Automotive and Discrete Group
Automotive Digital Division

Infotainment Business Unit

Low Power Management

Purpose of this document is to provide details about the Low Power Management library implemented in STA8089/STA8090 firmwares (TeseoIII family).

Contents

1.1 Index

1.2 List of Tables

1.3 List of Figures

2 Document Management

2.1 Revision History

Rev	Date	Author	Notes
1.0	08/04/2013	Michele Renna	
2.0	05/07/2013	Michele Renna	Add duty cycle features
2.1	17/12/2014	Antonio Furno	General review
3.0	31/03/2016	Jerome Durand Vincent Delaunay	Addition of Periodic mode
3.1	06/06/2016	Jerome Durand	Update after review
3.2	29/06/2016	Jerome Durand	Update for Standby Periodic mode and reworked APIs
3.3	02/11/2016	Jerome Durand	Update for STAGPS™

Table 1 Revision history

2.2 Acronyms

Keyword	Definition
BEIDOU	Chinese Navigation Satellite System
GALILEO	European Navigation Satellite System
GLONASS	GLObal NAvigation Satellite System (the GNSS operated by the Russian Aerospace Defence Forces)
GNSS	Global Navigation Satellite System - Satellite based system to calculate the position of the receiver on the earth surface.
GPS	Global Positioning System - United States Satellite Navigation System
STAGPS™	ST Autonomous assisted GPS solution
STANDBY	State where minimum hardware blocks are powered on

Table 2 Acronyms

2.3 Reference Documents

Reference	Title	Author	Version
HW_PwrStrgOvw	Power Strategy Overview	G. Peveraro	2.0
GNSS_API	GNSS library specification	A. Di Girolamo F. Boggia	4.6
FW_Configuration	STA8089-90 Firmware Configuration	A. Di Girolamo	1.2
OS20_Spec	OS20+ operating system specification	F. Boggia	3.3

Table 3 References

3 Introduction

3.1 Overview

The Low Power Management library implements different modes including the functionalities below:

- Active and Standby Periodic Low Power mode:
  - Report a fix at a given periodicity
  - Autonomous periodic ephemeris refresh
  - RTC calibration capability
  - Optional use of STAGPS™ (Standby mode only)
  - Different hardware power state between fixes are possible
- Fix on demand Low Power mode (Standby mode only):
  - Report a fix on demand triggered by an hardware pin
  - Autonomous periodic ephemeris refresh
  - RTC calibration capability

The periodic mode saves power when a fix is needed more than every 5 seconds and when accuracy degradation is acceptable. Two cases are depicted, corresponding to different hardware states between the fix activities. There is the active case and the standby case (maximum power saving). The periodic mode is described in chapter 4.1 and 6. As the Fix On Demand mode shares the parameters and algorithm, it is described in the same chapters as periodic mode. The usage of STAGPS™ feature allows to reduce the energy spent in the ephemeris refresh periods.

The chapter 5 describes how to setup the parameters of all modes.

The choice between the different modes is driven by the required fix periodicity.

Fix Periodicity	Appropriate mode
0.1s-1s	None
5s-24H (SDK)	Active Periodic mode
5s-24H (Binary + SDK)	Standby Periodic mode + optional STAGPS™
Asynchronous	Fix On Demand

Table 4 Low power mode versus Fix periodicity

3.2 Mode maturity

Feature	Maturity level	Platforms
Active Periodic mode	Validated (GPS Only)	STA8089, STA8090

Standby Periodic mode	Validated (GPS Only)	STA8089, STA8090
Standby Periodic mode w/ STAGPS™	Validated (GPS Only)	STA8089, STA8090
Fix On Demand mode (STANDBY)	Validated (GPS Only)	STA8089, STA8090

Table 5 Feature maturity

4 Algorithm description

4.1 Periodic mode

The periodic mode has different settings to control the FIX reporting, and other settings to control the low power hardware state.

