```

Mit dieser Kommandovariante kann eine Ein-/Ausgabeleitung des angegebenen Ports gelesen werden. Dabei steht der Buchstabe hinter **p** für den Port (**a** bis **k**) und die Zahl 0...15 für das Bit im Port.

Der Rückgabewert des Kommandos ist der Bitwert der IO-Leitung. Der Rückgabewert kann in einer bedingten Kommandoausführung verwendet werden.

Mit dem optionalen Schalter ‚**-s**' wird die gesamte Ausgabe unterdrückt (silent). Dieser Schalter wird bevorzugt bei der bedingten Kommandoausführung angewendet.

Beispiel 4:

```
@peb -s pb.12 && @echo Eingang 1 der Event-IO-Buchse 1 ist low
```

ist gleichbedeutend mit

```
@set io in1.1 >null && @echo ist low
```

Befehle im Detail

ping	TCP ping Befehl über Ethernet

Das Kommando ‚**ping**' ist ein Diagnosekommando, mit dem überprüft werden kann, ob der angegebene Host im Netzwerk erreichbar ist. Es wird eine Adressauflösung durchgeführt, falls eine URL anstelle einer IP-Adresse abgegeben wird.

Es wird die IP-Adresse und die gesamte Paketumlaufzeit (RTT) ausgegeben. Standardmäßig werden 3 Abfragen hintereinander ausgeführt.

Es ist identisch zu dem meist auf allen Betriebssystemen vorhandenen Kommando. Für die erfolgreiche Abfrage muss

1. Das eigene Netzwerk initialisiert sein (geschieht implizit beim ersten Aufruf von ‚ping').
2. Bei Angabe einer URL, anstelle einer IP-Adresse, muss ein Server für die Namensauflösung (DNS) erreichbar sein.
3. Der entfernte Rechner muss erreichbar sein.
4. Der entfernte Rechner muss das Protokoll unterstützen (Ping muss implementiert sein).

Beispiel:

```
ping google.de

64 bytes from 216.58.208.35: icmp_seq=1  ttl=  time=0.027533 s
64 bytes from 216.58.208.35: icmp_seq=2  ttl=  time=0.027092 s
64 bytes from 216.58.208.35: icmp_seq=3  ttl=  time=0.027057 s
--- google.de ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2.118 s
rtt min/avg/max = 0.027057/0.027227/0.027533 s
```

Parameter:

```
[-s] [-c <count>] [-i <interval_s>] [-w <max_wait_s>] {<ip_addr>|<host_name>}  time can be float
```

```
-s
```

Wird der optionale Schalter ‚-s' angegeben, dann werden Ausgaben unterdrückt. Das ist dann sinnvoll, wenn der Rückgabewert des Kommandos in einer bedingten Kommandoausführung verwendet werden soll.

```
-c <count>
```

Es werden die angegebene Anzahl von Ping-Anfragen gesendet. Ohne Angabe dieses Parameters werden 3 Ping-Anfragen gesendet.

```
-i <interval_s>
```

Zwischen den Ping-Anfragen wird die angegebenen Zeit gewartet. Die Zahl kann eine Zahl mit Komma sein. Ohne Angabe dieses Parameters beträgt die Wartezeit 1 s.

Befehle im Detail

ping	TCP ping Befehl über Ethernet
`-w <max_wait_s>` Das ist die maximale Wartezeit zum Aufbau der Verbindung, falls das Ethernet-Modul nicht schon initialisiert wurde. Und es ist die maximale Wartezeit auf die Ping-Antwort. Ohne Angabe wird eine maximale Wartezeit von 5 s eingestellt.	
`<ip_addr>|<host_name>` Der Ziel-Host kann als URL (z. B. google.de) oder als IP-Adresse in Punktnotation (z. B. 192.168.2.1) angegeben werden. Wird er als URL angegeben, dann muss im Netz, in dem sich das Gerät befindet, ein DNS oder ein Gateway verfügbar sein.	

Befehle im Detail

pob	Datenbytes an Speicheradressen schreiben IO-Ports (GPIO) einstellen und beschreiben

Das Kommando hat drei Verwendungsmöglichkeiten:

- Es wird zum Schreiben von Datenbytes an beliebige Speicheradressen benutzt. Das Schreiben von beliebigen 8-Bit-Werten bzw. Datensätzen an beliebige Speicheradressen kann mit einem Wiederholzähler durchgeführt werden.
- Zum Setzen/Abfragen/Einstellen von IO-Leitungen der GPIO-Ports des Microcontrollers. Dies ist erforderlich, wenn noch keine vordefinierten Shell-Kommandos für eine spezielle Konfiguration vorhanden sind.
- Die aktuelle Konfiguration aller IO-Pins kann abgefragt werden.

Die GPIO-Portleitungen können mit diesem Kommando in ihrer Wirkrichtung (Ein-/Ausgang) umgeschaltet werden. **Dies muss mit größter Vorsicht geschehen, da bei falscher Datenrichtung IO-Leitungen ‚gegeneinander' arbeiten, also einen Kurzschluss verursachen, was zur Beschädigung des Geräts führen kann.**

Das Kommando pob richtet sich an die Anwender, die mit der Programmierung des Gerätes umfassend vertraut sind. Bei falscher Anwendung treten Fehlfunktionen im Gerät auf, die einen Neustart erfordern oder die im Extremfall eine Beschädigung zur Folge haben können.

Das Kommando wird auch zur Kodierung von Eingaben in das Base64-Format verwendet, welches für die Kodierung von Nutzername und Passwort bei einer NTRIP-Verbindung (s. S. 44 ntrip.cfg) benötigt wird.

Parameter:

```
[-s] [-x] {{<adr>|-f [-a] <file>} [-c <repeat_cnt>] | b64} "<string>"| <byte> [<byte> | "<string>"]
               | p                      [clk] [idr] [odr] [od]      print port settings
               | p                      [mode] [type] [altf]        print port settings
               | p                      info                        print ports pin config
               | p{a|...|k}             [idr|odr|info]              read all pins or print pin config
               | p{a|...|k}             clk [on|off]                switch or display port bus clock
               | p{a|...|k} [-m <msk>] [lo|hi|tgl|<val>] [-d]      -d 100 µs delayed read IDR else ODR
               | p{a|...|k}.{0|...|15} [lo|hi|tgl]       [-d]      set pin value
               | p{a|...|k}.{0|...|15} [idr|odr]                    read one pin
               | p{a|...|k}.{0|...|15} info                         print current settings
               | p{a|...|k}.{0|...|15} out [pup|pdn] [od] [lo|hi]  set pin to output
               | p{a|...|k}.{0|...|15} in  [pup|pdn]                set pin to input
               | p{a|...|k}.{0|...|15} af  [pup|pdn] [od] <af>     set pin alternate fct.
               | p{a|...|k}.{0|...|15} an  [pup|pdn]                set pin to analog input
               | p{a|...|k}.{0|...|15} spd [2|25|50|100]            MHz pin switching speed
```

-s

Bei der Ausführung des Kommandos werden Ausgaben unterdrückt, was bei sich oft wiederholender bedingter Kommandoausführung sinnvoll ist. In dem Fall wird der Kommandorückgabewert in einer bedingten Kommandoausführung verwendet.

-x

Mit diesem Schalter werden alle Zahlen als hexadezimal interpretiert und sie werden ohne das sonst obligatorische Prefix 0x angegeben. Das hat den Zweck, dass man auf eine Kommandozeile mehr Daten eingeben kann als sonst.

Beispiel: 0x23 wird mit 23 angegeben.

Daraus ergibt sich, dass keine dezimalen Zahlen angegeben werden dürfen, da keine Unterscheidung möglich ist.

Befehle im Detail

pob	Datenbytes an Speicheradressen schreiben IO-Ports (GPIO) einstellen und beschreiben

<adr>

Adresse, ab der die Daten geschrieben werden. Die Angabe ist eine 32-Bit Zahl. Sie kann dezimal oder mit vorangestelltem ‚0x' gemacht werden. Es ist darauf zu achten, dass bei dem verwendeten Microcontroller große Teile des 32-Bit Adressraumes nicht ansprechbar sind und ein Zugriff in ein Reset führt.

Im 32-Bit Adressraum sind alle Adressen des Flash-ROM, des sRAM und der Peripheriemodule (UART, DMA, IIC, ADC, DAC, SDMMC, usw.) untergebracht. Ohne genaue Kenntnis des Referenzhandbuches des Microcontrollers STM32F429 sind hier keine sinnvollen Eingaben zu bewerkstelligen.

Hier sollten nur Speicherbereiche genutzt werden, die nicht für das PPM-OS reserviert sind. Die zu programmierenden Daten werden als Textstring in "..." eingeschlossen und/oder als dezimale oder hexadezimale Zahlen angegeben. Die Begrenzung für die Menge der Daten ist die Längenbegrenzung einer Kommandozeile (350 Zeichen inkl. Kommandoname).
Es wird empfohlen, alternativ das Memoryfile ohne Speicheradressenangabe für temporäre Zwischenspeicher in folgendem Format zu verwenden: mem-<name>-<len>-p

-f [-a] <file>

Der Schalter –f dient dem Schreiben in ein File. Das optionale -a bedeutet Daten am Ende eines eventuell vorhandenen Files anhängen. Ohne -a wird ein eventuell vorhandenes File vor dem Schreiben von Daten gelöscht. Die Daten, die geschrieben werden, können als eine beliebig gemischte Aufzählung von Zeichenketten, also Zeichen eingeschlossen in "...", und Zahlen angegeben werden. Beispiel (s. hierzu S. 59 „memory file"):

```
pob -f coma "hello world!" 0x0d 0x0a "Have a nice day." 13 10
pob -f mem-temp-200-p "hello world!" 0x0d 0x0a "Have a nice day." 13 10
cat mem-temp
```

[-c <repeat_cnt>]

Mit dem optionalen Schalter -c kann für das Schreiben von beliebigen 8-Bit-Werten bzw. Datensätzen an beliebige Speicheradressen oder in Files im Sinne des ppmOS ein Wiederholzähler eingestellt werden.
Beispiel:

```
pob 0x20000000 -c 40 0
pob 0x20000000 "hello world!" 0xd 10 0
```

pob b64 "<string>"| <byte> [<byte> | "<string>"]

Kodierung von eingegebenen Werten in das Base64-Format.

Beispiel:

Eingabe: **pob b64 "Nutzer:Passwort"**

Ausgabe: **TnV0emVyOlBhc3N3b3J0**

Werden die kodierten Werte in einem File erwartet, dann kann mit der Ein-/Ausgabeumleitung gearbeitet werden. Beispiel Erstellung der c:/sys/ntrip.cfg: (der erste Teil der c:/sys/ntrip.cfg wurde schon anderweitig vorbereitet)

```
echo -n "Authorization: " >>c:/sys/ntrip.cfg
pob b64 "user:password" >>c:/sys/ntrip.cfg
echo '"' >>c:/sys/ntrip.cfg
```

Befehle im Detail

pob	Datenbytes an Speicheradressen schreiben IO-Ports (GPIO) einstellen und beschreiben

```
"<string>"| <byte> [<byte> | "<string>"]
```

Liste der zu schreibenden Bytes. Die Angaben können dezimal, mit vorangestelltem ‚**0x**' hexadezimal oder in **"..."** eingeklammerte beliebige Zeichenfolgen sein. Die Angaben können gemischt und wiederholt werden.

Beispiel:

```
pob -f coma "hello world!" 0x0d 0x0a "Have a nice day." 13 10
```

```
pob p [clk] [idr] [odr] [od]
pob p [mode] [type] [altf]
```

Dieses Kommando dient der Ausgabe der aktuellen Einstellung aller IO-Ports des Microcontrollers sowie der Port Bus Clock (**clk**). Abfrage der aktuellen Ausgabe und der anliegenden Signale:

```
pob p
```

Oder

```
pob p odr idr
```

```
pob ODR 0100 0200 0001 0082 0070 0804 0c22 0000 0000 0000 0000
pob IDR a547 02f4 0001 82ff 1074 0bc7 ac3a 0000 0000 0000 0000
```

ODR bedeutet ‚output data register'. Ein gesetztes Bit zeigt an, dass das Port Pin auf high gesetzt wurde. IDR bedeutet ‚input data register' und zeigt den Signalpegel an dem jeweiligen Pin an.

```
pob p od
```

```
pob OD  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

OD bedeutet ‚open drain' und ein gesetztes Bit zeigt an, dass das jeweilige Pin als Open-Drain-Output programmiert ist. Anderenfalls ist das Pin als Push-Pull-Output programmiert.

```
pob p mode
```

```
pob MODE a8018028 0aa50240 02aa0aa9 000a6aa4 54541540 05500fd0 00504415 00000000 00000000 00000000 00000000
```

Abfrage der aktuellen Einstellung der Datenrichtung bzw. der alternativen Verwendung des GPIO-Pin für ein Peripheriemodul (z. B. UART, ...). Hier ist die Information in 2 Bit pro Portleitung kodiert.
Es bedeuten:
00 Pin ist Eingang
01 Pin ist Ausgang
10 alternative Pin-Verwendung für ein Peripheriemodul
11 analoger Mode (zum AD-Wandler durchgeschaltet)

```
pob p type
```

```
pob TYPE     0000     0000     0000     0000     0000     0000     0000     0000     0000     0000     0000
```

Abfrage der aktuellen Einstellung des Ausgangstyps (push-pull → 0 oder open-drain → 1) des jeweiligen Pins (1 Bit pro Pin). Diese Abfrage ist gleich der Abfrage mit ‚**od**'. (Siehe f.)

Befehle im Detail

pob	Datenbytes an Speicheradressen schreiben IO-Ports (GPIO) einstellen und beschreiben

"`pob p [clk] [idr] [odr] [od]`
`pob p [mode] [type] [altf]`" f.:

```
pob p altf

pob AF_L b0000bb0 00000000 88bb55b0 07777c00 00000088 00000000 00000000 00000000 00000000 00000000 00000000
pob AF_H 00000000 00bbb500 000ccccc 00000077 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Abfrage der eingestellten alternativen Funktionen für alle Port-Pins. Eine alternative Funktion ist aktiviert, wenn in ‚pob MODE ...' die jeweils zwei Bits pro Pin den Wert 10 (binär) haben.

Der alternative Funktionscode ist ein 4-Bit-Wert pro Pin. Die Kodierung der alternativen Funktion ist dem Datenblatt des Microcontrollers STM32F427 zu entnehmen.

`pob p{a|…|k} info`

Anzeige von Port-Pin-Einstellungen. Es werden die Einstellungen aller Pins des angegebenen Ports (hier Port A ‚p**a**') ausgegeben (ohne Angabe des Ports werden die Einstellungen aller Pins von jedem Port angezeigt):

```
pa.00 in     -  nop -
pa.01 af11   pp nop 100MHz
pa.02 af11   pp nop 100MHz
pa.03 in     -  nop -
pa.04 in     -  nop -
pa.05 in     -  nop -
pa.06 in     -  nop -
pa.07 af11   pp nop 100MHz
pa.08 out    pp nop   2MHz
pa.09 in     -  nop -
pa.10 in     -  nop -
pa.11 in     -  nop -
pa.12 in     -  nop -
pa.13 af00   pp pup 100MHz
pa.14 af00   pp pdn   2MHz
pa.15 af00   pp pup   2MHz
```

Darin bedeuten:

in	Pin ist Eingang
out	Pin ist Ausgang
af..	Pin ist auf eine alternative Funktion programmiert, z. B. TX UART
pp	Das Ausgabe-Pin arbeitet in der push/pull-Einstellung
nop	Kein Pullup oder Pulldown-Widerstand ist aktiv
pup	Ein Pullup-Widerstand ist aktiv, was bei einem offenen Eingang den definierten Signalzustand high (1) gewährleistet.
pdn	Ein Pulldown-Widerstand ist aktiv, was bei einem offenen Eingang den definierten Signalzustand low (0) gewährleistet.
2MHz, 100MHz	das Ausgabepin ist in der Lage, Zustandswechsel in der angegebenen Frequenz-normgerecht auszugeben. Die Einstellung wird mit 'pob p... clk {

Befehle im Detail

pob	Datenbytes an Speicheradressen schreiben IO-Ports (GPIO) einstellen und beschreiben

```
pob p{a|...|k} [idr|odr|info]
```

Einen Port abfragen. Der Schalter ‚**idr**' oder keine Angabe bedeutet Auslesen des ‚**input data register**'. Die Angabe ‚**odr**' bedeutet Auslesen des ‚**output data register**'.

Der Unterschied besteht darin, dass ‚**idr**' den Zustand der tatsächlichen Spannung am Pin wiedergibt (high oder low) und ‚**odr**' den eingestellten (gewünschten) Zustand, falls das Pin Ausgang ist.

Es kommt bei der Abfrage eines Ausgangspin nur dann zu einem unterschiedlichem Ergebnis, wenn durch äußere Belastung des Pins der eingestellte Zustand high oder low nicht erreicht wird.

Der angezeigte Wert ist auch der Rückgabewert des Kommandos, welcher für eine bedingten Kommandoausführung verwendet werden kann. Wird hinter ‚**pob**' noch der Schalter ‚**-s**' (silent) angegeben, dann wird die Ausgabe unterdrückt.

(Zu Schalter ‚**info**' siehe ‚pob p info'.)

Beispiele:

Alle Pins des Ports (hier Port B) lesen (die Ausgabe ist ein 16-Bit Wert):

Eingabe:

```
pob pb idr
```

Rückgabe:

```
pb 0x0280
```

Der Wert bedeutet, dass an Bit 7 und Bit 9 ein high Pegel anliegt, und an den anderen liegt low an.

Zustand der Ausgabepins des Ports (hier Port A) lesen (die Rückgabe ist ein 16-Bit Wert):

```
pob pa odr
```

```
pob p{a|...|k}          clk [on|off]
```

Anzeigen oder Ein– bzw. Abschalten der Port Bus Clock. Sämtliche Einstellungen und Ein-/Ausgaben von und zum GPIO-Port (general purpose IO) werden im GPIO-On-Chip-Modul des Microcontrollers realisiert. Die CPU des Microcontrollers liest oder schreibt nur Daten in das GPIO-Modul. Um diese 'teilautonome' Funktion realisieren zu können, braucht das GPIO-Modul permanent einen Bus-Takt. Dieser Takt wird nach Power On bei der Chip-Initialisierung eingeschaltet und er bleibt standardmäßig dauerhaft aktiv.

Wird der Takt abgeschaltet, dann kommen keine Ausgabebefehle mehr im GPIO-Modul an. Ein z.B. vom Programmablauf gesteuerter Pegelwechsel kommt nicht an, er wird quasi unterbunden.

Auf diese Weise könnte man in einen fest vorgegebenen Programmablauf (Programm läuft im Flash und ist nicht änderbar) per Skriptsteuerung eingreifen.

```
pob p{a|...|k} [-m <msk>] [lo|hi|tgl|<val>] [-d]
```

Einen Port einstellen. Es werden alle Pins des Ports oder eine Auswahl geschaltet. Mit dem optionalen Schalter ‚**-m <msk>**' werden die zu beeinflussenden Port-Pins eingestellt. Ein 1-Bit bedeutet einstellen und ein 0-Bit bedeutet Port-Pin bleibt unverändert. (f. nächste Seite)

Befehle im Detail

pob	Datenbytes an Speicheradressen schreiben IO-Ports (GPIO) einstellen und beschreiben

`„pob p{a|...|k} [-m <msk>] [lo|hi|tgl|<val>] [-d]“ f.:`

Mit ‚**lo**‘ werden die ausgewählten Port-Pins auf low gesetzt und ‚**hi**‘ bedeutet auf high setzen. Mit ‚**tgl**‘ wird der aktuelle Zustand umgekehrt, also high wird zu low und low wird zu high.

Die Angabe eines Wertes **<val>** stellt die betreffenden, auf Ausgabe programmierten Port-Pins entsprechend ein.

Nach der Einstellung wird das Output Data Register (odr) zur Kontrollausgabe erneut ausgelesen. Wird der Schalter ‚**-d**‘ angegeben, dann wird nicht das odr sondern das Input Data Register (idr) nach einer Verzögerung von 100 µs ausgelesen. Das ermöglicht die Kontrolle des tatsächlichen elektrischen Zustandes an den Port-Pins.

Der angezeigte Wert ist auch der Rückgabewert des Kommandos, welcher für eine bedingte Kommandoausführung verwendet werden kann. Wird hinter ‚**pob**‘ noch der Schalter ‚**-s**‘ (silent) angegeben, dann wird die Ausgabe unterdrückt.

Beispiele:

Die gelbe LED einschalten (sie ist lowaktiv):

```
pob pe -m 0x40 0
```

Die gelbe LED ausschalten (sie ist lowaktiv):

```
pob pe -m 0x40 0x40
```

`pob p{a|...|k}.{0|...|15} [lo|hi|tgl] [-d]`

Im Port **a ... k** das Port-Pin **0 ... 15** einstellen. Mit ‚**lo**‘ wird das Port-Pin auf low gesetzt und ‚**hi**‘ bedeutet auf high setzen. Mit ‚**tgl**‘ wird der aktuelle Zustand umgekehrt. High wird zu low und low wird zu high.

Nach der Einstellung wird das Output Data Register (odr) zur Kontrollausgabe erneut ausgelesen. Wird der Schalter ‚**-d**‘ angegeben, dann wird nicht das odr sondern das Input Data Register (idr) nach einer Verzögerung von 100 µs ausgelesen. Das ermöglicht die Kontrolle des tatsächlichen elektrischen Zustandes am Port-Pin.

Ohne `lo|hi|tgl` wird nur die aktuelle Einstellung angezeigt.

Der angezeigte Wert ist auch der Rückgabewert des Kommandos, welcher für eine bedingten Kommandoausführung verwendet werden kann. wird hinter ‚**pob**‘ noch der Schalter ‚**-s**‘ (silent) angegeben, dann wird die Ausgabe unterdrückt.

`pob p{a|...|k}.{0|...|15} [idr|odr]`

Im Port **a ... k** das Port-Pin **0 ... 15** abfragen. Der Schalter ‚**idr**‘ oder keine Angabe bedeutet Auslesen des ‚input data register‘. Die Angabe ‚odr‘ bedeutet Auslesen des ‚output data register‘.

Der Unterschied besteht darin, dass **idr** den Zustand der tatsächlichen Spannung am Pin wiedergibt (high oder low) und **odr** den eingestellten (gewünschten) Zustand, falls das Pin Ausgang ist.

Es kommt bei der Abfrage eines Ausgangspin nur dann zu einem unterschiedlichem Ergebnis, wenn durch äußere Belastung des Pins der eingestellte Zustand high oder low nicht erreicht wird.

Der angezeigte Wert ist auch der Rückgabewert des Kommandos, welcher für eine bedingten Kommandoausführung verwendet werden kann. wird hinter ‚**pob**‘ noch der Schalter ‚**-s**‘ (silent) angegeben, dann wird die Ausgabe unterdrückt.

Befehle im Detail

pob	Datenbytes an Speicheradressen schreiben IO-Ports (GPIO) einstellen und beschreiben

pob p{a|...|k}.{0|...|15} info

Die Funktionseinstellungen des angegebenen Port-Pins werden ausgegeben.

Beispiel:

pob pd.3 info

pd.3 af 7 pull no pull pp speed 2 MHz

Im Port ‚**D**' ist das Port-Pin **3** auf die alternative Funktion **7** (UART COMC CTS) eingestellt. Es ist kein Pullup- bzw. Pulldown-Widerstand aktiviert. Es ist ein push pull Ausgang, der beim Zustandswechsel die Flankensteilheit von äquivalenten 2 MHz aufweist.

---

pob p{a|...|k}.{0|...|15} out [pup|pdn] [od] [lo|hi]

Das angegebene Portpin wird auf Ausgang programmiert. Mit ‚**pup**' wird ein Pullup-Widerstand (typ. 40 kOhm) aktiviert. Mit ‚**pdn**' wird ein Pulldown-Widerstand (typ. 40 kOhm) aktiviert. Mit ‚**od**' ('open drain')wird der interne Pullup-Transistor der Pin-Ausgangsstufe deaktiviert. D. h. das Ausgangspin kann nur aktiv auf low ziehen.

Wird ‚**lo**' oder ‚**hi**' angegeben, dann wird der Ausgangspegel entsprechend gesetzt. Ohne Angabe bleibt der aktuelle Wert im ODR (Output Data Register) enthaltene Wert erhalten. Beispiel:

pob pe.6 out

=> Das Pin 6 im Port E (gelbe LED) wird auf Ausgang programmiert.

---

pob p{a|...|k}.{0|...|15} in [pup|pdn]

Das angegebene Port-Pin wird auf Eingang programmiert. Mit ‚**pup**' wird ein Pullup-Widerstand (typ. 40 kOhm) aktiviert. Mit ‚**pdn**' wird ein Pulldown-Widerstand (typ. 40 kOhm) aktiviert. Ohne Angabe von ‚**pup**' oder ‚**pdn**' bleibt das Pin hochohmig.

---

pob p{a|...|k}.{0|...|15} af [pup|pdn] [od] <af>

Für das angegebene Portpin wird die alternative Funktion mit Nummer **<af>** programmiert. Mit ‚**pup**' wird ein Pullup-Widerstand (typ. 40 kOhm) aktiviert. Mit ‚**pdn**' wird ein Pulldown-Widerstand (typ. 40 kOhm) aktiviert. Mit ‚**od**' wird der interne Pullup-Transistor der Pin-Ausgangsstufe deaktiviert. D. h. das Ausgangspin kann nur aktiv auf low ziehen. Ob das programmiert Portpin Ein- oder Ausgang ist, hängt von der jeweiligen alternativen Funktion ab. D. h. ein vorheriges einstellen ‚**in**' oder ‚**out**' ist nicht nötig.
Die Einstellungen der Flankensteilheit bei Ausgängen muss zur verwendeten alternativen Funktion passend programmiert werden.
Vor der Verwendung ist unbedingt das Reference Manual des Controller STM32F429 zu Rate zu ziehen. Bei fehlerhafter Programmierung kann das Gerät dauerhaft beschädigt und damit unbrauchbar werden.

Befehle im Detail

pob	Datenbytes an Speicheradressen schreiben IO-Ports (GPIO) einstellen und beschreiben

```
pob p{a|...|k}.{0|...|15} an [pup|pdn]
```

Das angegebene Port-Pin wird als analoger Eingang programmiert. Ohne weitere erforderliche Programmierung des dazugehörigen AD-Wandlers bleibt die Einstellung ohne Wirkung mit Ausnahme, dass das Port-Pin ein hochohmiger Eingang wird. Mit ‚**pup**' wird ein Pullup-Widerstand (typ. 40 kOhm) aktiviert. Mit ‚**pdn**' wird ein Pulldown-Widerstand (typ. 40 kOhm) aktiviert.

```
pob p{a|...|k}.{0|...|15} spd [2|25|50|100]
```

Falls das angegebene Portpin Ausgang ist, dann kann mit diesem Unterkommando die Flankensteilheit der Pegelwechsel eingestellt werden. Die anzugebende Zahl entspricht einer äquivalenten Steilheit in MHz. Z. B. sind die UART-Tx-Pins ebenso wie alle GPIO-Pins auf 2 MHz eingestellt. Hingegen sind die Pins des Ethernet- und des SD-Kartenmoduls auf 100 MHz eingestellt, da hier wegen der hohen Bitraten kleinere ‚**spd**'-Werte eine korrekte Funktion verhindern würde.
Die Firmware des Gerätes ist so programmiert, dass die Einstellung an den jeweiligen Pins immer so gering wie möglich ist um die elektromagnetische Störausstrahlung so gering wie möglich zu halten.

Befehle im Detail

pos, pow	Daten im Short oder Word-Format an Speicheradressen schreiben

Mit diesem Kommando können beliebige 16- bzw. 32-Bit-Werte bzw. Datensätze an beliebige Speicheradressen geschrieben werden. Das Schreiben kann mit einem Wiederholzähler programmiert werden. Der Adresszeiger wird bei einer Wiederholung weitergezählt.

Beide Kommandos gleichen sich in der Funktion, nur dass mit ‚**pos**' 16-Bit-Werte und mit ‚**pow**' 32-Bit-Werte geschrieben werden.

Beispielsweise werden mit ‚**pow 0x20000000 0**' 4 Byte geschrieben und mit ‚**pos 0x20000000 0**' zwei byte.

Der Microcontroller schreibt immer im ‚little endian'-Format, d. h. das Kommando ‚**pow 0x20000000 0x12345678**' erzeugt die folgende Speicherbelegung:

```
peb -a 0x20000000 4

memb: 20000000  78 56 34 12
```

aber mit

```
pew -a 0x20000000 1
```

Werden die Bytes wieder als zusammengehöriger 32-bit-Wert angezeigt:

```
memw: 20000000  12345678
```

Parameter:

```
pos <adr> [-c <repeat_cnt>] <short_to_poke> {<short_to_poke>}
pow <adr> [-c <repeat_cnt>] <word_to_poke> {<word_to_poke>}
```

<adr>

Adresse aber der die Werte geschrieben werden. Wird mit einem Wiederholzähler gearbeitet, dann wird der Adresszeiger immer weitergezählt.

Es ist zu beachten, dass der Zugriff auf nicht vorhandenen Speicher bzw. Peripherieregister zum Absturz der Software führt.

Ebenso kann beim Schreiben in bestimmte Peripherieregister eine Programmierung erzeugt werden, die das Gerät beschädigt und unbrauchbar macht.

Eine fundierte Kenntnis des Reference Manuals des verwendeten Microcontrollers STM32F429 ist also unbedingte Voraussetzung für die Anwendung dieser Kommandos.

-c <repeat_cnt>

Mit diesem optionalen Parameter werden die im Kommando angegebenen Werte wiederholt hintereinander geschrieben.

<short_to_poke> bzw. <word_to_poke>

Mindestens ein Wert ist anzugeben. Es können so viele Werte angegeben werden, wie auf die Kommandozeile (Länge der Kommandozeile max. 299 Zeichen) passen. Die Werte sind mit Leerzeichen oder TAB's zu trennen.

prog	Daten in den FlashROM programmieren, FlashROM löschen und Option-Bytes programmieren

Das Kommando wird verwendet um die Firmware im Flash-ROM des Microcontrollers zu speichern. Es kann auch zur Speicherung beliebiger anderer Daten im Flash-ROM verwendet werden.

Parameter:

```
 -u <file>
| {-ne|-e} <dest> [-m <src> <len> | [-o <offs>] [-l <len>] <file>] //-e erase only if no src is given
| tab               //print sector tab
| opt  [<word>]     //programm all option bytes in FLASH_OPTCR
| opt1 [<word>]     //programm all option bytes in FLASH_OPTCR1
```

```
prog -u <file>
```

Diese Form des Kommandos wird zum Speichern der Firmware im Flash-ROM verwendet. Im Flash-ROM (2 MByte) sind immer 2 Versionen (boot und main) gespeichert.

Dabei wird ein spezielles Fileformat in **<file>** vorausgesetzt. Das zu programmierende File wird im FAT-Dateisystem erwartet. Das Kommando holt die Informationen (Zieladresse, Länge und Prüfsumme) aus dem File und reagiert entsprechend. Vor der Programmierung wird das File geprüft. Es wird nur fortgesetzt, wenn die Prüfung fehlerfrei ist.

Im Flash-ROM des Microcontrollers (2 MByte) sind zwei identische Versionen der Firmware gespeichert. Vor dem Programmieren muss der Zielspeicherbereich gelöscht werden. Die aktuell laufende Firmware-Version darf sich nicht selbst löschen. Deshalb muss in diesem Fall in die zweite Version gewechselt werden. Dies geschieht über die implizite Ausführung des Kommandos ‚**call alternate <cmd>**' (siehe Beispiel 1 auf S. 211 und Kommando ‚**call**' S. 88). Wenn das implizite ‚**call alternate <cmd>**' fehlschlägt, wird das Kommando ‚**prog**' mit einer Fehlermeldung abgebrochen. Damit wird gesichert, dass ein Programmiervorgang über ‚**prog -u <file>**' mit höchst möglicher Sicherheit abläuft. Nach erfolgreicher Programmierung erfolgt ein Rücksprung in die neu programmierte Version.

Siehe f.

Befehle im Detail

prog	Daten in den FlashROM programmieren, FlashROM löschen und Option-Bytes programmieren

„**prog -u <file>**„ f.:

Beispiel 1:

```
prog -u c:/sys/40xx_2nd.bin

call alternate 'loop 1 "wait 2; prog -u c:/sys/40xx_2nd.bin && call alternate" &'
closing all files in c:
closing files finished
ppm40xx V 1.22 compiled Apr  4 2017 10:04:28 (168 MHz, CPUID 410fc241, GCC 6.2.1)
copyright ppm GmbH 2017     9F6 UID: 3334.3032.3532.470b.001c.002d '-....G252043'
---------------------------------- boot loader ----------------------------------
.
. (weitere Ausgaben während des Bootvorgangs weggelassen)
.

