L89&L26-DR&L26-P&L26-T AGNSS Application Note

GNSS Module Series

Rev. L89&L26-DR&L26-P&L26-T_AGNSS_Application_Note_V1.0

Date: 2020-03-17

Status: Released

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. For more information, please visit:
http://www.quectel.com/support/sales.htm

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm
Or email to: support@quectel.com

GENERAL NOTES

COPYRIGHT

About the Document

Revision History

Version	Date	Author	Description
1.0	2020-03-17	Berton PENG	Initial

Contents

Table Index

1 Introduction

This document mainly introduces the AGNSS function of Quectel GNSS modules, and describes the operation mechanism and procedures of this function. The main purpose of AGNSS is to improve TTFF of the modules.

The modules which support this feature are:

- L89
- L26-DR
- L26-P
- L26-T

1.1. Self-Trained Assisted GPS (STAGPS™)

L89, L26-DR, L26-P and L26-T are able to provide predicted ephemerides to the GNSS engine in a time frame less than the usual time (about 30 seconds) needed to download real ephemeris from the sky. This reduces considerably the time till the module gain fix especially in dense urban canyon environments.

The STAGPS™ provides the predicted ephemeris for each satellite.

The autonomous solution uses the past real ephemeris (downloaded from the sky and stored in its internal database) to extrapolate the parameters of future ephemeris (up to 5 days of prediction). For these reasons, the STAGPS™ autonomous performances (in terms of position accuracy using predicted ephemeris) are strictly dependent on the real ephemeris database content. During normal use of STAGPS™ autonomous, the system automatically uploads the real ephemeris into its database as soon as new ephemerides are downloaded from the sky, which means the global content of the real ephemeris input database is determined by the history of device running periods in the past.

1.2. Real-Time AGNSS (RT-AGNSS)

The Real-Time AGNSS is able to provide the approximate current time, the ephemerides, the almanacs and optionally the approximate position to the GNSS engine in a time frame less than the usual time (about 30 seconds) needed to download real ephemeris from the sky. This information is delivered from the host side to the module side. This reduces considerably the time to get fix especially in critical

environments where the ephemeris download time could be very long. Real-Time AGNSS requires a network connection to download assistance data from the server.

The assistance data includes the current time (if not available, from instance, from RTC), the ephemerides, the almanacs and optionally the approximate position.

All the assistance data can be injected into the device backup memory with NMEA commands. The following chapters of this document describe those NMEA commands which support Real-Time AGNSS, including the testing procedures to simulate different working scenarios. The NMEA commands and the testing procedures described in this document work for GPS and GLONASS, Galileo and BeiDou.

2 STAGPS™ Procedure

2.1. STAGPS™ Autonomous Procedure

The main feature introduced by a GNSS device with STAGPS™ capability is the ephemeris availability as soon as the device is turned on (and also GPS fix availability), even if the device is used in critical environments where the ephemeris downloading from the sky is difficult (e.g. urban canyon, low signal strengths, etc.). The STAGPS™ engine uses the old real ephemeris that is stored in its internal database to predict the future ephemeris. Having device ON with visible sky, the internal database is continuously updated with new ephemeris, predictions are updated as well and the expiration time of the prediction is moved forward. If the ephemeris database is not updated (e.g. the update is disabled), the predictions can be used for some days (usually 3 days) in the future and then they expire. The expiration time and the accuracy of GPS position based on predicted ephemeris rely on the real ephemeris (number of ephemeris per satellite and time distance between the ephemeris) used for prediction.

The most important and useful test to measure the STAGPS™ performance is the WARM start (device is restarted clearing the real ephemeris stored in the GPS backup memory), as manifested by the steps below. Under this condition, a device without STAGPS™ capability usually gets the fix in a time longer than 18 seconds, while the device with STAGPS™ capability will get the fix in the typical time of HOT start conditions.

1. Enable the Assisted GPS and save the configuration through reset. Once saved, the configurations will persist even after a power cycling. Therefore, this step is generally performed only once.

```
$PSTMCFGAGPS,1<CR><LF>          //Enable the Assisted GPS
$PSTMSAVEPAR<CR><LF>            //Save the configuration
$PSTMSRR<CR><LF>                //Reset the system
```

2. Enable STAGPS™ function.

```
$PSTMSTAGPSONOFF,1              //Enable the STAGPS™ function
```

3. Wait for STAGPS™ processing to complete. Then, query STAGPS™ processing status. When the status is 0, the STAGPS™ processing is completed.

```
$PSTMGETAGPSSTATUS              //Query the status of STAGPS™
```

2.2. NMEA Commands for STAGPS™

Table 1: NMEA Commands List for STAGPS™

Command	Description
$PSTMCFGAGPS	Enable/disable the Assisted GPS.
$PSTMSTAGPSONOFF	Turn on/off the STAGPS™ engine.
$PSTMSTAGPSINVALIDATE	Clear the data stored in the STAGPS™ internal database.
$PSTMGETAGPSSTATUS	Return the status of the STAGPS™ internal processing.

NOTE

The string ***<checksum>** is optional when inputting commands.

2.2.1. $PSTMCFGAGPS

This command configures the Assisted GPS. The setting can only take effect by issuing the saving command (i.e. **$PSTMSAVEPAR**) and, then, resetting the system (through **$PSTMSRR**). Once saved, the configurations will persist even after a power cycling.