The periodic mode can have two different hardware states between FIX activities:

- Wait For Interrupt state used in Active Periodic mode, where the system clock is set to the RING oscillator (a low power oscillator)
- Standby state used in Standby Periodic mode, where only Always ON domain is alive

Although the Wait For Interrupt hardware state ensure continuity of software execution and maintain data, the Standby hardware state is a reset and ARM Core state and on-board memories except backup RAM are lost.

4.1.1 State Machine

The periodic mode has basically two parts in its state machine – one to handle the fix (left) and one to handle the case of no fix (right). The transitions between both in case of fix loss or recovery is done according to the steady state condition. The steady state is the combination of the following information:

- The system is in Position Accurate condition (position fix available);
- Ephemeris available (5 each activated constellations);
- Almanac, Ephemeris or Health information collected for all satellites.

Generally at first start up (Full Cold Mode) this condition, in full sky is reached in 12.5 minutes for GPS constellation.

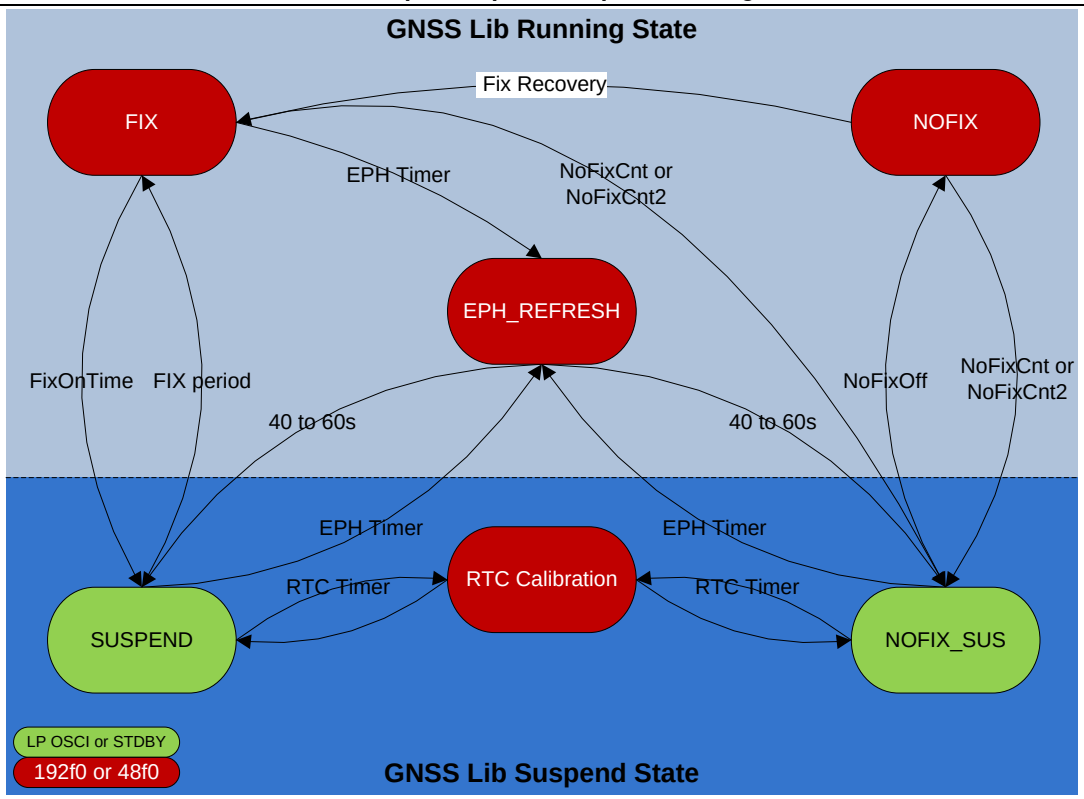


Figure 1 Periodic mode State Diagram

Here are details about the different states:

