UM3397

User manual

Teseo VI and Teseo APP2–NMEA observables description

Introduction

This document describes the raw GNSS measurements produced by STMicroelectronics multiband Teseo VI and Teseo APP2 GNSS receivers and the related NMEA proprietary messages.

1 Overview

Teseo VI and Teseo APP2 devices embed the Arm® Cortex®-M7.

For information on the Arm® Cortex®-M7, refer to the technical reference manuals, available from the www.arm.com website.

Note: Arm, Cortex, and the Arm logo are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

arm

1.1 Gathering data from NMEA

STMicroelectronics GNSS receivers can output all the data needed to feed generic PVT navigation algorithms.

These involve:

- Raw DSP measurements signal status indicators
- SVs position
- Velocity and corrections, related timestamps
- User position and velocity estimation

These data are encapsulated in specific NMEA proprietary messages ($PSTMTG and $PSTMTS messages), which are issued from NMEA port of the receiver.

This document describes the extended versions of the $PSTMTG/$PSTMTS sentences, which contain the information needed to be converted into a RINEX format.

2 NMEA specification

2.1 Overview

STMicroelectronics proprietary NMEA $PSTMTG/$PSTMTS formats implement the following requirements:

- Keep observable in accordance with well-established RINEX conventions
- $PSTMTS sorted to make multifrequency processing more efficient
- Split ionosphere and troposphere contributions in two separate fields
- Time group delay (TGD), user range accuracy (URA), leap-seconds
- Reserved fields for SSR corrections
- Added cycle-slip flag for more efficient slip detection

To provide RINEX compatible observables, the measurement epochs are propagated to the GPS TOW second boundary. The new observables replace the ones of the previous versions.

The front-end identifier added to the extended TG sentence allows this frequency to be corrected for the nominal residual clock drift, recovering the true satellite doppler.

Table 1. $PSTMTG/$PSTMTS fields description

Ext. Field	Description	TG/TS
ver	TG/TS version number and measurement clock steering indicator	$PSTMTG
Front-end	Analog front-end identifier	$PSTMTG
cp	Carrier phase	$PSTMTS
flags	Multipath and signal quality indicators	$PSTMTS

Field config_status (4x hexadecimal digits) is extended as follows:

- Bit #15:12 – Version number of the $PSTMTG/$PSTMTS sentences. Legacy messages have all bits set to zero. Extended messages report a progressive version number, starting from 1. Bit 15 is reserved to indicate clock steering (1 = steering, 0 = no), such that the actual available version numbers are in the range 0:7.
- Bit #11:8 – Analog front-end identifier. Legacy messages have this bit all set to zero. Extended messages report a number indicating the type of analog front-end, which was installed at the time of logging the NMEA stream. The following identifiers are defined.

Table 2. $PSTMTG/$PSTMTS front-end identifiers

ID	Description
0	Legacy front-end with 26 MHz TCXO
1	Legacy front-end with 48 MHz TCXO
15	Unknown or not specified.

2.1.1 Carrier phase

The carrier phase observable cp [cycle] is available in the $PSTMTS sentence. For a comprehensive description of the $PSTMTS sentence, refer to Section 2.5: $PSTMTS. When available, carrier-phase is reported in the $PSTMTS field, immediately following the frequency field. Carrier phase is expressed in cycles including a fractional part. Half cycle ambiguity information is reported in TS flags. The cp time derivative is consistent with doppler plus nominal IF clock drift bias plus residual clock drift (legacy $PSTMTG/$PSTMTS).

The time derivative of the carrier phase is consistent with doppler plus residual clock drift (that is, the IF clock drift should not appear anymore).

Table 3. **Carrier phase field position in the $PSTMTS sentence**

Field	Symbol	Description	Unit
46417.43	f_m	Frequency	Hz
12345.678	Φ	Carrier phase	Cycle

2.1.2 Pseudorange

The pseudoranges are modulus 200 ms with time derivative consistent with doppler plus residual clock drift.

2.1.3 Flags

Field Plf (preamble locked) in $PSTMTS is extended with additional flag bits, to indicate the level of estimated multipath error and other signal quality indicators. The bit encoding details of the flags are the following.

Table 4. **$PSTMTS flags encoding**

Bit	Description
0	1 = available
1	Preamble locked 1 = locked
3:2	Multipath indicator: • 0 = no • 3 = strong
4	Loss of lock indicator
6:5	Reserved
7	Preamble polarity: • 0 = normal; • 1 = reversed
8	Half-cycle ambiguity • 1 = not fixed • 0 = fixed
13:9	Reserved
14	• 1 = main freq. (L1) • 0 = dual freq. (L2, L5, etc.)
15	Cycle slip indicator (at current epoch vs. previous one) 1 = occurred
16	ddm_ext_validity
17	psr_int
18	cp_int
19	drop_detection_alarm
20	code_alarm [0 false, 1 high]
21	preamble_propagated
24:22	Code_alarm_intensity[0 min, 7 max]. Valid when the code alarm is high.
25	Spoofing: • 0 = no spoofing • 1 = possible spoofing

2.2 $PSTMTG

Time and global information.

NMEA message list bitmask (64 bits): 0000 0000 0000 0100.

Synopsis:

```
 $PSTMTG,<Week>,<TOW>,<Reserved>,<CPUTime>,<TimeBestValidity>,<NCO>,<config_status>,<constell
ation_mask>, <time_best_sat_type>,<time_master_sat_type>,<time_master_week_n>,  <time_master_
tow>,<time_master_validity>, <TG-aux-flags>,<clock-bias>,<mfreq_const_mask>, <leap-sec>,<pps_
edge>,<mtb_ms>,<mtb_timestamp>*<checksum><cr><lf>
```

Table 5. PSTMTG field descriptions

Parameter	Format	Description
Week	Decimal, 4 digits	Week number (system time, best estimate)
TOW	Decimal, 12 digits	Time of week (system time, best estimate)
Reserved	Decimal, 1 digit	
CPU-Time	Decimal, 10 digits	CPU time
TimeBestValidity	Decimal, 2 digits	• 0 = no time. The system has no sense of time. No RTC, GNSS, or stored (flash) time is available. • 1 = time read from flash The current time was read from a flash. It could be very old and it is essentially the same as no time. • 2 = TOW time (week number not available) The time of the week has been read from the navigation data, but not the week number (WN). • 3 = time set user Time was set by the USER through a command. • 4 = time set user RTC The RTC time was set by the user (this is depreciated). • 5 = RTC time The time difference from the last saved fix is higher than 1 hour. • 6 = RTC time accurate The time difference from the last saved fix is lower than or equal to 1 hour. • 7 = time approximate A time has been downloaded from the constellation, but the last ephemeris time is > 300 s old. • 8 = time accurate Either POSITION_TIME or EPHEMERIS_TIME (an accurate time). Accurate time is a status that is stored when the GNSS engine is restarted using a warm start command and the system had a time validity of 9 or 10. Anytime validity that is >= 8 can be considered an accurate time. • 9 = position time The time has been corrected using the position (an accurate time). • 10 = ephemeris time. Time set from downloaded ephemeris tow. Time is accurate (ms), but less than validity 9.
NCO	Floating	Estimated receiver clock drift [Hz]. If clock steering is enabled this value shall be used in the Doppler calculation (instead of the nominal values).
config_status	Hexadecimal, 4 digits	**Byte #0: Kalman filter configuration** For each bit: • 0 means feature disabled • 1 means feature enabled **Bit / Description** Bit #0 = Walking mode ON Bit #1 = Stop Detection ON