Synopsis:

```
$PSTMCFGAGPS,<en_agps>*<checksum><CR><LF>
```

Arguments:

Parameter	Format	Description
en_agps	Decimal	Enable/disable AGPS engine 0 = AGPS disabled (default) 1 = AGPS enabled
checksum	Hexadecimal, 2 digits	Checksum of the message bytes between but not including the **$** and ***** characters

Results:

- If no error occurs, the following message will be returned:

```
$PSTMCFGAGPSOK*<checksum><CR><LF>
```

- If there is any error, the following message will be returned:

```
$PSTMCFGAGPSERROR*<checksum><CR><LF>
```

2.2.2. $PSTMSTAGPSONOFF

Turn on/off the STAGPS™ engine.

Synopsis:

```
$PSTMSTAGPSONOFF,<param>*<checksum><CR><LF>
```

Arguments:

Parameter	Format	Description
param	Decimal, 1 digit	ON/OFF status of the STAGPS™ engine: 0 = The STAGPS™ engine is suspended. 1 = The STAGPS™ engine is started
checksum	Hexadecimal, 2 digits	Checksum of the message bytes between but not including the **$** and ***** characters

Results:

- If no error occurs, the following messages will be returned:

```
$PSTMPOLSTARTED*08                  //The engine is started.
```

Or

```
$PSTMPOLSUSPENDED*02                //The engine is suspended.
```

- If there is any error, the following message will be returned:

```
$PSTMPOLONOFFERROR*5f
```

2.2.3. $PSTMSTAGPSINVALIDATE

Clear the data stored in the STAGPS™ internal database.

Synopsis:

```
$PSTMSTAGPSINVALIDATE,<param>*<checksum><CR><LF>
```

Arguments:

Parameter	Format	Description
param	Decimal, 1 digit	The database should be erased: 1 = Clear the real ephemeris database (only autonomous).
checksum	Hexadecimal, 2 digits	Checksum of the message bytes between but not including the **$** and ***** characters

Results:

- If no error occurs, the following message will be returned:

```
$PSTMSTAGPSINVALIDATEOK,1*00
```

- If there is any error, the following message will be returned:

```
$PSTMSTAGPSINVALIDATEERROR*41
```

2.2.4. $PSTMGETAGPSSTATUS

Return the status of the STAGPS™ internal processing.

Synopsis:

```
$PSTMGETAGPSSTATUS*<checksum><CR><LF>
```

Arguments:

None

Results:

The system sends back the STAGPS™ status in the following message:

```
$PSTMAGPSSTATUS,<status>*<checksum><CR><LF>
```

The parameters included in the result above are listed below:

Parameter	Format	Description
status	Decimal, 1 digit	0 = The STAGPS™ processing is completed. Any number different from zero means that the STAGPS™ processing is ongoing and so the ephemeris prediction data has not been completely generated.
checksum	Hexadecimal, 2 digits	Checksum of the message bytes between but not including the **$** and ***** characters

3 RT-AGNSS Procedure

3.1. Real-time Assistance Data Uploading Procedure

The Real-Time AGNSS performances depend on the availability of a network connection in order to download assistance data, which include:

- the current time (if not available, from instance, from RTC)
- the ephemerides
- the almanacs
- the approximate position (optional)

The procedure is as follows:

1. Once those data have been downloaded from the server, the first thing to do is to inject the current time into the device (if the device has no RTC, or if it is set to a wrong time). This can be done either by using the **$PSTMINITTIME** command or, if also the approximate position is available, both current time and position can be injected by using the **$PSTMINITGPS** command.

2. Then the ephemerides can be injected into the device by using the **$PSTMEPHEM** command for each satellite (between two consecutive commands there must be at least a 20-millisecond delay).

3. After injecting ephemerides, the almanacs can be injected into the device by the **$PSTMALMANAC** command for each satellite (between two consecutive commands there must be at least a 20-millisecond delay).

4. Now the device is capable of achieving the fix very quickly, if enough satellites are in view.

3.2. NMEA Commands for RT-AGNSS

Table 2: NMEA Commands List for RT-AGNSS

Command	Description
$PSTMINITGPS	Initialize GPS position and time.
$PSTMINITTIME	Initialize GPS time.
$PSTMEPHEM	Set ephemeris data.
$PSTMALMANAC	Set almanac data.

NOTE

The string ***<checksum>** is optional when inputting commands.

3.2.1. $PSTMINITGPS

Initialize position and GPS time using UTC format. This command must be issued after a cold reset otherwise the execution will fail. The date issued with parameters Day, Month and Year must be after January 2018.