- SUSPEND: The GNSS Lib has previously managed to report a fix, steady state has been reached, so the SUSPEND state can be entered. Three timers are run: FixPeriod for next Fix occurrence, EPH refresh and RTC calibration. Expiration of the first two timers can trigger a transition to FIX or EPH_REFRESH states, while the RTC calibration is done in suspended mode.
- FIX: A new fix or a series of N fixes are expected. Go back to SUSPEND as soon as 1 or N fixes are reported. If the GNSS fix can't be calculated during NoFixCnt or NoFixCnt2 seconds (difference between both timers explained below), a transition to NOFIX_SUS, a suspended state, is triggered. If the ephemeris refresh timer occurs during the fix calculation, a transition to EPH_REFRESH occurs.
- EPH_REFRESH: Period where ephemeris are downloaded. If the signal is lost during NoFixCnt or NoFixCnt2 seconds, a transition to NOFIX_SUS is triggered, otherwise it goes to SUSPEND.
- NOFIX_SUS: Suspended state, but coming from a signal loss transition. Periodicities are different than in normal FIX condition to avoid losing too much energy in poor signal situation. Ephemeris download are anyway tried, so a transition to EPH_REFRESH can occur. A transition to NOFIX state occurs when NoFixOff timer occurs.
- NOFIX: The GNSS Lib wait for the configured number of seconds that the GNSS signal is recovered. If yes, a transition to FIX state occurs. In case no, the lib goes back to NOFIX_SUS.
- RTC Calibration: When configured in the settings, a RTC calibration is done on the first transition to SUSPEND state, and regularly reconfirmed every 5 minutes.

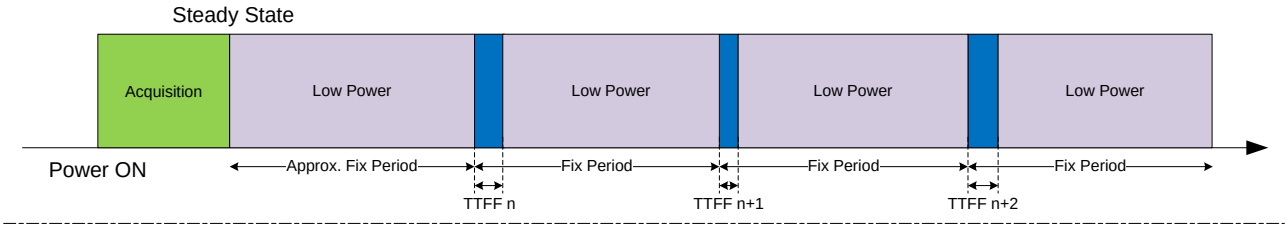
The two states concerned by the low power hardware states are SUSPEND and NOFIX_SUS. The RTC Calibration state occurs while the GNSS Lib is suspended, but is executed anyway at high frequency (48f0 or 192f0 according to frequency settings).

NoFixCnt is used in HOT conditions (Number of ephemeris and RTC are OK), while NoFixCnt2 is used in non-HOT conditions (start-up cases, obsolete ephemeris…). Their values are related to the expected sensitivity supported by the platform in bad RF conditions. Lower values gives a worst sensitivity.

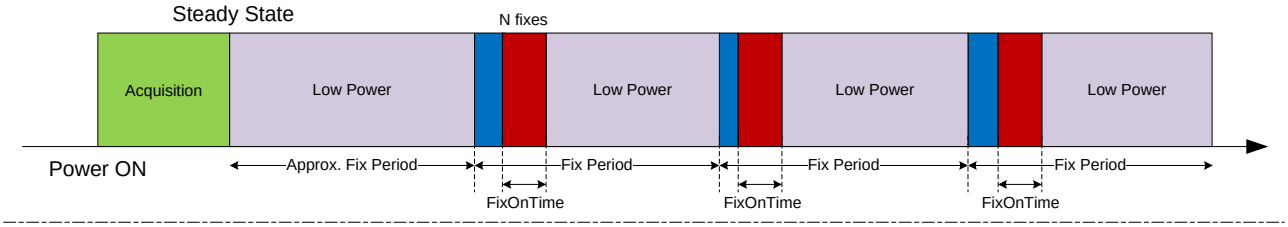
The EPH_REFRESH state aims at downloading ephemeris and almanacs before they become obsolete to ensure a certain level of fix accuracy. It is done approximately every 30 minutes, during 40 to 60 seconds. When the STAGPS™ feature is set and the GNSS receiver has downloaded an ephemeris for each satellite of the constellation, the STAGPS™ ephemeris predictions can replace real ephemeris and the ephemeris refresh interval is extended to about 10 hours and last 66 seconds.