Parameter	Format	Description
		Bit #2 = Frequency Ramp On Bit #3 = Velocity estimator model • 1 means MULTIPLE MODEL • 0 means SINGLE MODEL Bit #4 = Velocity estimator filter: • 1 means SLOW • 0 means FAST Bit #5 = FDE Status ON Bit #6 = TCXO jump • 1 tcxo jump applied • 0 tcxo jump not applied Bit #7 = IFLC auto mode • 1 IFLC auto mode • 0 Standard mode **Byte #1: Global configuration** **Bit / Description** Bit #3:0 Front end frequency • 1 means 48 MHz • 0 means 26 MHz Bit #6:4 = $PSTMTG, $PSTMTS version Bit #7 = clock steering indicator • 1 is steered • 0 is not steered
constellation_mask	Decimal	Refer to Teseo VI and Teseo APP2–Firmware configuration to get the full description of the bit mask.
time_best_sat_type	Decimal	Selected best time satellite type
time_master_sat_type	Decimal	Master time satellite type
time_master_week_n	Decimal	Master time week number
time_master_tow	Floating	Master time TOW
time_master_validity	Decimal	Master week number time validity. It can be different from time best validity.
TG_flags_aux	Unsigned decimal (32-bit)	**Byte #0: Additional flags** **Bit/Description** 0:TCXO jump detected 1:Earth rotation Corr. in sat pos: • 0 means legacy • 1 means w/o earth rotation 2 means spectral inversion: • 0 = legacy (BeiDou inverted) • 1 = spectral inversion internally compensated 3:Clock bias validity: • 0 means not valid (legacy); • 1 means valid 4:Indicates presence of pps_edge field 5: Indicates presence of sat data in TS • 1 means sat data not present; • 0 means sat data present 6: Indicates presence of MTB timestamp • 1 means MTB timestamp present; • 0 means MTB timestamp not present

Parameter	Format	Description
		7: OSCI32 OK status • 0 means not OK • 1 means OSCI32 output is OK 8: week_valid: • 0 means WN has not been downloaded from the sky • 1 means WN has been downloaded from the sky 9:31: reserved
clock_bias_mt	Floating	Estimated receiver clock bias [meter]
mfreq_const_mask	Decimal	Multifrequency configuration mask.
leap_sec	Signed decimal (8 bits)	Leap seconds (0 = unknown, -1 not valid)
pps_edge	Unsigned decimal (32-bit)	PPS edge counter at 64F0 clock resolution
mtb_ms	Signed decimal (32-bit)	Master time-base. Number of ms ($1*10^{-3}$ s) since time reference.
mtb_timestamp	Signed decimal (32-bit)	Master time-base. Fixed-point fractional precision in ps ($1*10^{-12}$ s)
checksum	Hexadecimal, 2 digits	NMEA message checksum

Example

```
 $PSTMTG,2122,228200.99999997,0,30470117,1,-47402.1515,a000,33423,0,0,2122,228200.99999997,1,
118,0.000,135,18,1370036460,21905,272472106*48
```

2.3 $PSTMCHMON

This message reports the correlation for a given satellite.

NMEA message list bitmask (64 bits): 0000 0000 0000 0200.

Synopsis:

```
$PSTMCHMON,<pool>,<sat_id>,<cpu_time1>,<cpu_time2>,<P1_VL>,<P1_L>,<P1_P>,<P1_E>,<P1_VE>,<P2_V
L>,<P2_L>,<P2_P>,<P2_E>,<P2_VE>*<checksum><cr><lf> or $PSTMCHMON,<pool>,<sat_id>,<cpu_time1>,
<cpu_time2>,<P1_VL>,<P1_L>,<P1_P>,<P1_E>,<P1_VE>,<P2_VL>,<P2_L>,<P2_P>,<P2_E>,<P2_VE>,<acm>*<
checksum><cr><lf>
```

Table 6. PSTMCHMON field descriptions

Parameter	Format	Description
pool	Decimal, 2 digits	Pool number being used
sat_id	Decimal, 2 digits	Satellite identifier
Cpu_time1,2	Decimal, 10 digits	The cpu tick count at the epoch when the correlation values have been collected from the baseband hardware (for debug purposes)
Corr_data (<P1_VL> to <P2_VE>)	Decimal, 10 digits	The power (in arbitrary units) of the very late correlation point for the channel nr. 1 etc. for the late, prompt, early, very early and for the remaining channels. Each channel can dump up to 5 points, so in this example 2 channels dump in total 10 correlation points.
acm	Floating	autocorrelation metric (parameter added when 1 or more pools are enabled in CDB Page 16 Line 2 Field 5)
checksum	Hexadecimal, 2 digits	NMEA message checksum

2.4 $PSTMSAT

This message is repeated for each satellite tracked and contains satellite iono and differential data.
NMEA message list bitmask (64 bits): 0000 0000 0000 0800.

Synopsis:

```
$PSTMSAT,<SatID>,<Satdat>,<Satx>,<Saty>,<Satz>,<Velx>,<Vely>,<Velz>,<Src>,<tgd>,<iono>,<tropo>,<rrc>,<dcb5>,<Difdat>,<Drc>,<Drrc>,<dC>,<dX>,<dY>,<dZ>,<predavl>,<predage>,<predeph>,<predtd>,<dcb2>,<GalRed>,<ssr1>,<ssr2>,<ssr3>,<ssr4>,<ssr5>,<ssr6>,<ssr7>,<diff1>,<diff2>,<diff3>,<diff4>,<diff5>,<diff6>,<diff7>,<dcb1c>*<checksum><cr><lf>
```

Table 7. PSTMSAT field descriptions

Parameter	Format	Description
Sat-ID	Decimal, up to 2 digits	Satellite identifier
Satdat	Decimal, 1 digit	Satellite data is available flag bit 0: • 0 = Sat. Ephemeris is not available or unhealthy sat. • 1 = Sat. Ephemeris available and healthy satellite
Satx	Floating	X ECEF Sat. Position at TX time
Saty	Floating	Y ECEF Sat. Position at TX time
Satz	Floating	Z ECEF Sat. Position at TX time
Velx	Floating	X ECEF satellite velocity
Vely	Floating	Y ECEF satellite velocity
Velz	Floating	Z ECEF satellite velocity
Src	Floating	Satellite range correction (incl. TGD)
tgd	Floating	Broadcast time group delay
iono	Floating	Ionospheric correction
tropo	Floating	Tropospheric correction
rrc	Floating	Range rate correction
dcb5	Floating	DCB for band 5
Difdat	Decimal, 1 digit	Differential data is available flag. This is a bit mask with more values bit set to 0 = not available, bit to 1 = available • Bit 0 = RTCM differential correction status • Bit 1 = IONO differential correction status • Bit 2 = SLOW differential correction status • Bit 3 = FAST differential correction status • Bit 4 = SLAS differential correction status • Bit 5 = SSR differential correction *Note: That also means that fields are not exclusive between OSSR and SSR.*
Drc	Floating	Differential Range Corr. (w/o tropo) [m]
Drrc	Floating	Differential range rate correction [m/s]
dC	Floating	SSR: clock correction
dX	Floating	SSR: ephemeris ECEF dX corrections
dY	Floating	SSR: ephemeris ECEF dY corrections
dZ	Floating	SSR: ephemeris ECEF dZ corrections
predavl	Decimal, 1 digit	Prediction available flag:

Parameter	Format	Description
		• 0 = Predicted ephemeris not available • 1 = Predicted ephemeris available
predage	Decimal, 3 digits	Age of predicted ephemeris (in hours)
predeph	Decimal, 1 digit	Number of satellites used for prediction (1 or 2)
predtd	Decimal, 2 digits	Time distance of ephemeris (in hours) calculated from 2 sats. Only valid if <predeph> = 2
dcb2	Floating	DCB for band 2
GalRed	Decimal, 1 digit	Galileo reduced ephemeris flag: • 0 = normal or predicted Galileo ephemeris • 1=Galileo reduced ephemeris (not predicted and not normal)
ssr1	Decimal, 3 digits	SSR: tracking modes of bias param 1
ssr2	Decimal, 3 digits	SSR: tracking modes of bias param 2
ssr3	Decimal, 3 digits	SSR: tracking modes of bias param 3
ssr4	Floating	SSR: reserved for code bias param 1
ssr5	Floating	SSR: code bias param 2
ssr6	Floating	SSR: code bias param 3
ssr7	Decimal, 2 digits	SSR: valid bit mask
diff1	Decimal, 2 digits	Differential data is available flag This is extended to a 2 parts fields. Bit 0 - 5 : bit mask for correction information MSB part / bit 8 - 10 : value for SSR source information MSB value : (mask 0xF0) • 0 = No SSR source • 1 = SSR differential correction available from E6HAS • 2 = SSR differential correction available from B2B_PPP • 3 = SSR differential correction available from RTCM3 injection • Bit 0 = dgps correction valid • Bit 1 = iono correction valid • Bit 2 = slow correction valid • Bit 3 = fast correction valid • Bit 4 = slas correction valid • Bit 4 = SSR correction valid
diff2	Floating	Differential range correction fast
diff3	Floating	Differential range correction slow orbit
diff4	Floating	Differential range correction slow clock
diff5	Floating	Differential range correction ionospheric
diff6	Floating	Differential range correction slas
diff7	Floating	Differential range correction dgps
dcb1c	Floating	DCB for band 1C
ssr8	Decimal, 3 digits	SSR: tracking modes of bias param 4
ssr9	Decimal, 3 digits	SSR: tracking modes of bias param 5
ssr10	Floating	SSR: code bias param 4

Parameter	Format	Description
ssr11	Floating	SSR: code bias param 5
checksum	Hexadecimal, 2 digits	NMEA message checksum

Note: <predxxx> fields are only valid if the AGPS software module has been included.

2.5 $PSTMTS

This message reports the satellite observables data.
NMEA message list bitmask (64 bits): 0000 0000 0000 0200.

Synopsis:

```
$PSTMTS,<dspdat>,<SatID>,<PsR>,<Freq>,<cp>,<flags>,<CN0>,<ttim>,<codeNoise>,<phaseNoise>,<cycle_slip_cnt>,<Gslot>,<elevation_deg>*<checksum><cr><lf>
```

Table 8. PSTMTS filed descriptions

Parameter	Format	Description
dsp-dat	Decimal, 1 digit	DSP data available • Bit #1 = main band • Bit #0 = the second band follows • Bit #2 = the third band follows • Bit #3 = the fourth band follows • Bit #4 = fifth band follows
Sat-ID	Decimal, 2 to 3 digits	Satellite identifier
PsR	Floating	Pseudorange (meters)
Freq	Floating	Satellite tracking frequency offset
cp	Floating	Carrier phase measurement (cycles)
dsp_flags	Decimal	**Bit/Description** Bit #0 • 1 means available Bit #1 = preamble locked • 1 means locked Bit #3:2 = multipath indicator • 0 means no • 3 means strong Bit #4 = loss of lock indicator Bit #6:5 = reserved Bit #7 = preamble polarity: • 0 means normal • 1 means reversed Bit #8 = half-cycle ambiguity • 1 means not fixed • 0 means fixed Bit #13:9 = reserved Bit #14 = indicates frequencies plan • 1 means main freq. (L1); • 0 means dual freq. (L2, L5, etc.) Bit #15 = cycle slip indicator (at current epoch vs. previous one): • 1 means occurred Bit #16 = ddm_ext_validity

Parameter	Format	Description
		Bit #17 = psr_int Bit #18 = cp_int Bit #19 = drop_detection_alarm Bit #20 = code alarm Bit #21 = preamble_propagated Bit #24:22 = code alarm value Bit #25 = spoofing Bit #63:26 = reserved
CN0	Floating	Satellite carrier to noise ratio (in dBHz)
Ttim	Decimal	Track time of satellite (in seconds)
codeNoise	Decimal	Moving average of the code-loop discriminator error in arbitrary units. Typical abs <2000.
phaseNoise	Decimal	Moving average of the error used to update the carrier loop in arbitrary units. Typical range 1-10k
cycle_slip_cnt	Decimal	Total cycle slip counter
GLONASS slot	Decimal, 1-2 digits	GLONASS satellite slot number (1 to 24), if available; otherwise 0;
elevation_deg	Signed decimal (8 bits)	Elevation degree
checksum	Hexadecimal, 2 digits	NMEA message checksum

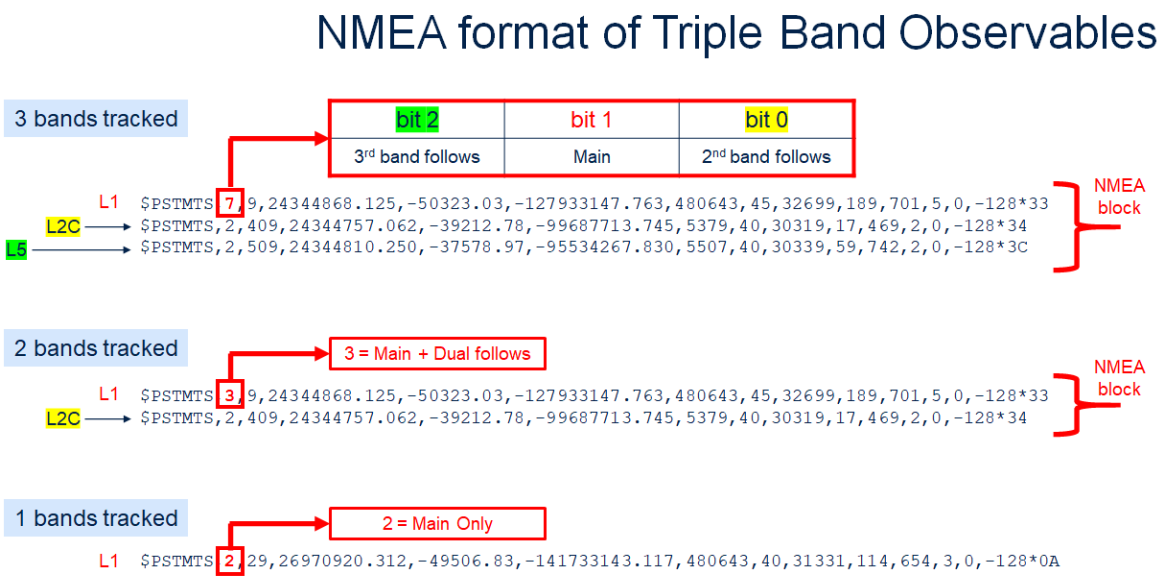
Example

```
$PSTMTS,3,5,23014252.000,-50964.86,-120940740.465,414863,25,735349,-328,4909,42,0,28*3A
```

```
$PSTMTS,2,405,23014149.625,-39710.89,-94239106.008,398727,25,555743,-175,11004,567,0,-128*18
```

Dsp-dat values examples

Figure 1. Dsp-dat values



Doppler preprocessing

The actual doppler measure (D^*) can be calculated from the *freq* field of the $PSTMTS message and from the *front-end frequency* bit in the *config_status* field of the $PSTMTG message in the following way:

$$D^* = Freq - clock_{GPS}$$

Where $clock_{GPS}$ can be obtained from Table 9. F.E. frequency bit meaning.