Synopsis:

```
$PSTMINITGPS,<Lat>,<LatRef>,<Lon>,<LonRef>,<Alt>,<Day>,<Month>,<Year>,<Hour>,<Minute>,<Second>*<checksum><CR><LR>
```

Arguments:

Parameter	Format	Description
Lat	DDMM.MMM	Latitude (Degree-Minute.Minute decimals)
LatRef	N or S	Latitude direction N = North S = South
Lon	DDDMM.MMM	Longitude (Degree-Minute.Minute decimals)
LonRef	E or W	Longitude Direction E = East W = West

Alt	dddd – Decimal, 4 digits	Altitude in meters (-1500 to 100000)
Day	D – Decimal, 2 digits	Day of month (01 to 31)
Month	MM – Decimal, 2 digits	Month (01 to 12)
Year	YYYY – Decimal, 4 digits	Year (2018 to …)
Hour	HH – Decimal, 2 digits	Hour (00 to 23)
Minute	MM – Decimal, 2 digits	Minute (00 to 59)
Second	SS – Decimal, 2 digits	Second (00 to 59)
checksum	Hexadecimal, 2 digits	Checksum of the message bytes between but not including the **$** and ***** characters

Results:

- If no error occurs, the GNSS receiver's position and time will be initialized and the following message will be returned:

```
$PSTMINITGPSOK*<checksum><CR><LF>
```

- In case of errors, the following message is returned:

```
$PSTMINITGPSERROR*<checksum><CR><LF>
```

Example:

```
$PSTMINITGPS,4811.365,N,01164.123,E,0530,23,02,2018,09,44,12
```

NOTES

1. The error between input time and real time should be less than 3 seconds.
   GPS time = UTC time + Leap second. In 2019, Leap second = 18s.
2. The error between input position and real position should be less than 30 kilometers.

3.2.2. $PSTMINITTIME

Initialize GPS time using UTC format. The date issued with parameters Day, Month and Year must be after January 2018.

Synopsis:

```
$PSTMINITTIME,<Day>,<Month>,<Year>,<Hour>,<Minute>,<Second>*<checksum><CR><LF>
```

Arguments:

Parameter	Format	Description
Day	DD – Decimal, 2 digits	Day of month (01 to 31)
Month	M – Decimal, 2 digits	Month (01 to 12)
Year	YYYY – Decimal, 4 digits	Year (2018 to …)
Hour	HH – Decimal, 2 digits	Hour (00 to 23)
Minute	MM – Decimal, 2 digits	Minute (00 to 59)
Second	SS – Decimal, 2 digits	Second (00 to 59)
checksum	Hexadecimal, 2 digits	Checksum of the message bytes between but not including the **$** and ***** characters

Results:

- If no error occurs, the time will be initialized and the following message will be returned:

```
$PSTMINITTIMEOK*<checksum><CR><LF>
```

- In case of errors, the following message is returned:

```
$PSTMINITTIMEERROR*<checksum><CR><LF>
```

Example:

```
$PSTMINITTIME,23,02,2018,09,44,12
```

NOTE

The error between inputted GPS time and real GPS time should be less than 3 seconds.
GPS time = UTC time + Leap second. In 2019, Leap second = 18s.

3.2.3. $PSTMEPHEM

Load ephemeris data. This command allows the user to load the ephemeris data into the backup memory.

Synopsis:

```
$PSTMEPHEM,<sat_id>,<ephem_data_size>,<ephem_data>*<checksum><CR><LF>
```

Arguments:

Parameter	Format	Description
sat_id	Decimal, up to 3 digits	Satellite PRN
ephem_data_size	Decimal, 2 digits	Size expressed in byte of the ephemeris data field.
ephem_data	Hexadecimal	Ephemeris data string
checksum	Hexadecimal, 2 digits	Checksum of the message bytes between but not including the **$** and ***** characters

The encoding of the **ephem_data** field is described in the following tables.

Table 3: Encoding for GPS Ephemeris

Bits	Structure Member	Description
16	week	Week number of the Issue of Data.
16	toe	Time of week for ephemeris epoch.
16	toc	Time of week for clock epoch
8	iode1	Issue of data 1.
8	iode2	Issue of data 2.
10	iodec	Issue of data clock.
14	i_dot	Rate of inclination angle.
8	reserved	
24	omega_dot	Rate of right ascension.
8	reserved	Must be 0.
16	crs	Amplitude of the sine harmonic correction to the orbit radius.
16	crc	Amplitude of the cosine harmonic correction to the orbit radius.
16	cus	Amplitude of the sine harmonic correction to the argument of latitude.
16	cuc	Amplitude of the cosine harmonic correction to the argument of latitude.
16	cis	Amplitude of the sine harmonic correction to the angle of inclination.
16	cic	Amplitude of the cosine harmonic correction to the angle of inclination.

16	motion_difference	Mean motion difference from computed value
16	reserved	Must be 0.
32	inclination	Inclination angle at reference time.
32	e	Eccentricity.
32	root_A	Square root of major axis.
32	mean_anomaly	Mean anomaly at reference time.
32	omega_zero	Longitude of ascending node of orbit plane at weekly epoch.
32	perigee	Argument of perigee.
8	time_group_delay	Estimated group delay differential.
8	af2	Second order clock correction.
16	af1	First order clock correction.
22	af0	Constant clock correction.
1	reserved	Reserved for use by GNSS library – must be 1
1	reserved	Reserved for use by GNSS library – must be 1
1	reserved	Reserved for use by GNSS library – must be 1
1	available	Contains 1 if ephemeris is available, 0 if not.
1	health	Contains 1 if the satellite is unhealthy, 0 if healthy.
1	reserved	Must be 0.
4	accuracy	Accuracy.