>loop 1 "wait 2; prog -u c:/sys/40xx_2nd.bin && call alternate"
>wait 2
start waiting 2000 ms
mounting file system c:
drv c: FAT32, cl 32 KB, sect 7774208, size 3878912 KB, ro rw, SDHC, crc 0, try 0, si 0
>c:/sys/autoexec.sh
-----------------------------------------------
.
. (weitere Ausgaben während der Ausführung der ‚autoexec.sh' weggelassen)
.
-----------------------------------------------
SD-card c: is inserted
end waiting
>prog -u c:/sys/40xx_2nd.bin
PROG: file crc is o.k.
PROG: erasing sector 16 (snr 20) @ 0x8110000
PROG: erasing sector 17 (snr 21) @ 0x8120000
PROG: erasing sector 18 (snr 22) @ 0x8140000
PROG: erasing sector 19 (snr 23) @ 0x8160000
PROG: ff test from 0x8110000 to 0x8170ea0 o.k.
PROG: dest 0x8110000 file c:/sys/40xx_2nd.bin, offset 0x0 (0), length 0x60ea4 (396964)
PROG: verify: crc is o.k.  2nd programm version
>call alternate
closing all files in c:
closing files finished
ppm40xx V 1.22 compiled Apr  4 2017 10:04:28 (168 MHz, CPUID 410fc241, GCC 6.2.1)
copyright ppm GmbH 2017     9F6 UID: 3334.3032.3532.470b.001c.002d '-....G252043'
-------------------------------------------------------------------------------
```

Bei der Programmierung der nicht aktiven Firmware-Version gibt es kein Reset. Das Update erfolgt im laufenden Betrieb, die aktive Firmware arbeitet während des Löschens und Neuprogrammierens ohne Besonderheit weiter.

Beispiel 2 Siehe f.

Befehle im Detail

prog	Daten in den FlashROM programmieren, FlashROM löschen und Option-Bytes programmieren

„**prog -u <file>**„ f.:

Beispiel 2:

```
prog -u c:/sys/40xx.bin

PROG: file crc is o.k.
PROG: erasing sector  0 (snr  0) @ 0x8000000
PROG: erasing sector  1 (snr  1) @ 0x8004000
PROG: erasing sector  2 (snr  2) @ 0x8008000
PROG: erasing sector  3 (snr  3) @ 0x800c000
PROG: erasing sector  4 (snr  4) @ 0x8010000
PROG: erasing sector  5 (snr  5) @ 0x8020000
PROG: erasing sector  6 (snr  6) @ 0x8040000
PROG: erasing sector  7 (snr  7) @ 0x8060000
PROG: ff test from 0x8000000 to 0x8060ea0 o.k.
PROG: dest 0x8000000 file c:/sys/40xx.bin, offset 0x0 (0), length 0x60ea4 (396964)
PROG: verify: crc is o.k.  boot loader
```

Mit ‚**ver -b**' und ‚**csm -m**' können beide Firmware-Versionen geprüft werden.

Befehle im Detail

prog	Daten in den FlashROM programmieren, FlashROM löschen und Option-Bytes programmieren

```
prog tab
```

Dieses Unterkommando gibt Auskunft (Adresse und Größe) über die einzelnen Sektoren im Flash-ROM.
Ausgabe:

```
PROG: table sector index:
PROG: six  0  adr 0x8000000  size  16 KB  snr  0
PROG: six  1  adr 0x8004000  size  16 KB  snr  1
PROG: six  2  adr 0x8008000  size  16 KB  snr  2
PROG: six  3  adr 0x800c000  size  16 KB  snr  3
PROG: six  4  adr 0x8010000  size  64 KB  snr  4
PROG: six  5  adr 0x8020000  size 128 KB  snr  5
PROG: six  6  adr 0x8040000  size 128 KB  snr  6
PROG: six  7  adr 0x8060000  size 128 KB  snr  7
PROG: six  8  adr 0x8080000  size 128 KB  snr  8
PROG: six  9  adr 0x80a0000  size 128 KB  snr  9
PROG: six 10  adr 0x80c0000  size 128 KB  snr 10
PROG: six 11  adr 0x80e0000  size 128 KB  snr 11
PROG: six 12  adr 0x8100000  size  16 KB  snr 16
PROG: six 13  adr 0x8104000  size  16 KB  snr 17
PROG: six 14  adr 0x8108000  size  16 KB  snr 18
PROG: six 15  adr 0x810c000  size  16 KB  snr 19
PROG: six 16  adr 0x8110000  size  64 KB  snr 20
PROG: six 17  adr 0x8120000  size 128 KB  snr 21
PROG: six 18  adr 0x8140000  size 128 KB  snr 22
PROG: six 19  adr 0x8160000  size 128 KB  snr 23
PROG: six 20  adr 0x8180000  size 128 KB  snr 24
PROG: six 21  adr 0x81a0000  size 128 KB  snr 25
PROG: six 22  adr 0x81c0000  size 128 KB  snr 26
PROG: six 23  adr 0x81e0000  size 128 KB  snr 27
```

```
prog opt  [<word>]
prog opt1 [<word>]
```

Mit diesen Unterkommandos werden die Optionbytes des Controllers angezeigt oder geändert.
Mit den Optionbytes werden grundlegende Einstellungen des Microcontrollers vorgenommen. Die Interpretation der ausgegebenen Werte setzt eine genaue Kenntnis des Reference Manual des STM32F429 voraus. Änderungen sollten hier nur vom Hersteller des Gerätes vorgenommen werden bzw. es sind alle notwendigen Einstellungen bereits bei Auslieferung gemacht worden.

Befehle im Detail

pwm	Pulsbreitenmoduliertes Signal erzeugen

Mit diesem Kommando lassen sich pulsbreitenmodulierte Signale an den Ausgängen out1.1, out1.2, out2.1 und out2.2 erzeugen. Damit können z. B. mit nachgeschalteter Filterung Digital-Analogausgänge realisiert werden oder es können Servoantriebe nach dem allgemeinen RC-Servostandard gesteuert werden.

In einer Spezialanwendung können vier Trigger-Signale erzeugt werden, die mit einer Auflösung von 6 ns eingestellt werden können.

Parameter:

```
  [init {event[1|2]|highspeed} [off] [onepulse] [hi|lo] [<per_s> [<hi_s>]] //dflt. per 5 ms
| start [once]
| isrcnt
| [<nr>]  //1 …

 [[-nc] [hi] <hi> [<per> [<prescl>]]]     -nc no check, <hi> und <per> in µs (float)
| per <per> [<prescl>]                    <per> in s (float)
| pscl <prescl>
| {min|max} [pos|<val_µs>]                <val_µs> (float)
| wide
| store | load
| stat [-s] [raw]
| echo [on|off]
```

```
pwm init highspeed ...
```

Diese Kommandovariante ist der Spezialanwendung für 4 synchrone Trigger-Signale vorbehalten. Dafür muss das Gerät intern einen Hardwarezusatz haben.

```
pwm init event[1|2] [off] [hi|lo] [<per_s> [<hi_s>]]
```

Mit dieser Kommandovariante wird ein PWM-Signal an den ausgewählten Eventausgängen erzeugt. Die Initialisierung bezieht sich auf 2 oder 4 Ausgangsleitungen. Mit ‚**event**' werden alle vier Ausgangsleitungen initialisiert, Mit ‚**event1**' die beiden von **out1** und mit ‚**event2**' die von **out2**.

Ein PWM-Signal ist durch eine Periodendauer und durch eine High-Dauer gekennzeichnet. Alle vier Ausgangsleitungen werden von demselben Timer gesteuert. Dieser Timer erhält einen Takt, der vom CPU-Clock, geteilt durch einen Vorteiler, abgeleitet wird. D. h. alle Timer-Ausgänge haben immer den Vorteiler der letzten Initialisierung.

Mit dem Zusatz ‚**off**' werden nur die Timer initialisiert, aber die Leitungen bleiben zunächst noch auf high falls ‚**hi**' oder ‚**hi**' nicht angegeben wurde. Erst mit ‚**pwm start**' wird das Signal an den Leitungen wirksam.

Zusätzlich zur Periode **<per_s>** und der Highdauer **<hi_s>** kann noch der Vorteiler **<prescl>** des Timers angegeben werden. Dabei ist zu beachten, dass alle Timer-Ausgänge von ein und demselben Vorteiler abgeleitet werden.

Beispiel 1: Siehe f.

Befehle im Detail

pwm	Pulsbreitenmoduliertes Signal erzeugen

"`pwm init event`" f.:

Beispiel 1:

```
pwm init event
```

Dies ist die Standardinitialisierung für Digitalservos. Auf allen Ausgängen erscheinen Highpulse mit einer Dauer von 1.5 ms (Neutral- oder Mittelstellung) mit einer Frequenz von 200 Hz.

Beispiel 2:

```
pwm init event off
pwm 1 1200; pwm 2 1300.2; pwm 3 1400.5; pwm 4 1600
pwm start
```

Dies ist wieder die Standardinitialisierung für Digitalservos (200 Hz, Nullimpuls 1.5 ms). Die Ausgänge sind zunächst alle high. Dann wird bei allen Ausgängen eine individuelle High-Dauer eingestellt. Und zum Schluss werden auf allen Ausgängen die Pulse ausgegeben.

Es ist zu beachten, dass hier die Eingabe in der Maßeinheit µs erfolgt, hingegen bei den anderen Eingaben in s. In jedem Fall können Zahlen mit oder ohne Dezimalpunkt oder in der wissenschaftlichen Notation (z. B. 1.52e-3) angegeben werden. Die jeweilige Maßeinheit kann der Kommandozeilenkurzhilfe entnommen werden.

Mit der Angabe ‚**hi**' oder ‚**lo**' wird die Pulsrichtung vorgegeben. Default ist hier der High-Puls (‚**hi**'). D. h. bei ‚**hi**' und Pulsbreite 0 würde sich ein durchgängiges Low-Signal ergeben. Bei ‚**lo**' und Pulsbreite 0 würde ein durchgängiges High-Signal erscheinen.

Beispiel 3:

```
pwm init event2 100e-6 50e-6
```

Es wird ein PWM-Signal mit 10 kHz und einem Tastverhältnis von 50 % auf beiden Ausgängen des Eventausgang 2 eingestellt. Mit einem RC-Filterglied, das die Rechteckpulse glättet, wird eine Spannung von ca. 1.7 V erzeugt. Es lässt sich somit ein analoges Signal um Bereich von 0 ... 3.5 V erzeugen. Die Auflösung beträgt dabei 14 Bit. Die Ausgangsspannung ist in ca. 16800 Stufen einstellbar.

Ein Ausgangssignal mit 1 MHz und einer Auflösung von 168 Stufen ist ebenfalls möglich. Dies ist ca. die obere Grenze.

Der Schalter `[onepulse]` ist dem High-Speed-Mode vorbehalten. Er ist im normalen PWM-Mode nicht anwendbar.

Befehle im Detail

pwm	Pulsbreitenmoduliertes Signal erzeugen

`pwm [1|2|3|4] [-nc] [hi] <hi>`

Mit diesem Kommando können die PWM-Pulsbreiten für jede Leitung einzeln oder zusammen für alle eingestellt werden.

Dies ist das ‚Arbeitskommando', es kann beliebig wiederholt werden, um dynamische Prozesse zu steuern. Wird die Ausgabe des Kommandos nicht benötigt, dann kann diese mit ‚**pwm [1|2|3|4] echo off**' dauerhaft unterdrückt werden. Dies ist besonders bei häufigen Änderungen sinnvoll.

Wird kein Ausgang angegeben, es werden also alle Leitungen auf denselben Wert eingestellt, dann ist der Schalter ‚**hi**' anzugeben.

Der Wert **<hi>** ist in µs-Einheiten anzugeben, gegebenenfalls auch mit Dezimalpunkt oder in der wissenschaftlichen Notation (z. B. 1600.5 oder 1.6005e3).

In der Standardservosteuerung wird das Signal mit 200 Hz ausgegeben und es sind für die Pulsbreite Standardlimits (900 µs ... 2100 µs) eingestellt. Eingaben außerhalb dieser Limits werden auf das jeweilige Limit begrenzt. Damit ist eine reguläre Funktion von Servos auch bei Fehleingaben sichergestellt. Soll ein beliebige Pulsbreite eingestellt werden, dann ist der Schalter ‚**-nc**' (no check) zu benutzen.

Wird bei der Initialisierung von der Standardperiode von 5 ms abgewichen oder soll der gesamte Tastverhältnisbereich von 0 % bis 100 % genutzt werden, dann muss immer ‚**-nc**' angegeben werden, um eine Begrenzung der Pulsbreite zu vermeiden.

`pwm per <psc> [<prescl>]`

Mit diesem Unterkommando kann die Periode des PWM-Signals geändert werden und optional der Vorteiler. Dieser ist ein 16-Bit-Wert.

Damit könnte man die Blinkfrequenz einer angeschlossenen Signal-LED je nach Bedarf ändern.

`pwm pscl <psc>`

Mit diesem Unterkommando kann der Vorteiler des Timers geändert werden. Der Vorteiler ist ein 16-Bit-Wert.

Die aktuelle Einstellung wird mit ‚**pwm stat raw**' angezeigt.

`pwm [1|2|3|4] {min|max} [pos|<val_µs>]`

Mit diesem Unterkommando kann in einer Servoanwendung die minimal und maximal mögliche Pulsbreite für alle oder einen Ausgang eingestellt werden. Die Verstellung der Grenzen mit **<val_µs>** ist nur im Bereich von 900 µs bis 2100 µs Pulsbreite möglich.

Mit ‚**pos**' wird der aktuell eingestellte PWM-Wert als der Min- oder Max-Wert übernommen. Dieser muss er vorher mit ‚`pwm [1|2|3|4] -nc <hi_µs>`' ohne Bereichslimitierung eingestellt worden sein.

Befehle im Detail

pwm	Pulsbreitenmoduliertes Signal erzeugen

```
pwm [1|2|3|4] wide
```

Die Bereichsgrenzen für eine gültige Pulsdauer werden auf 900 µs und 2100 µs gesetzt.

```
pwm [1|2|3|4] {store|load}
```

Die aktuellen Bereichsgrenzen für die Pulsbreite können in einem batteriegepufferten RAM dauerhaft gespeichert werden. Den Werten wird eine Prüfsumme hinzugefügt, die eine Veränderung erkennen lässt.

Mit '`pwm [1|2|3|4] load`' werden sie aus dem batteriegestützten RAM in den Arbeitsspeicher geladen. Dieses Kommando wird implizit mit ‚**pwm init event ...**' ausgeführt. Ein erfolgreiches Laden wird mit einem ‚o.k.' am ende der Zeilen angezeigt.

```
PWM:  1    0.0%   1500.0000 µs  lim: 1000.0 1800.0 µs (18000, 12000, 21600) o.k.
PWM:  2    0.0%   1500.0000 µs  lim:  969.3 2100.0 µs (18000, 11631, 25200) o.k.
PWM:  3    0.0%   1500.0000 µs  lim: 1050.0 1900.0 µs (18000, 12600, 22800) o.k.
PWM:  4    0.0%   1500.0000 µs  lim:  969.3 2100.0 µs (18000, 11631, 25200) o.k.
```

Anderenfalls erscheint ‚**error**' und es werden die Standardlimits (900µs ... 2100 µs) geladen.

```
pwm [1|2|3|4] stat [-s] [raw]
```

Die aktuellen Einstellungen anzeigen:

```
pwm stat

PWM:  1  -62.5%   1250.0000 µs  lim:  900.0 2100.0 µs (15000, 10800, 25200)
PWM:  2    0.0%   1500.0000 µs  lim:  900.0 2100.0 µs (18000, 10800, 25200)
PWM:  3 -150.0%    900.0000 µs  lim:  900.0  900.0 µs (10800, 10800, 10800)
PWM:  4    0.0%   1500.0000 µs  lim:  900.0 2100.0 µs (18000, 10800, 25200)
```

Die Prozentzahl bezieht sich auf folgende Werte:

```
    0%       1500 µs Pulsbreite
 100%1900 µs Pulsbreite
-100% 1100 µs Pulsbreite
```

Die Zahlen in Klammern sind die dazugehörigen dimensionslosen Timer-Registerwerte (alle 16-Bit).

Die ‚Short'-Version von ‚**pwm stat**'.

```
pwm stat -s

PWM:  1  -62.5%   1250.0000 µs
PWM:  2    0.0%   1500.0000 µs
PWM:  3 -150.0%    900.0000 µs
PWM:  4    0.0%   1500.0000 µs
```

Die Anzeige der Timer-Register: Siehe f.

Befehle im Detail

pwm	Pulsbreitenmoduliertes Signal erzeugen

"`pwm [1|2|3|4] stat [-s] [raw]`" f.:

Die Anzeige der Timer-Register:

```
pwm stat raw

TIM01.1: 0x40010000, PSC  14, ARR   60000, CCR     15000, T      5000.0000 µs, hi      1250.0000 µs (0.083333 µs 12000000.0000 Hz)
TIM01.2: 0x40010000, PSC  14, ARR   60000, CCR     18000, T      5000.0000 µs, hi      1500.0000 µs (0.083333 µs 12000000.0000 Hz)
TIM01.3: 0x40010000, PSC  14, ARR   60000, CCR     10800, T      5000.0000 µs, hi       900.0000 µs (0.083333 µs 12000000.0000 Hz)
TIM01.4: 0x40010000, PSC  14, ARR   60000, CCR     18000, T      5000.0000 µs, hi      1500.0000 µs (0.083333 µs 12000000.0000 Hz)
```

Es handelt sich hier also um den Timer 1 des Controllers mit seinen 4 Ausgängen. Das Prescalerregister (PSC) ist mit 14 geladen. Das Periodenregister (ARR) hat den Wert 60000. Am Ende sind in Klammern der ‚Timertick', also der kleinste Zeitschritt und davon der reziproke Wert, die Timerfrequenz (12 MHz), angegeben. Das Produkt aus PSC und Timerfrequenz ergibt die aktuelle Corefrequenz.

`pwm [1|2|3|4] echo [on|off]`

Nach der Eingabe zur Änderung der PWM-Einstellungen werden standardmäßig die neuen Einstellung ausgegeben. Mit diesem Kommando lässt sich diese Ausgabe an- und abschalten bzw. der ‚Echozustand' ausgeben.

Die Kontrollausgabe abzuschalten, hat besondere Bedeutung bei häufigen Änderungen mit ‚**pwm [1|2|3|4] <hi>**'.

Befehle im Detail

Reset	Gerät rücksetzen

Dieses Kommando löst einen Reset des Microcontrollers aus. Es dient dazu, den Controller und die Firmware in einen definierten Anfangszustand zu versetzen.

Dazu werden alle offenen Flashprogrammieraktionen abgewartet, Filepuffer werden geschrieben, alle Files geschlossen, alle Interrupts gesperrt, das Multitasking wird angehalten und im Standardfall der Microcontroller intern rückgesetzt. Dies entspricht dem Rücksetzvorgang beim Power on.

Alternativ kann der Microcontroller in eine Warteschleife geschickt werden, um damit dem Watchdogtimer das Reset zu überlassen. Diese Variante dient der Verifizierung der Wirksamkeit des Watchdogtimers.

Eine weitere Alternative ist ein Neustart der Firmware ohne Hardware-Reset. D. h. es läuft alles wie beim Standard-Reset ab, nur dass am Ende ein Sprung zur Anfangsadresse der Bootroutine gemacht wird. Dabei bleiben alle chipinternen Moduleinstellungen bis zu ihrer Neuinitialisierung erhalten.

Sollen beim Reset keine Dateipuffer auf Datenträger bzw. Streams geschrieben werden (flush) oder ist eventuell ein Problem mit dem Flushen von eventuell offenen Dateien zu rechnen, dann kann mit Angabe des Schalter ‚-nf' (no flush) dieser Zwischenschritt übersprungen werden.

Nach dem Reset wird der Bootvorgang durchlaufen, welcher die Ein-/Ausgabeleitungen in einen definierten Zustand versetzt, die SD-Karte und das Dateisystem initialisiert und das Shell-Skript **$(syspath)autoexec.sh** ausführt.

Die Systemvariable **$(syspath)** enthält den eingestellten Systempfad (Standard: ‚**c:/sys/**'). Es ist der Pfad, in dem die Standardsystemfiles

autoexec.sh
gps.cfg
firstfix.sh
ntrip.cfg
bestpos.cfg
upload.cfg
network.cfg
tcp.cfg

gesucht werden.
Der Standardsystempfad kann mit ‚**set syspath**' angezeigt und mit ‚**set syspath <path>**' neu gesetzt werden.

Parameter:

```
[-nf] [-j|-w]     -nf no flush open files
                  -j  jump to adress stored at flash address 4
                  -w  wait for watchdog reset
```

`-nf`

Vor dem Rücksetzen werden keine Dateipuffer geflusht bevor die Dateien geschlossen werden.

Befehle im Detail

Reset	Gerät rücksetzen
`-j` Mit diesem alternativen Parameter kann nach Abschluss der Reset-Prozedur ein Sprung zur Bootroutine gemacht werden. Dabei entfällt das komplette Chip-Reset. Nach dem Reboot der Firmware erscheint hinter der Ausgabe der Version eine Zeile, die mit ‚**RR:** ...' (reset reason) beginnt. In dieser Zeile wird die letzte Reset-Ursache mitgeteilt. Bei Verwendung dieses Schalters erscheint dort ‚**JMP**'.	
`-w` Mit diesem alternativen Parameter geht der Controller am Ende der Reset-Prozedur in eine unendliche Schleife. Nach dem Ablauf der eingestellten Watchdogtimer-Periode wird der unabhängige Watchdogtimer einen Watchdog-Reset auslösen, der identisch mit einem Hardware-Reset ist. Damit kann die Wirksamkeit des Watchdogtimers verifiziert werden. Nach dem Reboot der Firmware erscheint hinter der Ausgabe der Version eine Zeile, die mit ‚**RR:** ...' (reset reason) beginnt. In dieser Zeile wird die letzte Reset-Ursache mitgeteilt. Bei Verwendung dieses Schalters erscheint dort ‚**WDT**'. Das Watchdog-Intervall ist standardmäßig auf 500 ms eingestellt.	

Befehle im Detail

rm	Datei im FAT-Dateisystem löschen

Mit diesem Kommando können Dateien im FAT-Dateisystem sowie Pipes und Memoryfiles gelöscht werden. Dabei ist der komplette Pfad und der Dateiname anzugeben. Der Dateiname kann in der 8.3 Namenskonvention oder in der Langversion angegeben werden. Enthält der Pfad oder der Name Leerzeichen, dann muss Pfad und Name in **"..."** eingeschlossen werden.

Der Rückkehrwert des Kommandos ist 0, wenn die Datei gelöscht wurde und es wird ‚**removed**' ausgegeben. Der Rückkehrwert ist ungleich 0, wenn ein Fehler auftrat (z. B. wenn die Datei nicht existiert) und es wird ‚**error while removing**' ausgegeben.

Der Rückkehrwert kann für eine bedingte Ausführung von Shell-Kommandos verwendet werden.

Parameter:

`<file>`

`<file>`

Name des zu löschenden Files mit kompletter Pfadangabe. Enthält der Pfad oder der Dateiname Leerzeichen, dann kann bei diesem Kommando auf die sonst notwendige Klammerung in "..." verzichtet werden, da keine weiteren Parameter hinter dem Namen folgen können. Die Pfadangabe kann **$(syspath)** enthalten. Diese Systemvariable enthält den aktuell gesetzten Systempfad. In der Standardeinstellung ist das ‚**c:/sys/**'.

Befehle im Detail

sector	Sektoren der Micro-SD-Karte lesen oder schreiben (low level)

Dieses Kommando sollte nur von erfahrenen Benutzern verwendet werden. Bei unsachgemäßer Verwendung wird das FAT-Dateisystem beschädigt, sodass der Zugriff auf gespeicherte Dateien, auch auf die, die zum Systemstart erforderlich sind, unmöglich wird. Für Geräte die durch unsachgemäßen Gebrauch dieses Kommandos beschädigt wurden entfällt die Garantie für die dadurch notwendigen Reparaturen.

Das Kommando erlaubt das Lesen und Beschreiben von Sektoren der internen Micro-SD-Karte. Es arbeitet unabhängig vom FAT-Dateisystem. Die Daten werden in einem ‚raw mode' gelesen. Dieses Kommando ist in der Wirkung vergleichbar mit dem GNU/Linux-Kommando ‚**dd if=/dev/sda... of=... count=... bs=512**'.

Die gelesenen Sektoren werden in ein File geschrieben, werden als Hexdump angezeigt oder sie werden an eine andere Sektorposition geschrieben. Es werden immer ganze Sektoren gelesen bzw. geschrieben.

Das Kommando dient diagnostischen oder systemadministrativen Zwecken und im Normalbetrieb des Gerätes wird es nicht benötigt.

Parameter:

```
[-e] [-b {0|<bytes_per_line>}] [-crc[p]] <drive> <sect>[u][x] [-cpy {[-a] <file>|<dest_sect>}] [<n>]     //hex numbers ->
                                                                                                    0x..., -e ignore errors
| {-poke|-sim} <drv> <sect>[u][x] [<offs> [-x] [-c <repeat_cnt>] [<byte> | "<string>" [<byte> | "<string>"]]]
| {-poke|-sim} <drv> <sect>[u][x] -cpy [-t {-8|<tmot_ms>} [-s]] <file> [<max_sect>]
```

`sector c: <sector> [<n>]`

Es werden **<n>** (ohne Angabe von **<n>** wird 1 angenommen) Sektoren gelesen und hexadezimal angezeigt. Mit ‚**c:**' wird das Laufwerk (die interne Micro-SD-Karte) bezeichnet und **<sector>** gibt den Sektor an, ab dem gelesen wird (zählt von 0 ...).

`-e`

bei Lesefehlern auf der SD-Karte wird nicht abgebrochen. Es wird ebenfalls nicht abgebrochen, wenn beim Schreiben in ein File oder an eine andere Sektorposition ein Fehler auftritt.

`-b <bytes_per_line>`

Die Anzahl der Bytes, die pro Zeile hexadezimal ausgegeben werden, kann mit (ohne Angabe werden 32 angenommen) vorgegeben werden.

Wird hier 0 angegeben, dann erfolgt keine Ausgabe. Dies dient dazu, Lesefehler zu entdecken oder um die Lesegeschwindigkeit zu ermitteln.

Befehle im Detail

sector	Sektoren der Micro-SD-Karte lesen oder schreiben (low level)

`-crcp`

Es werden zusätzlich für jeden ausgelesenen Sektor eine 32-Bit CRC und eine 32-Bit Prüfsummen ermittelt und ausgegeben.
Damit kann die Integrität von Sektoren hinreichend verifiziert werden.

Beispiel:

```
sector -b 0 -crcp c: 0 10

crc32 0xa45455f2  csm 0x5460c641  sect         0, cluster  67108610
crc32 0xe151aab2  csm 0x00000000  sect         1, cluster  67108610
crc32 0xe151aab2  csm 0x00000000  sect         2, cluster  67108610
crc32 0xe151aab2  csm 0x00000000  sect         3, cluster  67108610
crc32 0xe151aab2  csm 0x00000000  sect         4, cluster  67108610
crc32 0xe151aab2  csm 0x00000000  sect         5, cluster  67108610
crc32 0xe151aab2  csm 0x00000000  sect         6, cluster  67108610
crc32 0xe151aab2  csm 0x00000000  sect         7, cluster  67108610
crc32 0xe151aab2  csm 0x00000000  sect         8, cluster  67108610
crc32 0xe151aab2  csm 0x00000000  sect         9, cluster  67108610
crc32 0xffffffff  csm 0x5460c641  sect         0, cluster  67108610  scnt         10 len       5120
```

Die letzte Zeile gibt eine Zusammenfassung. In der Zusammenfassung erscheint hinter crc32 immer **0xffffffff**. Damit wird angedeutet, dass es keine Gesamt-CRC und nur eine Gesamtprüfsumme gibt. Wird eine Gesamt-CRC benötigt, dann ist der Schalter ‚**-crc**' anzuwenden.

`-crc`

Es wird zusätzlich eine 32-Bit CRC und eine 32-Bit Prüfsummen über alle gelesenen Sektoren ermittelt und ausgegeben.
Damit kann die Integrität eines Abschnitts von Sektoren hinreichend verifiziert werden.

Beispiel:

```
sector -b 0 -crc c: 0 10

crc32 0xcba829f7  csm 0x5460c641  sect         0, cluster  67108610  scnt         10 len       5120
```

Befehle im Detail

sector	Sektoren der Micro-SD-Karte lesen oder schreiben (low level)

```
sector c: <sector>[u][x] -cpy [-a] <file> [<n>]
```

Es werden **<n>** (ohne Angabe von **<n>** wird 1 angenommen) Sektoren gelesen und in dem angegebenen File gespeichert. Mit ‚**c:**' wird das Quell-Laufwerk (die interne Micro-SD-Karte) bezeichnet und **<sector>** gibt den Sektor an, ab dem gelesen wird (zählt von 0 ...).
Das Zielfile kann ein beliebiges File im ppmOS sein (**c:/..., pipe{1|2|3}, coma, comb, comc, blth, null, dmy, can1-..., icom..., ips{a|b|c|d}..., mem-...**).

Wird der Schalter ‚**-a**' angegeben, dann werden die gelesenen Daten am Fileende angehängt.

Der Schalter **'-b <bytes_per_line>'** wird ignoriert.

u
Anwendbar auf Micro-SD-Karte und NAND-Flash-Speicher, aber auf Micro-SD-Karte haben sie keine Wirkung. Beim NAND -Flash-Speicher muß das ppmOS das sog. 'wear leveling' selbst erledigen, also das Aussortieren von fehlerhaften Speicher-blöcken. Ohne Angabe von 'u' wird ein logischer Sektor angesprochen, also so, wie man es erwartet. Im Normalfall ist der logische Sektor mit dem physischen identisch. Hat der physische Sektor nach häufigem Schreiben und Löschen Fehler, dann wird dafür einer der Reservesektoren genommen. Nach aussen hin bleibt es aber dieselbe (logische) Sektornummer. Gute Micro-SD-Karten machen das intern selbst. Mit der Angabe von 'u' wird der Sektor 'untranslated' angesprochen, es ist dann immer der physische Sektor, was nur zu Entwicklungs- und Testzwecken sinnvoll ist.

x
Anwendbar auf NAND-Flash-Speicher (wird ignoriert bei Micro-SD). Da es auf dieser niedrigen Dateiverwaltungsebene keine Verzeichnisse mit Längenangaben zu den Files gibt, wird mit 'x' wie extended, vor dem Schreiben bzw. Lesen das Fileende gesucht. Das Fileenende ist der erste vom Ende rückwärts gefundener Sektor der nicht komplett 0xff enthält. Beim Lesen wird bis dahin gelesen und beim (Append-)Schreiben wird im Sektor dahinter, dem ersten komplett gelöschten ff-Sektor, fortgesetzt.

Beispiel:

```
ls sector-d-0x1000x-100000
```

Ausgabe:

```
sector1-d-4096x-100000   0                0
```

Das Sektorfile kann maximal 51200000 Byte aufnehmen und es hat aktuell die Länge 0.

```
date >sector-d-0x1000x-100000
```
benötigt einen Sektor.

```
ls sector-d-0x1000x-100000
```
ergibt nun

```
sector1-d-4096x-100000   512              0
date >sector-d-0x1000x-100000
```
Hier wird immer im 'Append'-Mode geschrieben!

Und

```
ls sector-d-0x1000x-100000
```
ergibt danach

```
sector1-d-4096x-100000   1024             0
```

Befehle im Detail

sector	Sektoren der Micro-SD-Karte lesen oder schreiben (low level)

```
sector c: <sector> -cpy <dest_sect> [<n>]
```

Es werden **<n>** (ohne Angabe von **<n>** wird 1 angenommen) Sektoren gelesen und beginnend an dem angegebenen Sektor `<dest_sect>` gespeichert. Mit ‚**c:**' wird das Quell-Laufwerk (die interne Micro-SD-Karte) bezeichnet und **<sector>** gibt den Sektor an, ab dem gelesen wird (zählt von 0 ...).
Der Schalter **'-b <bytes_per_line>'** wird ignoriert.