4.1.2 Good GNSS coverage sequences

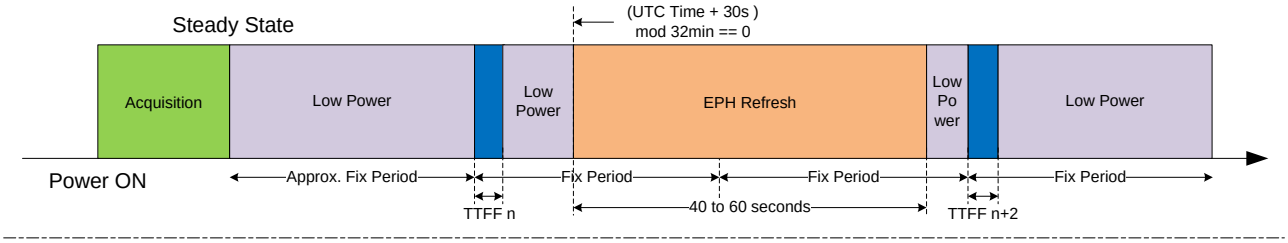
FIX - Good coverage – FixOnTime = 1



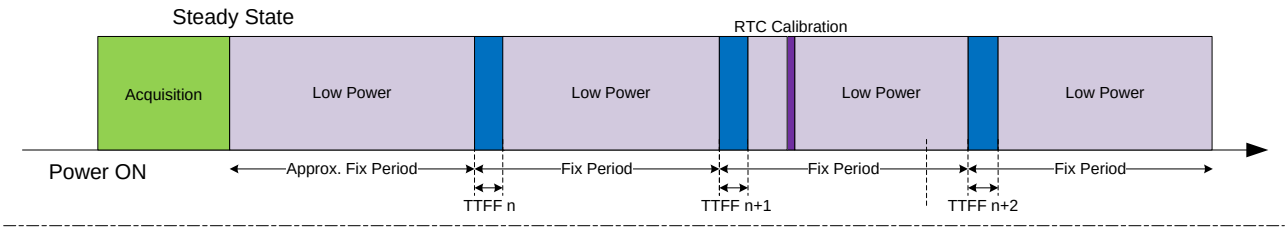
FIX - Good coverage – FixOnTime = N



EPH Refresh - Good coverage – FixOnTime = 1 – Eph Refresh = 1

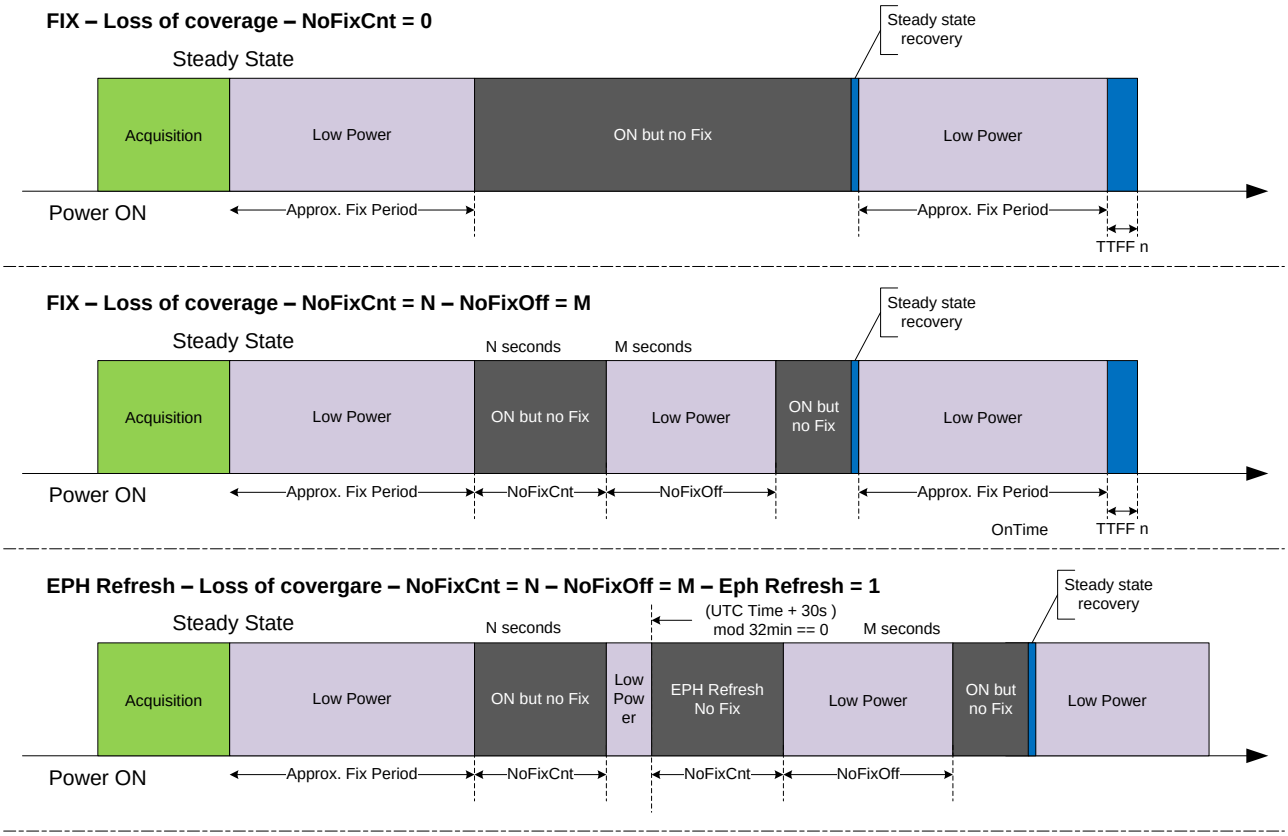


EPH Refresh - Good coverage – FixOnTime = 1 – RTC Calibration = 1



All sequences begin with an acquisition phase where all visible satellite ephemeris and almanacs are downloaded. The position of the first fix after the first Low Power period is approximate, but all next periods are regularly placed every “Fix Period”.
Sequence 3: Example of an ephemeris download period among the fixes.
Sequence 4: Example of the RTC calibration among the fixes.

4.1.3 Poor GNSS coverage sequences



In all sequences, the acquisition phase is ok and all ephemeris and almanacs are downloaded. The steady state in entered, but a loss of coverage occurs during the Low Power period.

Sequence 1: NoFixCnt = 0 means we don't alternate fix activities and low power periods. On the GNSS activation, the loss of coverage is detected and the GNSS will remain active until the recovery of the fix.

Sequence 2: As NoFixCnt is different from 0, the GNSS solution will remain active during N seconds and go back to low power state during M seconds. It will alternate this way until the fix is recovered.

Sequence 3: Despite the loss of coverage, the GNSS solution will try to decode the satellites when the ephemeris refresh activity is due. Instead of lasting 40 to 60s, the trial period will be only N seconds.