The GPS encoding above can be represented in C language with the following structure:

```
typedef struct gps_ephemeris_s
{
    unsigned int week :          16
    unsigned int toe :           16;
    unsigned int toc :           16;
    unsigned int iode1 :         8;
    unsigned int iode2 :         8;
    unsigned int iodc :          10;
    unsigned int i_dot :         14;
    unsigned int spare1 :        8;
```

```
    unsigned int omega_dot :        24;
    unsigned int reserved1 :        2; // must be 0
    unsigned int reserved2 :        6; // must be 0
    unsigned int crs :              16;
    unsigned int crc :              16;
    unsigned int cus :              16;
    unsigned int cuc :              16;
    unsigned int cis :              16;
    unsigned int cic :              16;
    unsigned int motion_difference : 16;
    unsigned int reserved3 :        10; // must be 0
    unsigned int spare3 :           6;
    unsigned int inclination :      32;
    unsigned int eccentricity :     32;
    unsigned int root_a :           32;
    unsigned int mean_anomaly :     32;
    unsigned int omega_zero :       32;
    unsigned int perigee :          32;
    unsigned int time_group_delay : 8;
    unsigned int af2 :              8;
    unsigned int af1 :              16;
    unsigned int af0 :              22;
    unsigned int reserved4 :        1; // must be 1
    unsigned int reserved5 :        1; // must be 1
    unsigned int reserved6 :        1; // must be 1
    unsigned int available :        1;
    unsigned int health :           1;
    unsigned int reserved7 :        1; // must be 0
    unsigned int accuracy :         4;
} gps_ephemeris_t;
```

Table 4: Encoding for GLONASS Ephemeris

Bits	Structure Member	Description
16	week	Week number of the Issue of Data.
16	toe	Time of week for ephemeris epoch.
4	toe_lsb	Time of week for ephemeris epoch (LSB).
11	NA	Calendar day number within the four-year period since the beginning of last leap year (almanac).
7	tb	Time of ephemeris index.

2	M	Type of satellite: 00=GLONASS 01=GLONASS-M
2	P1	Time interval between two adjacent tb parameters.
1	P3	Number of satellites for which almanac is transmitted within this frame 0=4 1=5.
1	P2	Flag of oddness ("1") or evenness ("0") of the value of tb
1	P4	Flag to show that ephemeris parameters are present.
2	KP	Notification on forthcoming leap second correction of UTC.
1	reserved	/
27	xn	Satellite PZ-90 x coordinate at epoch tb.
5	xn_dot_dot	Satellite PZ-90 x velocity at epoch tb.
24	xn_dot	Satellite PZ-90 x acceleration component at epoch tb.
5	n	Slot number (1 to 24).
3	Bn	Healthy flags.
27	yn	Satellite PZ-90 y coordinate at epoch tb.
5	yn_dot_dot	Satellite PZ-90 y acceleration component at epoch tb.
24	yn_dot	Satellite PZ-90 y velocity at epoch tb.
8	age_h	Age of predicted ephemeris (Unit: hours).
27	zn	Satellite PZ-90 z coordinate at epoch tb.
5	zn_dot_dot	Satellite PZ-90 z acceleration component at epoch tb.
24	zn_dot	Satellite PZ-90 z velocity at epoch tb.
8	reserved	Must be 0.
11	gamma_n	Satellite clock frequency drift at epoch tb.
5	E_n	Age of the ephemeris information.
4	freq_id	Frequency ID
12	reserved	/
22	tau_n	Satellite clock correction at epoch tb.
10	reserved	Must be 0.
32	tau_c	GLONASS to UTC(SU) time correction.

22	tau_GPS	GLONASS to GPS system time correction.
10	reserved	/
11	NT	Calendar day number of ephemeris within the four-year period since the beginning of last leap year.
5	N4	Four-year interval number starting from 1996.
12	tk	Satellite time referenced to the beginning of the frame.
4	FT	Predicted satellite user range accuracy at time tb
32	reserved	/
5	m_available	Must be 0x1F.
1	nvm_reliable	Must be 1.
26	spare	/
25	reserved	/
1	available	Contains 1 if ephemeris is available, 0 if not.
1	health	Contains 1 if the satellite is unhealthy, 0 if healthy.
1	reserved	Must be 0.
4	reserved	/

The GLONASS encoding above can be represented in C language with the following structure:

```
typedef struct glonass_ephemris_s
{
    unsigned int week :          16;
    unsigned int toe :           16;
    unsigned int toe_lsb :       4;
    unsigned int NA :            11;
    unsigned int tb :            7;
    unsigned int M :             2;
    unsigned int P1 :            2;
    unsigned int P3 :            1;
    unsigned int P2 :            1;
    unsigned int P4 :            1;
    unsigned int KP :            2;
    unsigned int spare0 :        1;
    unsigned int xn :            27;
    unsigned int xn_dot_dot :    5;
    unsigned int xn_dot :        24;
```

```
    unsigned int n :          5;
    unsigned int Bn :         3;
    unsigned int yn :         27;
    unsigned int yn_dot_dot : 5;
    unsigned int yn_dot :     24;
    unsigned int spare4 :     8;
    unsigned int zn :         27;
    unsigned int zn_dot_dot : 5;
    unsigned int zn_dot :     24;
    unsigned int reserved1 :  2; // must be 0
    unsigned int reserved2 :  6; // must be 0
    unsigned int gamma_n :    11;
    unsigned int E_n :        5;
    unsigned int freq_id :    4;
    unsigned int spare3 :     12;
    unsigned int tau_n :      22;
    unsigned int reserved3 :  10; // must be 0
    unsigned int tau_c :      32;
    unsigned int tau_GPS :    22;
    unsigned int spare5 :     10;
    unsigned int NT :         11;
    unsigned int N4 :         5;
    unsigned int tk :         12;
    unsigned int FT :         4;
    unsigned int spare7 :     32;
    unsigned int m_available : 5;
    unsigned int nvm_reliable : 1;
    unsigned int spare8 :     26;
    unsigned int spare9 :     25;
    unsigned int available :  1;
    unsigned int health :     1;
    unsigned int reserved4 :  1; // must be 0
    unsigned int spare10 :    4;
} glonass_ephemeris_t;
```