```
sector {-poke|-sim} <drv> <sect>[u][x] [<offs> [-x] [-c <repeat_cnt>] [<byte> | "<string>" [<byte> | "<string>"]]]
```

In den angegebenen Sector **<sect>** werden ab dem Byteoffset **<offset>** die folgenden Bytes bzw. Strings geschrieben.
Um bei der Anwendung eine Kontrollmöglichkeit vor dem Schreiben zu haben, ist der Schalter **‚-sim'** anzugeben. Es erscheint die Hex-Ausgabe des Sektors mit den gepatchten Bytes.
Wenn die Änderung verifiziert wurde, kann mit derselben Kommandozeile, in der **‚-sim'** mit **‚-poke'** ausgetauscht wurde, der Inhalt auf die SD-Karte geschrieben werden.
Damit ist ein gezieltes Überschreiben von Sektorinhalten möglich. Verwendung findet diese Kommandovariante beim Debugging von Verzeichniseinträgen oder von Dateiinhalten. Dies stellt den ‚low level'-Zugriff auf die interne Micro-SD-Karte dar. Damit eröffnen sich auch Spezialanwendungen, bei denen in von der Partitionierung ausgenommenen Sektoren besondere Daten gespeichert werden können.
Mit **<byte>** sind 8-Bit Werte zwischen 0 und 255 gemeint, die als Dezimalzahl oder mit vorangestelltem ‚**0x**' Hexadezimalzahl angegeben werden können. Es können auch in `"..."` eingeschlossene ASCII-Strings angegeben werden.
Wird der Schalter **‚-c <repeat_cnt>'** verwendet, dann werden alle dahinter angegeben Zeichen und ASCII-Strings mehrfach wiederholt. Würde dabei die Sektorgrenze überschritten werden, dann erscheint die Fehlermeldung **‚error 2 in values to poke in sector'**.
Die Anzahl, der auf der SD-Karte vorhandenen der Sektoren, kann mit **‚df c:'** ermittelt werden.

`-x`
Bei dem Poken von Werten in einen Sektor können Zahlen (Bytewerte) und/oder Zeichenketten (Strings in "...") angeben werden. Bei Hexadezimalzahlen ist üblicher Weise ein 0x... voranzustellen um sie von Dezimalzahlen unterscheiden zu können. –x bedeutet, dass alle Zahlen als Hexadezimalzahlen interpretiert werden. Dadurch kann man auf eine Kommandozeile deutlich mehr Byte angeben.

Befehle im Detail

sector	Sektoren der Micro-SD-Karte lesen oder schreiben (low level)

```
sector {-poke|-sim} <drv> <sect>[u][x] -cpy [-t {-8|<tmot_ms>} [-s]] <file> [<max_sect>]
```

Das File **<file>** wird beginnend am angegebenen Sector **<sect>** auf die SD-Karte kopiert.
Um bei der Anwendung eine Kontrollmöglichkeit vor dem Schreiben zu haben (z. B. ob das File lesbar ist), ist der Schalter **‚-sim'** anzugeben. Es wird nichts geschrieben.
Wenn der Ablauf verifiziert wurde, dann kann mit derselben Kommandozeile, in der **‚-sim'** mit **‚-poke'** ausgetauscht wurde, der Inhalt auf die SD-Karte geschrieben werden.
Das Verfahren, erst ‚**-sim**' und dann ‚-**poke**', ist bei dieser Kommandovariante aber nur auf Files anwendbar, bei denen sich der Dateizugriffszeiger an den Anfang zurückdrehen lässt, bzw. bei dem sich das File schließen und mit dem selben Inhalt erneut öffnen lässt. Das ist nur bei Dateien im FAT-Dateisystem, beim Filetyp **mem-...** und **zero-...** möglich. Bei allen anderen Files (**com..., blth, pipe..., vcom1, can1-..., icom..., ips[a|b|c|d}...)** ist das nicht möglich. Das sind Streams, bei denen einmal Gelesenes nicht ein zweites Mal gelesen werden kann.

Damit ist ein gezieltes Überschreiben von Sektorinhalten möglich. Verwendung findet diese Kommandovariante beim Debugging von Verzeichniseinträgen oder von Dateiinhalten. Dies stellt den ‚low level'-Zugriff auf die interne Micro-SD-Karte dar. Damit eröffnen sich auch Spezialanwendungen, bei denen in von der Partitionierung ausgenommenen Sektoren besondere Daten gespeichert werden können.

Befehle im Detail

set	Einstellung von Geräte- und Systemparametern

Dies ist ein Sammelkommando für diverse Geräte- und Systemeinstellungen. Im folgenden werden nur die Kommandozeilenhilfen für alle Unterkommandos aufgelistet. Hier werden lediglich Kurzerläuterungen gegeben. Die ausführlichen Erläuterungen erfolgen zu jedem Unterkommando separat. Die Eingabe ‚**set ?**' ergibt die Kommandozeilenhilfe und die Eingabe von ‚**set**' ergibt die Ausgabe der aktuellen Einstellungen sämtlicher Unterkommandos.

Parameter:

```
  echo [not-found] [<task_name>] [on|off]
| intp             [<task_name>] [on|off [<unlock_key>]]
| loop             [<task_name>] [on|off]    has effect if intp is off
| ext-syntax       [<task_name>] [on|off]
| close-sh-on-exec [<task_name>] [on|off]
| fread-as-container [on|off]
| {i|o}rd [{<com_name>|<com_alias>} [tee]
           {  [-ob|-ib|-r [-eq] [<baud_dest>]] <com_name>|<com_alias>
            | [-a] <file>
            | [-r] off [all]
           }
         ]
| rtc [utc | gps <leap_seconds>]        since 2017.1.1, gps = utc + 18 s
| phone [<ix> [clr|<name> <no>] | support [on|off]]
| bridge {<com_name>|<com_alias>} {<com_name>|<com_alias>}
| pwr [supply|blth|gps|sd|eth|gsm [-pwm]|io|5p|app [on|off]] ;any combination of sources
| led [red] [green] [yellow] [orange] [green2] [button] [on|off|tgl]
| comc  [rs232|rs485|rtscts] [on|off|tgl]
       | rts [-s] [hi|lo|tgl|gpio|uart]   -s silent, no print
       | cts [-s]                         returns 1 for high and 0 for low, -s no print
       | modbus [start [-led] <baud> <tcp_file>|stat|stop|{led|simu} [on|off]|rst]
| blth [init|scan|{rst|rtscts} [on|off|tgl]]
| io  [-s] [out1.1] [out1.2] [out2.1] [out2.2] [gpio5] [buzzer] [can-tx] [hi|lo|tgl]
| io  [-s] {[in1.1] [in1.2] [in2.1] [in2.2]} [irq [off|hl|lh|both|clr]] via && or || usable
| io  [-s] in-all                            [irq [off|hl|lh|both|clr]] via && or || usable
| io  [-s] event                             [irq [off|hl|lh|both|clr]] via && or || usable
| io  [-s] {event|pps|gps-led1|varfreq}      [out|in|hi|lo|tgl]         via && or || usable
| io  [-s] can-rx                                                       via && or || usable
| io  [-s] [rst-gps1|rst-imu] [hi|lo|tgl]                               low active
| usb [{rs232|stick|off} [now] |stat [full]]  ;rs232 and off means internally connected to SD card
| adc [stat [-s] | echo [on|off|esc]]
| adc chkrepeats <fail_repeats> [<ok_repeats>]    0...255, unit is 'set adc chk' calls
| adc chk [-v] <vin_mV>              return 0 when vin >= <vin_mV> else return 1
| adc {temp|vbatt}                  measure chip temp or vbatt
| syspath [<path>]
| path [<path>]
| fctl [rd|wr] <filename> [{clr|set} <val>]   get/clr/set file ctrl flags for rd/wr, bit 0=1 -> GF_BLOCKING_WR
```

Befehle im Detail

set	Einstellung von Geräte- und Systemparametern

Kommando	Erläuterung	Default
`echo`	Schaltet das Kommandozeilenecho an oder aus. (Mit „not-found“ bleibt die Fehlermeldung bei einem nicht gefundenen Kommando on/aus)	on
`intp`	Schaltet den Kommandointerpreter (Shell-Interpreter) an der betreffenden Schnittstelle an oder aus	on
`loop`	Schaltet einen Loopback-Modus an oder aus wenn der Shell-Interpreter für eine Schnittstelle deaktiviert ist	off
`ext-syntax`	Schaltet für einen Shell-Interpreter die erweiterten Syntaxmerkmale (> && || @ ;) an oder aus	on
`close-sh-on-exec`	Schaltet das temporäre Schließen eines Skriptfiles während Ausführung eines Kommandos aus dem Skriptfile an oder aus.	off
`fread-as-container`	Schaltet Erkennung u. Nutzung des Containerfiletyps (Micro-SD) an od. ab	on
`ird`	Setzt Eingabeumleitungen u./od. Verzweigungen für eine serielle Schnittstelle	off
`ord`	Setzt Ausgabeumleitungen u./od. Verzweigungen für eine serielle Schnittstelle	off
`phone`	Anzeige und Einstellungen für das Telefonbuch der erlaubten SMS-Telefonnummern	leer + 2 PPM-Servicenr.
`bridge`	Verbindet zwei serielle Schnittstellen um Daten ohne Kenntnis der Baudrate bzw. Baudratenumschaltung übertragen zu können.	off
`pwr`	Ermöglicht das Schalten der verschiedenen Stromversorgungsinseln (supply, gsm, gps, ethernet, bluetooth, GPS-Antennenspeisung, digitale IO)	supply on, SD-Karte u. coma on, comb, comc u. digitale IO on, alle anderen off
`led`	Erlaubt das Schalten der Geräte LED	
`comc`	Umschaltung des Hardwareprotokolls, RS485/RS232 und Modbus	RS232
`blth`	Schaltet die Bluetooth-Funktionalität an oder aus	off
`io`	IO-Signale können geschaltet werden (IO1-, IO2-Buchse, usw.)	
`usb`	Das Verhalten bei einer USB-Verbindung einstellen	USB-Stickmode
`vcom`	Virtuelle COM-Schnittstelle zum Controller via USB aktivieren/deaktivieren	off
`adc`	Einstellung der Anzeigeoptionen der Spannungsmessungen	off
`syspath`	Einstellung des Pfades für das Systemverzeichnis (Prefix für autoexec.sh usw.).	c:/sys/
`path`	Systemvariable. Wird sie mit `set path c:/sys/` gesetzt, dann kann beim Aufruf von Scripten auf der SD-Karte der Pfad weggelassen werden. Damit wäre es möglich ein internes Kommando auf ein Skript umzuleiten.	n.a.
`fctl`	Einstellung und Anzeige des Controlmodes (blocking oder nonblocking) für Schnittstellen und geöffnete Dateien.	

Befehle im Detail

set adc	AD-Wandler einstellen und Betriebsspannungen messen

Mit dem Unterkommando können die Eingangsspannung sowie interne Spannungen (3.3 V und 5 V GPS-Antennenspannung) angezeigt werden. Das Kommando dient auch zum Kontrollieren der Spannung der Stützbatterie für die Echtzeituhr und zur Messung der Geräteinnentemperatur.

Wenn die Eingangsspannung und die interne 3.3 V Betriebsspannung unter einen Grenzwert fällt, dann werden diese Einbrüche gezählt. Der AD-Wandler ist in der Lage, Spannungseinbrüche bis hinunter in den Mikrosekundenbereich zu erkennen und zu zählen. Fällt die Eingangsspannung unter 9 V, dann werden die Spannungen ausgegeben. Fällt sie weiter unter einen Grenzwert von 8.7 V und die interne 3.3 V Betriebsspannung fällt unter 2.9 V, dann werden alle Files geschlossen und das Gerät schaltet sich aus, da eine ordnungsgemäße Funktion nur im Bereich von 9 V bis 32 V gewährleistet ist.

Parameter:

```
  [stat [-s] | echo [on|off|esc]]
| chkrepeats <fail_repeats> [<ok_repeats>]    0...255, unit is 'set adc chk' calls
| chk [-v] <vin_mV>   return 0 when vin >= <vin_mV> else return 1
| {temp|vbatt}        measure chip temp or vbatt
```

```
set adc stat [-s]
```

Ausgabe des Status

```
ADC: 17.05.02 08:19:21.333 s  Vin 12096 mV  V5p 4997 mV  Vcc3 3277 mV  Vrefint 1210 mV  Tchip  32.6 °C  Vinusb off  Vincnt    89  Vcc3cnt   1
```

Wenn **‚set adc vbatt'** eingestellt wurde, erscheint an der Stelle, wo Tchip stand, die aktuelle Spannung der internen (auswechselbaren) Lithiumbatterie (CR 2032).

```
ADC: 17.05.02 08:29:44.384 s  Vin 12096 mV  V5p 4997 mV  Vcc3 3277 mV  Vrefint     mV  Vbatt  3176 mV  Vinusb off  Vincnt    89  Vcc3cnt   1
```

Ausgabe des Status in Kurzform ohne Maßeinheiten mit Schalter **‚-s'**:

```
ADC: 17.05.02 08:23:02.366 12096 4997 3276 1210  32.6 off    89     1
```

oder

```
ADC: 17.05.02 08:29:47.376 12096 4995 3278  3172 off    89     1
```

```
set adc echo [on|off|esc]
```

Periodische Ausgabe des ADC-Status (wie bei ‚**set adc stat**') anzeigen, ein-, ausschalten oder auf einem Terminal positionsfest anzeigen. Der Status wird fortlaufen angezeigt. Die Zeilen füllen den Bildschirm. Mit dem Schalter ‚**esc**' werden ESC-Sequenzen eingefügt, die in einem Terminal, das diese ESC-Sequenzen unterstützt, zu einer Anzeige in der obersten Bildschirmzeile führen.

Befehle im Detail

set adc	AD-Wandler einstellen und Betriebsspannungen messen
`set adc chk [-v] <vin_mV>` Abfrage, ob die Eingangsspannung über oder unter dem angegebenen Wert liegt. Damit ist die Möglichkeit gegeben, in einer bedingten Shell-Kommandoausführung auf Veränderungen der Eingangsspannung zu reagieren.	
`set adc {temp|vbatt}` Mit diesem Unterkommando wird eingestellt, ob die Chiptemperatur oder die Spannung der Stützbatterie (3 V Lithium, CR 2032, auswechselbar) gemessen werden soll. Standardmäßig wird die Chiptemperatur und damit die Gehäuseinnentemperatur des Gerätes gemessen.	

Befehle im Detail

set blth	Funktionen der Bluetooth-Schnittstelle einstellen

Mit diesem Kommando wird die Bluetooth-Schnittstelle aktiviert, deaktiviert oder Einstellungen vorgenommen. Standardmäßig ist die Bluetooth-Schnittstelle ausgeschaltet. Die Bluetooth-Schnittstelle wird über ein serielles Protokoll (SPP serial port profile) konfiguriert. Die Verbindungsaufnahme durch einen anderen Bluetooth-Teilnehmer erfordert eine PIN (Standard 1234). Wie die PIN geändert werden kann findet sich am Ende dieses Abschnitts.

Parameter:

```
[init|scan|{rst|rtscts} [on|off|tgl]]
```

```
set blth init
```

Die Stromversorgung für das Bluetooth-Modul wird eingeschaltet und die Schnittstelle ‚**blth**' ist ansprechbar. Das Modul wird mit ‚**set pwr blth off**' bei Bedarf abgeschaltet.

```
set blth scan
```

Das Bluetooth-Modul wird für andere Bluetooth-Geräte sichtbar und es kann mit einem neuen Teilnehmer eine Verbindung hergestellt werden. Ist dem Teilnehmer die Bluetooth-Device-Adresse aus einer vorherigen Verbindung bereits bekannt, dann braucht dieses Kommando nicht erneut ausgeführt werden.

```
set blth rst [on|off|tgl]
```

Das Bluetooth-Modul hat einen Reset-Eingang, mit dem das Modul in den Grundzustand versetzt werden kann. Die Funktion ist mit einem power on bei ‚**set blth init**' vergleichbar.

```
set blth rtscts [on|off|tgl]
```

Die Steuerung des Datenflusses zwischen Controller und Bluetooth-Modul erfolgt standardmäßig im Hardwareprotokoll RTS/CTS. D. h. Controller und Bluetooth-Modul signalisieren sich gegenseitig, ob Daten entgegengenommen werden können. Das ist deshalb nötig, da über die ‚Luftübertragungsstrecke' je nach Entfernung und Empfangsbedingung die Daten unterschiedlich schnell übertragen werden.

Befehle im Detail

set blth	Funktionen der Bluetoothschnittstelle einstellen

Bluetooth-PIN ändern über das Skript **bluetooth-cfg.sh** (braucht nur bei Änderungen aufgerufen werden):

Skriptdatei **bluetooth-cfg.sh** auf dem PC mit folgendem Inhalt anlegen. In der Zeile **...btkey** die **1234** durch die gewünschte PIN ersetzen (kann auch mehr als 4 Stellen haben). In der Zeile **...btname** kann **ppm40xx** durch einen selbstgewählten Namen ersetzt werden.

bluetooth-cfg.sh:

```
set blth init
@wait -s 0.4; echo at+btcancel>blth
@wait -s 0.4; echo at+btmode,0>blth
@wait -s 0.4; echo at+btfp,1>blth
@wait -s 0.4; echo 'at+btkey="1234"'>blth
@wait -s 0.4; echo 'at+btname="ppm40xx"'>blth
@wait -s 0.4; echo atz>blth
@wait -s 1;   echo at+btmode,3>blth
@wait -s 0.4; echo at+btinfo?>blth
#Anzeige 'STANDBY'

@wait -s 0.4; set blth rst on
@wait -s 1;   set blth rst off
@wait -s 0.4

#damit es von anderen Geräten gefunden werden kann
echo at+btscan>blth

echo at+btinfo?>blth
```

Nachdem die Datei im USB-Stickmode auf den 40xx übertragen wurde (Empfehlung: nach ‚**c:/sys/bluetooth-cfg.sh**') und das Kabel getrennt wurde kann das Shell-Skript gestartet werden.

Danach ist das Gerät für andere Geräte sichtbar und es kann eine Datenübertragung erfolgen (Aus- und Eingabe). Standardmäßig ist an der Schnittstelle ‚**blth**' ein Shell-Interpreter aktiv. Wird dieser nicht benötigt, weil z. B. NTRIP-Daten über Bluetooth ankommen und diese an das GPS-Board weitergeleitet werden sollen, dann kann der Shell-Interpreter mit ‚**set intp blth off**' abgeschaltet werden.

Befehle im Detail

set bridge	Zwei serielle Schnittstellen (UART) mit einer Brücke verbinden

Das Kommando ‚**set bridge**' verbindet softwaremäßig 2 UARTS des Controllers im ‚**Bit Banging**'-Mode. D. h. sie werden intern durch schnellstmögliches Polling miteinander verbunden. Damit ist es z. B. möglich, von einer extern zugänglichen Schnittstelle (**coma** und **comc**, nicht comb) zum GPS-Board über **gps1** oder **gps2** eine ‚quasi' Drahtverbindung herzustellen. Diese Art der Kopplung hat den Vorteil, dass eine externe Software die Baudrate, Stopbits oder die Parität beliebig ein- und während der Verbindung auch umstellen kann. Das ist z. B. der Fall, wenn ein GPS-Firmware-Update mit einem angeschlossenen PC durchgeführt werden soll. Die ‚Quasikopplung' funktioniert mit Baudraten von 0 Bd bis 921600 Bd.

Da diese Art der Kopplung mit Software realisiert wird und sie bis zu sehr hohen Baudraten funktioniert, müssen alle anderen Funktionen des Gerätes deaktiviert werden.

Nach der Ausführung dieses Kommandos werden keine weiteren Kommandos ausgeführt auch nicht die, die mit ‚;' getrennt in derselben Kommandozeile folgen.

Der Bridgemode kann durch Drücken der On/Off-Taste oder durch Trennen der Stromversorgung (min. 30 s vor Wiedereinschalten warten) beendet werden. Während des Bridgemodes ist nur noch der interne unabhängige Watchdogtimer aktiv.

Parameter:

```
{<com_name>|<com_alias>} {<com_name>|<com_alias>}
```

```
<com_name>|<com_alias>
```

Hier sind die Schnittstellenaliasnamen (**coma, comc, gsm, blth, gps1, gps2** – nicht **comb**) oder die systematischen Namen **com1, com2, com3, com6, com7** oder **com8** einzusetzen.

Beispiel für ein GPS-Firmware-Update:

```
set bridge coma gps1
```

Der Modus kann nur durch Unterbrechung der Stromversorgung beendet werden.

Befehle im Detail

set comc	Funktionen der Schnittstelle comc einstellen

Mit diesem Kommando werden die erweiterten Hardwaremöglichkeiten der Schnittstelle (UART) comc angezeigt und eingestellt.

Die Schnittstelle kann im RS232-Mode (spannungsgesteuert +/- 3-12 V) oder im RS485-Mode (spannungsdifferenziell) betrieben werden. eine spezielle Modbus-Konfiguration (Modbus over TCP/IP) kann eingestellt werden. Außerdem verfügt diese UART über Modemsteuerleitungen RTS und CTS, welche für das Hardware-Handshake oder als einfache GPIO (General-Purpose-IO-Leitungen) verwendet werden können.

Im Modbus-Mode ist der 40xx der Modbus-Master und an der Schnittstelle **comc** werden die Modbus-Slaves angeschlossen. Der Modbus-Master erhält seine Protokolldaten über TCP-Pakete. Die über die **comc** empfangenen Antwortdaten der Slaves werden wieder über die TCP-Verbindung versendet.

Parameter:

```
 [rs232|rs485|rtscts] [on|off|tgl]
| rts [-s] [hi|lo|tgl|gpio|uart]   -s silent, no print
| cts [-s]                         returns 1 for high and 0 for low, -s no print
| modbus [start [-led] <baud> <tcp_file>|stat|stop|{led|simu} [on|off]|rst]
```

```
set comc rs232|rs485|rtscts] [on|off|tgl]
```

Abfrage, Ein-, Aus- oder Umschaltung des RS232-Modes. die Standardeinstellung ist ‚**rs232 on**'.

Beispiel:

```
set comc rs232
```

Ausgabe:

```
comc: mode rs232, no flow control, rts controlled by uart, rts lo, cts lo
```

Der RS232 ist aktiv, es ist keine Hardware-Flusssteuerung eingestellt und die Leitungen RTS und CTS haben den angegebenen Pegel.

Das Kommando ‚**set comc rs232 off**' ist identisch mit ‚**set comc rs485 on**' und umgekehrt. Die Ausgabezeile enthält bei der Kommandovariante ‚**set comc rs485**' dieselben Informationen.

Befehle im Detail

set comc	Funktionen der Schnittstelle comc einstellen

```
set comc rtscts [on|off|tgl]
```

Abfrage, Ein-, Aus- oder Umschaltung der Hardware-Flusssteuerung. Die Standardeinstellung ist ‚**off**'.

Die Ausgabe ist identisch mit der Ausgabe bei ‚**rs232**' oder ‚**rs485**'.

Wenn die Hardware-Flusssteuerung aktiviert ist, aber das angeschlossene Gerät dieses nicht unterstützt, dann kann es zum Stillstand der Datenausgabe auf **comc** kommen, weil standardmäßig der Blockingmode aktiviert ist (siehe ‚**set fctl comc ...**' S. 242). D. h. ein Task wartet mit der Ausgabe neuer Daten solange bis er die Daten an den Übertragungspuffer übergeben kann. Dieser wird aber irgendwann voll, wenn das angeschlossene Gerät keine Empfangsbereitschaft (CTS - clear to send) meldet.

```
set comc rts [-s] [hi|lo|tgl|gpio|uart]
```

Die Ausgabeleitung RTS kann abgefragt oder eingestellt (‚**hi**' oder ‚**lo**') oder umgeschaltet (‚**tgl**') werden.

Eine Verwendung als GPIO-Ausgabeleitung (General Purpose IO-Leitung) ist möglich, wenn ‚**set comc rs232 on**' eingestellt und die Verbindung zur UART aufgehoben wird mit ‚**set comc rts gpio**'.

Die Standardeinstellung ist ‚**rts controlled by uart**' und ‚**rtscts off**'.

Der optionale Schalter ‚-s' (silent) unterdrückt die Ausgabe der Statuszeile.

Die Ausgabe ist identisch mit der Ausgabe bei ‚**rs232**' oder ‚**rs485**'.

```
set comc cts [-s]
```

Mit diesem Unterkommando kann die Hardwareleitung CTS (clear to send) abgefragt werden.

Beispielausgabe :

```
comc: cts is high
```

Das Kommando hat einen Rückkehrwert, der in einer bedingten Shell-Kommandoausführung verwendet werden kann.

Beispiel:

```
@set comc cts -s && @set led yellow on
@set comc cts -s || @set led yellow off
```

Diese Kommandos könnten in CRON-Jobs eingebaut werden. Es muss dabei beachtet werden, dass die Aktion immer ausgeführt wird, wenn abgefragt wird.

Der optionale Schalter **‚-s'** (silent) unterdrückt die Ausgabe der Statuszeile.

Befehle im Detail

set comc	Funktionen der Schnittstelle comc einstellen

```
set comc modbus [start [-led] <baud> <tcp_file>|stat|stop|{led|simu} [on|off]|rst]
```

Mit diesem Unterkommando wird das Modbus-Protokoll RTU over TCP aktiviert und gesteuert.

Die Verwendung beginnt mit dem Start:

```
set comc modbus start [-led] <baud> <tcp_file>
```

Dabei gibt **<baud>** die Baudrate der comc-UART an und **<tcp_file>** ein Ethernetfile vom Type **ips{a|b|c|d}1**. D. h. der 40xx ist der Server und ein anders Gerät/PC ist der Client. Prinzipiell wäre auch ein Rollentausch möglich. Die Schnittstelle **comc** wird in den RS485 Mode geschaltet.

Das Ethernetfile muss danach mit dem Kommando ‚**ip server start ips{a|b|c|d}:<port> -i**' geöffnet werden. Da in ‚**ip ...**' kein ‚**-n ...**' angegeben wurde, kann nur ein Client die Verbindung herstellen und die das File **<tcp_file>** lautet dann **ips{a|b|c|d}1**. Wichtig ist auch der Schalter ‚-i', der besagt, dass kein Shell-Interpreter die empfangenen Daten bekommt. Diese erhält der Task ‚**modbus**', der das empfangene Paket an LED 4 signalisiert und auf **comc** ausgibt und danach auf die Antwort des angeschlossenen Sensors wartet. Die ankommenden Daten werden durch Blinken der LED 3 signalisiert und in einem TCP-Paket zum Modbus-Client via Ethernet gesendet.

Mit dem optionalen Schalter ‚**-led**' wird die LED-Signalisierung der Modbus-Pakete auf der UART-Leitung abgeschaltet.

Um auch ohne Sensor testen zu können, kann man das Kommando ‚**set comc modbus simu on**' ausführen. Darauf sendet der Modbus-Task nach dem Senden des über Ethernet empfangenen Modbus-Requests ein Dummy-Antwortpaket nach **comc**. Wenn nun mit einem Teststecker die Signale Tx und Rx in **comc** gebrückt werden, dann empfängt der Modbus-Task das Dummy-Paket und sendet es weiter nach **ips{a|b|c|d}1**.

Die LED-Signalisation des Datenverkehrs kann mit ‚**set comc modbus led off**' abgeschaltet und mit ‚**... on**' wieder eingeschaltet werden.

Der Test (das Senden von Dummy-Paketen) wird abgeschaltet mit , **set comc modbus simu on**'.

Wenn die Ethernet-Verbindung unterbrochen wird bzw. der PC die Verbindung trennt, dann erscheint:

```
IP: ipsd1 break in shell client task, eventFlags 0x76, error 1000
IP: ipsd1 connection to 192.168.2.102:51319 closed
```

Mit ‚`set comc modbus stop`' wird der Modbus-Task gestoppt.

Befehle im Detail

set comc	Funktionen der Schnittstelle comc einstellen

```
set comc modbus stat
```

Mit diesem Unterkommando wird der Zustand der Modbus-Funktionalität abgefragt.

Beispiel:

```
modbus: tcp file 'ipsd1', in 447312, out 183966, cyc 0.030084 s, cmds dropped 36, RS485 crc err: req 1, resp 1
```

Darin bedeuten die Zahlen hinter ‚**in**' und ‚**out**' die übertragenen Datenmengen in Byte. Der Wert hinter ‚**cyc**' gibt die zuletzt gemessene Paketzykluszeit (via ethernet empfangen) an.

Die Anzahl der Modbus-Requests (**cmds**), die nicht beantwortet wurden, wird hinter ‚**cmds dropped**' angegeben. Es folgen die Zähler für CRC-Fehler auf der RS485-Leitung getrennt nach Requests und Antworten (Responses).

Befehle im Detail

set close-sh-on-exec	File-Behandlung für Shell-Skripte einstellen

Bei der Ausführung eines Shell-Skripts wird das Skriptfile geöffnet und die Kommandos werden zeilenweise gelesen und ausgeführt. Dieses Kommando stellt ein, ob das Skriptfile vor der Ausführung geschlossen und nach der Ausführung des Kommandos wieder an der alten Stelle geöffnet wird oder ob das File geöffnet bleibt. Die Standardeinstellung ist ‚**off**', also File bleibt offen. Da im **ppmOS** gleichzeitig nur 12 Files geöffnet sein können, könnte es in einer speziellen Anwendung nötig sein das Skriptfile zu schließen, weil vielleicht das auszuführende Kommando seinerseits auch Files öffnet und die Filetabelle unter Umständen voll werden könnte. Die Umstellung auf ‚on', also File während der Kommandoausführung schließen, bedeutet ein Performanceverlust, da zur Kommandoausführungszeit noch die Zeit zum Schließen und Öffnen des Skriptfiles hinzukommt.

Diese Einstellung gibt es für jeden Shell-Interpreter separat.

Parameter:

```
close-sh-on-exec [<task_name>] [on|off]
```

```
[<task_name>]
```

Diese optionale Angabe benennt den Task, dessen Shell-Interpreter eingestellt werden soll.

Ohne diese Angabe bezieht sich das Kommando auf den Shell-Interpreter, der dieses Kommando ausführt.

```
on
```

Das Skriptfile wird vor der Ausführung eines Kommandos geschlossen und danach wieder geöffnet.

```
off
```

Das Skriptfile wird vor der Ausführung eines Kommandos geschlossen und danach wieder geöffnet.

Ohne ‚**on**' oder ‚**off**' wird die aktuelle Einstellung angezeigt. Die Default-Einstellung ist ‚**off**'.

```
coma: close script file on exec is off
```

Befehle im Detail

set echo	Kommandoecho für den Shell-Interpreter einstellen

Empfängt ein Shell-Interpreter ein Kommando, dann gibt er dieses Kommando standardmäßig aus, bevor die Ausführung des Kommandos beginnt. Auf diese Weise kann in einem Terminal die Abfolge von Kommandos und deren Ausgaben besser verfolgt werden. Es finden sich dann die Kommandos und deren Ausgaben im Eingangslogfile des Terminalprogramms. Die Standardeinstellung ist dann besonders sinnvoll, wenn das Terminalprogramm eine extra Eingabezeile hat und im kanonischen Mode arbeitet.

Diese Einstellung gibt es für jeden Shell-Interpreter separat.

Parameter:

```
echo [not-found] [<task_name>] [on|off]
```

`[not-found]`

Mit dieser optionalen Angabe kann die Fehlermeldung '# not found' abgeschaltet werden. Diese Fehlermeldung wird z.B. dann ausgegeben, wenn ein dem ppmOS unbekanntes oder ein falsch geschriebenes Kommando gesendet wird.

`[<task_name>]`

Diese optionale Angabe benennt den Task, dessen Shell-Interpreter eingestellt werden soll.

Ohne diese Angabe bezieht sich das Kommando auf den Shell-Interpreter, der dieses Kommando ausführt.

`on`

Vor der Ausführung eines Kommandos wird das Kommando ausgegeben.

`off`

Das Kommando wird nicht ausgegeben.

Ohne ‚**on**' oder ‚**off**' wird die aktuelle Einstellung angezeigt. Die Default-Einstellung ist ‚**on**'.

```
coma: interpreter command echo is on
```

Befehle im Detail

set ext-syntax	Einstellung für die erweiterten Syntaxregeln in der Skriptsprache des ppmOS

Die Basisregeln des Shell-Interpreter umfassen die Ausführung von Kommandos, die in einer Zeile, die mit CR+LF, CR oder nur LF abgeschlossen ist, notiert sein müssen. Das erste Wort in der Zeile ist das Kommandowort, das in der Kommandoliste gesucht wird. Es muss mit mindestens einem Leerzeichen oder TAB von anderen folgenden Zeichen getrennt sein, außer es handelt sich um CR+LF, CR oder LF. Leerzeichen bzw. Tabulatoren vor dem Kommandowort werden ignoriert.
Die Zeichen hinter dem Kommando bis zum Zeilenende sind die Kommandoargumente. Kommandoargumente sind optional. Die erweiterte Syntax ist funktionsähnlich zur Shellscript-Sprache in Linux.
Diese Basisregeln sind in der Standardeinstellung erweitert um die Zeichenfolgen:

`@` oder `!`	Das Kommandoecho wird für dieses Kommando abgeschaltet, es wird unabhängig von ‚**set echo ...**' **nicht** ausgegeben.
`@@` oder `!!`	Das Kommandoecho **und** die erweiterten Syntaxregeln werden für dieses Kommando abgeschaltet. Das ist dann nötig, wenn die für die erweiterte Syntax reservierten Zeichen in den Kommandoargumenten vorkommen können (z. B. Kommando ‚**dnld ...**').
`;`	Trennzeichen zwischen zwei Kommandos in einer Kommandozeile. D. h. in einer Kommandozeile können mehrere Kommandos durch ‚;' getrennt aufgeführt werden. Sie werden von links nach rechts ohne Bedingung nacheinander ausgeführt.
`&&`	Diese beiden aufeinanderfolgenden Zeichen bedeuten eine bedingte Kommandoausführung. Das Kommando vor dem ‚**&&**' wird ausgeführt. Ist des Rückkehrcode 0 (hat in der Regel die Bedeutung fehlerfrei), dann wird auch das hinter ‚**&&**' folgende Kommando ausgeführt. Ist der Rückkehrcode des ersten Kommandos ungleich 0, dann wird das folgende Kommando nicht ausgeführt.
`||`	Diese beiden aufeinanderfolgenden Zeichen bedeuten eine bedingte Kommandoausführung. Das Kommando vor dem ‚**||**' wird ausgeführt. Ist des Rückkehrcode ungleich 0 (hat in der Regel die Bedeutung Fehler bei der Ausführung), dann wird das hinter ‚**||**' folgende Kommando ausgeführt. Ist der Rückkehrcode des ersten Kommandos gleich 0, dann wird das folgende Kommando nicht ausgeführt.
`>` oder `>>`	Das ist das Zeichen für die Ausgabeumleitung. Damit wird die Ausgabe des Kommandos (Standardausgabe) in das dahinter angegebene File umgeleitet. Mit ‚**>**' wird das File angelegt oder falls es existiert, wird es erst gelöscht. Mit ‚**>>**' wird die Ausgabe an ein eventuell existierendes File angehängt. Anderenfalls wird das File erzeugt.

Ein wesentlicher Unterschied zur Linuxshell ist die Vorrangregel, dass im **ppmOS alle** hinter ‚**&&**' oder ‚**||**' folgenden durch ‚;' getrennte Kommandos als ein Kommando gelten. In Linux ist es nur das unmittelbar auf ‚**&&**' oder ‚**||**' folgende.

Parameter:

```
[<task_name>] [on|off]
```

Befehle im Detail

set ext-syntax	Einstellung für die erweiterten Syntaxregeln in der Skriptsprache des ppmOS

`[<task_name>]`

Diese optionale Angabe benennt den Task, dessen Shell-Interpreter eingestellt werden soll.

Ohne diese Angabe bezieht sich das Kommando auf den Shell-Interpreter, der dieses Kommando ausführt.

`on`

Die erweiterten Syntaxregeln werden angewendet, sie können in Skripten verwendet werden.

`off`

Die erweiterten Syntaxregeln werden nicht angewendet, die Sonderzeichen sind in den Argumenten eines Kommandos verwendbar.

Ohne ‚**on**' oder ‚**off**' wird die aktuelle Einstellung angezeigt. Die Default-Einstellung ist ‚**on**'.