Table 9. F.E. frequency bit meaning

Front-end frequency bit	$clock_{GPS}$
0 (= 26 MHz)	-47122.395833492279 Hz
1 (= 48 MHz)	-40526.315789699554 Hz

When clock steering is active (bit 3 in the version field of the TG sentence set to '1'), then the rate of the fundamental time frame used to trigger the measurement epochs is adjusted (steered), such that no millisecond jump happens in the observables. This rate is reported in the NCO field of the $PSTMTG sentence and can be used as an accurate value of $clock_{GPS}$.

Note: BeiDou signals received by the integrated front-end are with an inverted spectrum. For this reason, the carrier phase and doppler must be sign-inverted. The formulas to compute the doppler and the corrected carrier phase for a BeiDou satellite shall be modified as follows (B1 = 1561.098 MHz, L1 = 1575.42 MHz):

$$D^* = -(Freq + clock_{GPS} * B1/L1)$$

GLONASS Sat-iD

The index for GLONASS frequency channels is a function of the SV identifier *Sat-iD* (which is reported in the measurement $PSTMTS message). It can be determined using the following lookup table.

Table 10. GLONASS sat ID vs. frequency channel ID (K) association

K	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
id	65	66	67	68	69	70	71	72	73	74	75	76	77	78
id	79	80	81	82	83	84	85	86	87	88	89	90	91	92

BeiDou sat-ID

When the satellite ID reported on $PSTMTS message is included in the range 141 to 177 or 978 to 1003, then that sentence refers to a BeiDou satellite. In that case the BeiDou PRN numbers can be determined by subtracting 140 or 940 to the satellite ID got from the $PSTMTS sentence.

- For BeiDou B2A satellites, 850 is added to Sat-iD, Sat-iD numbers are from 851 to 913.
- For BeiDou B1C satellites, 1540 is added to Sat-iD, Sat-iD numbers are from 1541 to 1603.
- For BeiDou B2i satellites, 540 is added to Sat-iD, Sat-iD numbers are from 541 to 599.
- For BeiDou B2b satellites, 1140 is added to Sat-iD, Sat-iD numbers are from 1141 to 1203.
- For BeiDou B3i satellites, 1040 is added to Sat-iD, Sat-iD numbers are from 1041 to 1103.

GPS L1 C/A Sat-iD

For GPS L1 C/A satellites, the Sat-iD to specify that the measurements refer to L1 C/A band for that satellite (for example, if GPS PRN = 16 then L1 C/A data is reported as Sat-iD = 16 in the $PSTMTS messages for that satellite).

GPS L2C and GLONASS L2 Sat-iD

For GPS and GLONASS L2 satellites, 400 is added to the Sat-iD to specify that the measurements refer to L2 band for that satellite (for example, if GPS PRN = 16 then L2 data is reported as Sat-iD = 416 in the $PSTMTS messages for that satellite; or, if GLONASS Sat-iD = 73 then L2 data is reported as Sat-iD = 473 for that satellite).

GPS L5 Sat-iD

For GPS L5 satellites, 500 is added to the Sat-iD to specify that the measurements refer to the L5 band for that satellite (for example, if GPS PRN = 16 then L5 data is reported as Sat-iD = 516 in the $PSTMTS messages for that satellite).

GPS L1C Sat-iD

For GPS L1C satellites, 1300 is added to the Sat-iD to specify that the measurements refer to the L1C band for that satellite (for example, if GPS PRN = 16 then L1C data is reported as Sat-iD = 1316 in the $PSTMTS messages for that satellite).

Galileo E1 Sat-iD

For GALILEO E1 signals, 300 is added to the Sat-iD to specify that the measurements refer to the E1 band for that satellite (for example, if GALILEO PRN = 16 then E1 data is reported as Sat-iD = 316 in the $PSTMTS messages for that satellite).

Galileo E5 Sat-iD

For GALILEO E5a(Q) signals, 600 is added to the Sat-iD to specify that the measurements refer to E5a band for that satellite (for example, if GALILEO PRN = 16 then E5a data is reported as Sat-iD = 616 in the $PSTMTS messages for that satellite).

For GALILEO E5b(Q) signals, 650 is added to the Sat-iD to specify that the measurements refer to the E5b band for that satellite.

Galileo E6 Sat-iD

For GALILEO E6 signals, 700 is added to the Sat-iD to specify that the measurements refer to the E6 band for that satellite (for example, if GALILEO PRN = 16 then E6 data is reported as Sat-iD = 716 in the $PSTMTS messages for that satellite).

QZSS sat-ID

The following sat-ID to PRN conversion table applies to the QZSS constellation:

Table 11. QZSS sat-ID to PRN mapping

Signal	Sat-ID	PRNs
L1 C/A	193 to 202	193 to 206
L1 C/B	203 to 206	
L2C	213 to 226	
L5	233 to 246	
L1C	253 to 266	
L1S	183 to 191	183 to 191

IRNSS L5 Sat-iD

For IRNSS L5 signals, 800 is added to the Sat-iD to specify that the measurements refer to the L5 band for that satellite (for example, if IRNSS PRN = 16 then L5 data is reported as Sat-iD = 816 in the $PSTMTS messages for that satellite).

2.6 $PSTMRF

It provides “satellite signal data” for each tracked satellite. A single message contains the relevant fields for three satellites. For all satellites, the message is repeated with the data of the other satellites.

NMEA message list bitmask (64 bits): 0000 0000 0000 0080.

Synopsis:

```
$PSTMRF,<MessgAmount>,<MessgIndex>,<used_sats>, [<Sat1ID>,<Sat1PhN>,<Sat1Freq>,<Sat1CN0>], .
.. [<SatNID>,<SatNPhN>,<SatNFreq>,<SatNCN0>]*<checksum><cr><lf>
```

N max 3

Table 12. PSTMRF field descriptions

Parameter	Format	Description
MessgAmount	Decimal, 1 digit	Number of consecutive $PSTMRF messages
MessgIndex	Decimal, 1 digit	Current number in the sequence of messages

Parameter	Format	Description
Used_sats	Decimal, 2 digits	Number of satellites used in the fix
SatxID	Decimal, 2 digits	Satellite x number (PRN)
SatxPhN	Decimal, 5 digits	Satellite x phase noise
SatxFreq	Decimal, 6 digits	Satellite x frequency
SatxCN0	Floating	Satellite x carrier to noise ratio (in DB)
…		
checksum	Hexadecimal, 2 digits	NMEA message checksum

Note: <predxxx> fields are only valid if the AGPS software module has been included.

2.7 Sample code

This section contains code snippets to show how to decode version 1 raw measurements from STMicroelectronics proprietary NMEA sentences.