Table 5: Encoding for Galileo Ephemeris

Bits	Structure Member	Description
16	week	Week number of the Issue of Data.
14	toe	Time of week for ephemeris epoch.
2	reserved	/

16	toc	Time of week for clock epoch.
10	iod_nav	Issue of data.
8	SISA	Signal In Space Accuracy.
10	reserved	Must be 0.
10	BGD_E1_E5a	E1-E5a Broadcast Group Delay.
10	BGD_E1_E5b	E1-E5b Broadcast Group Delay.
2	E1BHS	E1-B Signal Health Status.
32	inclination	Inclination angle at reference time
32	eccentricity	Eccentricity.
32	root_a	Square root of major axis.
32	mean_anomaly	Mean anomaly at reference time.
32	omega_zero	Longitude of ascending node of orbit plane at weekly epoch.
32	perigee	Argument of perigee.
14	i_dot	Rate of inclination angle.
1	available	Contains 1 if ephemeris is available, 0 if not.
1	health	Contains 1 if the satellite is unhealthy, 0 if healthy.
16	motion_difference	Mean motion difference from computed value.
16	crs	Amplitude of the sine harmonic correction to the orbit radius.
16	crc	Amplitude of the cosine harmonic correction to the orbit radius.
16	cus	Amplitude of the sine harmonic correction to the argument of latitude.
16	cuc	Amplitude of the cosine harmonic correction to the argument of latitude.
16	cis	Amplitude of the sine harmonic correction to the angle of inclination.
16	cic	Amplitude of the cosine harmonic correction to the angle of inclination.
24	omega_dot	Rate of right ascension.
6	SVID	Satellite Identification.
1	E1BDVS	E1-B Data Validity Status.

1	reserved	Must be 0.
8	reserved	Must be 0.
16	reserved	Must be 0.
6	af2	Second order clock correction.
21	af1	First order clock correction.
5	word_available	Must be 0x1F.
31	af0	Constant clock correction.
1	reserved	/
6	reserved	Must be 0.
26	reserved	Reserved for use by GNSS library – must be 1.
1	reserved	Must be 0.

The Galileo encoding above can be represented in C language with the following structure:

```
typedef struct galileo_ephemeris_s
{
    unsigned int week :                 16;
    unsigned int toe :                  14;
    unsigned int reserved1 :            2; // must be 0
    unsigned int toc :                  14;
    unsigned int iod_nav :              10;
    unsigned int SISA :                 8;
    unsigned int reserved2 :            10; // must be 0
    unsigned int BGD_E1_E5a :           10;
    unsigned int BGD_E1_E5b :           10;
    unsigned int E1BHS :                2;
    unsigned int inclination :          32;
    unsigned int eccentricity :         32;
    unsigned int root_a :               32;
    unsigned int mean_anomaly :         32;
    unsigned int omega_zero :           32;
    unsigned int perigee :              32;
    unsigned int i_dot :                14;
    unsigned int available :            1;
    unsigned int health :               1;
    unsigned int motion_difference :    16;
    unsigned int crs :                  16;
```

```
    unsigned int crc :                  16;
    unsigned int cus :                  16;
    unsigned int cuc :                  16;
    unsigned int cis :                  16;
    unsigned int cic :                  16;
    unsigned int omega_dot :            24;
    unsigned int SVID :                 6;
    unsigned int E1BDVS :               1;
    unsigned int reserved3 :            1; // must be 0
    unsigned int af2 :                  6;
    unsigned int af1 :                  21;
    unsigned int word_available :       5;
    unsigned int af0 :                  31;
unsigned int spare0 :                   1;
unsigned int reserved4 :                6; // must be 0
unsigned int spare1 :                   26;
} galileo_ephemeris_t;
```

Table 6: Encoding for BeiDou Ephemeris

Bits	Structure Member	Description
32	inclination	Inclination angle at reference time.
32	eccentricity	Eccentricity.
32	root_a	Square root of major axis.
32	mean_anomaly	Mean anomaly at reference time.
32	omega_zero	Longitude of ascending node of orbit plane at weekly epoch.
32	perigee	Argument of perigee.
17	toe	Time of week for ephemeris epoch.
10	time_group_delay	Estimated group delay differential.
5	aode	Issue of data, ephemeris.
24	omega_dot	Rate of right ascension.
8	A0	Ionospheric Delay Model Parameter α0
24	af0	Constant clock correction.
8	A1	Ionospheric Delay Model Parameter α1