```
coma: interpreter parses extended syntax "> && || ;" is on
```

Befehle im Detail

set fctl	Einstellen und anzeigen der Controlflags eines Files

Jedem File sind zwei 16-Bit-Control-Worte zugeordnet, eins für Schreiben und eins für Lesen. Darin werden Flags gespeichert, die das Verhalten bei Schreib- und Leseaktionen bestimmen.

Hier wird die Bedeutung des Bit 0 ‚**BLOCKING_WRITE**' beschrieben. Alle anderen Bits dürfen nicht verändert werden.

Wenn z. B. Daten in ein File geschrieben werden, dann gibt es einen Task (Datenerzeuger), der die Daten in einen Puffer schreibt und ein Hardwaremodul bzw. ein weiterer Task (Datenabnehmer), der die Daten physikalisch in das File überträgt. Beide Mitwirkende sind durch jeweils eine maximale Datenrate gekennzeichnet. Ist die Datenrate der Datenentstehung kleiner als die Datenrate, mit der der Datenabnehmer die Daten in das File schreiben kann, dann gibt es keine Verzögerung.

Ist die Erzeugerdatenrate größer als die Abnehmerdatenrate, dann kommt es zu einem Konflikt. Der Erzeuger muss warten bis er alle Daten los wird oder er wartet nicht, es gehen aber Daten verloren. Um einen kurzzeitigen Datenburst aufnehmen zu können, sind die meisten Datenfiles gepuffert. Dieser Puffer kann aber nur kurzzeitig eine höhere Datenrate aufnehmen. Wird der Puffer voll, dann ist der Konflikt wieder da.

Wenn der Datenerzeuger wartet, bis er seine Daten vollständig in das File schreiben kann, dann nennt man das ‚**blocking write**'. Das Gegenteil, also nicht warten und es gehen eventuell Daten verloren, heißt ‚**nonblocking write**'.

Mit diesem Kommando kann diese Eigenschaft für ein File angezeigt und geändert werden (Bit 0).

Beispiele:

Die Files **coma, comb, comc, gps1, gps2, gsm** und **blth** werden standardmäßig im ‚**blocking mode**' betrieben.
Die Files **pipe1, pipe2 und pipe2** werden standardmäßig im ‚**nonblocking mode**' betrieben.
Die Files **icom{1|2|3|4}** werden standardmäßig im ‚**nonblocking mode**' betrieben.
Die Files **ips{a|b|c|d}{1|2|3|4}** werden schreibend im ‚**blocking mode**' und lesend im ‚**nonblocking mode**' geöffnet.
Wird aber ein File **ips{a|b|c|d}*** geöffnet, dann wird für Schreiben für alle auf ‚**nonblocking mode**' umgestellt.

Parameter:

```
[rd|wr] <filename> [{clr|set} <val>]   get/clr/set file ctrl flags for rd/wr, bit 0=1 -> GF_BLOCKING_WR
```

`[rd|wr]`

Der optionale Parameter bestimmt für welche Datenrichtung das Flag Control Word angezeigt bzw. geändert werden soll.

`<filename>`

Name des Files

Befehle im Detail

set fctl	Einstellen und anzeigen der Controlflags eines Files

```
{clr|set} <val>
```

Mit ‚**clr**' werden die Bits im Flag Control Word gelöscht, die in **<val>** gesetzt sind.

Mit ‚**set**' werden die Bits im Flag Control Word gesetzt, die auch in **<val>** gesetzt sind.

Beispiel:

```
set fctl wr ipsa clr 1
```

Ausgabe:

```
fctl: wr ipsa1 0x8000
```

Befehle im Detail

set fread-as-container	Einstellungen für das spezielle Containerfile anzeigen und verändern

Im FAT-Dateisystem auf der Mikro-SD-Karte kann ein spezielles Containerfile angelegt werden. Ein Containerfile hat die Eigenschaft, dass Daten bis zu einer angegebenen Größe geschrieben werden können. Ist die Maximalgröße erreicht, dann werden die ältesten Daten wieder überschrieben. Von der Funktion her ist es ein Ringspeicher. Die Länge des Containerfiles ist bei der Erzeugung 512 Byte. Es wächst erst wären des Beschreibens auf die Maximalgröße an.

Die Eigenschaften des Containerfiles werden in einem 512 Byte langen Fileheader gespeichert.

Der Fileheader:

Signatur	8 Byte	"kx<Y-N\n"
Offset	4 byte, unsigned 32 Bit, little endian	Offset des nächsten schreibenden Zugriffs. Gleichbedeutend mit Offset der ältesten Daten, wenn die Maximalgröße erreicht wurde.
Maximalgröße	4 Byte, unsigned 32 Bit, little endian	maxsize
Zähler für Vollschreibereignisse	4 Byte, unsigned 32 Bit, little endian	loopcount

Die gesamte in den Container geschriebene Datenmenge ist **loopcount*maxsize+offset**. Davon sind aber immer nur **maxsize** Byte abrufbar wenn **loopcnt** > 0 ist anderenfalls nur die Datenmenge **offset**.

Das Containerfile wird wie andere Files behandelt. Wenn es angezeigt oder z. B. kopiert wird, dann ist der Fileheader transparent.

Mit diesem Kommando kann der Zugriffsmode umgestellt werden. die Standardeinstellung ist **on**.

Beispiel:

Containerfile anlegen mit Maximallänge 1024000 Byte (1 Mbyte) und kopieren des Textes „Hier beginnt das File“:

```
wrfile -xb -c 1024000 c:/cntnr/c1 "Hier beginnt das File" 13 10
```

Ausgabe mit ‚**hexdump**‘:

```
hexdump c:/cntnr/c1

00000000  48 69 65 72 20 62 65 67 69 6e 6e 74 20 64 61 73  Hier beginnt das
00000010  20 46 69 6c 65 0d 0a                              File..
```

Umstellung des Zugriffs-Modes: Siehe f.

Befehle im Detail

set fread-as-container	Einstellungen für das spezielle Containerfile anzeigen und verändern

Umstellung des Zugriffs-Modes:

```
set fread-as-container off

Ausgabe:

hexdump -f32 -b 8  c:/cntnr/c1


00000000  593c786b 0a1f4e2d 00000017 000fa000 00000000 00000000 00000000 00000000  kx<Y-N..........................
00000020  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  ................................
00000040  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  ................................
00000060  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  ................................
00000080  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  ................................
000000a0  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  ................................
000000c0  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  ................................
000000e0  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  ................................
00000100  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  ................................
00000120  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  ................................
00000140  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  ................................
00000160  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  ................................
00000180  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  ................................
000001a0  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  ................................
000001c0  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  ................................
000001e0  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  ................................
00000200  72656948 67656220 746e6e69 73616420 6c694620 000a0d65                    Hier beginnt das File...
```

Es ist der 512 Byte lange Fileheader zu erkennen und die initial gespeicherten Daten.

Nicht vergessen wieder zurückzustellen, weil sonst weitere Zugriffe dieses File als Normalfile interpretieren!

```
set fread-as-container off
```

Hinweis:

Ein Containerfile kann auf dem 40xx in ein reguläres File kopiert werden mit:

cp <container-file> <non-container-file>

Das Containerfile bleibt unverändert und das **<non-conatiner-file>** kann dann im USB-Stickmode auf den PC kopiert werden, wo es dann als normales File weiterverarbeitet werden kann.

Wird ein Containerfile im USB-Stickmode auf einen PC übertragen, dann muss auf dem PC mit spezieller Software, die Kenntnis von dem Header hat, auf die Daten im Container zugegriffen werden, weil unter Windows und Linux der Dateityp nicht bekannt ist.

Parameter:

```
[on|off]
```

```
set fread-as-container on
```

Das **ppmOS** kennt diesen Dateityp und behandelt Schreib- und Leseaktionen entsprechend.

Befehle im Detail

set fread-as-container	Einstellungen für das spezielle Containerfile anzeigen und verändern

```
set fread-as-container off
```

Im ppmOS sind die Erkennung und extra Behandlung von Containerfiles deaktiviert. Containerfiles werden wie normale Files behandelt.

Wird auf Containerfiles nur lesend zugegriffen, dann bleibt ihre innere Struktur erhalten. Nach einem erneuten ‚**set fread-as-container on**' ist das File wieder als Containerfile verwendbar.

Befehle im Detail

set intp	Kommandozeileninterpreter für eine Schnittstelle aktivieren oder deaktivieren

An verschiedenen Schnittstellen werden die Eingangsdaten von einem sogenannten Shell-Interpreter oder Kommandointerpreter ausgewertet. Bei diesem werden die empfangenen Daten, die als ASCII-Zeichen erwartet werden und die durch die Zeichen CR+LF, CR oder nur LF getrennt sind, interpretiert. Das Datenformat ist also zeilenorientiert. An diesen Schnittstellen ist mindestens bis zur Version 1.46 des **ppmOS** eine grundlegende Terminalimplementierung realisiert. Grundlegend deshalb, weil erweiterte Terminalzeichen und Terminalsteuerungen nicht ausgewertet werden.

Diese Schnittstellen sind:

- coma, comb, comc, blth,
- ips{a|b|c|d}{1|2|3|4} nur wenn nicht mit ‚**-i**' gestartet
- GSM-TCP-Verbindung
- SMS-Empfang über das GSM-Modem

Diese Kommandointerpreter ermöglichen die Konfiguration und Steuerung des 40xx durch angeschlossene Geräte bzw. von entfernter Software.

Ist die Fernsteuerung nicht erforderlich bzw. aufgrund der Konfiguration nicht möglich, dann kann mit dem Kommando ‚**set intp off**' der Shell-Interpreter abgeschaltet werden. Die ankommenden Daten können dann weitergeleitet oder anderweitig verwendet werden.

Wird an einer Schnittstelle, die standardmäßig keinen Shell-Interpreter hat, trotzdem einer benötigt, dann besteht die Möglichkeit, die empfangenen Daten über eine Ausgabeumleitung an **stdin** oder über eine Eingabeumleitung an den Eingabepuffer einer der Schnittstellen **coma, comb, comc** oder **blth** weiterzuleiten.

Beispiel:

Die an **icom1** (hat keinen Shell-Interpreter) ankommenden Daten sollen von einem Shell-Interpreter ausgeführt werden:

```
cp -t -8 -s icom1 stdin&
```

Sollen die Ausgabedaten der Shell auch in **icom1** erscheinen, dann kann die Umleitung so erfolgen:

```
set ord coma tee icom1
```

die Ausgabedaten erscheinen in coma **und** icom1.

Oder:

```
set ord coma icom1
```

die Ausgabedaten erscheinen **nur** an icom1.

Parameter:

```
[<task_name>] [on|off [<unlock_key]]
```

Befehle im Detail

set intp	Kommandozeileninterpreter für eine Schnittstelle aktivieren oder deaktivieren

`[<task_name>]`

Der optionale Parameter bestimmt den Task, dessen Interpreter-Einstellung angezeigt oder verändert werden soll.

`[on|off [<unlock_key]]`

Ohne ‚**on**' oder ‚**off**' wird die aktuelle Einstellung angezeigt.

Mit ‚**on**' wird der Interpreter ab- und mit ‚**off**' wird er eingeschaltet.

Hinweis:
Bei Tasks, die ohne eigenen Interpreter gestartet wurden, wird der Zustand des Tasks angezeigt oder geändert, der diesen Task gestartet hat, was in der Regel der Task ‚**coma**' ist.

<unlock_key>
Der Interpreter wird abgeschaltet und nach Erkennen der <unlock_key>-Zeichenfolge (CR+LF nicht erforderlich) an derselben Schnittstelle, wird er wieder aktiviert. Dabei muss <unlock_key> zwischen zwei Punkten „.<unlock_key>." stehen.

Befehle im Detail

set io	Digitale Ein- und Ausgänge des Gerätes abfragen und einstellen

Das Gerät 40xx verfügt über eine Reihe von Ein- und Ausgabeleitungen, die mit diesem Kommando angezeigt und eingestellt werden können. Es gibt die beiden IO-Buchsen auf der Rückseite, die je 2 Ausgabe- und 2 Eingabeleitungen dem Anwender zur Verfügung stellen. In der Buchse der Schnittstelle coma sind 2 Ausgabeleitungen, die für eine externe Anwendung verwendet werden können.

Parameter:

```
          io  [-s] [out1.1] [out1.2] [out2.1] [out2.2] [gpio5] [buzzer] [can-tx] [hi|lo|tgl]
        | io  [-s] {[in1.1] [in1.2] [in2.1] [in2.2]} [irq [off|hl|lh|both|clr]] via && or || usable
        | io  [-s] in-all                            [irq [off|hl|lh|both|clr]] via && or || usable
        | io  [-s] quadrature-encoder {all|in1|in2}  [irq [off|hl|lh|both|clr]] via && or || usable
        | io  [-s] {event|varfreq|pps}               [irq [off|hl|lh|both|clr]] via && or || usable
        | io  [-s] {event|varfreq|pps|gps-led1}      [out|in|hi|lo|tgl]         via && or || usable
        | io  [-s] can-rx                                                       via && or || usable
        | io  [-s] [rst-gps1|rst-imu] [hi|lo|tgl]                               low active
```

`[-s]`

Mit diesem optionalen Schalter werden alle Textausgaben unterdrückt. Das ist dann nützlich, wenn nur der Rückgabewert des Kommandos oder die Änderung einer Ausgabeleitung von Bedeutung ist.

Die Alternative hierzu ist die allgemein anwendbare Ausgabeumleitung nach ‚**null**', also ‚**set io ... >null**'.

Der Vorteil bei der Verwendung von ‚**-s**' gegenüber ‚... **>null**' besteht in der geringeren Laufzeit des Kommandos (z. B. ca. 70% bei ‚**set io in1.1 -s**' und 50% bei ‚**set io in-all -s**').

```
set io [-s] [out1.1] [out1.2] [out2.1] [out2.2] [gpio5] [buzzer] [can-tx] [hi|lo|tgl]
```

Mit diesem Kommando werden die in der Kommandozeile aufgeführten Ausgabesignale abgefragt (ohne **hi|lo|tgl**) oder entsprechend eingestellt (mit **hi|lo|tgl**).

Der Schalter ‚**tgl**' bedeutet umschalten in den anderen Signalzustand, ‚**hi**' auf high stellen und ‚**lo**' auf low stellen.

Wenn mehr als ein Signal angezeigt oder eingestellt werden soll, dann müssen die Signalnamen in der hier angegebenen Reihenfolge aufgeführt werden.

Beispiel 1:

```
set io
io: out 0 0  0 0  in 0 0  0 1  gpio5 0  buzzer 0  can tx 1 rx 1  rst-gps1 0  rst-gps2 0
```

Der Zustand aller Signale wird angezeigt.

Beispiel 2: Siehe nächste Seite.

Befehle im Detail

set io	Digitale Ein- und Ausgänge des Gerätes abfragen und einstellen

”`set io [-s] [out1.1] …`” f.:

Beispiel 2:

```
set io out1.1 out2.2
io: out1.1   0
io: out2.2   0
```

Alle angegebenen Pins werden dabei genau zum selben Zeitpunkt eingestellt.

Beispiel 3:

```
set io buzzer hi
```

Der Signaltongeber geht an.

```
set io [-s] {[in1.1] [in1.2] [in2.1] [in2.2]} [irq [off|hl|lh|both|clr]]
set io [-s] in-all                            [irq [off|hl|lh|both|clr]]
```

Die Eingabeleitungen der beiden IO-Buchsen werden abgefragt. Hier ist mindestens ein Signalleitung anzugeben oder ‚**in-all**‘ für alle vier Signalleitungen.

Alle angegebenen Pins werden genau zum selben Zeitpunkt abgefragt.

Das Kommando hat einen Rückgabewert, der in einer bedingten Kommandoausführung genutzt werden kann, um je nach Signalzustand eine bestimmte Aktion auszulösen. Jede Eingabeleitung ist mit einem Bit im Rückgabewert repräsentiert, das gesetzt ist, wenn ein High-Pegel an der Eingabeleitung anliegt:

in1.1 → Bit 0

in1.2 → Bit 1

in2.1 → Bit 2

in2.2 → Bit 3

Mit dem Zusatz ‚**irq**‘ wird eine Interrupt-basierte Zählung von Flankenwechsel für diese Pins eingestellt:

`off`	Die Interrupt-Zählung wird abgeschaltet.
`hl`	Die Interrupt-Zählung für high-low-Übergänge wird aktiviert.
`lh`	Die Interrupt-Zählung für low-high-Übergänge wird aktiviert.
`both`	Es werden high-low- und low-high-Übergänge Interrupt-basiert gezählt.
`clr`	Der Impulszähler wird auf 0 gesetzt.

Der Rückkehrwert ist immer die Summe der Interrupt-Ereignisse aller abgefragten Eingabeleitungen. Wird nur eine Eingabeleitung angegeben, was die Standardanwendung sein dürfte, dann ist der Rückgabewert die Zahl der Interrupt-Ereignisse an dieser Leitung.

Siehe nächste Seite.

Befehle im Detail

set io	Digitale Ein- und Ausgänge des Gerätes abfragen und einstellen

```
"set io [-s] {[in1.1] [in1.2] [in2.1] [in2.2]} [irq [off|hl|lh|both|clr]]
set io [-s] in-all                              [irq [off|hl|lh|both|clr]]" f.:
```

Mit ‚**set io ... irq clr**' werden die Zähler der angegebenen Leitungen ausgelesen und anschließend auf 0 gesetzt (read before clear). Der Rückgabewert ist die Zahl der Interrupt-Ereignisse, die bis dahin gezählt wurden. Es ist gesichert, dass keine Interrupts während aufeinanderfolgender Abfragen mit anschließenden Löschen verloren gehen.

Beispiel 1:

```
set io in-all
io: in 0 0 0 0 irq both lh   di   di   cnt        2        2        0        0        0        2        0
```

Hinter ‚**irq**' ist die Einstellung der Interrupt-Zählung ersichtlich. Hinter ‚**cnt**' folgen die vier Interrupt-Zähler der Eingänge.

Die letzte Zahl in der Ausgabezeile gibt die Anzahl der Eventinterrupts (SUB-D25 Pin 23, Systemkabel COMA SUB-D9 Pin 8) an.
Die vorletzte Zahl gibt die Anzahl der USB-Kabel-Steck- und Trennvorgänge an.
Die drittletzte Zahl gibt die Anzahl der Micro-SD-Karten-Steck- und Auswerfvorgänge an.

Beispiel 2:
Dazu in der IO-Buchse 1 die Signale **out1.1** und **in1.1** brücken.

Initialisierung:

```
set io in1.1 irq both
loop -8 -t 1000 "@set io -s in1.1 irq clr || @stat -e; @echo Es traten Interrupts auf"&
```

Testen:

```
time loop 1000 "@set io -s out1.1 hi;@set io -s out1.1 lo"
```

Ausgabe:

```
>loop 1000 "@set io -s out1.1 hi;@set io -s out1.1 lo"
cpu time 0.109158629 s
last command return code: 2000 (0x7d0)
Es traten Interrupts auf
```

Es ist ein Rechtecksignal mit ca. 9 kHz und 1000 Perioden erzeugt worden, welches korrekt mit ‚**set io -s in1.1 irq clr**' gezählt wurde.

Pulse sind bis minimal ca. 40 ns Breite detektierbar.

Hinweis:
Die Interrupt-Impulszählung ist für high/low oder low/high-Flanken bis höchstens zu ca. 1 MHz geeignet. Für die Einstellung ‚**... irq both**' sind es ca. 500 kHz. Werden höhere Frequenzen angelegt, dann kann es zu einem Watchdog-Reset kommen, da die CPU-Belastung durch die Interrupt-Bearbeitung so hoch wird, dass das Multitasking zu stark verzögert und die Laufzeitüberwachung ein Reset herbeiführt.

Befehle im Detail

set io	Digitale Ein- und Ausgänge des Gerätes abfragen und einstellen

```
io  [-s] quadrature-encoder {all|in1|in2}  [irq [off|hl|lh|both|clr]] via && or || usable
```

Die Event_Ins der I/O-Ports können, wenn mit diesem Kommando entsprechend konfiguriert, als “Rotary Decoder” mit vorzeichenbehafteter Drehrichtung interpretiert und genutzt werden. D.h. man kann daran einen rotierenden Weg– und/ oder Winkelsensor oder einen anderen beliebigen Impulsgeber anschließen und eine Pulszählung durchführen. Diese Pulszählung kann dann für bedingte Kommandoausführungen genutzt werden.

Die Auswertung bzw. Weiterverwendung der Pulszählung kann per Skriptkommandos realisiert werden, d.h. eine bedingte Ausführung triggern, die Daten in Echtzeit anzeigen (Geschwindigkeit, Weg) oder die Daten in ein File für spätere Auswertung speichern.

Durch zwei I/O-Ports am 40xx stehen dem Anwender zwei unabhängige Decoder zur Verfügung.

Im 40xx gibt es nur die Signale A/B pro konfiguriertem Decoder, also ohne Referenzimpuls. Wenn ein Referenzimpuls erforderlich ist, dann kann dafür der Eventeingang oder in2 verwendet werden. Das Besondere an der Impulsaufnahme an zwei Kontakten oder Lichtschranken ist die versetzte Anordnung (90° Phasenlage: während ein Signal einen Flankenwechsel hat, hat das zweite Signal high oder low). Das ermöglicht neben der Impulszählung die zusätzliche Erkennung der Dreh- bzw. Bewegungsrichtung. Die Impulszählung während der Bewegung ermöglicht bei Kenntnis der geometrischen Verhältnisse eine Ermittlung der zurückgelegten Wegstrecke oder Geschwindigkeit.

Als Beispiel könnte ein Peiselerrad mit 2000 Pulsen pro Umdrehung und ca. 2 m Umfang (26") (https://www.peiseler-gmbh.de/p5rad.html) theoretisch bis zu einer Geschwindigkeit von ca. 100 m/s (360 km/h) verwendet werden!

Es bedeuten:

-s	silent, keine Ausgabe, Verwendung mit bedingter Kommandoausführung && oder ||
all|in1|in2	Auswahl des Eingangs
irq	Steuerung der Impulszählung über Interrupts
off	Abschaltung bzw. keine Interruptzählung
hl	nur Zählung der high low Flanken an in1.1 in1.2 bzw. in2.1 und in2.2
lh	nur Zählung der low high Flanken
both	Zählung aller Flanken, ergibt doppelte Impulszahl gegenüber hl oder lh
clr	Impulszähler auf 0 setzen

Bei der Einstellung **irq hl** oder **irq lh** ist die maximale Zählfrequenz ca. 100 kHz und bei **irq both** ist diese max. ca. 50 kHz.

Beispiel 1:

Ausgabe ist ein Geschwindigkeitsäquivalent:

```
cron put cntin1 * * * * * * * -i "@set io quadrature-encoder in1 irq clr"
```

oder

Ausgabe ist eine Wegäquivalent, also ohne 'clr'

```
cron put cntin1 * * * * * * * -i "@set io quadrature-encoder in1 >comb"
```

Befehle im Detail

set io	Digitale Ein- und Ausgänge des Gerätes abfragen und einstellen

” `io [-s] quadrature-encoder ...`” f.: ,

Beispiel 2:

In diesem Beispiel wird die in einem Zeitabschnitt gezählte Pulszahl (Wegstrecke, Drehwinkel) gegen einen Grenzwert verglichen. Ist die Pulszahl größer oder gleich (gte - greater or equal), dann passiert nichts, die innere loop-Schleife wird nicht verlassen. Ist die Pulszahl in 1000 ms kleiner als 1900, dann wird der Text "Pulslimit unterschritten" ausgegeben. Anstelle von echo kann natürlich ein beliebiges Kommando/Skript ausgeführt werden. Die Grenzfrequenz in diesem Beispiel ist 950 Hz.

set io quadrature-encoder in1 irq hl

loop -8 -t 1000 "@loop -8 -t 1000 -gte 1900 '@set io -s quadrature-encoder in1 irq clr'; @echo Pulslimit unterschritten"&

`io [-s] {event|varfreq|pps} [irq [off|hl|lh|both|clr]] via && or || usable`

Mit diesem Unterkommando werden Einstellung zur Interrupt-Zählung an den folgenden externen Ein– bzw. Ausgängen vorgenommen:

- EVENT_in (SUB-D25 Pin 23, Systemkabel COMA SUB-D9 Pin 8)
- VARF (SUB-D25 Pin 11, Systemkabel COMA SUB-D9 Pin 7)
- PPS out (SUB-D25 Pin 15, Systemkabel COMA SUB-D9 Pin 9)

Der aktuelle Signalpegel kann ebenfalls angezeigt werden.

Die Parameter sind dieselben wie für die Signale **in1.1, in1.2, in2.1** und **in2.2**.

Befehle im Detail

set io	Digitale Ein- und Ausgänge des Gerätes abfragen und einstellen

```
set io [-s] {event|pps|gps-led1|varfreq} [out|in|hi|lo|tgl]
```

Mit diesem Unterkommando werden Signale, die für die GPS-Board-Steuerung verwendet werden, eingestellt.

Signal	Standardeinstellung der Controller-Pins	am SUB-D25-Steckverbinder (nicht änderbar)	Anschluss
`event`	Eingang	Eingang	SUB-D25 Pin 23 und im PPM-Systemkabel COMA SUB-D9 Pin 8
`pps`	Eingang	Ausgang	SUB-D25 Pin 15 und im PPM-Systemkabel COMA SUB-D9 Pin 9
`gps-led1`	Ausgang	-	GPS-Adapterboard X1.15 und X2.11 => Event2-Eingang (Novatel)
`varfreq`	Eingang	Ausgang	SUB-D25 Pin 11 und im PPM-Systemkabel COMA SUB-D9 Pin 7

Das Kommando hat bei fehlerfreier Ausführung einen Rückgabewert, der für eine bedingte Shell-Kommandoausführung genutzt werden kann. Es ist der Zustand entweder des angelegten Signals, wenn auf ‚**in**' programmiert oder der ausgegebene Zustand, wenn auf ‚**out**' programmiert wurde.

Mit ‚**out**' wird das angegebene Signal aus der Sicht des Controllers auf Ausgang und mit ‚**in**' auf Eingang programmiert.

Wenn das Signal auf Ausgang programmiert wurde, kann mit ‚**hi**' oder ‚**lo**' der Ausgang entsprechend gesetzt oder mit ‚**tgl**' umgeschaltet werden.

Am externen Eventeingang, der mit dem GPS-Board verbunden ist, wird üblicher Weise extern ein Trigger-Signal angelegt, um z. B. vom GPS-Board eine extra Positionsberechnung genau zum Zeitpunkt des Signals berechnen zu lassen. Das **ppmOS** kann bei entsprechender Skriptprogrammierung dieses Signal ebenfalls erkennen und eine weitere Aktion auslösen.

Es ist aber auch möglich, das Event-Controller-Pin auf Ausgang (‚**out**') zu programmieren. Dann ist das **ppmOS** in der Lage, über eine entsprechende Skriptprogrammierung selbst ein Trigger-Signal an das GPS-Board zu geben.

Beispiel: Nutzung einer FSAS-IMU an einem 40xx ohne GPS-Board (200Hz)

```
set io varfreq out
loop -8 -t 5 -f "@set io -s varfreq hi; @wait -s 0.0025; @set io -s varfreq lo"&
```

Befehle im Detail

set io	Digitale Ein- und Ausgänge des Gerätes abfragen und einstellen

```
set io [-s] can-rx
```

Abfrage der CAN-Rx-Leitung. Das Kommando dient Testzwecken (z. B. ob die Verkabelung fehlerfrei ist).

Beispiel:

```
set io can-tx lo; @set io can-rx

io: can tx   0
io: can rx   0

set io can-tx hi; @set io can-rx

io: can tx   1
io: can rx   1
```

Die Leitungen CAN-H und CAN-L (CAN-Buchse am Backpanel) müssen mit Abschlusswiderständen (je 120 Ohm an den Enden) verbunden sein.
Wenn beide Male der gleiche Signalzustand (tx = rx) angezeigt wird, dann ist das ein Hinweis, dass die Verkabelung in Ordnung sein kann. Dies ist ein notwendiger Test aber noch kein hinreichender. Schlägt der Test fehl, dann das Kabel trennen und mit einem Prüfstecker o. ä. die beiden Anschlüsse in der Buchse direkt mit einem Widerstand brücken und den Test wiederholen.

Hinweis:
Der Test ist nur möglich, wenn das CAN-Modul noch nicht initialisiert wurde (‚**can init bps ...**').

```
set io [-s] [rst-gps1|rst-imu] [hi|lo|tgl]
```

Abfrage oder Einstellung der Reset-Leitungen, die zu den GPS-Boards geführt sind. Das GPS-Board bzw. die IMU wird zurückgesetzt, wenn das Signal auf low (lo) gesetzt wird.
Damit ist es möglich, das/die eingebaute GPS-Board/IMU zurückzusetzen.

Beispiel:
GPS-Board zurücksetzen

```
set io -s rst-gps1 lo; @set io -s rst-gps1 hi
```

Befehle im Detail

set ird	Eingabeumleitung für UART-Schnittstellen anzeigen und einstellen

Mit diesem Kommando können Umleitungen der UART-Schnittstellen des Controllers in Eingaberichtung angezeigt und geändert werden.

Standardmäßig ist das Rx-Pin einer UART des Controllers mit dem Eingabepuffer des UART-Treibers verbunden. Dieser gibt bei Empfang von Zeichen ein Signal indem er ein Signalbit im Event FLag Word des damit verbundenen Task setzt. Dieser Task liest die Daten aus und verarbeitet sie.

Im UART-Treiber sind Möglichkeiten implementiert, die Daten alternativ oder zusätzlich an andere Ziele weiterzuleiten:

1. in den Empfangspuffer einer anderen UART

2. in den Ausgangspuffer einer anderen UART

3. in das Ausgaberegister einer anderen UART (ungepuffert)

4. in ein anderes File des **ppmOS** (**icom1{1|2|3|4|5}, ips{a|b|c|d}{*|1|2|3|4}, pipe{1|2|3}, can1-**..., **null**, FAT-Dateisystem nur über pipe möglich)

Daten, die an den Empfangspuffer einer zweiten UART weitergeleitet werden, werden von dem Besitzertask der zweiten UART so verarbeitet, als ob sie am Rx-Pin der zweiten UART empfangen wurden.

Diese Weiterleitung erfolgt zeichenweise in der Interrupt Service Routine der betreffenden UART.

Die Daten können maximal in 3 Streams weitergeleitet werden (bei der Anwendung von ‚**tee**' und ‚**-r**'). Im Minimum erfolgt keine Weiterleitung (‚**set ird ... null**'), die Daten werden verworfen.

Parameter:

```
ird [{<com_name>|<com_alias>} [tee]
     {  [-ob|-ib|-r [-eq] [<baud_dest>]] <com_name>|<com_alias>
      | [-a] <file>
      | [-r] off [all]
     }
    ]
```

```
set ird
```

Anzeige der Eingabeumleitung aller UARTS:

```
ird com1 (blth)  ->
ird com2 (comc)  -> ib_com2
ird com3 (coma)  -> ib_com3
ird com4 (comb)  -> ib_com4
ird com5 (    )  ->
ird com6 (gps2)  -> ib_com6                          tee pipe2
ird com7 (gsm )  -> ib_com7
ird com8 (gps1)  -> ib_com8                          tee pipe1
```

Bei allen UART's werden die empfangenen Zeichen in den dazugehörigen Empfangspuffer geschrieben (→ **ib_...**).

Die Zeichen, die die UART gps1 und gps2 empfängt werden zusätzlich nach pipe1 bzw. pipe2 kopiert.

Befehle im Detail

set ird	Eingabeumleitung für UART-Schnittstellen anzeigen und einstellen

`{<com_name>|<com_alias>}`

Angabe der UART als systematischer Name (**com1, com2, com...**) oder als Aliasname (**coma, comb, comc, ...**).

`[tee]`

Der optionale Schalter gibt an, dass sich das Kommando auf eine zusätzliche Weiterleitung bezieht. Man kann die Weiterleitung wie das freie Ende eines T-Stücks betrachten, dass in einer Leitung eingebaut ist.

Ohne diesen Parameter wird der normale Pfad (**ib_...**) geändert bzw. wieder hergestellt.

`set ird {<com_name>|<com_alias>} [tee] [-ob|-ib|-r [-eq] [<baud_dest>]] <com_name>|<com_alias>`

Weiterleitung an eine andere UART, die mit dem systematischen Namen (**com1, com2, com...**) oder mit dem Aliasnamen (**coma, comb, comc, ...**) benannt wird.

Die Weiterleitung ist in 4 Varianten möglich:

`-ob`	Weiterleitung in den Ausgabepuffer der Ziel-UART. Weiterleitung unabhängig von einer Umleitung.
`-ib`	Weiterleitung in den Eingabepuffer der Ziel-UART, Zeichen wird vom Shell-Interpreter (wenn aktiviert) der Ziel-UART interpretiert. Weiterleitung unabhängig von einer Umleitung.
`-r`	Weiterleitung durch direktes Schreiben des empfangenen Zeichens in das Ausgaberegister der UART. Diese Variante ist die schnellste, aber eine eventuelle weitere Ausgabe an der Ziel-UART wird gestört. Wird noch ‚**-eq**' angegeben, dann wird zunächst die Baudrate der Quell-UART an der Ziel-UART eingestellt (angeglichen). Wird nur **<baud_dest>** angegeben, dann wird die Ziel-UART auf die angegebene Baudrate eingestellt. Die Quell-UART bleibt unverändert. Wird ‚**-eq**' und **<baud_dest>** angegeben, dann werden Quell- und Ziel-UART auf die angegebene Baudrate eingestellt. Die Weiterleitung ‚**... -r comb**' ist nicht möglich.
ohne Schalter	Weiterleitung an den Eingang der Weiterleitungslogik der Ziel-UART. D. h. wenn die Ziel-UART ausgabeseitig umgeleitet ist, dann werden auch diese Zeichen dort hin umgeleitet. Diese Umleitung benötigt mehr CPU-Zeit als ‚**-ob**' und wesentlich mehr Zeit als die Variante ‚**-r**'.

In der Regel sollte für eine UART-UART-Umleitung ‚**-ob**' verwendet werden.

Die ‚**-r**'-Umleitung wird bei hohen Datenraten und wenn keine weiteren Ausgaben an der Ziel-UART vorkommen empfohlen. Die Baudrate an der 2. UART muss größer oder gleich der Baudrate der originalen UART sein.

Befehle im Detail

set ird	Eingabeumleitung für UART-Schnittstellen anzeigen und einstellen

```
set ird {<com_name>|<com_alias>} [tee] [-a] <file>
```

Die Weiterleitung erfolgt in ein anderes File des **ppmOS** (**icom1{1|2|3|4|5}, ips{a|b|c|d}{*|1|2|3|4}, pipe{1|2|3}, can1-..., null**).

Beispiel 1:

```
set ird gps2 tee ipsb*
```

Die an **gps2** empfangenen Daten werden zusätzlich dem IP-Server ‚ipsb' zur Verfügung gestellt. Alle verbunden Clients erhalten diesen Datenstrom (Jokerzeichen ‚*****' am Ende von ‚**ipsb**').

Die Weiterleitung an ein File im FAT-Dateisystem muss über eine Zwischenpufferung in **pipe{1|2|3}** erfolgen.

Beispiel 2:

```
set ird gps1 tee pipe1; cp -t -8 pipe1 c:/data/gps1-logfile&
```

Der als Task gestartete copy-Befehl arbeitet mit einem unendlichen Timeout (‚**-t -8**') und kopiert die Daten aus **pipe1** in das File im FAT-Dateisystem auf der Micro-SD-Karte.

```
set ird {<com_name>|<com_alias>} [tee] [-r] off [all]
```

Die Datenweiterleitung wird beendet. Dabei bedeutet ‚**off**' Zurückschaltung der Weiterleitung an den für diese UART vorgesehenen Eingabepuffer. Der Schalter ‚**all**' bedeutet, dass ‚**-r**', und ‚**tee**' beendet und die originale Umleitung reaktiviert wird.

Befehle im Detail

set led	Signal-LED des Gerätes ein-, ausschalten und Zustand anzeigen

Mit diesem Kommando werden die Signal-LED des Gerätes gesteuert.

Einige LED haben in der Standardanwendung des Gerätes eine Funktion, die vom **ppmOS** gesteuert wird. Die LED in der Ein-/Austaste blinkt konstant im Sekundentakt. Sie Signalisiert eine ordnungsgemäße Funktion des Multitaskings.

Die LED 1 ist eine Zweifarb-LED (rot und grün). Die LED 2 leuchtet gelb. Die LED 3 leuchtet orange und die LED 4 leuchtet grün.

Die LED 1 wird in der Regel verwendet, um eine gültige Position und das Aufzeichnen eines Logfiles zu signalisieren.

Mit diesem Kommando kann jede LED separat geschaltet werden. Die Einstellung ist so lange gültig, bis das **ppmOS** eventuell eine Umschaltung vornimmt.

Parameter:

```
[red] [green] [yellow] [orange] [green2] [button] [on|off|tgl]
```

```
[red] [green] [yellow] [orange] [green2] [button]
```

Benennung der LED, die geschaltet werden soll. Die LED ‚**green**' ist eine der beiden LED in LED 1 und ‚**green2**' ist die LED 4.

Es können mehrere LED in einem Kommando geschaltet werden. Dazu sind die Namen in der oben angegebenen Reihenfolge anzugeben.