Codes are provided for reference only.

$PSTMTG decoding example

```
typedef struct {
  int TG_TS_version;
  int TG_TS_front_end;
  int clock_steering;
  int week_n;
  double tow;
  int sats_tracked;
  unsigned int cpu_time;
  unsigned int time_validity;
  double clock_drift;
  double clock_bias_mt;
  int flags_aux;
  int leap_sec;
} TG_data_t;

int nmea_read_TG_msg(char *line, TG_data_t *tg_data)
{
  int num_fields_tg;
  int kf_config;
  int const_mask = 0;

  if( strncmp(line, "$PSTMTG,", strlen("$PSTMTG,") ))
  {
    return -1;
  }

  num_fields_tg = sscanf(line, "$PSTMTG,%d,%lf,%d,%u,%d,%lf,%x,",
   &tg_data->week_n, &tg_data->tow, &tg_data->sats_tracked,
   &tg_data->cpu_time, &tg_data->time_validity, &clock_drift, &kf_config);

  if( num_fields_tg != 7)
  {
    return -1;
  }

  tg_data->TG_TS_version   = (kf_config >> 12) & 0xF;
  tg_data->TG_TS_front_end = (kf_config >> 8) & 0xF;
  tg_data->clock_steering  = (tg_data->TG_TS_version & 0x8) ? 1:0;
  tg_data->TG_TS_version   &= 0x7;  // remove steering bit from version

  if (tg_data->TG_TS_version != 2)
  {
    printf("TG/TS version not supported : %d\n", tg_data->TG_TS_version);
    return -1;
```

```
  }
  else // (tg_data->TG_TS_version == 2)
  {
    int time_master_sat_type;
    int time_master_week_n;
    double time_master_tow;
    int time_master_validity;
    unsigned int mfreq_constellation_mask;

    num_fields_tg = sscanf(line,
        "$PSTMTG,%d,%lf,%d,%d,%d,%lf,%x,%u,%u,%d,%d,%lf,%d,%d,%lf,%d,%d",
          &tg_data->week_n,
          &tg_data->tow,
          &tg_data->sats_tracked,
          &tg_data->cpu_time,
          &tg_data->time_validity,
          &tg_data->clock_drift,
          &kf_config,
          &const_mask,
          &tg_data->time_best_sat_type,
          &time_master_sat_type,
          &time_master_week_n,
          &time_master_tow,
          &time_master_validity,
          &tg_data->flags_aux,
          &tg_data->clock_bias_mt,
          &mfreq_constellation_mask,
          &tg_data->leap_sec
    );
  }

  return 0;
}
```

$PSTMTS decoding example

```
typedef struct {
  int dsp_avail;
  int sat_id;
  double PseudoRange;
  double Doppler;
  double CarrierPhase;
  int PreambleLocked;
  int CN0;
  int TrackedTime;
  int glonass_slot;
  int cycle_slip_cnt;
  int cp_valid;
} TS_data_t;

int nmea_read_TS_msg(char *line, TS_data_t *ts_data)
{
  int num_fields;

  if( strncmp(line, "$PSTMTS,", strlen("$PSTMTS,") ))
  {
    return -1;
  }

  num_fields = sscanf(line, "$PSTMTS,%d,%d", &ts_data->dsp_avail,
    &ts_data->sat_id);

  if(num_fields != 2)
  {
    return -1;
  }

  ts_data->pred_available = 0;
  ts_data->pr_corr = 0.0;
```

```
  ts_data->cp_corr = 0.0;
  ts_data->glonass_slot = 0
  ts_data->cycle_slip_cnt = 0;
  dsp_data_flags_t dsp_flags;
  unsigned long long int dsp_flags_64;
  int code_noise, ave_phase_noise;
  int dsp_available = 0;
  int checksum;

  num_fields = sscanf_s(line,
"$PSTMTS,%d,%d,%lf,%lf,%lf,%llu,%d,%d,%d,%d,%d,%d,%d*%X",
    &dsp_available,
    &ts_data->sat_id,
    &ts_data->PseudoRange,
    &ts_data->Doppler,
    &ts_data->CarrierPhase,
    &dsp_flags_64,
    &ts_data->CN0,
    &ts_data->TrackedTime,
    &code_noise,
    &ave_phase_noise,
    &ts_data->cycle_slip_cnt,
    &ts_data->glonass_slot,
    &elevation_deg,
    &checksum);

  if ((num_fields != 33))
  {
    return -2;
  }

  dsp_flags.REG = dsp_flags_64;
  ts_data->PreambleLocked = dsp_flags.BIT.preamble_locked;
}
```

Raw measurements calculation example

```
#define GPS_TRANSMITTED_FREQ       1575420000.0     // Hz
#define GLONASS_TRANSMITTED_FREQ   1602000000.0     // Hz
#define GLONASS_CHANNEL_SPACING    562500.0         // Hz
#define COMPASS_TRANSMITTED_FREQ   1561098000.0     // Hz
#define GLONASS_SAT_ID_HALF(sat_id) ((((int)(sat_id)-72)>6)?1:0)
#define SAT_ID_TO_K(sat_id) (GLONASS_SAT_ID_HALF(sat_id)==1 ? ((int)(sat_id)-86):((int)(sat_i
d)-72))

void meas_calc(const TG_data_t *tg_data, TS_data_t *ts_data, double *prange, double *cphase,
double *doppler)
{ double clock_drift = 0.0;
  double GPS_CLOCK_OFFSET = 0.0;
  int bds_sat_id = 0;
  if (tg_data->TG_TS_front_end == 0)
  {GPS_CLOCK_OFFSET = -47122.395833492279;
  }
  else if (tg_data->TG_TS_front_end == 1)
  {GPS_CLOCK_OFFSET = -40526.315789699554;
  }
  if( tg_data->clock_steering)
  {gps_clock_drift = tg_data->clock_drift;
  }
    else
  {gps_clock_drift = GPS_CLOCK_OFFSET;
  }
  if( sat_type == SAT_TYPE_GPS )
  {clock_drift = gps_clock_drift;
  }
  else if( sat_type == SAT_TYPE_GLONASS )
  {int freq_id = SAT_ID_TO_K(this_ts->sat_id);
    clock_drift =  (gps_clock_drift * (GLONASS_TRANSMITTED_FREQ + GLONASS_CHANNEL_SPACING*(fr
eq_id)) / GPS_TRANSMITTED_FREQ);
```

```
  }
  else if( sat_type == SAT_TYPE_COMPASS )
  {clock_drift = (-1.0) * gps_clock_drift * COMPASS_TRANSMITTED_FREQ / GPS_TRANSMITTED_FREQ;
  }
  // raw measurements with corrections
  *prange  = this_ts->PseudoRange;
  *cphase  = -(this_ts->CarrierPhase);
  *doppler = this_ts->Doppler - clock_drift;
}
```

3 NMEA signal and observations quality metrics

3.1 $PSTMMG

This message reports the time week number, the time of week, the validity, and the constellation type of the current best time, in addition to signal and observation quality metrics masks.

The best time is the time with the better validity.

The masks specify which metrics are reported in the subsequent $PSTMMS messages.

The bit#0 in the mask is the lsb.