20	sow	Seconds of week
11	af2	Second order clock correction.
1	is_geo	1 for Geostationary satellites, otherwise 0
22	af1	First order clock correction.
10	subframe_avail	Must be 0x3FF.
16	motion_difference	Mean motion difference from computed value.
8	A2	Ionospheric Delay Model Parameter α2
8	A3	Ionospheric Delay Model Parameter α3
18	crs	Amplitude of the sine harmonic correction to the orbit radius.
8	B2	Ionospheric Delay Model Parameter β2.
4	urai	User range accuracy index.
2	reserved	Must be 0.
18	crc	Amplitude of the cosine harmonic correction to the orbit radius.
8	B3	Ionospheric Delay Model Parameter β3
5	aodc	Issue of data, clock
1	spare	/
18	cus	Amplitude of the sine harmonic correction to the argument of latitude.
14	i_dot	Rate of inclination angle.
18	cuc	Amplitude of the cosine harmonic correction to the argument of latitude.
8	B0	Ionospheric Delay Model Parameter β0.
6	spare	/
18	cis	Amplitude of the sine harmonic correction to the angle of inclination.
8	B1	Ionospheric Delay Model Parameter β1
6	reserved	Must be 0.
18	cic	Amplitude of the cosine harmonic correction to the angle of inclination.
1	nvm_reliable	Must be 1.

11	reserved	Must be 0.
2	spare	/
17	toc	Time of week for clock epoch
13	week	Week number of the Issue of Data
1	available	Contains 1 if ephemeris is available, 0 if not.
1	health	Contains 1 if the satellite is unhealthy, 0 if healthy.

The BeiDou encoding above can be represented in C language with the following structure:

```
typedef struct beidou_ephemeris_s
{
    unsigned int inclination :          32;
    unsigned int eccentricity :         32;
    unsigned int root_a :               32;
    unsigned int mean_anomaly :         32;
    unsigned int omega_zero :           32;
    unsigned int perigee :              32;
    unsigned int toe :                  17;
    unsigned int time_group_delay :     10;
    unsigned int aode :                 5;
    unsigned int omega_dot :            24;
    unsigned int A0 :                   8;
    unsigned int af0 :                  24;
    unsigned int A1 :                   8;
    unsigned int sow :                  20;
    unsigned int af2 :                  11;
    unsigned int is_geo :               1;
    unsigned int af1 :                  22;
    unsigned int subframe_avail :       10;
    unsigned int motion_difference :    16;
    unsigned int A2 :                   8;
    unsigned int A3 :                   8;
    unsigned int crs :                  18;
    unsigned int B2 :                   8;
    unsigned int urai :                 4;
    unsigned int reserved1 :            2;
    unsigned int crc :                  18;
    unsigned int B3 :                   8;
    unsigned int aodc :                 5;
    unsigned int spare0 :               1;
```

```
    unsigned int cus :              18;
    unsigned int i_dot :            14;
    unsigned int cuc :              18;
    unsigned int B0 :               8;
    unsigned int spare1 :           6;
    unsigned int cis :              18;
    unsigned int B1 :               8;
    unsigned int reserved2 :        6;
    unsigned int cic :              18;
    unsigned int nvm_reliable :     1;
    unsigned int reserved3 :        1;
    unsigned int reserved4 :        10;
    unsigned int spare4 :           2;
    unsigned int toc :              17;
    unsigned int week :             13;
    unsigned int available :        1;
    unsigned int health :           1;
} beidou_ephemeris_t;
```

NOTES

1. A little-endian architecture has been assumed. For example, in the case of GPS PRN #1 ephemeris where week number is 1831 (= 0x0727) and toe is 15750 (= 0x3D86), the NMEA command string will begin with **$PSTMEPHEM,1,64,*2707863d*…**.
2. For the scale of data in the table for each constellation, please refer to respective ICDs.
3. The **$PSTMDUMPEPHEMS** command can be used to verify if an ephemeris, which has been injected using the **$PSTMEPHEM** command, has been correctly sent.

Results:

- The ephemeris data string is decoded and stored into the backup memory.
- If more than one ephemeris is uploaded for each satellite, the oldest ephemeris will be overwritten.
- No message will be sent as reply.
- If more than one **$PSTMEPHEM** command needs to be issued between two consecutive commands, there must be at least a 20-millisecond delay.

C Language Sample Code:

The following example shows how to use the above structures. The example below is given for GPS ephemeris, but it also applies to GLOANSS, Galileo and BeiDou with very few changes.

```
void convert_gps_eph(int sat_id, gps_ephemeris_t *ephem_ptr, char nmea_output[512])
{
    unsigned char* eph_ptr = (unsigned char*)(ephem_ptr);
    int index, index2;
```

```
    index = sprintf((char *)nmea_output,"$PSTMEPHEM,%i,%i,", sat_id, sizeof(gps_ephemeris_t));
    for (index2 = 0; index2 < sizeof(gps_ephemeris_t); index2++)
    {
        index += sprintf((char *)(nmea_output + index),"%02x",eph_ptr[index2]);
    }
}
```

NOTE

The **$PSTMDUMPEPHEMS** command can be used to verify if an ephemeris, which has been injected using the **$PSTMEPHEM** command, has been correctly sent.

3.2.4. $PSTMALMANAC

Load almanacs data. This command allows the user to load the almanacs data into the backup memory.