```
[on|off|tgl]
```

Dieser optionale Parameter stellt den Zustand ein. Der Parameter ‚**tgl**' bedeutet umschalten, also von ‚**on**' nach ‚**off**' oder umgekehrt.

Ohne diesen Parameter wird der aktuelle Zustand angezeigt.

Das Kommando hat einen Rückgabewert, in dem ein gesetztes Bit bedeutet, dass die betreffende LED an ist.

Bit 0	red
Bit 1	green
Bit 2	yellow
Bit 3	orange
Bit 4	green2
Bit 5	button

Der Rückgabewert ‚**-1**' bedeutet Kommandozeilenfehler.

Befehle im Detail

set loop	Loopback-Mode für Shell-Interpreter anzeigen und einstellen

Mit diesem Kommando wird der Loopback-Mode eines Shell-Interpreters angezeigt bzw. eingestellt.

Loopback-Mode bedeutet, dass die Daten, die an einer Task-Shell ankommen, unmittelbar wieder ausgegeben werden. Der Loopback-Mode ist nur wirksam, wenn der Shell-Interpreter deaktiviert ist. Siehe dazu auch ‚**set intp ...**' S. 247 und ‚**set echo ...**' S. 239.

Das Kommando wird in der Regel nur zu Testzwecken angewendet.

Parameter:

```
[<task_name>] [on|off]    has effect if intp is off
```

```
[<task_name>]
```

Der optionale Parameter gibt den Task an, bei dem der Shell-Interpreter-Loopback-Mode eingestellt bzw. angezeigt werden soll.

Ohne diese Angabe wird die Task-Shell angesprochen, bei der diese Eingabe ankommt.

```
[on|off]
```

Der optionale Parameter gibt an, ob der Loopback-Mode ein- oder ausgeschaltet werden soll.

Ohne ‚**on**' oder ‚**off**' wird der aktuelle Zustand angezeigt.

Befehle im Detail

set ord	Ausgabeumleitung für UART-Schnittstellen anzeigen und einstellen

Mit diesem Kommando können Umleitungen der UART-Schnittstellen des Controllers in Ausgaberichtung angezeigt und geändert werden.

Standardmäßig ist das Tx-Pin einer UART des Controllers mit dem Ausgabepuffer des UART-Treibers verbunden. Wenn Daten zu senden sind, wird der Ausgabeinterrupt der UART aktiviert. Die Interrupt Service Routine holt dann Zeichen für Zeichen aus dem Ausgangspuffer und schreibt sie in das Senderegister der UART.

Die Ausgabeumleitung setzt beim Schreiben von Zeichen in den Ausgabepuffer an. Im Gegensatz dazu setzt die Eingabeumleitung in der Interrupt Service Routine beim Schreiben von einzelnen Zeichen in den Empfangspuffer an.

Es sind Möglichkeiten implementiert, die Daten alternativ oder zusätzlich an andere Ziele weiterzuleiten:

1. in den Empfangspuffer einer anderen UART

2. in den Ausgangspuffer einer anderen UART

3. in das Ausgaberegister einer anderen UART (ungepuffert)

4. in ein anderes File des **ppmOS** (FAT-Dateisystem , **icom1{1|2|3|4|5}, ips{a|b|c|d}{*|1|2|3|4}, pipe{1|2|3}, can1-**..., **null, ...**) ´

Diese Weiterleitung erfolgt blockweise beim Schreiben von Daten in den Ausgabepuffer der UART.

Die Daten können maximal in 3 Streams weitergeleitet werden (bei der Anwendung von ‚**tee**' und ‚**-r**'). Im Minimum erfolgt keine Weiterleitung (‚**set ord ... null**'), die Daten werden verworfen.

Parameter:

```
ord [{<com_name>|<com_alias>} [tee]
     {  [-ob|-ib|-r [-eq] [<baud_dest>]] <com_name>|<com_alias>
      | [-a] <file>
      | [-r] off [all]
     }
    ]
```

```
set ord
```

Anzeige der Ausgabeumleitung aller UARTS.

Beispiel:

```
ord com1 (blth) <-
ord com2 (comc) <-  ob_com2
ord com3 (coma) <-  ob_com3              -r com2
ord com4 (comb) <-  ob_com4
ord com5 (    ) <-
ord com6 (gps2) <-  ob_com6
ord com7 (gsm ) <-
ord com8 (gps1) <-  ob_com8                          tee comb
```

Bei allen UART's werden die zu sendenden Zeichen aus dem dazugehörigen Ausgangspuffer gelesen (<- **ob_...**). Die Zeichen, die an **coma** ausgegeben werden, werden zusätzlich per Registerumleitung an **comc** ausgegeben. Die UART **gsm** ist noch nicht initialisiert und die Ausgabedaten der UART **comc** werden zusätzlich an **comb** ausgegeben.

Befehle im Detail

set ord	Ausgabeumleitung für UART-Schnittstellen anzeigen und einstellen

`{<com_name>|<com_alias>}`

Angabe der UART als systematischer Name (**com1, com2, com...**) oder als Aliasname (**coma, comb, comc, ...**).

`[tee]`

Der optionale Schalter gibt an, dass sich das Kommando auf eine zusätzliche Weiterleitung bezieht. Man kann die Weiterleitung wie das freie Ende eines T-Stücks betrachten, dass in einer Leitung eingebaut ist.

Ohne diesen Parameter wird der normale Pfad (**ob_...**) geändert bzw. wieder hergestellt.

`set ord {<com_name>|<com_alias>} [tee] [-ob|-ib|-r [-eq] [<baud_dest>]] <com_name>|<com_alias>`

Weiterleitung an eine andere UART, die mit dem systematischen Namen (**com1, com2, com...**) oder mit dem Aliasnamen (**coma, comb, comc, ...**) benannt wird.

Die Weiterleitung ist in 4 Varianten möglich:

`-ob`	Weiterleitung in den Ausgabepuffer der Ziel-UART. Weiterleitung unabhängig von einer Umleitung.
`-ib`	Weiterleitung in den Eingabepuffer der Ziel-UART, Zeichen wird vom Shell-Interpreter (wenn aktiviert) der Ziel-UART interpretiert. Weiterleitung unabhängig von einer Umleitung.
`-r`	Weiterleitung durch direktes Schreiben des empfangenen Zeichens in das Ausgaberegister der UART. Diese Variante ist die schnellste, aber eine eventuelle weitere Ausgabe an der Ziel-UART wird gestört. Wird noch ‚**-eq**' angegeben, dann wird zunächst die Baudrate der Quell-UART an der Ziel-UART eingestellt (angeglichen). Wird nur **<baud_dest>** angegeben, dann wird die Ziel-UART auf die angegebene Baudrate eingestellt. Die Quell-UART bleibt unverändert. Wird ‚**-eq**' und **<baud_dest>** angegeben, dann werden Quell- und Ziel-UART auf die angegebene Baudrate eingestellt. Die Weiterleitung ‚**... -r comb**' ist nicht möglich.
ohne Schalter	Weiterleitung an den Eingang der Weiterleitungslogik der Ziel-UART. D. h. wenn die Ziel-UART ausgabeseitig umgeleitet ist, dann werden auch diese Zeichen dort hin umgeleitet. Diese Umleitung benötigt mehr CPU-Zeit als ‚**-ob**' und wesentlich mehr Zeit als die Variante ‚**-r**'.

In der Regel sollte für eine UART-UART-Umleitung ‚**-ob**' verwendet werden.

Die ‚**-r**'-Umleitung wird bei hohen Datenraten und wenn keine weiteren Ausgaben an der Ziel-UART vorkommen empfohlen. Die Baudrate an der 2. UART muss größer oder gleich der Baudrate der originalen UART sein.

Befehle im Detail

set ord	Ausgabeumleitung für UART-Schnittstellen anzeigen und einstellen

```
set ord {<com_name>|<com_alias>} [tee] [-a] <file>
```

Die Weiterleitung erfolgt in ein anderes File des **ppmOS** (FAT-Dateisystem, **icom1{1|2|3|4|5}, ips{a|b|c|d}{*|1|2|3|4}, pipe {1|2|3}, can1-**..., **blth, null**).

Beispiel 1:

```
set ord comb tee ipsb*
```

Die an **comb** ausgebenen Daten werden zusätzlich dem IP-Server ‚**ipsb**' zur Verfügung gestellt. Alle verbunden Clients erhalten diesen Datenstrom (Jokerzeichen ‚*****' am Ende von ‚**ipsb**').

```
set ord {<com_name>|<com_alias>} [tee] [-r] off [all]
```

Die Datenweiterleitung wird beendet. Dabei bedeutet ‚**off**' Zurückschaltung der Ausgabe an das für diese UART vorgesehene Tx-Pin. Der Schalter ‚**all**' bedeutet, dass ‚**-r**', und ‚**tee**' beendet und die originale Umleitung reaktiviert wird.

set path	Einstellen eines festen Systempfads

```
set path [<path>]
```

Mit diesem Kommando kann ein fester Systempfad bzw. Dateiverzeichnis festgelegt werden, in welchem nach aufgerufenen Skripten gesucht wird. Es kann dann beim Aufruf von Skripten auf der SD-Karte der Pfad weggelassen werden. Damit wäre es möglich ein internes Kommando auf ein Script umzuleiten.
Zum Beispiel:

set path c:/sys/
#Der Aufruf eines Scriptes (hier z.B. bestpos.cfg) muss dann nicht mehr mit **c:/sys/bestpos.cfg** erfolgen. Es reicht dann #einfach:
bestpos.cfg.

Default für 'set path' ist "c:/sys/".
Es können also Kommandos (Skripte), die in c:/sys/ liegen, ohne Voranstellen des Pfades gestartet werden. Die Einstellung wird im batteriegestützten RAM gespeichert. Sie steht nach Power on oder Reset zur Verfügung.

Befehle im Detail

set phone	Telefonbucheinträge für die SMS-Fernsteuerung anzeigen und ändern

Das eingebaute GSM-Modem ist für Datenübertragung (GSM und EDGE) und SMS (short message service) geeignet. Für die Anmeldung und Autorisierung ist eine SIM-Karte eines Providers erforderlich. Das Gerät besitzt dann eine Mobilfunknummer, unter der es per SMS erreichbar ist. Das Gerät ist über das Internet nicht direkt erreichbar. Eine Internetverbindung kann nur das Gerät selbst herstellen.

Um dennoch eine universelle Fernsteuerung zu ermöglichen, wurde eine Weiterleitung von empfangenen SMS an einen Shell-Interpreter implementiert. Die Weiterleitung an den Shell-Interpreter bzw. die Ausführung einer SMS (auch „Kommando-SMS“) ist von einer Autorisierung abhängig.

Die Autorisierung ist durch ein Telefonbuch realisiert, in dem alle Absendertelefonnummern (maximal 8) verzeichnet sind, von denen empfangene SMS als Shell-Kommandos interpretiert werden. Die Autorisierung basiert also auf der Aussage bzw. dem Standard der Netzbetreiber, dass die Absendertelefonnummer immer die des Absenders ist und sie nicht geändert werden kann, wie das im Gegensatz dazu bei der Absenderadresse einer E-Mail-Versendung möglich ist.

Empfangene SMS werden immer in einer Protokolldatei (**c:/sms/smsprot.txt**) mit einem Ausführungs- bzw. Nichtausführungsvermerk gespeichert (siehe Kommando ‚**sms**‘ S. 278). Neue SMS werden an der ‚**smsprot.txt**‘ angehangen.

Das Telefonbuch wird auch dafür verwendet, um beim SMS-Versand mit dem Kommando ‚**sms <name> ...**‘ die zu **<name>** zugehörige Telefonnummer zu ermitteln.

Das Telefonbuch wird ausfallsicher über Power off und Reset in einem batteriegestützten Parameterspeicher gespeichert (Lithiumzelle CR2032 – auswechselbar).

Parameter:

```
[<ix> [clr|<name> <no>] | support [on|off]]
```

```
set phone
```

Ohne weitere Parameter werden die gespeicherten Telefonnummern angezeigt.

Beispiel:

```
set phone

phone  0 Monteur1    004915212345678
phone  1 empty
phone  2 empty
phone  3 empty
phone  4 empty
phone  5 operator1   004917212345678
phone  6 empty
phone  7 empty
```

Befehle im Detail

set phone	Telefonbucheinträge für die SMS-Fernsteuerung anzeigen und ändern

set phone <ix> <name> <no>

Eine neue Telefonnummer in das Telefonbuch eintragen.

Der Index muss im Bereich 0...7 liegen. Der Name kann max. 15 Zeichen lang sein. Er darf keine Leerzeichen enthalten.

Die Telefonnummer darf 19 Zeichen lang sein. Sie darf keine Leerzeichen enthalten. Ein ‚+' vor der Telefonnummer ist erlaubt.

set phone <ix> clr

Mit diesem Unterkommando wird eine Telefonnummer aus dem Telefonbuch entfernt.

Der Index muss im Bereich 0...7.

set phone support [on|off]

Mit '**off**' wird die Möglichkeit des SMS-Supports durch die PPM GmbH abgeschaltet und mit '**on**' wird der SMS-Support durch die PPM GmbH wieder ermöglicht. Die Standardeinstellung ist ‚**on**'.

```
set phone support on

phone: factory support is enabled
```

Befehle im Detail

set pwr	Stromversorgung der Gerätefunktionseinheiten ein- und ausschalten

Mit dem Kommando ‚**set pwr ...**' werden die einzelnen Stromversorgungen innerhalb des 40xx geschaltet oder deren Schaltzustand angezeigt.

Anzeige des aktuellen Zustands:

```
set pwr

pwr: supply on
pwr: blth   off
pwr: gps    on gps1 /RST released, gps2 /RST released
pwr: sd     on
pwr: eth    off
pwr: gsm    off
pwr: io     on
pwr: 5p     on
```

`supply`	Hauptstromversorgung des Gerätes über die Buchse SUB-D25 (Vin+ Pin 24 und 25, GND Pin 12 und 13) am Backpanel.
`blth`	Stromversorgung für das Bluetooth-Modul.
`gps`	Stromversorgung für das bzw. die GPS-Boards, hier wird zusätzlich der Zustand der beiden Reset-Signale für das bzw. die GPS-Boards ausgegeben.
`sd`	Stromversorgung für die interne Micro-SD-Karte und den RS232-Treiber für coma.
`eth`	Stromversorgung für das Ethernet-Modul.
`gsm`	Stromversorgung für das GSM-Modem.
`io`	Stromversorgung für die RS232-Treiber von comc und comb, sowie für die beiden IO-Buchsen, CAN, PPS_out, Event_in, VarFreq. Die genannten Buchsen bzw. Signale werden über eine Trennisolation an den Anwenderbuchsen zur Verfügung gestellt.
`5p`	Stromversorgung für die GPS-Antennenversorgung bzw. das GPS-Board.

Die Stromversorgungen werden bei der Initialisierung der betreffenden Funktionsmodule des 40xx mit eingeschaltet. Es besteht in der Standardanwendung keine Notwendigkeit die Teilbetriebsspannungen separat einzuschalten.

Sie können zum Ausschalten eines Moduls verwendet werden, wenn das betreffende Modul keine explizites Deaktivierungskommando hat.

Hinweis: Wenn die Stromversorgung ‚**set pwr io off**' abgeschaltet wird, kann keine Kommunikation mehr über **comc** bzw. **comb** erfolgen. D. h. eine Einschaltung über ein Kommando ist dann nicht mehr möglich. Das gleiche gilt für ‚**set pwr sd off**'. Danach ist keine Kommunikation über **coma** möglich. Zusätzlich ist hierbei die Micro-SD-Karte ‚abgehängt'.

Parameter:

```
[supply|blth|gps|sd|eth|gsm [-pwm]|io|5p|app [on|off]] ;any combination of sources
```

Befehle im Detail

set pwr	Stromversorgung der Gerätefunktionseinheiten ein- und ausschalten

`supply|blth|gps|sd|eth|gsm [-pwm]|io|5p|app`

Benennung der Stromversorgungsinsel. Es können mehrere Stromversorgungen in einem Kommando aufgeführt werden.

Die Stromversorgung des GSM-Modems wird standardmäßig über eine PWM-Steuerung ‚langsam' hochgefahren, um die Stromspitze gering zu halten. Diese PWM-Einschaltsteuerung kann mit ‚**-pwm**' abgeschaltet werden.

Mit dem Schalter ‚app' ist ein Schalter, mit dem **alle** relevanten Stromversorgungen ein- oder ausgeschaltet werden können.

Mit ‚**set pwr app off**' wird eine minimale Stromaufnahme eingestellt. Das Gerät kann nur noch über **coma** kommunizieren. Der Controller hat weiterhin Zugriff auf die SD-Karte. Damit ergeben sich ca. 80 mA Stromaufnahme aus einer 12 V Versorgungsspannung.

Bei Nutzung aller Gerätekomponenten ergibt sich je nach Typ des eingebauten GPS-Boards eine Stromaufnahme von 200 bis 400 mA bei 12 V Versorgungsspannung. Bei 32 V Versorgungsspannung ergeben sich minimal ca. 50 mA und maximal 95 mA bis 150 mA.

Eine weitere Stromabsenkung ist nur noch über das Kommando ‚sleep ...' möglich. Während der Sleepphase wird kein und nur alle 30 Minuten für ca. 0,8 s Strom aufgenommen um interne Speicher (Super-Caps oder Gold-Caps) aufzuladen. Diese Art der ‚sleep'-Stromaufnahme ist der isolierten Stromversorgung geschuldet, bei der es über die Trennisolation hinweg keine Stromabsenkung gibt. Die mittlere Stromaufnahme aus einer 12 V Spannungsquelle beträgt ca. 110 µA.

Befehle im Detail

set rtc	Einstellung der RTC auf GPS-Zeit oder UTC

Die batteriegestützte RTC des Gerätes kann beliebig gestellt werden. Es ist jedoch üblich sie entweder auf die UTC (Universal Time Coordinated, auch Weltzeit oder Zulu-Zeit) oder auf die GPS-Zeit einzustellen. Die numerische Einstellung wird mit dem Kommando ‚**date ...**' vorgenommen. Die GPS-Zeit weicht von der UTC um einen Betrag ab.

Es gilt:

GPS-Zeit = UTC + <leap_seconds>

Dieser Betrag ist nicht konstant. Seit dem 1. Januar 2017 ist der Betrag 18 s.

Die RTC des Gerätes wird für die Zeitstempel bei Dateioperationen (FAT-Dateisystem) und andere Datumsanfragen (CRON-Manager) verwendet. Bei GPS-Empfang wird sie auf den Sekundentick der GPS-Zeit synchronisiert, d. h. die RTC wird getrimmt.

Parameter:

```
[utc | gps <leap_seconds>]        since 2017.1.1, gps = utc + 18 s
```

```
[utc | gps <leap_seconds>]
```

Ohne diesen Parameter wird die aktuelle Einstellung angezeigt.

```
set rtc

rtc: utc
```

oder z. B.

```
set rtc

rtc: gps, leap seconds 18
```

Wenn auf GPS-Zeit umgestellt werden soll, dann ist der Parameter **<leap_seconds>** Pflicht.

Befehle im Detail

set syspath	Suchpfad für Systemdateien anzeigen und einstellen

Mit diesem Kommando wird der eingestellte Systempfad angezeigt oder neu eingestellt.

Der Systempfad ist der Speicherort (in der Regel ein Verzeichnis), in dem die folgenden Systemdateien gesucht werden:

autoexec.sh
gps.cfg
firstfix.sh
ntrip.cfg
tcp.cfg
network.cfg
crontab.cfg

Damit ist die Möglichkeit gegeben, verschiedene Gerätekonfigurationen unabhängig voneinander im Gerät zu speichern. Es braucht nur der Systempfad neu eingestellt und anschließend das Gerät zurückgesetzt werden, um eine andere Konfiguration zu aktivieren.

Der Systempfad, der beim Öffnen dieser Dateien vor den Dateiname gesetzt wird. Er wird in der Systemvariable ‚**syspath**' gespeichert.

Die Systempfadvariable kann in selbstgeschriebenen Skripten oder direkt auf der Kommandozeile verwendet werden.

Beispiel:

Eingabe auf der Kommandozeile im Terminal:

```
set syspath c:/testconfig/
```

Kommandozeile in der autoexec.sh:

```
$(syspath)init-etc.sh
```

Nach dem Reboot und dem impliziten Aufruf der ‚**autoexec.sh**' ersetzt der Shellinterpreter den Aufruf **$(...)** der Systemvariable **syspath** mit dem aktuellen Wert. Zur Ausführung kommt dann

```
c:/testconfig/init-etc.sh
```

Die Standardeinstellung ist ‚**c:/sys/**'. Der Speicherort ist das FAT-Dateisystem auf der Mikro-SD-Karte.

Die Einstellung wird in einem batteriegestützten RAM ausfallsicher gespeichert. Sie bleibt auch nach einem Power Off oder Reset erhalten.

Parameter:

```
[<path>]
```

Befehle im Detail

set syspath	Suchpfad für Systemdateien anzeigen und einstellen

set syspath [<path>]

Ohne den optionalen Parameter wird die aktuelle Einstellung angezeigt:

```
set syspath

syspath: c:/sys/
```

Bei der Eingabe eines neuen Systempfades im FAT-Dateisystem ist auf das abschließende ‚/' zu achten.

Befehle im Detail

set usb	Zuordnung der externen USB-Schnittstelle für die interne Verwendung anzeigen und ändern

Die externe USB-Schnittstelle (Mini-USB-Buchse) kann mit der Mikro-SD-Karte auf dem Controllerboard verbunden werden.

Diese Verbindung mit der Mikro-SD-Karte wird im weiteren **USB-Stickmode** genannt.

Der USB-Stickmode ist die Standardeinstellung. Dabei verhält sich der 40xx wie ein USB-Stick (USB 2.0 High Speed). Dieser Modus ist auch ohne externe Stromversorgung verwendbar, d.h. es ist nur das USB-Kabel zum PC erforderlich. Dies ist aber nur dann der Fall, wenn der 40xx durch Beenden der Stromversorgung ausgeschaltet wurde. Ohne externe Stromversorgung bleiben die anderen Funktionselemente des Gerätes, außer die Konsole ‚coma' ausgeschaltet. Mit externer Stromversorgung arbeitet das Gerät weiter wie vor dem Anschluss mit der Ausnahme, dass keine Dateioperationen durch den Micro-Controller mit der SD-Karte möglich sind. Eventuell offene Dateien werden beim Anschluss eines USB-Kabels vor der Umschaltung zum PC geschlossen.

Nach dem Trennen geht der 40xx per Default für 3 Sekunden in den sleep Mode, d.h. er startet neu, d.h. die ‚**autoexec.sh**' wird erneut ausgeführt, da davon ausgegangen wird, dass die Konfiguration geändert wurde. Es muss also bei der Programmierung der ‚**autoexec.sh**' berücksichtigt werden, dass sie beim Abziehen des USB-Kabels wieder aufgerufen wird.

Hinweis:

Zum Beispiel sollten CRON-Jobs in der ‚**autoexec.sh**' und in den von ihr aufgerufenen Skripten aus diesem Grund als ‚benannte' Cron-Jobs programmiert werden:

```
cron put mein-job1 ...
```

oder

```
cron put ix <ix> ...
```

hierbei wird ein gleichnamiger (eventuell bereits bestehender) Cron-Job überschrieben, also ersetzt.

Mit einfach nur ‚**cron put ...**' wird der nächste freie Platz in der Cron-Tabelle gesucht und der Cron-Job wird dort eingetragen. Er wäre damit mehrfach vorhanden.

Parameter:

```
usb [{stick|off} [now] |stat [full]]
on-stick-disconnect [do-nothing|run-autoexec|sleep-3s]          sleep-3s is default after factory reset
```

```
set usb stick [now]
```

Beim nächsten Anstecken des USB-Kabels schließt das ppmOS alle offenen Dateien im FAT-Dateisystem und danach wird die Verbindung Mikro-SD-Karte zur USB-Buchse hergestellt. Der angeschlossene Host (PC, Tablett o. ä.) sieht einen Massenspeicher (USB-Stick).
Mit **‚set usb stick now'** wird die Umschaltung sofort ausgeführt auch wenn noch kein USB-Kabel angeschlossen ist.
Das **ppmOS** kann bis zum Trennen der Verbindung bzw. bis zur Umschaltung nach ‚**off**' keine Dateizugriffe zum FAT-Dateisystem machen. Nach der Umschaltung **ohne** Trennung des USB-Kabels muss das FAT-Dateisystem wieder gemountet werden mit ‚**df -i c:**', um Zugriff auf Dateien des FAT-Dateisystems zu bekommen.

Befehle im Detail

set usb	Zuordnung der externen USB-Schnittstelle für die interne Verwendung anzeigen und ändern

```
set usb off [now]
```

Beim **nächsten Anstecken** des USB-Kabels wird keine Umschaltung vorgenommen. Die Verbindung Controller Mikro-SD-Karte bleibt unberührt. Diese Einstellung hat dann Bedeutung, wenn verhindert werden muss, dass beim USB-Kabelanschluss eine Aktion ausgeführt wird.

Wird das kombinierte Kommando

```
set usb off; set usb off now
```

am Terminal eingegeben während eine PC-USB-Kabelverbindung besteht, dann wird die Mikro-SD-Karte **sofort** mit dem Controller verbunden. Es erfolgt jedoch kein (re-)mount des Dateisystems, da das USB-Kabel noch angesteckt ist.

Wenn erforderlich, kann das (re-)mount des Dateisystems mit ‚**df -i c:**' am Terminal explizit initiiert werden. Danach kann das **ppmOS** wieder auf das FAT-Dateisystem zugreifen, die ‚**autoexec.sh**' wird dabei **nicht** ausgeführt. Falls das GPS-Board initialisiert ist, wird die USB-Verbindung zur USB-RS232-Verbindung.

Mit ‚**set usb stick now**' wird die Mikro-SD-Karte unmittelbar getrennt und sie wird wieder mit dem PC verbunden. Dieser sieht dann wieder einen USB-Stick.

Mit dieser Methode kann das Gerät sehr effektiv, ohne das USB-Kabel abziehen zu müssen, programmiert und getestet werden.

Hierbei muss jedoch die externe Stromversorgung ständig verbunden sein.

```
[now]
```

Mit diesem optionalen Parameter wird die Umschaltung sofort ausgeführt.

Standardmäßig wird der Parameter nicht verwendet, da die vorgenommene Einstellung erst beim Anschluss des USB-Kabels wirksam werden soll.

Befehle im Detail

set usb	Zuordnung der externen USB-Schnittstelle für die interne Verwendung anzeigen und ändern

```
set usb stat [full]
```

Der Status des USB-Umschalters wird angezeigt.

```
set usb stat
```

```
USB: SD-card present, mode actual: SD-card, on next connect: Stick
```

Es wird der Status der SD-Karte ausgegeben (vorhanden, nicht vorhanden), dann folgt die aktuelle Einstellung des USB-Umschalters (jetzt ist die SD-Karte mit dem Controller verbunden). Am Ende der Zeile wird der Verbindungsmode, der beim nächsten Verbinden des USB-Kabels hergestellt wird angezeigt (der USB-Stickmode)..

```
set usb stat full
```

```
USB: SD-card present, mode actual: SD-card, on next connect: SD-card
USB: mode  0
USB: kvbus 0
USB: /busy 0
USB: /err  1
USB: /rst  1
USB: 5V    1
sd : det   0
```

Nach der Statusausgabe werden die Zustände der digitalen Steuerleitungen des USB-Umschalters, der USB-Vbus-Spannung (‚**5V 1**' → USB-Kabel steckt) und der SD-Kartenerkennung (‚**sd : det 0**' → invertierte Logik, Karte vorhanden) ausgegeben.

Befehle im Detail

set vcom	USB-Schnittstelle des Controllers mit der USB-Buchse verbinden

Die USB-Buchse am Frontpanel kann standardmäßig mit der internen Micro-SD-Karte (USB-Stickmode) verbunden werden.

Als zusätzliche Möglichkeit kann der Controller des Gerätes einen virtuellen COM-Port über USB bereitstellen. Die Nutzung ist mit der Einschränkung verbunden, dass das GPS-Board deaktiviert wird. **Für die Nutzung ist ein Hardware-Zusatz erforderlich, der standardmäßig nicht vorhanden ist.**

Dieser zusätzliche Schnittstelle hat den Vorteil, dass kein RS232-USB-Schnittstellenwandler nötig ist beim Anschluss des 40xx an einen PC ohne serielle Schnittstelle.

Ein weiterer Vorteil ist die wesentlich höhere Datenübertragungsgeschwindigkeit (bis zu 1 Mbyte/s).

Das Kommando muss in Kombination mit anderen Kommandos verwendet werden. Diese werden zweckmäßiger Weise in dem Skript ‚**c:/sys/vcom-start**' zusammengefasst:

```
set usb rs232
set usb rs232 now
set pwr gps on
set io rst-gps1 lo
set io rst-gps2 lo
set vcom init
```

Die Deaktivierung kann durchgeführt werden mit dem Skript ‚**c:/sys/vcom-stop**' mit dem Inhalt:

```
set vcom deinit
set usb off now
gps init&
```

Hier bleibt der USB-Mode auf ‚rs232' stehen. Das verhindert, falls das USB-Kabel angeschlossen bleibt, die Trennung des Controllers von der SD-Karte und den Übergang in den USB-Stickmode.

Parameter:

```
[{init [-u]|reinit} [<rx_len>]|deinit|stat [clr]
```

```
set vcom init [-u]
```

Der VCOM-Treiber wird initialisiert. Es wird der Task ‚**vcom1**' gestartet und das Ein- Ausgabefile ‚**vcom1**' ist verfügbar. Die Eingabedaten gelangen zu einem Shell-Interpreter. Dieser kann mit ‚**set intp vcom1 off|on**' ab- oder angeschaltet werden.

Wurde das Kommando ‚**set vcom init**' bereits ausgeführt, dann ergibt eine erneute Ausführung die Meldung ‚**USB: vcom1 alreday initialized**'. Mit Angabe des optionalen Schalters ‚-u' (unconditional) wird die Initialisierung ohne Bedingung wiederholt.

Hinweis:

Der virtuelle COM-Port über USB (**vcom1**) steht nur zur Verfügung, wenn das GPS-Board nicht benötigt wird und ein entsprechender Patch auf dem GPS-Adapterboard durchgeführt wurde.

Befehle im Detail

set vcom	USB-Schnittstelle des Controllers mit der USB-Buchse verbinden

```
set vcom reinit [<rx_len>]
```

Der VCOM-Treiber kann mit diesem Unterkommando reinitialisiert werden.

Wird dabei **<rx_len>** angegeben, dann wird der alte Empfangspuffer geschlossen und ein neuer mit der angegebenen Länge alloziert. Standardmäßig werden 2048 Byte Empfangspuffer alloziert. Für große Datenmengen, die zum Controller schnell übertragen werden müssen, kann der Empfangspuffer auf die Größe des verbleibenden Arbeitsspeichers (Kommando ‚**stat**') vergrößert werden. Ein gewisse Speichermenge sollte jedoch frei bleiben (min. ca. 10 kByte).

```
set vcom deinit
```

Der VCOM-Treiber kann mit diesem Unterkommando entfernt werden. Der von dem Empfangs- und Sendepuffer belegte Speicher wird freigegeben. Der Task ‚**vcom1**' bleibt erhalten, er verbraucht keine weitere Rechenzeit. Wenn erforderlich, dann kann er mit ‚**tsk kill vcom1**' entfernt werden.