NMEA message list bitmask (64 bits): 0000 0000 2000 0000

Synopsis:

```
$PSTMMG,<Week>,<Tow>,<Val>,<Sat_type>,<SigQM mask>,<ObsQM mask>,<SigQM mask2>,<ObsQM mask2>*<
checksum><cr><lf>
```

Table 13. PSTMMG field descriptions

Parameter	Format	Description
Week	Decimal, 4 digits	Best time week number
TOW	Floating	Best time of the week
Val	Decimal	Best time validity
Sat_type	Decimal	Best time constellation
SigQM mask	Decimal	Signal quality metrics mask
ObsQM mask	Decimal	Observation quality metrics mask
SigQM mask2	Decimal	Signal quality metrics mask 2
ObsQM mask2	Decimal	Observation quality metrics mask 2
checksum	Hexadecimal, 2 digits	NMEA message checksum

The signal quality metrics mask bits have the following meaning:

Table 14. Signal quality metrics mask description

Bit	Metrics	Freq.	Res.
0	Signal strength (CN0)	1	0.25 dB
1	Carrier loop discriminator output std estimate	1	1
2	Code loop discriminator output std estimate	1	1
3	Double-delta metric, based on 5x correlations	1	1
4	Auto-correlation metric #1, based on 10x correlations (added if CDB Page 16 Line 2 Field 5 has 1 pool or more enabled)	1	1
5	DSP flags (main freq.)	1	-
9:6	Reserved	--	--
10	Signal strength (CN0)	2	0.25 dB
11	Carrier loop discriminator output std estimate	2	1
12	Code loop discriminator output std estimate	2	1
13	Double-delta metric, based on 5x correlations	2	1
14	Auto-correlation metric #1, based on 10x correlations	2	1

Bit	Metrics	Freq.	Res.
15	DSP flags (dual freq.)	2	-
19:16	Reserved	--	
20	Signal strength (CN0)	3	0.25 dB
21	Carrier loop discriminator output std estimate	3	1
22	Code loop discriminator output std estimate	3	1
23	Double-delta metric, based on 5x correlations	3	1
24	Auto-correlation metric #1, based on 10x correlations	3	1
25	DSP flags (triple freq.)	3	-
31:26	Reserved	--	

The observation quality metrics mask bits have the following meaning:

Table 15. Observable quality metrics mask description

Bit	Metrics	Freq.	Res.
0	Code-carrier divergence (main freq.)	1	1
1	Code-carrier divergence (dual freq.)	2	1
2	Code-carrier divergence (triple freq.)	3	1
3	Doppler to range-rate divergence (main freq.)	1	1
4	Doppler to range-rate divergence (dual freq.)	2	1
5	Doppler to range-rate divergence (triple freq.)	3	1
6	Dual frequency carrier phase delta gradient (main, dual)	1.2	1
7	Dual frequency carrier phase delta gradient (main, triple)	1.3	1
8	Interfrequency code bias residual (main, dual)	1.2	1
9	Interfrequency code bias residual (main, triple)	1.3	1
31:10	Reserved	--	--

The signal quality metrics mask2 bits have the following meaning.

Table 16. Signal quality metrics mask2 description

Bit	Metrics	Freq.	Res.
0	Signal strength (CN0)	4	0.25 dB
1	Carrier loop discriminator output std estimate	4	1
2	Code loop discriminator output std estimate	4	1
3	Double-delta metric, based on 5x correlations	4	1
4	Auto-correlation metric #1, based on 10x correlations	4	1
5	DSP flags (main freq.)	4	-
9:6	Reserved	--	--
10	Signal strength (CN0)	5	0.25 dB
11	Carrier loop discriminator output std estimate	5	1
12	Code loop discriminator output std estimate	5	1
13	Double-delta metric, based on 5x correlations	5	1
14	Auto-correlation metric #1, based on 10x correlations	5	1

Bit	Metrics	Freq.	Res.
15	DSP flags (dual freq.)	5	-
31:16	Reserved	--	

The observation quality metrics Mask2 bits have the following meaning:

Table 17. Observable quality metrics mask2 description

Bit	Metrics	Freq.	Res.
0	Code-carrier divergence (fourth freq.)	4	1
1	Code-carrier divergence (fifth freq.)	5	1
2	Reserved	--	--
3	Doppler to range-rate divergence (fourth freq.)	4	1
4	Doppler to range-rate divergence (fifth freq.)	5	1
5	Reserved	--	--
6	Dual frequency carrier phase delta gradient (main, fifth)	1.5	1
7	Reserved	--	--
8	Interfrequency code bias residual (main, fifth)	1.5	1
21:9	Reserved	--	--
22	Dual frequency carrier phase delta gradient (main, fourth)	1.4	1
23	Interfrequency code bias residual (main, fourth)	1.4	1
31:24	Reserved	--	--

Example

```
$PSTMMG,2258,142533.00000008,9,0,48281663,524287,47150,1103327099*08
```

3.2 $PSTMMS

This message reports the valid signal and observation quality metrics for each tracked satellite.

The metrics reported are the ones specified in the preceding $PSTMMG message masks.

If a metric bit is set in the corresponding $PSTMMG mask, but its value is not valid then it will be reported as empty field.

NMEA message list bitmask (64 bits): 0000 0000 2000 0000.

Synopsis:

```
$PSTMMS,<SatID>,[<SigM #31>],[<SigM #30>],…,[<SigM #0>],[<ObsM #31>],[<ObsM #30>],…,[<ObsM #0>]*<checksum><cr><lf>
```

Table 18. PSTMMS field descriptions

Parameter	Format	Description
Sat-ID	Decimal, 2 digits	Satellite number (PRN)
SigM #0	Decimal	1st signal metric (only if its bit is set in the preceding $PSTMMG signal metric mask)
SigM #1	Decimal	2nd signal metric (only if its bit is set in the preceding $PSTMMG signal metric mask)
…	Decimal	…
SigM #31	Decimal	32nd signal metric (only if its bit is set in the preceding $PSTMMG signal metric mask)

Parameter	Format	Description
ObsM #0	Decimal	1st observation metric (only if its bit is set in the preceding $PSTMMG signal metric mask)
ObsM #1	Decimal	2nd observation metric (only if its bit is set in the preceding $PSTMMG signal metric mask)
…	Decimal	…
ObsM #31	Decimal	32nd observation metric (only if its bit is set in the preceding $PSTMMG signal metric mask)

Example

```
$PSTMMS,9,5255,76,-356,691,21635,,39,274,538,40,1187,-149,2516,275,652,75*1B
```

3.3 $PSTMMSE

This message reports the valid signal and observation quality metrics for each tracked satellite.

The metrics reported are the ones specified in the preceding $PSTMMG message masks 2.

If a metric bit is set in the corresponding $PSTMMG mask 2, but its value is not valid then it will be reported as empty field.

NMEA message list bitmask (64 bits): 0000 0000 2000 0000.