Synopsis:

```
$PSTMALMANAC,<sat_id>,<N>,<byte1>,…..,<byteN>*<checksum><CR><LF>
```

Arguments:

Parameter	Format	Description
sat_id	Decimal, 2 digits	Satellite number
N	Decimal, 1 digit	Number of the almanac data bytes
byte1	Hexadecimal, 2 digits	First byte of the almanac data
byteN	Hexadecimal, 2 digits	Last byte of the almanac data
checksum	Hexadecimal, 2 digits	Checksum of the message bytes between but not including the **$** and ***** characters

The N Bytes that are in the parameters are the dump of a structure that contains all the information of the almanac. Data formats are constellation-dependent, which are listed in the following tables.

Table 7: Encoding for GPS Almanac

Bits	Structure Member	Description
8	satid	The satellite number.
16	week	The week number for the epoch

8	toa	Reference time almanac.
16	e	Eccentricity.
16	delta_i	Rate of inclination angle.
16	omega_dot	Rate of right ascension.
24	root_A	Square root of semi-major axis.
24	omega_zero	Longitude of ascending node of orbit plane at weekly epoch.
24	perigee	Argument of perigee.
24	mean_anomaly	Mean anomaly at reference time.
11	af0	Constant clock correction.
11	af1	First order clock correction.
1	health	Contains 1 if the satellite is unhealthy 0 if healthy.
1	available	Contains 1 if almanac is available 0 if not.

The GPS encoding above can be represented in C language with the following structure:

```
typedef struct gps_almanac_s
{
unsigned int satid : 8;
unsigned int week : 16;
unsigned int toa : 8;
unsigned int eccentricity : 16;
unsigned int delta_i : 16;
unsigned int omega_dot : 16;
unsigned int root_a : 24;
unsigned int omega_zero : 24;
unsigned int perigee : 24;
unsigned int mean_anomaly : 24;
unsigned int af0 : 11;
unsigned int af1 : 11;
unsigned int health : 1;
unsigned int available : 1;
unsigned int spare0 : 32;
unsigned int spare1 : 32;
} gps_almanac_t;
```

Table 8: Encoding for GLONASS Almanac

Bits	Structure Member	Description
8	satid	The satellite number.
16	week	The week number for the epoch
8	toa	Reference time almanac.
5	n_A	Slot number (1 to 24).
5	H_n_A	Carrier frequency channel number.
2	M_n_A	Type of satellite 00=GLONASS 01=GLONASS-M.
10	tau_n_A	Satellite clock correction.
15	epsilon_n_A	Eccentricity.
21	t_lambda_n_A	Time of the first ascending node passage.
21	lambda_n_A	Longitude of ascending node of orbit plane at almanac epoch.
18	delta_i_n_A	Inclination angle correction to nominal value.
7	delta_T_n_dot_A	Draconian period rate of change.
22	delta_T_n_A	Draconian period correction.
16	omega_n_A	Argument of perigee.
1	health	Contains 1 if the satellite is unhealthy 0 if healthy.
1	available	Contains 1 if almanac is available 0 if not.
32	Tau_c	/
11	NA	/
5	N4	/
16	Spare	/

The GLONASS encoding above can be represented in C language with the following structure:

```
typedef struct glonass_almanac_s
{
unsigned int satid : 8;
unsigned int week : 16;
unsigned int toa : 20;
```

```
unsigned int n_A : 5;
unsigned int H_n_A : 5;
unsigned int M_n_A : 2;
unsigned int tau_n_A : 10;
unsigned int epsilon_n_A : 15;
unsigned int t_lambda_n_A : 21;
unsigned int lambda_n_A : 21;
unsigned int delta_i_n_A : 18;
unsigned int delta_T_n_dot_A : 7;
unsigned int delta_T_n_A : 22;
unsigned int omega_n_A : 16;
unsigned int health : 1;
unsigned int available : 1;
unsigned int tau_c : 32;
unsigned int NA : 11;
unsigned int N4 : 5;
unsigned int spare0 : 16;
} glonass_almanac_t;
```

Table 9: Encoding for Galileo Almanac

Bits	Structure Member	Description
16	satid	The satellite number.
6	svid	Space Vehicle Identification.
16	week	The week number for the epoch
20	toa	Reference time almanac.
13	delta_a	Delta of semi-major axis.
11	e	Eccentricity.
16	perigee	Argument of perigee.
11	delta_i	Rate of inclination angle.
16	omega_zero	Longitude of ascending node of orbit plane at weekly epoch.
11	omega_dot	Rate of right ascension.
16	mean_anomaly	Mean anomaly at reference time.
16	af0	Constant clock correction.

13	af1	First order clock correction.
2	E5b_HS	E5 Signal Health Status
2	E1B_HS	E1-B Signal Health Status
4	ioda_1	Issue of data Almanac 1
4	ioda_2	Issue of data Almanac 2
1	health	Contains 1 if the satellite is unhealthy 0 if healthy.
2	reserved	Reserved for use by GNSS library.
1	health	Contains 1 if the satellite is unhealthy, 0 if healthy
1	available	Contains 1 if almanac is available 0 if not.