```
set vcom stat [clr]
```

Den Status des VCOM-Treibers anzeigen und die Bytezähler für Empfang, Senden, Empfangs- und Sendeüberlauf anzeigen und löschen (‚**clr**').

Beispiel:

```
set vcom stat
```

```
vcom1: HS rxlvl 0 (0 89999) rxd 102400097 txd 40002855 ovfl rx 0 tx 0
```

Die nichtverarbeitete Datenmenge (**rxlvl**) ist 0, der Empfangspuffer ist 90000 groß. Es wurden **rxd** Byte empfangen und **txd** Byte gesendet. Es gab keinen Überlauf.

Wird der optionale Schalter ‚**clr**' angefügt, dann wird der Status zum Abfragezeitpunkt ausgegeben und anschließend werden die Zähler auf 0 gesetzt.

Befehle im Detail

sleep	Gerät in den Schlafzustand versetzen

Mit diesem Kommando kann die Firmware (das **ppmOS**) für eine Wartezeit gestoppt werden. Dabei geht die Hardware in einen stromsparenden Modus, d. h. die Stromaufnahme ist erheblich reduziert. Es wird keine Software mehr abgearbeitet. Die Wartezeit kann als Dauer in Sekunden angegeben oder als Aufwachdatum vorgegeben werden.

Während das Gerät im Schlafzustand ist, kann es durch Drücken der ON-/Off-Taste eingeschaltet werden. Damit wird der Schlafzustand abgebrochen.

Das Gerät verfügt über eine batteriegestützte Echtzeituhr (RTC). Mit Hilfe dieser RTC werden beide Schlafmodi realisiert. Die interne Stützbatterie (Typ CR2032, auswechselbar) hat eine geschätzte Laufzeit von ca. 3-5 Jahren.

Da die Stromversorgung des Gerätes isoliert, also über eine Trennisolation realisiert wurde, ergibt sich eine Besonderheit bei der Stromaufnahme während des Schlafzustandes. Der Strombedarf während des Schafzustandes wird aus internen Super-Caps (Kondensatoren mit sehr hoher Kapazität) gedeckt. D.h. zunächst fließt kein Strom aus der Versorgungsquelle in das Gerät, aber alle 30 Minuten schaltet sich das Geräte für ca. 0,8 Sekunden ein, um die Super-Caps nachzuladen. In dieser kurzen Zeit fließen ca. 250 mA (bei 12 V Versorgungsspannung). Über die gesamte ‚Schlafzeit' gerechnet, ergibt sich ein mittlerer Strom von ca. 110 µA aus einer 12 V Versorgungsquelle. Ein Spannungsquelle, die diesen Anforderungen genügt, ist also in der Schlafzeit erforderlich. D. h. auch, dass nach Einnahme des Schlafzustandes die Stromversorgung zum Gerät für weniger als eine halbe Stunde vollständig getrennt werden könnte.

Nach Ablauf der Wartezeit bzw. bei Erreichen des Aufwachzeitpunktes schaltet sich das Gerät mit einem Reset ein und die Firmware startet den Bootvorgang, die **$(syspath)autoexec.sh** wird ausgeführt.

Parameter:

```
[[-nc] {<sleep_sec> | -d <yy.mm.dd hh:mm:ss>}]   // -nc sleep not closing any open file
```

-nc

Mit diesem optionalen Schalter wird angezeigt, dass bei der Einnahme des Schlafzustandes eventuell geöffnete Dateien nicht geschlossen werden sollen. Dieser Schalter sollte nur dann verwendet werden, wenn dadurch kein Datenverlust eintreten kann oder wenn mit diesem Befehl eine Art ‚emergency' Sleep eingenommen werden soll.

<sleep_sec>

Schlafzeit in ganzen Sekunden.

Befehle im Detail

sleep	Gerät in den Schlafzustand versetzen

```
-d <yy.mm.dd hh:mm:ss>
```

Einstellung des Aufwachzeitpunktes mit der Angabe von Jahr, Monat, Tag, Stunde, Minute und Sekunde. Die Angabe der Stunde erfolgt im Bereich 0 ... 23.

Beispiel:

```
sleep -d 17.5.1 0:0:0

closing all files in c:
closing files finished
17.04.10 08:18:26.123 s going to sleep for 1784495 s now
```

Befehle im Detail

sms	SMS-Versand über das GSM-Modem

Mit dem sms-Kommando kann über den Short-Message-Service (SMS), der über das GSM zur Verfügung gestellt wird, Kurznachrichten ausgetauscht werden. Das Gerät ist in der Lage, SMS zu verschicken und zu empfangen.

Das Kommando ist erst nach dem Kommando ‚**gsm init**' verfügbar. Mit ‚**gsm init**' wird der Cron-Job

```
CRON: job[ 0] sms-ls      :   *+30/7 * * * * * * @sms -ls
```

gestartet. Das Modem empfängt und speichert unabhängig vom **ppmOS** bis zu 25 SMS. Mit Hilfe des Cron-Jobs wird 30 s nach der Initialisierung des GSM-Modems alle 7 s das Kommando ‚**sms -ls**' ausgeführt. Es fragt das GSM-Modem auf empfangene SMS ab. Wenn das GSM-Modem eine andere Datenübertragung (ftp, tcp) durchführt, dann erfolgt in dieser Zeit keine SMS-Abfrage.

Empfangene SMS werden vom Shell-Interpreter von **coma** ausgeführt, aber nur dann, wenn die Absendertelefonnummer im **ppmOS**-internen Telefonbuch gespeichert ist. Dadurch ist eine Fernbedienung durch unautorisierte Absendertelefonnummern ausgeschlossen. Das Telefonbuch wird mit dem Kommando ‚**set phone ...**' verwaltet.

Die SMS wird ausgeführt, indem die SMS zeilenweise per Systemmessage an den **coma**-Task geschickt wird, d. h. die Kommandos werden nacheinander mit der jeweiligen Laufzeit ausgeführt. Es ist jedoch möglich das Kommando durch Anhängen von ‚**&**' als Task zu starten. Es sind alle Syntaxelemente wie auf der normalen Kommandozeile möglich.

Über alle empfangenen SMS wird ein Protokoll geführt. Die Protokolldatei ‚**smsprot.txt**' wird im Pfad **c:/sms/** gespeichert. Sie wird permanent fortgeschrieben.

Wird eine SMS ausgeführt, dann wird in der Protokolldatei ein Datensatz angelegt, der mit einer Trennlinie beginnt und mit dem OK der Modemantwort auf das Auslesen endet:

Beispiel:

```
----------------------
16.12.01 09:24:07
^SLMS: "ME",25,1
becomes executed:
+CMGR: "REC UNREAD","+491729805693",,"16/12/01,10:23:55+04"
ip server start ipsb:12000

OK
```

Darin ist das Datum und die Uhrzeit des Empfangs, die Behandlungsinformation ‚`becomes executed:`', die Absendertelefonnummer und der Inhalt der SMS ‚`ip server start ipsb:12000`' protokolliert.

Als Behandlungsinformation kann erscheinen:

```
not executed (not in phone book)
not executed (error in reception)
not executed (disabled)
```

Die Ausführung kann mit ‚**set intp sms [on|off]**' angezeigt, disabled → ‚`not executed (disabled)`' oder enabled → ‚`becomes executed`' werden.

Siehe nächste Seite.

Befehle im Detail

sms	SMS-Versand über das GSM-Modem

Die zweite Verwendung des Kommandos ist das Versenden von SMS. Der Inhalt kann ein auf der Kommandozeile angegebener Text oder eine Datei sein, von der ein max. 160 Byte langer Abschnitt gesendet wird. Zum Übertragen einer längeren Datei kann diese in mehreren SMS verschickt werden. Dazu ist bei jeder SMS der Offset anzugeben, der in 160 Byte-Schritten erhöht werden sollte.

Zu beachten ist, dass nur ASCII-Dateiinhalte (7-Bit Werte) gesendet werden können und dabei ist noch auf das jeweils eingestellte GSM-Alphabet zu achten. Es kommt hier teilweise zum Umkodieren von Zeichen.

Das erfolgreiche Versenden wird mit ‚**SMS: transmitted**' oder mit z. B. ‚**SMS: tx failed, error 24001**' quittiert.

Das Kommando hat einen Rückkehrcode, der in einer bedingten Kommandoausführung verwendet werden kann.

Die Bedeutung der Codes sind:

0	Die Aktion wurde fehlerfrei ausgeführt.
	Beim Einstellen der SIM-PIN bzw. bei der Abfrage ist ein Fehler aufgetreten.
10540	- Modem ist ausgeschaltet oder nicht vorhanden.
10550	- Abfrage, ob SIM-PIN schon eingegeben wurde, schlägt fehl.
10560	- Auslesen der SIM-PIN aus ‚**c:/sys/upload.cfg**' und Übertragung zum Modem schlägt fehl, die SIM-PIN ist falsch oder das Modem ist nach einem Power On noch nicht bereit bzw. nocht nicht im Netz eingebucht.
+1 oder +2	1 → Modem meldet ‚**ERROR**' 2 → Timeout, Modem antwortet nicht
12200	Das zu übertragende File konnte nicht geöffnet werden.
12300	Der Dateizugriffszeiger des zu übertragenden Files konnte nicht auf den angegebenen Wert (Offset) eingestellt werden. Das File ist evtl. zu kurz.
20000	Das GSM-Modem ist gerade belegt (SMS auslesen, FTP-Transfer oder TCP-Verbindung).
21001 21002	Fehler bei der Modemeinstellung (1 → Modem meldet ‚**ERROR**', 2 → Timeout (2 s), Modem antwortet nicht)
22000	Der anstelle der Telefonnummer angegebene Name wurde nicht im Telefonbuch gefunden.
23000	Modem antwortet nicht mit der Eingabebereitschaft (Prompt ‚>').
24000	Länge des zu übertragenden File ab dem Anfang bzw. dem angegebenen Offset ist 0.
24100	Der SMS-Text <text> beginnt mit " oder '. Es fehlt das schließende " bzw. ', da angenommen wird, dass der Text in "..." bzw. in '...' eingeschlossen sein soll.
25001 25002	Fehler beim Abschluss der SMS-Eingabe (Ctrl Z). (1 → Modem meldet ‚**ERROR**', 2 → Timeout (2 s), Modem antwortet nicht)

Befehle im Detail

sms	SMS-Versand über das GSM-Modem
Parameter: `<nr> {<text>| POS | SYSSTAT | {@|!}<file> [<offset>]} | -l[s]`	
`<nr>` Telefonnummer des Adressaten. Diese muss nach den Regeln des jeweiligen Dienstanbieters notiert sein und sie darf keine Leerzeichen enthalten. Anstelle der Telefonnummer kann ein Name aus dem im **ppmOS** gespeicherten Telefonbuch angegeben werden. Das Telefonbuch wird mit dem Kommando ‚**set phone ...**' verwaltet. Es wird in einem batteriegestützten Speicher gespeichert.	
`sms <nr> <text>` Der auf der Kommandozeile angegebene Text bis zum Zeilenende wird als SMS an die angegebene Nummer verschickt. Wenn im zu sendenden Text die Zeichen ‚**;**', ‚**&&**' oder ‚**||**' oder führende Leerzeichen vorkommen, dann muss der Text in `"..."` bzw. in `'...'` geklammert werden.	
`sms <nr> POS` Beginnt der Text mit dem Schlüsselwort ‚**POS**', dann werden die aktuellen GPS-Datensätze GPGGA und GPRMC versendet.	

Befehle im Detail

sms	SMS-Versand über das GSM-Modem

sms <nr> SYSSTAT

Beginnt der Text mit dem Wort ‚**SYSSTAT**', dann wird der Systemstatus übermittelt.

Beispiel:

```
9F6,DHD13402482,L6X010201RN0000,1.22,utc,17.04.11,10:01:43,1480,1,13,,,,12.085,PON,0x00,0,0,84,267,480,834,480,262,2,3,5,0,1087
```

Darin bedeuten:

9F6

3-Buchstabenkennung des Gerätes. Sie wird aus der Seriennummer des GPS-Boards abgeleitet.

DHD13402482

Seriennummer des GPS-Boards.

L6X010201RN0000

Optionscode des GPS-Boards. Hieraus werden die Eigenschaften des GPS-Boards ersichtlich.

1.22

Versionsnummer der Firmware des Gerätes.

utc

Die Systemzeit ist auf UTC eingestellt (alternativ gps – GPS-Zeit).

17.04.11

Datum der Erzeugung des SYSSTAT-Datensatzes.

10:01:43

Uhrzeit der Erzeugung des SYSSTAT-Datensatzes.

1480

Zeit in Sekunden seit dem letzten Reset bzw. Power on (uptime).

1

Quality Indicator. Zahl ist aus dem $GPGGA-Datensatz

13

Anzahl der genutzten Satelliten. Zahl ist aus dem $GPGGA-Datensatz

,,,

Drei leere Datenfelder aus dem Power-Status. Derzeit nicht genutzt. Die Kommas dienen als Platzhalter. Ein Datenfeld wird durch ein vorangestelltes Komma eingeleitet.

Siehe nächste Seite.

Befehle im Detail

sms	SMS-Versand über das GSM-Modem

“`sms <nr> SYSSTAT`“ f.:

```
12.085
```

Versorgungsspannung des Gerätes in Volt.

```
PON,0x00,0,0,84,267,480,834,480,262,2,3,1,0,1087
```

Reset-Ursache (PON – power on), gefolgt vom dem Registerwert RCC_CSR (0x00, obere 8 Bit) und den Zählern der einzelnen Reset-Ursachen:

`0`	LPW – low power reset
`0`	WDG – window watchdog reset
`84`	IWD – independent watchdog
`267`	SWR – software reset
`480`	POR – power reset
`834`	EXR – extern reset (
`480`	BOR – brown out reset (Unterspannung)
`262`	RST – reset (Kommando ‚**Reset**‘, SWR und EXR werden inkrementiert)
`2`	JMP – jump reset (Kommando ‚**Reset -j**‘, dies ist kein echter Reset, sondern ein Sprung zur Startadresse)
`3`	WDT – wdt reset (Kommando ‚**Reset -w**‘, IWD und EXR werden inkrementiert)
`1`	CSH - Reset nach Absturz nach der Bootphase
`0`	CSB – Reset nach Absturz in der Bootphase
`1087`	PON – power on (Einschaltung der Betriebsspannung)

```
sms <nr> {@|!}<file> [<offset> [<len>]]
```

Von dem auf der Kommandozeile angegebene File werden max. 160 Zeichen oder max. die Anzahl **<len>** an Zeichen als SMS an die angegebene Nummer verschickt. Optional kann ein Fileoffset angegeben werden, ab dem 160 Zeichen ausgelesen und gesendet werden.

Es ist zu beachten, dass nur 7-Bit Daten versendet werden und durch das jeweilige GSM-Alphabet einige ASCII-Zeichen umkodiert werden.

Befehle im Detail

sms	SMS-Versand über das GSM-Modem

```
sms -l
sms -ls
```

Mit diesem Kommando wird das GSM-Modem auf empfangene SMS abgefragt. Hat das GSM-Modem SMS empfangen, dann werden diese einzeln ausgelesen und dem Shell-Interpreter von **coma** über eine Messagepipe übermittelt. Das Interpretieren kann mit ‚**set intp sms off**' abgeschaltet und mit ‚**set intp sms on**' wieder angeschaltet werden. Empfangene SMS werden im Logfile ‚**c:/sms/smsprot.txt**' mit Datum und Uhrzeit gespeichert.

Der Schalter ‚**-ls**' unterdrückt eventuelle Statusmeldungen während der Abfrage (silent).

Vor der Ausführung wird ‚**sms batch starts**' angezeigt und es wird 1 s vor der Ausführung gewartet, damit das Modem wieder frei ist für einen eventuellen SMS-Versand oder eine GSM-FTP-Verbindung, die Bestandteil der Kommandosequenz in der SMS sein können.

Befehle im Detail

SNAP	Erzeugung von synchronen Pulsen definierter Länge und Empfang von Pulsen

Mit dem Kommando **‚SNAP'** kann die Erzeugung von synchronen Pulsen definierter Länge sowie der Empfang von Pulsen konfiguriert werden, also z.B. zum Triggern und/oder empfangen von Pulsen von (Luftbild-) Kameras oder anderen Sensoren.

‚SNAP' steuert Ausgabe-Events und zählt Eingabe-Events, wobei die einzelnen EventIn-Signale separat gezählt werden.

Es kann - unter anderem - ein Event unmaskiert werden, sodass dieses permanent gezählt wird und ein RECO-Log die Aufrufsyntax schreibt.

Hinweis:

Da es sich um ein sehr spezifisches Kommando handelt, werden die Parameter des Kommandos hier nur gelistet (s. unten). Falls Sie nähere Informationen oder Support zu diesem Kommando benötigen, so senden Sie bitte eine Email mit dem Betreff „40xx Kommando SNAP" an support@ppmgmbh.com

Parameter:

```
   [<ev_list>]                                 make a snap, <ev_list> can be 'all' or
                                              one of (1,2,3,4) in any order # or #,# or #,#,# or #,#,#,#
 | [<ev_list>] init {highactive|lowactive}   [<puls_time_ms> [<response_timeout_ms>]
                                                      default puls time       : 10 ms
                                                      default response timeout: 400 ms
 | [<ev_list>] deactivate                    don't generate pulses when only 'SNAP' arises
 | [<ev_list>] activate                      generate pulses with preinitialized parameters
 | [<ev_list>] active-tm     [<time_ms>]     set active time on given event lines
 | [<ev_list>] response-tmot [<tmot_ms>]     set response timeout, started after last pulse
 | [<ev_list>] accept-irq [always|windowed] set behavior of accepting camera events
                                            always   - at least one 'SNAP' is required, thereafter always
                                            windowed - accept it only while timout window after 'SNAP'
 | [<ev_list>] open                          set shutter signal to active level
 | [<ev_list>] close                         set shutter signal to inactive level
 | [<ev_list>] reset-counter                 set snap and irq counter to 0
 | reco-file   [<file>|close]                write RECO-sentences to <file>
 | reco-cycle  [off|on|<divider>]            control output of cyclic RECO sentence
 | echo        [on|off]                      print infos about whats going on
 | stat        [-v]                          print status information, -v be more verbose
```

Befehle im Detail

stat	Geräte-, Speicher- und Schnittstellenstatus ausgeben

Mit dem Kommando ‚**stat**' kann der Zustand des Gerätes ermittelt werden. Mit dem Unterkommando ‚**-f**', ‚**-wdt**' und ‚**heap malloc**' werden auch Einstellungen vorgenommen bzw. Aktionen ausgelöst. Bei den anderen Unterkommandos werden nur Ausgaben gemacht.

Parameter:

```
  -sys                                      system status
| -sys2                                     gps board status
| -sys2-nmea                                gps board status as PPMS3-NMEA sentence
| -e                                        last command return code
| -f                                        flush open files
| -cz [-nf] {<ix>|-1}             close zombie file <ix> or all files -> -1, -nf no flush before
| -i                                        NVIC infos
| -scb                                      system control block infos
| -ic [-d|-D] [-nz | <irq> [...]]          print interrupt counter, all or only for <irq>
| -ic  -l [<irq> [...]]                     print address of ISR
| <task_name>
| heap [malloc <size>|free <adr>]
| -wdt [init|set [[std|max] [<pr> <rld>]]]  pr 0...6, rld 0...4095, T=(rld<<(pr+2))/32kHz
```

```
stat
```

Es werden Informationen zum vorhandenen und belegten Speicher ausgegeben (**heap**). Es folgen Informationen zur Schnittstelle des benutzten Shell-Interpreters. In der Zeile ‚**RR: ...**' sind die Reset-Ursache und die persistenten Zähler der verschiedenen Ursachen aufgelistet. Danach folgt die Tabelle der geöffneten Files im FAT-Dateisystem.

```
heap: 0x20012000 free 124928 min 1888, txt 0x81710e8, dat 2524, bss 27740, ctor 60
coma: com3 rxlvl 1 (1999) ovfl rx 0 tx 0 lines 20 0 unknown 0  intp_ovfl 0 0
RR: RST (0x14) counts [LPW 0, WDG 0, IWD 91 | SWR 314, POR 517, EXR 925, BOR 517] RST 304, JMP 3, WDT 4, CSH 10, CSB 0, PON 1182
open files: 2 max 12
 0 C:51.2MB       r   51200000    7357952  72 00 0 16.07.20 21:01:32  16.07.20 21:01:29.090 '51.2mb'
 1 C:TESTFI~1.12M w      13312      13312  77 00 1 17.04.13 12:29:42  17.04.13 12:29:42.500 'testfile-5.12 MB'
```

heap:
Der frei verwendbare Speicher für temporäre Alloziierungen beginnt bei der Adresse **0x20012000** es sind zum Zeitpunkt der Abfrage **124928** Byte frei. Bei aufwendigen Konfigurationen kann dieser freie Speicher klein werden (wenige Kilobyte), was eine kritische Speichersituation sein kann. Hierauf ist bei der Konfiguration zu achten. Der freie Speicher sollte in der Regel nicht kleiner als 10 kByte sein.
Die Informationen hinter ‚**txt**', ‚**dat**', ‚**bss**' und ‚**ctor**' beziehen sich auf das laufende Programm. Sie sind für den Betrieb des Gerätes

coma:
Name des Shelltasks, der dieses Kommando ausführt (kann auch **ipsa**..., **ipsb**..., **blth**, usw. sein). Dahinter Informationen zum Eingangspufferspeicher (**rxlvl**). Hinter ‚**lines**' ist die Anzahl der bereits ausgeführten Shell-Kommandos (20) angegeben. Es folgt ein Zähler für unbekannte Kommandos und Überläufe des Interpreterpuffers für zu lange Kommandozeilen.

RR:
Die Reset-Ursachen (reset reasons)

Befehle im Detail

stat	Geräte-, Speicher- und Schnittstellenstatus ausgeben

```
stat -sys
```

Hier wird der Systemstatus in einer verdichteten Form ausgegeben.

Dieser Datensatz kann auch mit dem Remote-SMS-Kommando ‚**sms {<nr>|<name>} SYSSTAT**' angefordert und übermittelt werden.

Möchte man diesen Systemstatus aus der Firmware heraus, also nicht per Remote-SMS, als SMS versenden, dann empfiehlt sich ‚**stat -sys >pipe1; sms {<nr>|<name>} !pipe1**'.

Beispiel:

```
sys: DLF,DMMU18260115L,OM7MR0800RN0000,1.46,utc,21.08.20,09:19:05,2171, 465563.316704,7923,4,37,,,,11.588,POR,0x07,0,0,446,624,3574,4545,3574,227,0,0,445,289,2398
```

Darin bedeuten:

DLF

3-Buchstabenkennung des Gerätes. Sie wird aus der Seriennummer des GPS-Boards abgeleitet.

DMMU18260115L
Seriennummer des GPS-Boards.

OM7MR0800RN0000
Optionscode des GPS-Boards. Hieraus werden die Eigenschaften des GPS-Boards ersichtlich.

1.46

Versionsnummer der Firmware des Gerätes.

utc

Die Systemzeit ist auf UTC eingestellt (alternativ gps – GPS-Zeit).

21.08.20

Datum der Erzeugung des SYSSTAT-Datensatzes.

09:19:05

Uhrzeit der Erzeugung des SYSSTAT-Datensatzes.

2171

GPS-Woche

465563.316704
GPS-Wochensekunden

7923

Zeit in Sekunden seit dem letzten Reset bzw. Power on (uptime).

4

Quality Indicator. Zahl ist aus dem $GPGGA-Datensatz (s. nächste Seite)

Befehle im Detail

stat	Geräte-, Speicher- und Schnittstellenstatus ausgeben

"`stat -sys`" f.:

`37`

Anzahl der genutzten Satelliten. Zahl ist aus dem $GPGGA-Datensatz

`,,,,`

Drei leere Datenfelder aus dem Power-Status. Derzeit nicht genutzt. Die Kommas dienen als Platzhalter. Ein Datenfeld wird durch ein vorangestelltes Komma eingeleitet.

`11.588`

Versorgungsspannung des Gerätes in Volt.

`POR,0x07,0,0,446,624,3574,4545,3574,227,0,0,445,289,2398`

Reset-Ursache (POR – power reset), gefolgt vom dem Registerwert RCC_CSR (0x07, obere 8 Bit) und den Zählern der einzelnen Reset-Ursachen:

`0`	LPW – low power reset
`0`	WDG – window watchdog reset
`446`	IWD – independent watchdog
`624`	SWR – software reset
`3574`	POR – power reset
`4545`	EXR – extern reset (wird auch beim Softwarereset hochgezählt)
`3574`	BOR – brown out reset (Unterspannung)
`227`	RST – reset (Kommando ‚**Reset**', SWR und EXR werden inkrementiert)
`0`	JMP – jump reset (Kommando ‚**Reset -j**', dies ist kein echter Reset, sondern ein Sprung zur Startadresse)
`0`	WDT – wdt reset (Kommando ‚**Reset -w**', IWD und EXR werden inkrementiert)
`445`	CSH – Absturz nach der Bootphase (crash)
`289`	CSB – Absturz in der Bootphase (crash)
`2389`	PON – power on (Einschaltung der Betriebsspannung)

Befehle im Detail

stat	Geräte-, Speicher- und Schnittstellenstatus ausgeben

stat -sys2

Mit diesem Unterkommando können Informationen zum ppmOS sowie zum GPS-Board abgefragt werden.

Beispiel Rückgabe:

```
sys2: ppm40xx,1.46,utc,21.06.02,10:23:34,2160, 296632.512908,5798,GSNLNNTMN,DMMU18260115L,OEM7720-1.00,OM7MR0800RN0000
```

Darin bedeuten:

ppm40xx	Gerätetyp
1.46	Version des installierten ppmOS
utc	Verwendetes Zeitsystem
21.06.02	Datum
10:23:34	Uhrzeit (hier: UTC)
2160	GPS Woche
296632.512908	GPS Wochensekunden
5798	Uptime
GSNLNNTMN	Auf internem NovAtel-Receiver aktives Firmware-Model (-Optionen)
DMMU18260115L	Seriennummer des NovAtel-Receiver-Boards
OEM7720-1.00	NovAtel Receiver-Board-Typ
OM7MR0800RN0000	Auf internem NovAtel-Receiver installierte Firmware-Version

stat -sys2-nmea

Mit diesem Unterkommando können Informationen zum ppmOS sowie zum GPS-Board abgefragt werden. Die Rückgabe erfolgt in Form des PPMS3-NMEA Datensatzes und enthält die selben Informationen wie die Rückgabe auf ‚stat –sys2' aber mit zusätzlicher Checksum.

Beispiel Rückgabe:

```
$PPMS3,ppm40xx,1.46,utc,21.06.02,12:38:57,2160, 304755.009240,36,GSNLNNTMN,DMMU18260115L,OEM7720-1.00,OM7MR0800RN0000*6A
```

stat -e

Gibt den Rückkehrwert des letzten Kommandos aus. Hierbei bedeutet 0 in den meisten Fällen fehlerfreie Ausgabe und ein Wert ungleich 0 zeigt einen Fehler an. Bei manchen Kommandos ist der Rückkehrwert auch eine normale Zahleninformation (z. B. ‚**set io in1.1**' oder ‚**pob pb.3**'), die zu einer bedingten Kommandoausführung verwendet wird.

Befehle im Detail

stat	Geräte-, Speicher- und Schnittstellenstatus ausgeben

```
stat -f
```

Falls Dateien im FAT-Dateisystem zum Schreiben geöffnet sind, dann werden gepufferte Inhalte auf die Micro-SD-Karte geschrieben und die Verzeichniseinträge werden aktualisiert. Normaler Weise geschieht das erst beim Schließen der Datei. Eine Ausnahme ist die GPS-Log-Datei (‚**gps log on ...**'), bei der der GPS-Task das periodische ‚Flushen' übernimmt (siehe Kommando ‚**gps flush ...**' S. 155).

Bei Anwendungen, bei denen es darauf ankommt, mit hoher Sicherheit weitere Daten zu speichern, kann dieses Kommando zu bestimmten Zeitpunkten explizit ausgeführt werden (z. B. als Cron-Job).

```
-cz [-nf] {<ix>|-1}
```

Mit diesem Kommando kann entweder ein bestimmtes (**<ix>**) oder alle (**„-1"**) „Zombie Files" gelöscht werden.

Wird der Schalter „**-nf**" angegeben, so werden die zu schließenden Files vor dem Schließen nicht geflusht.

```
stat -i
```

Es werden Informationen zu den Interrupt-Einstellungen des Microcontrollers **STM32F427xx** ausgegeben.

Diese Informationen haben für den Normalbetrieb keine Bedeutung.

Um Informationen zur Interrupt-Häufigkeit bestimmter Interrupt-Quellen zu bekommen, kann das Kommando ‚**stat -ic ...**' (interrupt counts) verwendet werden.

```
SCnSCB->ICTR    ro  0xe000e004  96  max. interrupts of this Cortex M4 implementation
SCnSCB->ACTLR   rw  0xe000e008  0x00000000                  aux. control reg.
DBGMCU->IDCODE  ro  0xe0042000  0x10036419                  bit 31..16 RevID, bit 11..0 devID
NVIC->ISER[3]   rw  0xe000e100  00040542 20c201d0 000c0080  set enable reg.
NVIC->ICER[3]   rw  0xe000e180  00040542 20c201d0 000c0080  clear enable reg.
NVIC->ISPR[3]   rw  0xe000e200  00000000 00000000 00020000  set pending reg.
NVIC->ICPR[3]   rw  0xe000e280  00000000 00000000 00020000  clear pending reg.
NVIC->IABR[3]   ro  0xe000e300  00000000 00000000 00000000  active bit reg.
NVIC->IP 4 bit  rw  0xe000e400  00 d0 00 00 00 00 60 00 b0 00 40 00 00 00 00 00
NVIC->IP 4 bit  rw  0xe000e410  00 00 c0 00 00 00 00 00 00 00 00 00 00 00 00 00
NVIC->IP 4 bit  rw  0xe000e420  00 00 00 00 30 00 30 30 90 00 00 00 00 00 00 00
NVIC->IP 4 bit  rw  0xe000e430  00 20 00 00 00 00 80 f0 00 00 00 00 00 50 00 00
NVIC->IP 4 bit  rw  0xe000e440  00 00 00 00 00 00 00 30 00 00 00 00 00 00 00 00
NVIC->IP 4 bit  rw  0xe000e450  00 00 30 30 00 00 00 00 00 00 00 00 00 00 00 00
NVIC->STIR      wo  0xe000ef00                       software trigger interrupt reg.
SYSCFG->EXTICR[4]   0x40013808  0401 0006 3300 0003  gpio interrupt line config., 4 bit/exti
EXTI->IMR           0x40013c00  010c15               int. mask, 1 -> enabled reg.
EXTI->EMR           0x40013c04  000000               event mask, 1 -> enabled reg.
EXTI->RTSR          0x40013c08  000c00               rising trigger selection reg.
EXTI->FTSR          0x40013c0c  010c15               falling trigger selection reg.
EXTI->SWIER         0x40013c10  000000               software interrupt event reg.
EXTI->PR            0x40013c14  000000               pending reg.
```

Befehle im Detail

stat	Geräte-, Speicher- und Schnittstellenstatus ausgeben

```
stat -scb
```

Es werden Informationen zum ‚System Control Block' des Microcontrollers **STM32F427xx** ausgegeben.

Diese Informationen haben für den Normalbetrieb keine Bedeutung.

```
SCB->   adr e000ed00
   ->  offs val
CPUID     00 410fc241
ICSR      04 00400000
VTOR      08 08110000
AIRCR     0C fa050000
SCR       10 00000000
CCR       14 00000200
SHPR[12] 18 00 00 00 00 00 00 00 00 00 00 00 10
SHCSR     24 00000000
CFSR      28 00000000
HFSR      2C 00000000
DFSR      30 00000000
MMFAR     34 e000ed34
BFAR      38 e000ed38
AFSR      3C 00000000
PFR[2]    40 00000030 00000200
DFR       48 00100000
ADR       4C 00000000
MMFR[4]   50 00100030 00000000 01000000 00000000
ISAR[5]   60 01101110 02112000 21232231 01111131 01310132
CPACR     88 00f00000
```

```
stat -ic [-d|-D] [-nz | <irq> [...]]
```

```
stat -ic
```

Es werden die Interrupt-Zähler aller Interrupt-Quellen des Controllers STM32F427xx ausgegeben. In der letzten Zeile wird der Zeitraum angegeben, in dem diese Werte gezählt wurden (Uptime).

```
IRQ_cnt[  0...  7]         0        23         0         0         0         0      2231         0
IRQ_cnt[  8... 15]         2         0      7323         0         0         0         0         0
IRQ_cnt[ 16... 23]         0         0       111         0         0         0         0         0
IRQ_cnt[ 24... 31]         0         0         0         0         0         0         0         0
IRQ_cnt[ 32... 39]         0         0         0         0     24297         0         0     27831
IRQ_cnt[ 40... 47]         0         0         0         0         0         0         0         0
IRQ_cnt[ 48... 55]         0    501662         0         0         0         0      5211    881909
IRQ_cnt[ 56... 63]         0         0         0         0         0       704         0         0
IRQ_cnt[ 64... 71]         0         0         0         0         0         0         0    832799
IRQ_cnt[ 72... 79]         0         0         0         0         0         0         0         0
IRQ_cnt[ 80... 87]         0         0      1023    368621         0         0         0         0
IRQ_cnt[ 88... 90]         0         0         0
IRQ_cnt uptime 5212.795428
```

Befehle im Detail

stat	Geräte-, Speicher- und Schnittstellenstatus ausgeben

```
stat -ic -d | -D
```

Es werden die Interrupts aller Interrupt-Quellen des Controllers STM32F427xx ausgegeben, die seit der letzten Abfrage mit ‚**-d**' (delta) aufgetreten sind. In der letzten Zeile wird der Zeitraum angegeben, in dem diese Werte gezählt wurden (Deltatime seit der letzten Abfrage).

```
IRQ_cnt[  0...  7]         0         0         0         0         0         0        19         0
IRQ_cnt[  8... 15]         0         0         0         0         0         0         0         0
IRQ_cnt[ 16... 23]         0         0         0         0         0         0         0         0
IRQ_cnt[ 24... 31]         0         0         0         0         0         0         0         0
IRQ_cnt[ 32... 39]         0         0         0         0         0         0         0      2204
IRQ_cnt[ 40... 47]         0         0         0         0         0         0         0         0
IRQ_cnt[ 48... 55]         0         0         0         0         0         0        19     18156
IRQ_cnt[ 56... 63]         0         0         0         0         0         1         0         0
IRQ_cnt[ 64... 71]         0         0         0         0         0         0         0      6954
IRQ_cnt[ 72... 79]         0         0         0         0         0         0         0         0
IRQ_cnt[ 80... 87]         0         0         0      2945         0         0         0         0
IRQ_cnt[ 88... 90]         0         0         0
IRQ_cnt cycle 18.559300
```

Wird ‚**–D**' anstelle von ‚**–d**' angegeben, dann entfällt die Angabe über den Zeitraum in welchem die Werte gezählt wurden.

```
stat -ic -nz
```

Es werden die Zähler der Interrupt-Quellen des Controllers STM32F427xx ausgegeben, die seit power on verschieden von 0 sind (not zero).

In der vorletzten Zeile wird die Zahl der Interrupt-Quellen angegeben, die wenigstens einmal einen Interrupt ausgelöst haben.

In der letzten Zeile wird der Zeitraum angegeben, in dem diese Werte gezählt wurden (Uptime).

```
IRQ_cnt[  1]          23 PVD
IRQ_cnt[  6]        2265 EXTI0
IRQ_cnt[  8]           2 EXTI2
IRQ_cnt[ 10]        7323 EXTI4
IRQ_cnt[ 18]         111 ADC
IRQ_cnt[ 36]       24297 SPI2
IRQ_cnt[ 39]       32236 USART3
IRQ_cnt[ 49]      501662 SDIO
IRQ_cnt[ 54]        5245 TIM6_DAC
IRQ_cnt[ 55]      915095 TIM7
IRQ_cnt[ 61]         705 ETH
IRQ_cnt[ 71]      845443 USART6
IRQ_cnt[ 82]        1023 UART7
IRQ_cnt[ 83]      373891 UART8
IRQ's with non zero counts: 14
IRQ_cnt uptime 5245.359643
```

Befehle im Detail

stat	Geräte-, Speicher- und Schnittstellenstatus ausgeben

```
stat -ic -d -nz
```

Es werden die Zähler der Interrupt-Quellen des Controllers STM32F427xx ausgegeben, die seit der letzten Abfrage verschieden von 0 sind (not zero), also die Interrupt-Anzahl seit der letzten Abfrage.

In der vorletzten Zeile wird die Zahl der Interrupt-Quellen angegeben, die wenigsten einmal einen Interrupt ausgelöst haben.

In der letzten Zeile wird der Zeitraum angegeben, in dem diese Werte gezählt wurden (Deltatime seit der letzten Abfrage).

```
IRQ_cnt[  6]           5 EXTI0
IRQ_cnt[ 39]         590 USART3
IRQ_cnt[ 54]           4 TIM6_DAC
IRQ_cnt[ 55]        4381 TIM7
IRQ_cnt[ 71]        1898 USART6
IRQ_cnt[ 83]         775 UART8
IRQ's with non zero counts:  6
IRQ_cnt cycle 4.489115
```

```
stat -ic <irq> [...]
```

Es werden die Zähler der aufgelisteten Interrupt-Quellen des Controllers STM32F427xx ausgegeben.
In der letzten Zeile wird der Zeitraum angegeben, in dem diese Werte gezählt wurden (Uptime).