Synopsis:

```
$PSTMMSE,<SatID>,[<SigM #31>],[<SigM #30>],…,[<SigM #0>],[<ObsM #31>],[<ObsM #30>],…,[<ObsM #0>]*<checksum><cr><lf>
```

Table 19. PSTMMSE field descriptions

Parameter	Format	Description
Sat-ID	Decimal, 2 digits	Satellite number (PRN)
SigM #0	Decimal	1st signal metric (only if its bit is set in the preceding $PSTMMG signal metric mask2)
SigM #1	Decimal	2nd signal metric (only if its bit is set in the preceding $PSTMMG signal metric mask2)
…	Decimal	…
SigM #31	Decimal	32nd signal metric (only if its bit is set in the preceding $PSTMMG signal metric mask2)
ObsM #0	Decimal	1st observation metric (only if its bit is set in the preceding $PSTMMG signal metric mask2)
ObsM #1	Decimal	2nd observation metric (only if its bit is set in the preceding $PSTMMG signal metric mask2)
…	Decimal	…
ObsM #31	Decimal	32nd observation metric (only if its bit is set in the preceding $PSTMMG signal metric mask2)

Example

```
$PSTMMSE,23,1,,,,,398467,10,10,306,,,-408312,-10,,115,,119,,106,,,,,,323,,88 *47
```

4 NMEA navigation data support

4.1 $PSTMEPHEM

This message contains the ephemeris data for each available satellite. Refer to Teseo VI and Teseo APP2–NMEA specifications and commands for "*ST GNSS NMEA specification and commands*" document for further details about $PSTMEPHEM message.

```
$PSTMEPHEM,<sat_id>,<ephem_data_size>,<ephem_data>*<cr><lf>
```

The encoding for GPS ephemeris from $PSTMEPHEM message payload (<*ephem_data*>) 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 age_h                : 10;
  unsigned int l2_codes             : 2;
  unsigned int spare3               : 4;
  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 subframe1_available  : 1;
  unsigned int subframe2_available  : 1;
  unsigned int subframe3_available  : 1;
  unsigned int available            : 1;
  unsigned int health               : 1;
  unsigned int predicted            : 1;
  unsigned int accuracy             : 4;
} gps_ephemeris_t;
```

The encoding for GLONASS ephemeris from $PSTMEPHEM message payload (<*ephem_data*>) 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 delta_tau_n          : 5;
  unsigned int spare4               : 3;

  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 age_h                : 10;

  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 predicted            : 1;
  unsigned int spare10              : 4;
} glonass_ephemeris_t;
```

The encoding for BeiDou ephemeris from $PSTMEPHEM message payload (<*ephem_data*>) 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 age_h             : 6;

  unsigned int cis               : 18;
  unsigned int B1                : 8;
  unsigned int time_distance_h     : 6;

  unsigned int cic               : 18;
  unsigned int nvm_reliable      : 1;
  unsigned int predicted         : 1;
  unsigned int tgd2              : 10;
  unsigned int spare4            : 2;

  unsigned int toc               : 17;
  unsigned int week              : 13;
  unsigned int available         : 1;
  unsigned int health            : 1;

} beidou_ephemeris_t;
```

The encoding for Galileo ephemeris from $PSTMEPHEM message payload (<*ephem_data*>) 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 ephems_n             : 2; // must be 0

  unsigned int toc                  : 14;
  unsigned int iod_nav              : 10;
  unsigned int SISA                 : 8;

  unsigned int age_h                : 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 predicted            : 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 time_distance_h      : 6; // must be 0
  unsigned int spare1               : 26;
} galileo_ephemeris_t;
```

The following snippet shows how to use the above structures to convert the payload from $PSTMEPHEM message into the decoding structure itself.

```
void decode_gps_eph(unsigned char* payload, gps_ephemeris_t* eph)
{    memcpy (eph, payload, sizeof(gps_ephemeris_t));
}
void decode_glo_eph(unsigned char* payload, glo_ephemeris_t* eph)
{    memcpy (eph, payload, sizeof(glonass_ephemeris_t));
}
void decode_bds_eph(unsigned char* payload, beidou_ephemeris_t* eph)
{    memcpy (eph, payload, sizeof(beidou_ephemeris_t));
}
void decode_gal_eph(unsigned char* payload, galileo_ephemeris_t* eph)
{    memcpy (eph, payload, sizeof(galileo_ephemeris_t));
```

Besides, the $PSTMEPHEM message payload (<*ephem_data*>), is provided as hexadecimal string encoded over ASCII, which means that it has to be converted into binary format before invoking the decoding functions described above.

For example, to extract the ASCII payload from the string:

```
char command[] = { "$PSTMEPHEM,9,64,9d07ae60ae605858588c050000a6ff006503fa1d5c0faf02edfffbffe
b360000c6eedc26f4bf630028060da1c61cbbd82b3f51f8517f094f0300460005fdcb03*55" };
```

The following C standard library function can be used:

```
sscanf(command, "$PSTMEPHEM,%d,%d,%s", &sat, &len, hex_buf);
```

Where:

- *sat* contains the satellite ID (int);
- *len* contains the length of the payload (int);
- *hex_buf* contains the ASCII payload (char array);

Now, the content of *hex_buf* has to be converted to binary.

```
for (i = 0; i < len; i++)
{
    sscanf(&hex_buf[i * 2], "%02x", &payload[i]);
}
```

Where:

- *payload* is the binary payload that can be passed to the decoding functions described above (unsigned char array).
- *i* is the for-loop index (int).

Note: All the values in the ephemeris structures are expressed in the format and scale reported in the official interface control documents for the respective constellations.

4.2 $PSTMSBASM

Output the SBAS satellite message frame. See AN_TeseoVI-APP2_GNSS_NMEA_Interface, 2024, Appendix A: Acronyms and reference documents document for further details about this message.

Synopsis:

```
$PSTMSBASM,<SatID>,<Msg>*<checksum><cr><lf>
```

4.3 $PSTMNAVM

Navigation data frame. Refer to Teseo VI and Teseo APP2–NMEA specifications and commands for further details about this message.

Synopsis:

```
$PSTMNAVM,<msg_id>,<prn>,<nav_frame>*<checksum><cr><lf>
```

Details

The navigation frame parameter depends on the constellation. Table 20. Navigation frame data types describes its meaning (refer to Teseo VI and Teseo APP2–NMEA specifications and commands to detail each constellation ICD):

Table 20. Navigation frame data types

Constellation	Type	Length (bits)	Length (bytes)	Note
GPS	Subframe	300	40 (10 words)	For each 32-bit word 30 bits are used (the 2 msb are ignored)
GLONASS	1 or 2 strings	85 or 170 (85+85)	11 or 22 (11+11 bytes)	One string for each message for strings from 1 to 5. Two strings for each message for strings from 6 to 15. For the first byte of each string the 3 msb are ignored and the fourth is always zero. The payload is 84 bits long
GALILEO I/NAV payload	payload	128	16 (4 words)	Each message contains the data payload from the I/NAV message (see Note for details)
GALILEO full I/NAV	Even and odd pages	240	32 (8 words)	Each message contains the full I/NAV message (see Note for details)
BEIDOU	Subframe	300	40 (10 words)	For each 32-bit word 30 bits are used (the 2 msb are ignored)
IRNSS	Frame	300	40 (10 words)	

Note: Only frames with correct CRC are reported.

In Table 20. Navigation frame data types, "word" means a 32-bit little endian encoded word, while "msb" means most significant bits.