The Galileo encoding above can be represented in C language with the following structure:

```
typedef struct galileo_almanac_s
{
unsigned int satid : 16;
unsigned int svid : 6;
unsigned int week : 16;
unsigned int toa : 20;
unsigned int delta_a : 13;
unsigned int eccentricity : 11;
unsigned int perigee : 16;
unsigned int delta_i : 11;
unsigned int omega_zero : 16;
unsigned int omega_dot : 11;
unsigned int mean_anomaly : 16;
unsigned int af0 : 16;
unsigned int af1 : 13;
unsigned int E5b_HS : 2;
unsigned int E1B_HS : 2;
unsigned int ioda_1 : 4;
unsigned int ioda_2 : 4;
unsigned int word_available : 2;
unsigned int health : 1;
unsigned int available : 1;
unsigned int spare0 : 2;
unsigned int spare1 : 32;
unsigned int spare2 : 32;
} galileo_almanac_t;
```

Table 10: Encoding for BeiDou Almanac

Bits	Structure Member	Description
8	satid	The satellite number.
16	week	The week number for the epoch.
8	toa	Reference time almanac.
17	eccentricity	Eccentricity.
11	af0	Constant clock correction.
1	is_geo	Contains 1 if it is a geostationary satellite.
3	spare0	Spare.
17	omega_dot	Rate of right ascension.
11	af1	First order clock correction.
4	spare1	Spare.
24	root_a	Square root of semi-major axis.
24	omega_zero	Longitude of ascending node of orbit plane at weekly epoch.
24	perigee	Argument of perigee.
24	mean_anomaly	Mean anomaly at reference time.
16	delta_i	Rate of inclination angle.
1	health	Contains 1 if the satellite is unhealthy 0 if healthy.
1	available	Contains 1 if almanac is available 0 if not.
14	spare2	Spare.

The BeiDou encoding above can be represented in C language with the following structure:

```
typedef struct compass_almanac_s
{
unsigned int satid : 8;
unsigned int week : 16;
unsigned int toa : 8;
unsigned int eccentricity : 17;
unsigned int af0 : 11;
unsigned int is_geo : 1;
```

```
unsigned int spare0 : 3;
unsigned int omega_dot : 17;
unsigned int af1 : 11;
unsigned int spare1 : 4;
unsigned int root_a : 24;
unsigned int omega_zero : 24;
unsigned int perigee : 24;
unsigned int mean_anomaly : 24;
unsigned int delta_i : 16;
unsigned int health : 1;
unsigned int available : 1;
unsigned int spare2 : 14;
} compass_almanac_t;
```

NOTES

1. A little-endian architecture has been assumed.
2. For the scale of data in the table for each constellation, please refer to respective ICDs.
3. The **$PSTMDUMPALMANAC** command can be used to verify if an almanac, which has been injected using the **$PSTMALMANAC** command, has been correctly sent.

Results:

- The almanac will be stored into backup memory.
- No message will be sent as reply.

Example:

```
$PSTMALMANAC,12,32,0c1a06907c1a971160fd0800fa0da141ae9f0600d912e90075669700490f8000
*75
```

3.3. Real-Time AGNSS Testing Procedure

The main feature introduced by a GNSS device with Real-Time AGNSS capability is the ephemeris availability (and also GNSS fix availability) as soon as the device downloads the assistance data from the network, even if the device is used in critical environments where the ephemeris downloading from the sky is difficult (e.g. urban canyon, low signal strengths, etc.).

The procedures for both COLD and WARM start tests are reported in this section.

3.3.1. Simple COLD Start Test

By the standard NMEA command **$PSTMCOLD**, the device is restarted in COLD start conditions, then the assistance data will be downloaded from the real time server and sent to the device. The procedure is described below:

- Assistance data are downloaded from the server.
- The current time is injected into the device by the **$PSTMINITTIME** command.
- In the case that the approximate position is also available, both current time and the position can be injected into the device by the **$PSTMINITGPS** command instead.
- Then, the ephemeris data (downloaded from the real time server) are injected into the device by using **$PSTMEPHEM** command for each satellite.
- Then, the almanacs data (downloaded from the real time server) are injected into the device by using **$PSTMALMANAC** command for each satellite.

Now the time to first fix (TTFF) and the position accuracy are evaluated - the first fix should be achieved shortly after the injection of the ephemeris data.

This test can be performed several times in order to get a good statistical result.

3.3.2. Simple WARM Start Test

By the standard NMEA command **$PSTMWARM**, the device is restarted in WARM start conditions, then ephemeris data will be downloaded from the real time server and injected into the device by using the **$PSTMEPHEM** command (in the WARM start condition, the device is supposed to have already downloaded the time and the almanacs from the sky).

Now the TTFF and the position accuracy are evaluated - the first fix should be achieved shortly after the injection of the ephemeris data.

This test can be performed several times in order to have a good statistical result.

4 Appendix A References

Table 11: Related Documents

SN	Document Name	Remark
[1]	Quectel_L26-DR_GNSS_Protocol_Specification	L26-DR GNSS protocol specification
[2]	Quectel_L26-T&L26-P_GNSS_Protocol_Specification	GNSS protocol specification of L26-T and L26-P modules
[3]	Quectel_L89_GNSS_Protocol_Specification	L89 GNSS Protocol Specification

Table 12: Terms and Abbreviations

Abbreviation	Description
AGNSS	Assisted GNSS
GNSS	Global Navigation Satellite System.
GPS	Global Positioning System
ICD	Interface Control Document
LSB	Least Significant Bit
NMEA	National Marine Electronics Association
PRN	Pseudorandom Noise
TTFF	Time To First Fix
UTC	Universal Time Coordinated