```
stat -ic 55 71

IRQ_cnt[ 55]    1802374 TIM7
IRQ_cnt[ 71]    1190678 USART6
IRQ_cnt uptime 6456.800021
```

```
stat -ic -d <irq> [...]
```

Es werden die Zähler der aufgelisteten Interrupt-Quellen des Controllers STM32F427xx ausgegeben.
In der letzten Zeile wird der Zeitraum angegeben, in dem diese Werte gezählt wurden (Deltatime seit der letzten Abfrage).

```
stat -ic -d 55 71

IRQ_cnt[ 55]         982 TIM7
IRQ_cnt[ 71]         315 USART6
IRQ_cnt cycle 1.001602
```

```
stat -ic -l [<irq> [...]]
```

Es wird eine Liste der aktuellen ISR (Interrupt Service Routine) mit Adressen ausgegeben.

Wird hinter <irq> eine Ziffer angegeben ('[...]'), so wird nur die Adresse der ISR mit dem der angegebenen Ziffer entsprechenden IRQ Count ausgegeben (Interrupt Request).

Befehle im Detail

stat	Geräte-, Speicher- und Schnittstellenstatus ausgeben

```
stat <task_name>
```

Es werden Zahlen zum Ein-/Ausgabepuffer der UART, falls der Task eine UART als Ein-/Ausgabefile hat, und zum Shell-Interpreter der mit diesem Task verbunden ist, ausgegeben.

```
stat coma
coma shell: com3 rxlvl 1 (1999) ovfl rx 0 tx 0 lines 18 0 unknown 0  intp_ovfl 0 0
```

Der Task ist mit der com3 verbunden. Im Empfangspuffer dieser UART befindet sich 1 Zeichen. Der Puffer hat eine Nettogröße von 1999 Zeichen. Es gab keinen Empfangs- bzw. Sendeüberlauf. Es wurden 18 Kommandozeilen ausgeführt und 0, die über eine Message-Pipe als Stream eingekommen sind. Es sind 0 unbekannte Kommandos aufgetreten und 0 mal war die Kommandozeile zu lang für den Interpreterzeilenpuffer (350 Zeichen).

```
stat heap [malloc <size>|free <adr>]
```

Mit dem Kommando ‚stat heap' wird der feie Speicher im Heap bestimmt. Die Zahl gibt die Summe alle Speicherfragmente an. D. h. diese Zahl ist nicht unbedingt ein zusammenhängender Speicherbereich. Diese Zahl ist identisch mit der Zahl in der Ausgabe auf das Kommando ‚stat'.

Mit 'stat heap malloc <size>' kann ein Speicherblock mit der Größe <size> bytes reserviert werden. Das ppmOS wird diesen Speicherblock nicht verwenden. Die Reservierung wird mit der Ausgabe der Anfangsadresse des Speicherblocks bestätigt.

Beispiel:

```
>stat heap malloc 20000
malloced  20000 @ 0x2001b5b0
```

Der Rückkehrcode des Kommandos (0 für reserviert und ungleich 0 für Fehler) kann in einer bedingten Kommandoausführung mit && oder || verwendet werden.

Der Speicherbereich steht ab der Reservierung für Anwenderzwecke zur Verfügung. Er bleibt bis zum nächsten Reset bzw. Power off reserviert.

Mit 'stat heap free <adr>' kann ein zuvor reservierter Speicherbereich wieder dem ppmOS zur Verfügung gestellt werden. Dabei muss <adr> genau die Zahl sein, die bei ‚malloc' ausgegeben wurde.

Befehle im Detail

stat	Geräte-, Speicher- und Schnittstellenstatus ausgeben

```
stat -wdt [init|set [[std|max] [<pr> <rld>]]]  pr 0...6, rld 0...4095, T=(rld<<(pr+2))/32kHz
```

Mit diesem Kommando wird die Einstellung des Watchdogtimers abgefragt bzw. dessen Einstellungen verändert.

Im **ppmOS** ist standardmäßig der Watchdogtimer aktiviert. Er lässt sich nicht deaktivieren, jedoch lässt sich das Watchdogintervall verändern. Die Abfrage mit ‚**stat -wdt**' ergibt:

```
wdt pr 1 rlr 2000   period  500.0 ms
```

Sollte die Firmware durch eine Fehlfunktion länger als 500 ms nicht reagieren, dann wird ein Watchdogreset ausgelöst. Dieser ist in seiner Wirkung mit dem Befehl ‚Reset' und dem Power-On-Reset identisch.

```
stat -wdt init
stat -wdt set std
```
Der Watchdogtimer wird in seiner Standardeinstellung initialisiert (500 ms).

```
stat -wdt set max
```
Der Watchdogtimer wird auf das maximal mögliche Intervall von 32 s eingestellt.

```
stat -wdt <pr> <rld>
```
Der Watchdogtimer wird mit den angegebenen Werten eingestellt. Die Formel zur Berechnung eines Intervalls lautet:

```
T=(rld<<(pr+2))/32kHz
```
Hierin kann **pr** die Werte 0...6 annehmen und **rld** ist ein 12-Bit Wert, der sinnvoller Weise im Bereich 1...4095 liegen muss. Der Wert 0 für **rld** führt unmittelbar zum Watchdogreset.

Mit dem Wert 3 für **pr** sind die Werte für **rld** die Millisekunden des sich ergebenden Watchdog-Intervalls.

Beispiel:
```
stat -wdt set 3 500

wdt pr 3 rlr 500   period  500.0 ms
```

Befehle im Detail

tail	Ausgabe des Endteils einer Datei

Mit diesem Kommando kann, ähnlich wie beim Kommando ‚**cat**', der Inhalt einer Datei auf der Konsole ausgegeben werden. Im Gegensatz zu ‚**cat**' wird mit ‚**tail**' das Ende der Datei ausgegeben. In der Standardeinstellung werden die 200 letzten Bytes der Datei ausgegeben. Mit dem Schalter ‚**-c**' kann eine Anzahl der auszugebenden Bytes eingestellt werden. Der Ausgabebeginn ergibt sich aus der Filelänge minus der angegeben Zahl. Prinzipiell können alle Filetypen des ppmOS angegeben werden. Da beim Zugriff auf das Ende der Datei der Dateizugriffszeiger neu gesetzt werden muss, hat dieses Kommando jedoch nur bei Dateien aus dem FAT-Dateisystem und bei Dateien vom Typ ‚**mem-[<adr>]-<len>-{r|w|p}[v]**' die gewünschte Wirkung.

Die Möglichkeiten der Ausgabeumleitung sind bei diesem Kommando ebenfalls anwendbar. Äquivalent zum Kommando ‚**tail ...**' ist das Kommando ‚**cat -o <offs> ...**' mit negativen Werten für **<offs>**.

Parameter:

```
[-c <bytes>] <file>
```

```
[-c <bytes>]
```

Dieser optionale Parameter gibt die Anzahl der Bytes an, die ausgegebene werden sollen. Ist die Datei kürzer, dann wird die gesamte Datei ausgegeben.

```
<file>
```

Name der Datei, die ausgegeben werden soll.

Befehle im Detail

tcp	TCP-Verbindung über das GSM-Modem herstellen

Mit diesem Kommando wird eine TCP-Verbindung im Raw-Mode über das GSM-Modem zu einem Server aufgebaut. Es dient der Fernbedienung und der Online-Datenübertragung in beide Richtungen. Die Verbindung kann nur vom **40xx** aufgebaut werden. Dies geschieht entweder durch einen in der **autoexec.sh** gestarteten Cron-Job oder durch eine SMS-Fernbedienung (siehe dazu das Kommando ‚**sms ...**' S. 278). Das Kommando ist erst nach der Ausführung von ‚gsm init' verfügbar.

In der Default-Einstellung werden ankommende Daten vom Shell-Interpreter ausgeführt und Daten, die an **coma** ausgegeben werden, werden in den Stream dupliziert. Das Ein- und Ausgabefile kann beim Öffnen der Verbindung und während der Verbindung geändert werden.

Die notwendigen Angaben zur Einwahl in das GSM-Netz stehen in **$(syspath)upload.cfg** (in der Regel hat **$(syspath)** den Inhalt ‚**c:/sys/**').

Die Angaben zum verwendeten Server (IP-Adresse und Port) stehen in **$(syspath)tcp.cfg**. In der Default-Einstellung ist **$(syspath)** gleich ‚**c:/sys/**'.

Die Datei ‚tcp.cfg' enthält im Minimum nur die Zeilenanzahl

address socktcp://<ip-address>:<port>

Alle darauf folgenden nichtleere Zeilen werden bei Herstellung der Verbindung an den Server übertragen. Hier können also Login-Daten o. ä. stehen. Zeilen, die mit ‚**exec**' beginnen werden nicht übertragen. Alles, was in der Zeile hinter ‚**exec**' folgt, wird an den Shell-Interpreter übergeben und dort ausgeführt.

Beispiel für **tcp.cfg** mit dem Ziel, eine Terminalverbindung aufzubauen:

```
address socktcp://87.165.217.89:52734

exec @echo connected:
exec @ver
exec @stat
exec @echo ------------------------------------------------------------
```

Die Zeilen hinter **exec** werden ausgeführt und auf dem entfernten Terminal erscheint nach Herstellung der Verbindung:

```
connected:
ppm40xx V 1.22 compiled Mar 30 2017 16:20:47 (168 MHz, CPUID 410fc241, GCC 6.2.1)
copyright ppm GmbH 2017     9F6 UID: 3334.3032.3532.470b.001c.002d '-....G252043'
------------------------------------------------------------------------------
heap: 0x20030000 free 109568 min 1264, txt 0x8170488, dat 2524, bss 26920, ctor 60
coma: com3 rxlvl 1 (1999) ovfl rx 0 tx 0 lines 44 0 esc_lines 0 0 unknown 0  intp_ovfl 0 0
RR: RST (0x14) counts [LPW 0, WDG 0, IWD 57 | SWR 168, POR 337, EXR 565, BOR 337] RST 163, JMP 1, WDT 1, CSH 5, CSB 0, PON 757
open files: 1 max 12
 0 C:TCP.CFG      r        183        110  72 00 0 17.03.30 17:12:36  16.02.26 17:02:41.900 'tcp.cfg'
------------------------------------------------------------
```

Parameter:

```
  [-v] [-s] open [{@|!}<cfg_file> | <ip_adr>:<port>] [-rx[a] <file>] [-tx {<file>|gpgga-tx}]
| [-v] close [wait [<wait_s>]]    ohne <wait_s> -> forever
| stat
| echo [on|off]
| -rx[a] {<file>|close}           received data   write to   ..., a -> append
| -tx    {<file>|close|gpgga-tx}  data to be tx'd read  from …
```

Befehle im Detail

tcp	TCP-Verbindung über das GSM-Modem herstellen

-v

Bei der Ausführung werden zusätzliche Informationen während der Verbindung ausgegeben (verbose). Diese können später wieder mit ‚**tcp echo off**' abgeschaltet werden. Der Parameter ist nur für Diagnosezwecke erforderlich.

-s

Während der gesamten Verbindungszeit werden keine Verbindungs- bzw. Diagnoseinformationen ausgegeben (silent). Das ist für dieses Kommando die Default-Einstellung.
Informationen zur übertragenen Datenmenge in beide Richtungen können jederzeit mit dem Kommando ‚**tcp stat**' ausgegeben oder die periodische Ausgabe mit ‚**tcp echo [on|off]**' ein- oder ausgeschaltet werden.

tcp open

Das ist das Standardkommando zur Herstellung einer TCP-Verbindung im Raw-Mode.

Hierbei werden implizit die Ausgabedaten von **coma** übertragen und die ankommenden Daten vom Shell-Interpreter von **coma** ausgeführt.

Die Einwahl- und Serverinformationen werden aus ‚**upload.cfg**' und ‚**tcp.cfg**' gelesen.

{@|!}<cfg_file>

Anstelle des Files ‚c:/sys/tcp.cfg' wird <cfg_file> für die Spezifizierung der TCP-Verbindung verwendet.

<ip_adr>:<port>

Anstelle der im File ‚c:/sys/tcp.cfg' angegebenen IP-Adresse und Portnummer werden die hier angegebenen verwendet. Im Minimum reicht also zur Herstellung einer Verbindung das Kommando:

tcp open <ip_addr>:<port_nr>

-tx {<file>|close|gpgga-tx}

Der optionale Parameter ermöglicht die Einstellung eines Files, aus dem Daten zum Server übertragen werden. Als Default ist das **pipe1**. Die Einstellung kann bei ‚**tcp open -tx ...**' oder jederzeit während der Verbindung vorgenommen werden.

[-tx {<file>|close|gpgga-tx}]

Der optionale Parameter ermöglicht die Einstellung eines Files, aus dem periodisch Daten zum NTRIP-Server übertragen werden. Per Default (ohne **-tx**) sind das GPGGA-Datensätze, die im Zyklus von 20 s übertragen werden. Wenn eine abweichende Einstellung ‚**ntrip -tx file_xyz**' gemacht wurde, dann kann mit (s. nächste Seite)

Befehle im Detail

tcp	TCP-Verbindung über das GSM-Modem herstellen

```
ntrip -tx gpgga-tx
```

wieder auf die als Default eingestellte GPGGA-Übertragung zurückgeschaltet werden.

Es ist zu beachten, dass, wenn der Schalter bei ‚**ntrip open -tx ...**' angegeben wird, auch die Login-Daten im angegebenen File erwartet werden.

Sollen keine GPGGA-Datensätze und nichts weiter als die Login-Daten übertragen werden, dann ist folgende Kommandosequenz auszuführen:

```
ntrip open; ntrip -tx close
```

Mit

```
ntrip -tx close
```

wird das File geschlossen und es werden keine Daten zum NTRIP-Server übertragen und wenn vorher ‚**ntrip echo on**' eingeschaltet war, dann erscheint folgende Zusammenfassung:

```
TCPio terminate signal received
 --------------------------------------
 >   102 s     0     61659    605 bps
<    102 s     0        89      1 bps
ntrip: success!
```

```
-rx[a] [<file>|close]
```

Anstelle der impliziten Ausführung der empfangenen Daten vom Shellinterpreter **coma** werden die empfangenen Daten in das File **<file>** geschrieben. Das kann eine COM-Schnittstelle (**coma, comb, comc , gps1, gps2** oder **blth**), ein TCP-Stream (**ips..., icom...**), die CAN-Schnittstelle (**can1...**), eine Pipe (**pipe1, pipe2** oder **pipe3**), ein Memoryfile (**mem-...-w**) oder ein reguläres File im FAT-Dateisystem sein.

Die Default-Einstellung ist ‚**stdin**'.

Hinweis:
Wenn während einer Terminalsitzung mit ‚**tcp -rx file_xyz**' der Eingabedatenstrom in ein anderes File geschrieben werden, dann kann in dieser Terminalsitzung kein Kommando mehr ausgeführt werden. Ein Zurückschalten auf ‚**stdin**' müsste dann eventuell mit einem Cron-Job (z. B. ‚**cron put *+30. * * * * * * tcp -rx stdin**' Rückschaltung nach 30 s) oder mit Hilfe eines ‚Stellvertreters' erledigt werden.

Mit

```
tcp -rx close
```

wird das Ausgabefile geschlossen. Bei Umschaltung auf ein neues File mit ‚**tcp -rx <file>**' wird das alte File implizit geschlossen. Anmerkung: serielle Schnittstellen (**coma, comb, comc, can1-..., blth**, ...) werden nur formal geschlossen, sie bleiben weiterhin aktiv.

Befehle im Detail

tcp	TCP-Verbindung über das GSM-Modem herstellen

```
tcp stat
```

Es wird eine Zwischeninformation über das Ein- und Ausgabefile, die Datenrichtung, die Verbindungsdauer, die übertragene Datenmenge seit letzter Abfrage, die gesamte übertragene Datenmenge und die Datenrate in Byte/s ausgegeben:

```
tcp: received          178 bytes,       178 written to stdin
tcp: transmitted     30588 bytes,     30588 read  from pipe1
 >    167 s   128       178      1 bps
<     167 s 25299     30588    183 bps
```

Der Status ist auch noch nach dem Schließen der TCP-Verbindung abrufbar.

```
tcp close [wait [<wait_s>]]
```

Mit diesem Kommando wird die TCP-Verbindung geschlossen. Ohne den optionalen Schalter ‚**wait**' kehrt das Kommando sofort zurück. Es wartet also nicht auf die Beendigung des GSM-Verbindungstasks. Mit ‚**wait**' ohne Sekundenangabe wird gewartet bis das Kommando beendet ist und mit Sekundenangabe wird maximal die angegebene Zeit gewartet.

Ausgabe:

tcp: success!

```
tcp echo [on|off]
```

Mit diesem Kommando wird die periodische Ausgabe von Zwischeninformationen an- oder ausgeschaltet bzw. der on/off-Zustand mitgeteilt.

Die Ausgabe erfolgt immer, wenn sich die empfangene oder gesendete Datenmenge ändert, jedoch nicht häufiger als einmal pro Sekunde.

Beispiel (Ausschnitt):

```
 >    140 s   400     59969
 >    141 s   400     60369
<     141 s    74       681
 >    143 s   857     61226
 >    144 s   400     61626
 >    145 s   400     62026
 >    146 s   400     62426
```

Dabei bedeuten ‚**>**' eingehende Daten und ‚**<**' ausgehende Daten.

Es folgen die Dauer der Verbindung in Sekunden, Datenmenge seit letzter Ausgabe und gesamte Datenmenge in der jeweiligen Richtung.

Befehle im Detail

time	Shell-Kommando zur Zeitmessung

Mit dem Kommando ‚**time**' wird die Laufzeit des angegebenen Kommandos gemessen.

Parameter:

`<cmd>`

`<cmd>`

Kommando, dass ausgeführt wird.

Wenn die Laufzeit von einer Kommandosequenz bestimmt werden soll, dann muss die Sequenz in **"..."** oder **'...'** eingeklammert werden.

Beispiel 1:

```
time ntrip open

reading from ntrip.cfg
at^moni
mrx0:
mrx0: Serving Cell                                          I Dedicated channel
mrx0: chann rs  dBm MCC MNC  LAC cell NCC BCC PWR RXLev  C1 I chann TS timAdv PWR  dBm Q ChMod
mrx0: 16 52  -58 262  02 0CB2 1E53   4   1  33  -107  48 I    No connection
mrx0:
mrx1: OK
ntrip: connected
transmitting: GET /LEIJ1 HTTP/1.0
transmitting: User-Agent: NTRIP XS/1.14
transmitting: Authorization: Basic d2xhbXBlOnN1cGVy
ntrip.cfg scan completed
transmitting:
NTRIP initialized, waiting for data
 >     1 s     0         0
<      1 s    89        89
---------------------------------------
cpu time 1.166671505 s
```

Die Kommandoausführung dauerte 1.166 s.

Beispiel 2:

```
>time ls -r -ss c:/
>ls -r -ss c:/
summary: c:/
   3218310800 bytes in 4548 files,  235 directories
cpu time 0.976591742 s
```

Den Speicherverbrauch aller Dateien auf der Micro-SD-Karte zu ermitteln, dauerte 0.976 s.

Befehle im Detail

tsk	Multitasking verwalten und Status ausgeben

Mit diesem Kommando kann das Multitasking des **ppmOS** diagnostiziert und gesteuert werden.

Parameter:

```
  ls  [-s|-m] [-a] [<index> | <name>] [mp [isr]]  -s summary, -m stack size, -a all matching pattern
| {term|kill|freeze|thaw}
                  {<index> | <name>} [...| <name> ]  compare from begin "^<name>" or partially "<name>"
| {fout|fin} [-f <file>] {-all| -dflt |             set for all tasks or default for new tasks
                  {<index> | <name>} [...| <name> ]} set input or output file for task(s)
| exist [-v]      {<index> | <name>}                  -v verbose
| inf [-d|-e|-m]  {<index> | <name>} [...| <name> ]  -d no stack dump, -e exist task?, -m min. free stack
| ev [or|and|clr|xor] [<flag_word>|term|freeze]
                  {<index> | <name>} [...| <name> ]
| tsl             {<index> | <name>|-all|-dflt} <max_time_slice_us>
| dflt-stkl       [<default_stack_length>]
| -s                                               show scheduler loops
```

```
tsk ls [-s|-m] [-a] [<index> | <name>] [mp [isr]]
```

Es wird die Taskliste oder die Information zu einem Task dessen Taskindex bzw. Taskname angegeben ist, ausgegeben.

Beispiel:

```
  pid  context cycle %  total %        cycle s        total s tsl µs  event_f stk_base       sp stklen p name
    0 20005630  95.680   77.935      11.221023     8431.011895   1000        1 1000f5a0 1000fd70   2400   idle
    1 200039c8   0.009    0.009       0.001068        1.021407   1000 80000001 100057e4 10005e90   2000 ! adc
    9 20001230   0.171    0.192       0.020061       20.875606   1000 80000010 10007148 10008088   5200 ! gps
   10 2000f398   0.009    0.004       0.001063        0.507855   1000 82000000 2000efb0 2000f268   1320   tcp_stack_tick
   11 20015530   0.000    0.000       0.000022        0.011285   1000        0 20014ef0 20015458   1920   tcp_stack_rx
    2 20003810   0.066    0.620       0.007831       67.086418   1000        1 10000000 10001298   5000 ! coma
    3 200041b0   0.035    0.036       0.004141        3.972708   1000        0 10001388 10001e70   3000 ! cron
    4 20005118   0.000    0.000       0.000000        0.100912   1000        0 10001f40 10002f98   4500 ! event
    5 20003cf0   0.000    0.000       0.000000        0.000001   1000        0 1000445c 100056f0   5000 ! comb
    6 20004068   0.000    0.000       0.000000        0.000001   1000        0 100030d4 10004368   5000 ! comc
    7 20005bc0   1.562    1.763       0.183230      190.815642   1000        0 10005fb4 100066b0   2000 ! gsm
rm           0   0.000   11.572       0.000000     1140.590339   1000 dflt tsl, total rm tasks: 3
-------------------------------------------------------------
cpu load     %   1.900   14.458       0.217414     1424.982174 cycle       11.727601 s, uptime    10817.951158 s
```

Die Spalten in der Tabelle bedeuten:

Siehe nächste Seiten.

Befehle im Detail

tsk	Multitasking verwalten und Status ausgeben

"`tsk ls [-s|-m] [<index> | <name>] [mp [isr]]`" (f.):

`pid`	Prozess-ID oder Task-ID. Diese Nummer kennzeichnet den jeweiligen Task eineindeutig, d. h. zu jeder Nummer gehört genau ein Task und jeder Task hat genau eine Nummer. Beendet sich ein Task, dann wird es diese **pid** nicht ein zweites Mal geben. Alle neu gestarteten Tasks erhalten eine neue **pid**. Die Zahl wird bei Systemstart (Power on oder Reset) von 0 aufwärts gezählt.
`context`	Die Verwaltung eines Task erfordert eine Datenstruktur, die Taskkontext genannt wird. An der angegebenen Addresse befindet sich diese Task-Datenstruktur. Sie enthält unter anderem die ‚**pid**', den Tasknamen bzw. die Kommandozeile, die als Task gestartet wurde, die Eventflags des Tasks, die Ein- und Ausgabefiles des Tasks, den Verweis auf den Shell-Interpreter, der für diesen Task zuständig ist und weitere Daten.
`cycle %`	CPU-Last in Prozent, die der Task seit letzter ‚**tsk ls**'-Abfrage verursacht hat. Wenn ein Task nicht eventgesteuert sondern im Polling arbeitet, dann wird eine relative hohe CPU-Last angezeigt. Diese ist in der Regel aber eine ‚weiche' Last, da das Polling bzw. die Polling-Frequenz zunimmt je geringer die CPU-Gesamtlast wird.
`total %`	CPU-Last in Prozent, die der Task seit Taskstart verursacht.
`cycle s`	CPU-Zeit in s, die der Task seit der letzten ‚**tsk ls**'-Abfrage aktiv war.
`total s`	CPU-Zeit in s, die der Task seit dem Taskstart aktiv war.
`tsl µs`	maximale Zeitdauer, auch Zeitscheibe genannt, die der Task, ohne vom Scheduler unterbrochen zu werden, hintereinander arbeiten kann. Der Worst Case für die Zeitdauer, die der Task die CPU nicht bekommt (pausiert), ergibt sich aus der Summer der Zeitscheiben aller anderen Tasks. Für eine Priorisierung der Tasks steht die taskbezogene Einstellung der maximalen Zeitscheibe zur Verfügung. Die Einstellung der max. Zeitscheibe pro Task ist im Bereich von 10 µ bis 100 ms sinnvoll. Die meisten Tasks sind eventgesteuert und sie geben nach der Eventbehandlung ihre Zeitscheibe wieder ab.
`event_f`	Das ist das Event-Flag-Wort des Tasks. Es ist ein 32-Bit-Wert. In diesem Flag-Wort haben einige Bits eine feste Bedeutung und die übrigen können bei der Programmierung bestimmten Events frei zugeordnet werden. Wenn das Event-Flag-Wort eines Tasks 0 wird, dann wird der Task suspendiert. Er verbraucht dann keine weitere CPU-Zeit. Ein Sonderfall ist der, wenn das Bit 31 gesetzt ist. Es bedeutet, dass der Task eingefroren ist (‚**tsk freeze ...**'), er verbraucht ebenso keine Rechenzeit. Für die Steuerung der Rechenzeitzuweisung ist der Taskscheduler, der im Namen des ‚**idle**'-Task arbeitet, zuständig.
`stk_base`	Jeder Task hat seinen eigenen Stack. Die Anfangsadresse des Stack-Bereiches ist in dieser Spalte hexadezimal angegeben.
`sp`	Der aktuelle Stackpointer zeigt auf die in dieser Spalte angegebene Adresse (hexadezimal).
`stklen`	Dies ist die zum Abfragezeitpunkt kleinste freie Stack-Länge des Tasks (dezimal). Die Angabe ist das Ergebnis einer permanenten Minimumsuche.
`p`	Hier ist ein Leerzeichen oder '**!**' möglich. Das '**!**' sagt aus, dass der Stack-Speicherbereich beim Taskstart schon vorgegeben wurde. Ohne '**!**' heißt, der Stack wurde durch den Taskscheduler alloziiert.
`name`	Angabe des Tasknamen bzw. der Kommandozeile die mit '&' am Ende als Task gestartet wurde.

Siehe nächste Seite.

Befehle im Detail

tsk	Multitasking verwalten und Status ausgeben

”`tsk ls [-s|-m] [<index> | <name>] [mp [isr]]`” f.:

Unter der Liste folgt eine Zeile, die mit '**rm ...**' beginnt. Die erste Zahl gibt die Anzahl der Tasks an, die sich seit dem letzten '**tsk ls**'-Aufruf selbst beendet haben bzw. die anderweitig terminiert wurden. Bei Power on bzw. Reset wird dieser Zähler mit 0 initialisiert. In dieser Zeile steht hinter '**tsl µs ... dflt tsl**' die max. Zeitscheibe für Tasks, die zukünftig gestartet werden. Hinter '**total rm tasks:** ...' steht die Anzahl der Tasks, die sich seit Power On bzw. Reset selbst beendet haben bzw. die anderweitig terminiert wurden. Bei Power On bzw. Reset wird dieser Zähler mit 0 initialisiert.

Unter der Trennlinie folgt eine Zusammenfassung der CPU-Last aller Tasks in Prozent und Sekunden. Hinter '**cycle** ...' steht die Zeit, die seit dem letzten '**tsk ls**'-Aufruf vergangen sind. Dahinter folgt noch die Laufzeit seit Power On bzw. Reset (uptime).

Mit dem Schalter '**-s**' wird nur die zusammenfassende letzte Zeile angezeigt.

Mit dem Schalter '**-m**' wird in der Spalte '**stklen**' die Größe des Stack-Speicherbereiches anstelle des minimal freien Stacks angezeigt.

Mit dem Schalter '**-a**' werden alle Tasks ausgegeben, die in das angegebene Namensschema passen.

Beispiel:

```
tsk ls -a "loop"
```

Es werden nur die Tasks gelistet, die in der Kommandozeile ‚loop‘ enthalten.

```
tsk ls -s -a "loop"
```

Es werden nur die Tasks gelistet, die in der Kommandozeile ‚loop‘ enthalten und es werden die Tabellenüberschrift und die Zusammenfassung weggelassen.

Befehle im Detail

tsk	Multitasking verwalten und Status ausgeben

```
tsk ls [<index> | <name>] mp [isr]
```

Mit dem Schalter '**mp**' werden Informationen zu den Messagepipes der Tasks ausgegeben. Wird noch der Schalter '**isr**' angegeben, dann werden die Messagepipes, die von Interrupt-Routinen verwendet werden, angezeigt.

```
 pid owner_task   pipe-adr    event  ev_ovfl bufsze  avail ovfl_c   msg_cnt     lock pipebase  write_pt  read_pt  rd_last  pipe_end
   0 no pipe
   1 no pipe
   9 no pipe
  10 no pipe
  11 no pipe
   2 coma         2000b708 2000b70c 2000b714   1200      0      0      3253        0 2000b748  2000b7fc 2000b7fc 2000b7e8  2000bbf8
   3 no pipe
   4 no pipe
   5 comb         2000cbb8 2000cbbc 2000cbc4   1200      0      0         0        0 2000cbf8  2000cbf8 2000cbf8 2000cbf8  2000d0a8
   6 comc         2000d0b8 2000d0bc 2000d0c4   1200      0      0         0        0 2000d0f8  2000d0f8 2000d0f8 2000d0f8  2000d5a8
   7 no pipe
```

Die Spalten in der Tabelle bedeuten:

pid	Prozess-ID oder Task-ID. Diese Nummer ist dieselbe, die bei ‚**tsk ls**' ausgegeben wird.
owner_task	Name des Tasks, dem diese Pipe gehört.
pipe-adr	Speicheradresse (hexadezimal) des jeweiligen Pipe-Objekts.
event	Speicheradresse eines Pointers, der auf das Eventflag-Wort des Besitzertasks zeigt. Wenn Messages eintreffen, dann wird hierüber der Task mit dem Setzen eines Eventflags benachrichtigt. Der Task wird aktiv und liest die Message.
ev_ovfl	Speicheradresse eines Pointers, der auf das Eventflag-Wort des Besitzertasks zeigt. Wenn ein Überlauf an Messages eintritt, d. h. der Message-Puffer ist voll, weil er nicht bzw. nicht schnell genug geleert wird, dann wird hierüber der Task mit dem Setzen eines Eventflags benachrichtigt.
bufsze	Die Länge der Messagepipe in Byte. Messages werden in diesem Puffer zirkular ober ohne Umbruch gespeichert. Das hat den Vorteil, dass Messages ‚on place' verarbeitet werden können aber eventuell auch den Nachteil, dass unter Umständen eine Message nicht mehr hineinpasst, obwohl in der Summe noch Platz wäre.
avail	Anzahl der noch nicht ausgelesenen Messages
ovfl_c	Anzahl der erfolglosen Versuche, eine Message in die Pipe zu schreiben.
msg_cnt	Anzahl der erfolgreichen Message-Übermittlungen.
lock	Wenn der Wert (hexadezimal) nicht 0 ist, dann wird diese Messagepipe gerade von einem Task bearbeitet, dessen Taskadresse hier angegeben ist.
pipebase	Basisadresse des zirkularen Message-Puffers.
write_pt	Schreibzeiger in den Message-Puffer
read_pt	Lesezeiger im Message-Puffer
rd_last	Adresse der ältesten noch nicht ausgelesenen Message. Neue Messages werden nur bis dorthin geschrieben, falls si in die verbleibende Lücke passen.
pipe_end	Endadresse, bis zu der Daten geschrieben werden können.

Befehle im Detail

tsk	Multitasking verwalten und Status ausgeben

```
tsk {term|kill|freeze|thaw} {<index> | <name>} [...| <name> ]
```

Das Unterkommando dient zum Terminieren, Killen, Einfrieren und Auftauen von Tasks. Es kann ein Task oder mehrere Tasks angegeben werden. Ein Task wird durch seine **PID** oder durch den vollständigen Namen oder durch ein Fragment der Kommandozeile angegeben. Wird ein Fragment angegeben, dann ist dies ein Teilstring, der in `"..."` einzuschließen ist. Dieser Teilstring wird in der Kommandozeile, die den Namen des Tasks darstellt, gesucht. Soll der Teilstring der Anfang des Tasknamens sein, dann muss er in `"^..."` eingeschlossen werden.

`term`	Es wird das Bit 28 im Eventflag-Wort gesetzt. Wenn der Task entsprechend programmiert ist, dann wird er beim Setzen dieses Bits sich ordnungsgemäß destruieren. Das ist bei den Kommandos, die eine lange Laufzeit haben können (**loop, cp, cat, hexdump,** ...) der Fall.
`kill`	Der Taskscheduler wird dem Task keine weitere Rechenzeit geben und ihn aus der Taskliste entfernen. Es wird der Destruktor des Task aufgerufen, um ein möglichst ‚rückstandsfreies' Entfernen zu erreichen. Je nach Interaktion dieses Tasks mit anderen Tasks kann ein ‚rückstandsfreies' Entfernen jedoch nicht garantiert werden.
`freeze`	Der Taskscheduler wird dem Task keine weitere Rechenzeit geben. Der Task bleibt in der Taskliste.
`thaw`	Ein zuvor mit ‚**freeze**' eingefrorener Task wird wieder aufgetaut. Er läuft exakt an der Stelle weiter, an der er gestoppt wurde. Die Folgen von eventuellen Puffer- oder Zeitüberläufen sind im einzelnen nicht kalkulierbar.

Beispiel:

Kommando:

```
loop -8 -t 1000 hexdump comc&
```

Zeile in der Taskliste:

```
  15 20017b3c   0.000    0.000       0.006337       0.006337   1000 80000001 20016ae8 200179b8   2784   loop -8 -t 1000 hexdump comc
```

Task beenden, gleichwertige Varianten unter der Voraussetzung, dass es keine weiteren Tasks mit den Teilstrings im Namen gibt:

```
tsk term 15
tsk term "hexd"
tsk term "^loo"
```

Befehle im Detail

tsk	Multitasking verwalten und Status ausgeben

```
tsk {fout|fin} [-f <file>] {-all| -dflt | {<index> | <name>} [...| <name> ]}
```

Jeder Task hat ein Eingabe- und ein Ausgabefile. Mit diesem Kommando werden die IO-Files angezeigt oder geändert. Werden beim Start eines Tasks keine expliziten IO-Files angegeben, dann erbt der neue Task von dem Task der ihn gestartet hat. In der Regel sind dies die IO-Files vom Task ‚**coma**', d. h. insbesondere die Ausgaben erscheinen an der Schnittstelle ‚**coma**'.

Ermitteln welches Ausgabefile der ‚**adc**'-Task hat:

```
tsk fout adc
```

Ausgabe:

```
task pid  1 writes to com3
```

Ausgabefiles aller Tasks ausgeben:

```
tsk fout -all

task pid  0 writes to com3
task pid  1 writes to com3
task pid  9 writes to com3
task pid  2 writes to com3
task pid  3 writes to com3
task pid  4 writes to com3
task pid  5 writes to comb
task pid  6 writes to com2
task pid  7 writes to com3
```

Ausgabefiles aller zukünftig gestarteten Tasks auf ‚comb' setzen:

```
tsk fout -f comb -dflt
```

Ausgabe des ADC-Tasks von **coma** auf **comb** umschalten:

```
set adc echo on
set fout -f comb adc
```

```
tsk exist [-v] {<index> | <name>}
```

Mit diesem Unterkommando wird die Existenz eines Tasks mit dem angegebenen Namen oder PID abgefragt. Das Kommando dient der bedingten Shell-Kommandoausführung. Der Rückgabewerte ist 0 wenn der Task existiert und 1 wenn er nicht existiert. Wird der Schalter ‚**-v**' (verbose) angegeben, dann wird ein Text mit dem Ergebnis der Suche ausgegeben.

Befehle im Detail

tsk	Multitasking verwalten und Status ausgeben

```
tsk inf [-d|-e|-m] {<index> | <name>} [...| <name> ]
```

Das Unterkommando ‚**inf**' wird verwendet, um Informationen über den Zustand des oder der angegebenen Tasks zu bekommen. Es wird in erster Linie zur Diagnose verwendet.

Sollte in der Software ein fehlerhafter Zustand eintreten (Absturz), dann wird ein Exceptionhandler (fault handler) aktiv, der ebenfalls die Informationen dieses Unterkommando ausgibt und anschließend die Firmware durch einen Reset neu startet.

Beispiel 1:

Ausgabe Taskdump ohne Stackdump.