It means that, in a little-endian architecture system, the navigation frame (converted to binary format) can be directly copied into a C language 32-bit unsigned integer words array. In other words:

- *For strings for #1 to #5 just the first 11 bytes will be used, while for strings from #6 to #15 all 22 bytes will be used by storing two consecutive strings (for example, strings #7 and #6). In this latter case, the first string (for example, string #n) will be stored in the second part of the array (that is, from byte #12 to #22), and the second string (for example, string #n+1) will be stored in the first part of the array (that is, from byte #1 to #11).*
- *For the GALILEO payload, the navigation frame can be copied in a C language variable defined according to Table 21. Galileo I/NAV payload, 128[bit], 32-bit packing type of definition:*

```
typedef tU32 gal_subframe_t [4];
```

The GALILEO payload navigation frame contains the message payload only, encoded according to the following table:

Table 21. Galileo I/NAV payload, 128[bit], 32-bit packing

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Data k 0-31 (112-bit)																															
1	Data k 32-63 (112-bit)																															
2	Data k 64-95 (112-bit)																															
3	Data k 96-111 (112-bit)																Data j (16-bit)														

- For GALILEO full I/NAV, the message reports both the even and odd pages of the I/NAV message (E1-b), encoded as specified in the GALILEO ICD document, on the first 240 bits, according to the following table:

Table 22. Galileo I/NAV full (even+odd)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	EO	P	Data k 0-29 (112-bit)																													
1	Data k 30-61 (112-bit)																															
2	Data k 62-93 (112-bit)																															
3	Data k 94-111 (112-bit)																	EO	P	Data j 0-11 (16-bit)											
4	Data j 4-LSB												Reserved1																			
5	Reserved1													SAR 0-19 (22-bit)																		
6	SAR		Spare										CRCi 24-bit																			

- For BEIDOU, the navigation frame can be copied in a C language variable defined according to the following type definition:

```
typedef tU32 bds_subframe_t [10];
```

where *tU32* is a 32-bit unsigned integer type and *tU08* is an 8-bit unsigned integer type.

Table 23. BeiDou common frame (D1/D2)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			Preamble 11-bit											Rev 4-bit				FraID			SOW 8-MSB								P 4-bit			
1			SOW 12-LSB																						P 8-bit							
2																									P 8-bit							
3																									P 8-bit							
4																									P 8-bit							
5																									P 8-bit							
6																									P 8-bit							
7																									P 8-bit							
8																									P 8-bit							
9																									P 8-bit							

- For IRNSS the navigation frame can be copied in a C language variable defined according to the following type definition:

```
typedef tU32 irnss_subframe_t [10];
```

Table 24. **IRNSS payload**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		TLM							TOWC																	A	A	ID		S		
1																																
2																																
3																																
4																																
5																																
6																																
7																																
8		PRN					CRC																							Tail		
9		Tail																														

Example

```
 $PSTMNAVM,0,4,00AFC32268A9BD26337FF43AC40B60D1B8B80018C8EE0B0330BDA238AF711D185E1000C0887907
81*23
```

5 Receiver configuration

5.1 Frequency plan

For information about the frequency plan see (AN firmware configuration, 2024) and firmware release note.

5.1.1 Configuring with NMEA

The following NMEA command can be used to change the multi-frequency configuration at run time:

Synopsis

```
$PSTMSETCONSTMASK,<constellation mask>,<multi-freq. mask><cr><lf>
```

or

```
$PSTMSETCONSTMASK,<constellation mask>,<multi-freq. mask>,<usage mask><cr><lf>
```

Arguments

Table 25. PSTMSETCONSTMASK field descriptions

Parameter	Format	Description
constellation_mask	Decimal, 1 – 15728639	It is a bit mask where each bit enables/disables a specific constellation independently of the others. Refer to Teseo VI and Teseo APP2–Firmware configuration for details.
multi-freq. mask	Decimal, 1 – 27795599	It is a bit mask where some bits configure multi-frequency, pilot/data or tracking mode, refer to Teseo VI and Teseo APP2–Firmware configuration.
usage_mask	Decimal, 1 – 122543	Specify which constellation is used for PVT (optional), refer to Teseo VI and Teseo APP2–Firmware configuration.

Note: When only 2 arguments are specified then usage_mask == constellation_mask

All masks values must be converted in decimal.

Example of usage:

```
$PSTMSETCONSTMASK,157,27262976,141 → GPS/QZSS L1CA, GAL E1, BDS B1I, SBAS
```

5.1.2 Configuring with FW configurator

Both the requested constellation mask and multi-frequency configuration can be set using the FW Configurator utility. This will allow having the device already working as for the selected operative scenario.

Refer to Teseo VI and Teseo APP2–Firmware configuration, to get more details about these parameters.

5.2 GLONASS interchannel bias

GLONASS pseudorange interchannel biases (ICB) are compensated by the measurement engine through tables of corrections terms. Two ICB tables are defined, for GLONASS L1 and L2 respectively. Each table has 14 elements for all the GLONASS FDM channels (from -7 to +6), relative to the center carrier frequency f01 (or f02). Each element is expressed in decimeters units.

GLONASS ICB tables can be dumped or modified thanks to some dedicated NMEA commands ($PSTMDUMPGLONASSICB and $PSTMGLONASSICB). Refer to Teseo VI and Teseo APP2–NMEA specifications and commands.

5.3 Interfrequency biases

Interfrequency pseudorange biases (IFB) are compensated by the measurement engine by application of corrections terms. IFB(s) are represented by an array of corrections. Each element is expressed in decimeters and indicates the pseudorange bias between a pair of carrier frequencies of the same constellation.

IFB table can be dumped or modified thanks to some dedicated NMEA commands ($PSTMDUMPIFB and $PSTMIFB). Refer to Teseo VI and Teseo APP2–NMEA specifications and commands.

5.3.1 IFB residuals

IFB values shall be intended as “nominal” values that the measurement engine applies to pseudoranges of the dual carrier. The receiver never tries to modify these values however it continuously estimates one or more IFBs. The residuals of this estimation for each satellite can be dumped on NMEA with the $PSTMIFBRES sentence. Refer to Teseo VI and Teseo APP2–NMEA specifications and commands.

The IFB residuals should be close to zero in optimal receiver operation, which is indicating that one or more estimated IFBs are matching the nominal values stored in NVM and applied to the pseudoranges. The residuals are dumped in decimeters units.

Appendix A Acronyms and reference documents

Table 26. Acronyms

Keyword	Definition
CN0	Carrier to noise density ratio [dB/Hz]
DCB	Differential code biases
ECEF	Earth centered Earth fixed
GLONASS	Global'naja navigacionnaja sputinkovaja sistema
GNSS	Global navigation satellite system
GPS	Global positioning system
HDOP	Horizontal dilution of precision
LS	Least squares
NMEA	National marine electronics association
PDOP	Position dilution of precision
PVT	Position, velocity, time
SV	Space vehicle
TCXO	Temperature-controlled oscillator
VDOP	Vertical dilution of precision
MTI	Message transmission interval

Table 27. Reference documents

Document name	Document title
UM3407	Teseo VI and Teseo APP2–NMEA specifications and commands
UM3428	Teseo VI and Teseo APP2–Firmware configuration
UM3432	Teseo VI–Assisted GNSS interface specification

Revision history

Table 28. **Document revision history**

Date	Revision	Changes
26-Nov-2024	1	Initial release.
28-Jan-2025	2	Updated *Section 2.2: $PSTMTG* and *Table 7. PSTMSAT field descriptions*.
01-Jul-2025	3	Updated *Section 2.2: $PSTMTG; Table 11. QZSS sat-ID to PRN mapping; Table 27. Reference documents.*
11-Nov-2025	4	Updated *Table 7. PSTMSAT field descriptions*.
24-Nov-2025	5	Updated Table 8. PSTMTS filed descriptions and $PSTMTS decoding example. Minor text changes.

Contents

List of tables

List of figures

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

The purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of the purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers' market segment, the purchasers shall contact ST for more information.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.