```
tsk inf -d coma

task:
  pid  context cycle %  total %        cycle s        total s tsl µs  event_f stk_base       sp stklen p name
    2 20003810   0.100    0.311       0.030532      84.473016   1000        1 10000000 10001298    984 ! coma
register:
r0-r3             00000000 00000000 20003810 00000000
r4-r7             20003464 20003468 20005610 00000000
r8-r11            00000000 00000000 00000000 00000000
r12, sp, lr, pc   2000e2b4 10001298 08123d4b 08124124
xpsr              410f0000
```

Es werden die Task-Informationen wie in ‚**tsk ls**' und die Registerinhalte der CPU ausgegeben.

Beispiel 2:

Ausgabe Taskdump mit Stackdump. Zusätzlich wird ein Hexdump des Speicherbereiches ausgegeben, auf den der Stackpointer zeigt.

```
tsk inf coma

task:
  pid  context cycle %  total %        cycle s        total s tsl µs  event_f stk_base       sp stklen p name
    2 20003810   0.165    0.311       0.008212      84.481228   1000  8000001 10000000 10001298    984 ! coma
register:
r0-r3             00000000 00000000 20003810 00000000
r4-r7             20003464 20003468 20005610 00000000
r8-r11            00000000 00000000 00000000 00000000
r12, sp, lr, pc   2000e2dc 10001298 08123d4b 08124124
xpsr              410f0000
stack:
10001298  00000000 00000000 00000000 00000000  ................
100012a8  00000000 00000000 00000000 00000000  ................
100012b8  00000000 00000000 00000000 00000000  ................
100012c8  00000000 00000000 00000000 00000000  ................
100012d8  20003464 20003468 20005610 00000000  d4. h4. .V. ....
100012e8  00000000 00000000 00000000 00000000  ................
100012f8  00000000 00000000 20003810 00000000  .........8. ....
10001308  2000e2dc 08123d4b 08124124 410f0000  ... K=..$A.....A
10001318  00000137 447a0000 deadface deadface  7.....zD........
10001328  deadface deadface deadface deadface  ................
10001338  deadface deadface deadface deadface  ................
10001348  deadface deadface 20003464 20003468  ........d4. h4.
10001358  20005610 08124145 deadface 00000000  .V. EA..........
10001368  00000009 2000b854 deadface 00000000  ....T.. ........
10001378  00000000 00000000 00000000 081431a5  ..............1..
10001388  deadface deadface deadface deadface  ................
```

Siehe nächste Seite.

Befehle im Detail

tsk	Multitasking verwalten und Status ausgeben

”`tsk inf [-d|-e|-m] {<index> | <name>} [...| <name> ]`” f.:

Beispiel 3:

Ausgabe der Taskadresse und der Stack-Informationen Minimum, aktuell und Stack-Größe in Byte.

Dies ist eine zusammenfassende Information über den Stack-Bereich des Tasks.

```
tsk inf -m coma comb comc

stack info: 20003810 10000000 10001298   984  4760  5000 ! coma
stack info: 20003cf0 1000445c 100056f0  4756  4756  5000 ! comb
stack info: 20004068 100030d4 10004368  4756  4756  5000 ! comc
```

`ev [or|clr|and|xor] [<flag_word>|term|freeze] {<index> | <name>} [...| <name> ]`

Mit diesem Unterkommando kann das Eventflag-Wort des oder der Tasks abgefragt oder geändert werden. Die Änderung erfolgt durch bit-weise logische Verknüpfung. Die Bits, die beeinflusst werden, werden in einem 32-Bit-Wort angegeben oder mit vordefinierten symbolischen Werten (**term** – Bit 28, **freeze** – Bit 31) angegeben.

`or`	Das angegebene **<flag_word>** wird bit-weise mit dem Eventflag-Wort ODER-verknüpft, d. h. es werden Bits im Eventflag-Wort gesetzt.
`clr`	Die Bits, die in **<flag_word>** gesetzt sind, werden im Eventflag-Wort gelöscht, d. h. der Task beendet die Aktivitäten, die mit den jeweiligen Bits verbunden sind.
`and`	Das angegebene **<flag_word>** wird bit-weise mit dem Eventflag-Wort UND-verknüpft, d. h. es werden Bits im Eventflag-Wort gelöscht, die in **<flag_word>** nicht gesetzt sind.
`xor`	Das angegebene **<flag_word>** wird bit-weise mit dem Eventflag-Wort exklusiv-oder-verknüpft, d. h. es werden Bits im Eventflag-Wort umgeschaltet (von 0 auf 1 bzw. von 1 auf 0), die in **<flag_word>** gesetzt sind.

`tsk tsl {<index> | <name>|-all|-dflt} <max_time_slice_us>`

Das im **ppmOS** realisierte Multitasking is preemptiv, eventgesteuert und es wird von einem tickless Scheduler kontrolliert. Damit ist gesagt, dass es kein festes Zeitscheibenraster für die Tasks gibt und dass es ein unterbrechbares Laufrecht für Tasks gibt. Anders ausgedrückt heißt das, dass kein Task beliebig viel Rechenzeit beanspruchen kann, er kann nicht andere Tasks blockieren. Es ist gesichert, dass jeder Task CPU-Zeit bekommt und zwar in vorherbestimmbarer Weise. Jedem Task wird eine maximale Rechenzeit in jedem Scheduler-Loop zugebilligt. Der ungünstigste Fall (worst case) lässt sich berechnen und entsprechend können mit diesem Kommando Einstellungen vorgenommen werden. Die Aktionen, die mit wesentlich kürzeren Reaktionszeiten ausgeführt werden müssen, werden in Interrupt-Serviceroutinen erledigt, die dann ihrerseits dem dazugehörigen Task Eventbits setzen. Dieser wird dann aktiv und erledigt dann die Weiterverarbeitung.

Mit diesem Kommando wird die maximale Zeitscheibe eines Tasks geändert. Mit dem Schalter ‚**-all**‘ kann die maximale Zeitscheibe für alle Tasks geändert werden. Mit dem Schalter ‚**-dflt**‘ wird die Zeitscheibe für zukünftig gestartete Tasks voreingestellt.

Als Standardwert ist 1 ms maximale Zeitscheibe für alle Tasks eingestellt. Werte bis hinunter zu ca. 20 µs sind noch sinnvoll, wobei dabei der Scheduler dann selbst schon 23 % CPU-Zeit verbraucht. Bei dem Standardwert von 1 ms verbraucht der Scheduler ca. 0.4 % CPU-Zeit.

Befehle im Detail

tsk	Multitasking verwalten und Status ausgeben

```
tsk dflt-stkl [<default_stack_length>]
```

Jeder Task verfügt über einen eigenen Stack-Speicherbereich. Dieser wird beim Start des Tasks vorgegeben. Die nach Power on bzw. Reset standardmäßig gestarteten Tasks erhalten einen bei der Entwicklung festgelegten Bereich. Für weitere gestartete Tasks wird ein Stack von 5000 Byte eingerichtet. Sollte sich zeigen, dass in einer besonderen Anwendung die Stack-Größe unpassend ist, dann wird mit diesem Kommando eine andere Default-Stack-Größe eingestellt. Diese Einstellung muss vor dem Starten des Tasks geschehen. Eine nachträgliche Änderung der Stack-Größe ist nicht möglich.

Die Stack-Größen der einzelnen Tasks kann mit ‚**tsk ls -m**' ermittelt werden.

```
tsk -s
```

Gibt Informationen zum Verhalten des Taskschedulers aus. Der Taskscheduler ist die Instanz in der Firmware des **ppmOS**, die dafür sorgt, dass alle Tasks Rechenzeit bekommen bzw. ihnen die Rechenzeit entzogen wird, um andere Tasks nicht zu benachteiligen. Da es sich beim Taskscheduler im **ppmOS** um einen ‚tickless scheduler' handelt, also einen Scheduler, der nicht nach einem festen Zeittaktschema arbeitet, sondern ereignisorientiert, kann die Zahl der Taskwechsel pro Zeiteinheit sehr unterschiedlich sein. Hier im Beispiel wurden in den letzten 107 Sekunden 1636 Taskwechsel vorgenommen und in der gesamtem Zeit seit power on waren es im Durchschnitt 10875 pro Sekunde.

```
task scheduler loops cycle:           176578,   1636 per s, cycle 107.868566 s
task scheduler loops total:        494311001,  10875 per s
```

Diese Informationen haben für den Normalbetrieb nur wenig Bedeutung.

Befehle im Detail

ulink	Shell-Kommando aus der Kommandoliste entfernen

Mit dem Kommando ‚**ulink**' können Kommandos aus der Liste der zur Verfügung stehenden Shell-Kommandos entfernt werden. Wenn ein Kommando entfernt wurde, dann kann es bis zum nächsten Power on bzw. Reset nicht mehr aufgerufen und auch nicht mehr neu in die Kommandoliste aufgenommen werden. Ein Laufzeit- oder Speichervorteil durch das Löschen von Kommandos ergibt sich nicht.

Parameter:

```
<cmd_to_unlink>
```

```
ulink <cmd_to_unlink>
```

Name des Kommandos, dass entfernt werden soll.

Beispiel:

```
ulink ping
```

Ausgabe:

```
ping unlinked
```

Befehle im Detail

ver	Version der Gerätefirmware ausgeben

Das Kommando ‚**ver**' gibt einen Informationstext aus, der zur Identifizierung des Gerätes dient.

Ausgabe nach ‚**ver**' ohne weitere Kommandoargumente:

```
ppm40xx V 1.22 compiled Apr 26 2017 12:45:08 (168 MHz, CPUID 410fc241, GCC 6.2.1)
copyright ppm GmbH 2017     9F6 UID: 3334.3032.3532.470b.001c.002d '-....G252043'
--------------------------------------------------------------------------------
```

In der ersten Zeile erscheinen die Versionsnummer der Firmware, das Compilier-Datum mit Uhrzeit, die Taktfrequenz des Controllers, die CPU-ID (bezeichnet den Typ) und die Version des C/C++-Compilers (GNU-C/C++).

In der zweiten Zeile erscheinen eine Copyright-Information, der 3-Buchstaben-Code des GPS-Boards und die einmalige ID des Controllerchips.

Die dritte Zeile stellt eine Unterstreichung dar, die auch die folgende Form haben kann:

```
--------------------------------- boot loader ---------------------------------
```

Damit wird unterschieden, ob das Hauptprogramm aktiv ist oder der Bootloader. Die Software läuft immer in der Hauptversion. Beim Update der Hauptversion wird temporär in die Bootloaderversion gewechselt. Beide Versionen haben exakt die gleiche Funktionalität. Vor dem Start der Hauptversion wird deren interne CRC geprüft. Ist diese falsch, dann bleibt die Bootversion aktiv.

Mit den optionalen Parametern können weitere spezielle Informationen ausgegeben werden.

Parameter:

```
[-a|-b|boot|main]     -a    all available information
                      -b    both program's version
                     boot   returns 0 if it is the boot programm else 1
                     main   returns 0 if it is the main programm else 1
```

Befehle im Detail

ver	Version der Gerätefirmware ausgeben

```
ver -a
```

Zusätzlich zu den allgemeinen Versionsinformationen werden spezielle Prozessorregister ausgegeben, die aber für den allgemeinen Gebrauch des Gerätes keine Bedeutung haben. Hier kann z. B. entnommen werden (5. Zeile), dass der Controller über 2 MB Flashspeicher verfügt.

Ausgabe:

```
ppm40xx V 1.22 compiled Apr 26 2017 12:45:08 (168 MHz, CPUID 410fc241, GCC 6.2.1)
copyright ppm GmbH 2017     9F6 UID: 3334.3032.3532.470b.001c.002d '-....G252043'
--------------------------------------------------------------------------------
unique ID    @ 0x1fff7a10: hex 3334.3032.3532.470b.001c.002d  asc: '-....G252043'
flash size   @ 0x1fff7a22: hex      800 (2048 kB) code begins @ 0x8110000
--------------------
ROM table    @ 0xe00fffd0: hex        0    0    0    0
ROM table    @ 0xe00fffd4: hex       11    4    a    0
ROM table    @ 0xe00fffd8: hex        d   10    5   b1
--------------------
ROM table    @ 0xe00ff000: hex fff0f003 fff02003 fff03003 fff01003 fff41003 fff42003 0
ROM table    @ 0xe00fffcc: hex        1
ROM table    @ 0xe000efd0: hex        4
--------------------
ROM table    @ 0xe000efe0: hex        c   b0    b    0    d   e0    5   b1
--------------------
Debug Fault Status                  rw @ 0xe000ed30: 0x00000000
Debug Halting Control and Status    rw @ 0xe000edf0: 0x03010000
Debug Core Register Selector        wo @ 0xe000edf4: 0x00000000
Debug Core Register Data            rw @ 0xe000edf8: 0x00000000
Debug Exception and Monitor Control rw @ 0xe000edfc: 0x00000000
--------------------------------------------------------------------------------
```

```
ver -b
```

Ausgabe der Kurzfassung der Versionsinformationen für die Boot- und Hauptprogrammversion. Bei Auslieferung eines Gerätes sind beide Versionen gleich.

```
version boot: ppm40xx V 1.22 compiled Apr 11 2017 18:19:04
version 2nd : ppm40xx V 1.22 compiled Apr 26 2017 12:45:08
```

```
ver boot
```

Abfrage, ob die Bootversion aktiv ist. Der Hauptzweck besteht in der Verwendung des Rückgabewertes des Kommandos in einer bedingten Shell-Kommandoausführung. Der Rückgabewert ist 0 wenn die angegebene Version aktiv ist, anderenfalls ist der Rückgabewert 1.

Ausgabe:

```
main programm is running
```

Befehle im Detail

ver	Version der Gerätefirmware ausgeben

```
ver main
```

Abfrage, ob die Hauptversion aktiv ist. Der Hauptzweck besteht in der Verwendung des Rückgabewertes des Kommandos in einer bedingten Shell-Kommandoausführung. Der Rückgabewert ist 0 wenn die angegebene Version aktiv ist, anderenfalls ist der Rückgabewert 1.

Ausgabe:

```
main programm is running
```

Wenn der Ausgabetext nicht benötigt wird, das Kommando also nur in einer bedingten Shell-Kommandoausführung verwendet wird, dann kann die Ausgabeumleitung nach ‚null' angewendet werden.

Beispiel:

```
ver main >null && @echo ja, die Hauptversion ist aktiv
```

Befehle im Detail

wait	Wartezeiten zwischen Shell-Kommandos einfügen

Das ‚**wait**'-Kommando dient zum Einfügen von Wartezeiten zwischen Shell-Kommandos. Das ist immer dann nötig, wenn ein definierter Zeitabstand zum Ende des vorherigen Kommandos eingestellt werden soll. Das Warten in der Standardversion (ohne Schalter ‚**-l**' bzw. ‚**-l3**') erfordert keine Rechenzeit der CPU. Der Task, der das ‚**wait**' ausführt, geht in dieser Zeit in den inaktiven Zustand. Wenn keine weiteren Tasks aktiv sind, dann geht die CPU in einen stromärmeren Stopzustand.

Die anzugebende Zeit kann als Gleitkommazahl mit dem Punkt als Dezimaltrennzeichen eingegeben werden.

Beispiel 1:

```
ip config
wait 5
```

Das Kommando ‚**ip config**' ist relativ schnell ausgeführt und kehrt nach ca. 25 ms zurück, aber die Herstellung des Links zum Router dauert danach noch ca. 2 s. Möchte man nun ein Folgekommando ausführen nachdem der Link hergestellt ist, dann kann das durch simples Warten mit diesem Kommando erledigt werden.

Beispiel 2:

```
loop 5 -t 400 "@set led yellow on >null; @wait -s 0.02; @set led yellow off >null"
```

Die gelbe LED wird 5 mal im Zeitabstand von 400 ms jeweils für 20 ms eingeschaltet.

Anmerkung:

Auch wenn bei der Eingabe Zahlen im Mikrosekundenbereich und kleiner möglich sind, es können aber mit diesem Kommando keine hochpräzisen Zeitsteuerungen im Mikrosekundenbereich realisiert werden.

Die Wirksamkeit dieses Kommandos kann unter Zuhilfenahme des Kommandos ‚**time**' getestet werden.

Beispiel 3:

```
time wait 1.234
```

Ausgabe:

```
start waiting 1234 ms
end waiting
cpu time 1.235286146 s
```

Parameter:

```
  [-s]     <delay_s>         //sec as float number, -s -> silent, no info
| [-s] -l  <delay_us>        //simple delay loop in us
| [-s] -l3 <delay_3clkticks> //simple delay loop in 3 CPU clock ticks (3/168000000 s)
```

Befehle im Detail

wait	Wartezeiten zwischen Shell-kommandos einfügen

```
@wait -s <delay_s>
```

Der Schalter ‚**-s**' (silent) zusammen mit dem vorangestellten ‚**@**' kann genutzt werden, wenn keine Ausgabe erfolgen soll.

```
wait -l <delay_us>
```

Die Angabe der Wartezeit erfolgt in Mikrosekunden. Der Unterschied zur Variante ohne ‚**-l**' besteht darin, dass hier die Wartezeit durch Ausführung von NOP's (no operation) realisiert wird. Angewendet wird diese Variante gewöhnlich nur für sehr kurze Wartezeiten.

Die in der Wartezeit auftretenden Interrupts verlängern die Wartezeit.

Es wird empfohlen die Variante ‚**wait <delay_s>**' zu verwenden.

Die Wartezeit in der Variante ‚**wait -l ...**' wird durch Hardwareinterrupts verlängert. Indirekter Weise kann damit die aktuelle Task- und Interrupt-Belastung ermittelt werden.

```
@time @wait -s -l 10000000
```

Ausgabe:

```
cpu time 10.049785362 s
```

Es ist kein weiter Task mit höherer Interrupt-Belastung aktiv.

```
wait -l3 <delay_3clkticks>
```

Die Angabe der Wartezeit erfolgt in der Einheit 3 CPU-Taktperioden. Bei 168 MHz CPU-Takt ist die Einheit also 17.86 ns. Der Unterschied zur Variante ohne ‚**-l3**' besteht auch hier darin, dass die Wartezeit durch Ausführung von NOP's (no operation) realisiert wird. Angewendet wird diese Variante gewöhnlich nur für sehr kurze Wartezeiten.

Die in der Wartezeit auftretenden Interrupts verlängern die Wartezeit.

Es wird empfohlen die Variante ‚**wait <delay_s>**' zu verwenden.

Befehle im Detail

wrfile	synthetische Testfiles oder Sektoren schreiben

Mit dem Kommando ‚**wrfile**' kann ein Testdatenfile erzeugt werden. Das dient in erster Linie dazu, Datenübertragungen zu simulieren, sie auszutesten, um in der realen Anwendung die Anforderungen zu erfüllen. Die Testdaten können auf der Kommandozeile als ASCII-Text in **"..."** oder **'...'** eingeschlossen oder als beliebige binäre Werte hexadezimal oder dezimal angegeben oder implizit vorgegeben werden. Die Testdaten werden in Form eines Testdatensatzes vorgegeben und dieser Datensatz wird mehrfach geschrieben, um die gewünschte Datenmenge zu erhalten. Zwischen den Datensätzen können Pausen eingelegt werden, um einen pulsierenden Datenstrom zu simulieren. Es kann eingestellt werden, dass erst nach einer gewissen Anzahl von Datensätzen die Pause eingelegt wird.

Als Zielfile sind alle im **ppmOS** möglichen Filetypen, die zum Schreiben geöffnet werden können oder Sektoren auf der Mikro-SD-Karte, möglich.

Das sind:

FAT-Dateisystem	c:/...
coma, comb, comc, gps1, gps2, gsm	UART-Schnittstelle
stdout	Standardausgabe (default ist coma)
blth	Bluetooth-Schnittstelle
pipe1, pipe2, pipe3	Pipe zur temporären Datenpufferung (Ringpuffer)
ips{a|b|c|d}{*|1|2|3|4}	IP-Server
icom{1|2|3|4}	IP-Clients
can1-...	CAN
mem-[<name>|<adr>][-<len>-[p][d|k][f|e][t][v]]	Speicherfile (RAM-Disk)
vcom1	USB-VCOM
null	‚schwarzes Loch' für Daten

Das Schreiben der Testdaten kann kontinuierlich oder in einem Zeittakt erfolgen.

Parameter:

```
[-t <ms> [-f[f]] [-c <n>]] [-a| -o <offset>] \
[-x{b|s|w|512}] [-m[a] <cnt>] {[-c[a] <size>] <file>|-s <drive> <sector>} <text>
```

Befehle im Detail

wrfile	synthetische Testfiles oder Sektoren schreiben

`[-t <ms> [-f[f]] [-c <n>]]`

Diese optionalen Parameter steuern den Zeittakt der Datensatzausgabe. Ohne Angabe werden die Datensätze so schnell wie möglich, also ohne Pause dazwischen, geschrieben.

Mit ‚**-t <ms>**' wird der Zeitabstand der Datenausgabe in Millisekunden eingestellt. Diese Zeit wird über die Suspendierung des betreffenden Tasks realisiert (s. a. Kommando ‚**wait**' S. 314). Mit dem Zusatz ‚**-f**' bzw. ‚**-ff**' wird der Zeittakt durch Polling realisiert, was bei sehr kleinen Zeit günstiger sein kann, aber auch eine höhere CPU-Last bedeutet.

Ist die Datensatzausgabedauer größer als die hier angegebene Zeit, dann gibt es keine Pause zwischen den Datensätzen.

Mit dem zusätzlichen optionalen Schalter ‚**-c**' kann eingestellt werden nach wie viel Datensätzen die Pause eingelegt wird. Der Default-Wert hierfür ist 1.

`[-a| -o <offset>]`

Mit diesen Schaltern wird angegeben, dass die Datensätze am Ende (‚**-a**') eines eventuell bestehenden Files geschrieben werden oder ab dem Offset <n> (‚**-o <n>**').

Bei seriellen Files (UART, CAN, Bluetooth, usw.) gibt es keinen Fileoffset. Daher ist bei diesen Files die Offsetangabe wirkungslos.

-xb

Mit diesem Schalter wird das Datenformat der Eingaben für **<text>** auf Byte (8 Bit) festgelegt.

-xs

Mit diesem Schalter wird das Datenformat der Eingaben für **<text>** auf short (16 Bit) festgelegt. Angaben in **<text>**, die in **"..."** oder **'...'** eingeschlossen sind, werden als Bytesequenz gespeichert.

Beispiel:

Eine Eingabe für **<text>** von 0 1 2 3 wird in der Bytefolge (little endian)

00 00 01 00 02 00 03 00

gespeichert.

Befehle im Detail

wrfile	synthetische Testfiles oder Sektoren schreiben

-xw

Mit diesem Schalter wird das Datenformat der Eingaben für **<text>** auf word (32 Bit) festgelegt. Angaben in **<text>**, die in **"..."** oder **'...'** eingeschlossen sind, werden als Bytesequenz gespeichert.

Beispiel:

Eine Eingabe für **<text>** von 0 1 2 3 oxdeadface wird in der Bytefolge (little endian)

00 00 00 00 01 00 00 00 02 00 00 00 03 00 00 00 ce fa ad de

gespeichert.

-x512

Mit diesem Schalter werden die Eingaben für **<text>** auf einen 512-Byte-Datenblock durch wiederholtes Kopieren aufgefüllt. Wenn kein **<text>** eingegeben wird, dann wird folgender Default-Datenblock erzeugt:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUV
WXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQR
STUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMN
OPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJ
KLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEF
GHIJKLMNOP
```

Er enthält die lückenlose Folge des ASCII-Alphabets von A bis Z. Am Ende ist ein CR und LF (0xd 0xa) eingefügt.

-m[a] <cnt>

Der Datensatz **<text>** wird **<cnt>** mal wiederholt kopiert.

Wird der Schalter ‚**-ma**' angewendet, dann hat das nur bei ‚**-x512**' eine Auswirkung. Die ersten 8 Zeichen des 512-Byte-Datenblocks werden mit einer achtstelligen Hexadezimalzahl überschrieben, die den Offset des jeweiligen Datenblocks, der mit diesem Kommando geschrieben wird, in Byte darstellt.

-c[a] <size>

Die Daten werden in ein spezielles Containerfile geschrieben. Ein Containerfile beginnt mit einem 512 Byte Headerblock, in dem die Ziellänge, die aktuelle Länge und der aktuelle Schreiboffset stehen. Darauf folgt der Datenblock, der maximal **<size>** Byte lang werden kann. Zu Beginn ist dieser Datenblock 0 Byte lang. Das Containerfile wird, wenn es seine vorgegebene Endlänge **<size>** erreicht hat, wie ein Ringpuffer wieder von vorne beginnend beschrieben. Mit der Variante ‚**-ca**' wird zunächst der aktuelle Schreibzeiger auf eine 512-Byte-Grenze durch Auffüllen mit 0-Bytes gebracht.

Befehle im Detail

wrfile	synthetische Testfiles oder Sektoren schreiben

```
-s <drive> <sector>
```

Dieser alternative Parameter bestimmt die Ausgabe der erzeugten Daten in Sektoren der Micro-SD-Karte. Der erste Sektor hat die Nummer 0. Bei der Anwendung dieser Ausgabevariante ist größte Umsicht gefordert, da bei unpassender Sektorangabe das Dateisystem beschädigt wird, und ein vollständiger Datenverlust wahrscheinlich ist. Falls es für Spezialanwendungen notwendig ist in dieser Form Daten zu speichern, wird empfohlen, die SD-Karte zu partitionieren und in der nicht genutzten Partition diese speziellen Files anzulegen.

Anhang A

Definition des binären Datensatzes 'ppmpos'

Definition des binären Datensatzes 'ppmpos' - Hinweise für Programmierer

```
//================
class BestPosBinTyp1_t              //little endian (Die GPS-Informationen werden dem NovAtel BESTPOS Log entnommen)
//================
{
   enum { CAN_TECH_SYNC     =0x4f50, //Synchronwort
          CAN_TECH_SYNC_MASK=~7      //die unteren 3 Bits sind ein frei laufender Zähler
        };

   static uint16_t bpos_pak_cnt[2];  //für die von gps1 und gps2, wird über msrc festgelegt

   public:

   uint16_t sync_cnt;       //Synchronwort und in den unteren 3 Bit ein frei laufender Zähler

   uint16_t gps_week;
   uint32_t week_ms;
   double   lat;

   double   lon;

   double   hgt_wgs84;      //Höhe über dem WGS84 Ellipsoid

   uint8_t  solution_state; //Referenz: 'OEM6 Firmware Reference Guide_Rev8_om-20000129_2015-01.pdf'
   uint8_t  position_type;
   uint8_t  nSat_solution_L1E1B1;
   uint8_t  nSat_solution_multifs;
   uint8_t  ext_solution_status;
   uint8_t  reserved;
   uint16_t paket_crc;

   public:

   uint16_t crc16_calc()  const { return CRC16Block((const uint8_t *)this, (uint8_t *)&paket_crc-(uint8_t *)this); }
   void     crc16_patch()       {        paket_crc =crc16_calc(); }
   bool     crc16_check() const { return paket_crc!=crc16_calc(); }
//1 -> error
   void     get_from(const GPS_time_t *gpstm, const
Novatel_msg_bestpos_t *src, const MsgSrcInfo_t *msrc);
   void     bestpos1_print() const;
}  __attribute__((packed));



//-------------------------------------
unsigned CRC16Block(const uint8_t *src, unsigned len)
//-------------------------------------
{ uint16_t crc=~0;

   while (len--)
     crc=CRC16(crc, *src++);

   return crc;
}


//-------------------------------------
uint16_t CRC16(uint16_t crc_, uint8_t val)
//-------------------------------------
{
   /*
   Quelle: "Modbus Protocol Reference Guide PI_MBUS_300 June 1996.pdf"
```

Anhang A

Definition des binären Datensatzes 'ppmpos'

```
          Modicon
          Modbus Protocol
          Reference Guide
          PI-MBUS-300 Rev. J

   hier modifiziert: zwei 8 Bit Tabellen in eine 16 Bit Tabelle umgewandelt

   */
   static const uint16_t ModBusCrcTab[]={
     0x0000, 0xc1c0, 0x81c1, 0x4001, 0x01c3, 0xc003, 0x8002, 0x41c2,
0x01c6, 0xc006, 0x8007, 0x41c7, 0x0005, 0xc1c5, 0x81c4,
     0x4004, 0x01cc, 0xc00c, 0x800d, 0x41cd, 0x000f, 0xc1cf, 0x81ce,
0x400e, 0x000a, 0xc1ca, 0x81cb, 0x400b, 0x01c9, 0xc009,
     0x8008, 0x41c8, 0x01d8, 0xc018, 0x8019, 0x41d9, 0x001b, 0xc1db,
0x81da, 0x401a, 0x001e, 0xc1de, 0x81df, 0x401f, 0x01dd,
     0xc01d, 0x801c, 0x41dc, 0x0014, 0xc1d4, 0x81d5, 0x4015, 0x01d7,
0xc017, 0x8016, 0x41d6, 0x01d2, 0xc012, 0x8013, 0x41d3,
     0x0011, 0xc1d1, 0x81d0, 0x4010, 0x01f0, 0xc030, 0x8031, 0x41f1,
0x0033, 0xc1f3, 0x81f2, 0x4032, 0x0036, 0xc1f6, 0x81f7,
     0x4037, 0x01f5, 0xc035, 0x8034, 0x41f4, 0x003c, 0xc1fc, 0x81fd,
0x403d, 0x01ff, 0xc03f, 0x803e, 0x41fe, 0x01fa, 0xc03a,
     0x803b, 0x41fb, 0x0039, 0xc1f9, 0x81f8, 0x4038, 0x0028, 0xc1e8,
0x81e9, 0x4029, 0x01eb, 0xc02b, 0x802a, 0x41ea, 0x01ee,
     0xc02e, 0x802f, 0x41ef, 0x002d, 0xc1ed, 0x81ec, 0x402c, 0x01e4,
0xc024, 0x8025, 0x41e5, 0x0027, 0xc1e7, 0x81e6, 0x4026,
     0x0022, 0xc1e2, 0x81e3, 0x4023, 0x01e1, 0xc021, 0x8020, 0x41e0,
0x01a0, 0xc060, 0x8061, 0x41a1, 0x0063, 0xc1a3, 0x81a2,
     0x4062, 0x0066, 0xc1a6, 0x81a7, 0x4067, 0x01a5, 0xc065, 0x8064,
0x41a4, 0x006c, 0xc1ac, 0x81ad, 0x406d, 0x01af, 0xc06f,
     0x806e, 0x41ae, 0x01aa, 0xc06a, 0x806b, 0x41ab, 0x0069, 0xc1a9,
0x81a8, 0x4068, 0x0078, 0xc1b8, 0x81b9, 0x4079, 0x01bb,
     0xc07b, 0x807a, 0x41ba, 0x01be, 0xc07e, 0x807f, 0x41bf, 0x007d,
0xc1bd, 0x81bc, 0x407c, 0x01b4, 0xc074, 0x8075, 0x41b5,
     0x0077, 0xc1b7, 0x81b6, 0x4076, 0x0072, 0xc1b2, 0x81b3, 0x4073,
0x01b1, 0xc071, 0x8070, 0x41b0, 0x0050, 0xc190, 0x8191,
     0x4051, 0x0193, 0xc053, 0x8052, 0x4192, 0x0196, 0xc056, 0x8057,
0x4197, 0x0055, 0xc195, 0x8194, 0x4054, 0x019c, 0xc05c,
     0x805d, 0x419d, 0x005f, 0xc19f, 0x819e, 0x405e, 0x005a, 0xc19a,
0x819b, 0x405b, 0x0199, 0xc059, 0x8058, 0x4198, 0x0188,
     0xc048, 0x8049, 0x4189, 0x004b, 0xc18b, 0x818a, 0x404a, 0x004e,
0xc18e, 0x818f, 0x404f, 0x018d, 0xc04d, 0x804c, 0x418c,
     0x0044, 0xc184, 0x8185, 0x4045, 0x0187, 0xc047, 0x8046, 0x4186,
0x0182, 0xc042, 0x8043, 0x4183, 0x0041, 0xc181, 0x8180,
     0x4040
                                        } ;

   __u16 crc(crc_);
   uint8_t i=crc.b0^val;

   __u16 tabval(ModBusCrcTab[i]);

   crc.b0=crc.b1^tabval.b1;
   crc.b1=       tabval.b0;

   return crc.s;
}
```


ALLES RUND UM GPS
ppm