5 Control Interfaces

5.1 NMEA Interface

5.1.1 $PSTMLOWPOWERONOFF

A Command is defined to control the low power status. The table below summarizes the command supported by the ST NMEA layer:

Syntax	Description
`$PSTMLOWPOWERONOFF`	Command to set the low power features

Synopsis:

```
$PSTMLOWPOWERONOFF <low power enable/disable>, <constellation mask>,
<EHPE threshold>, <Max tracked sats>, <Switch constellation features >, <Duty
Cycle enable/disable>, <Duty Cycle ms signal off>, <Periodic mode>, <Fix
period>, <Number of fix>, <Ephemeris refresh>, <RTC refresh>, <No Fix
timeout>, <No Fix timeout Off duration><cr><lf>
```

Arguments:

Parameter	Format	Description
low power enable/disable	Decimal, 1 digit	General Low Power features Enable/Disable 0: OFF, 1: ON
Reserved settings		
Constellation mask	Decimal, 3 digits	Reserved
EHPE threshold	Decimal, 3 digits	Reserved
Max tracked sats	Decimal, 2 digits	Reserved
Switch constellation features	Decimal, 1 digit	Reserved, must be 0
Reserved settings		
Duty Cycle enable/disable	Decimal, 1 digit	Reserved, must be 0
Duty Cycle signal off	Decimal, 3 digits	Reserved
Periodic mode settings		

Periodic mode	Decimal, 1 digit	Setup Active or Standby periodic mode 0: OFF 1: Active Periodic mode 3: Standby Periodic mode
FixPeriod	Decimal, 5 digits	Interval between two fixes [s]
FixOnTime	Decimal, 2 digits	Number of fixes reported for each interval
Ephemeris refresh	Decimal, 1 digit	Enable/Disable the refresh of ephemeris data 0: OFF, 1: ON
RTC calibration	Decimal, 1 digit	Enable/Disable the RTC calibration 0: OFF, 1: ON
NoFixCnt	Decimal, 2 digits	Time to declare fix loss [s] in HOT conditions
NoFixOff	Decimal, 2 digits	Period of off period after a fix loss [s]

Parameter “low power enable/disable” in CDB-ID 200 must be set to “Enable” to have any of the low power features to be active.

5.2 Firmware Configuration

5.2.1 CDB-ID 200 - Application ON/OFF

Allow enabling/disabling low power features.

For each bit:

0 means feature disabled

1 means feature enabled

Bit	Bitmask	Function	Default
32	0x80000000	Low power enable	Not set

5.2.2 CDB-ID 257 – Periodic operating mode setting 1

Configure the periodic low power mode. This CBD has to be combined with CBD-258.This parameter includes different fields as reported in the following table:

Bits	Values	Description	Default
From B0 to B7	0/1 for each feature	Periodic feature set Enable/Disable: B0-B1: 00: Periodic mode OFF 01: Active Periodic mode 11: Standby Periodic mode B2: Ephemeris refresh required B3: RTC calibration required B4 to B7 are reserved for further usage.	B0-B1: 00 B2: 1 B3: 1
From B8 to B24	0..86400	FixPeriod [s]. 0 means the Fix will be given only on WAKEUP pin activation. Value 0 is only valid in Standby Periodic mode.	10
From B25 to B31	1..127	FixOnTime - Number of fix to report every fix wakeup.	1

5.2.3 CDB-ID 258 – Periodic operating mode setting 2

Configure the periodic low power mode. This CBD has to be combined with CBD-257.This parameter includes different fields as reported in the following table:

Bits	Values	Description	Default
From B0 to B7	0..255	NoFixCnt [s] - Time to declare fix loss in HOT conditions.	15
From B8 to B19	0..4095	NoFixOff [s] - Off duration time after a fix loss event.	180
From B20 to B28	0..300	NoFixCnt2 [s] – Time to declare fix loss in non-HOT conditions – startup case, obsolete ephemeris …	90

5.2.4 CDB-ID 259 – Low Power Mode HW Setting

Describe the state of each power supplies in the TESEO. The TESEO has a Backup LDO, LDO1, LDO2 and SMPS. Two different states are possible, the High and the Low frequency states, basically related to the TCXO ON or OFF state. The value 0 means OFF, any other values represent a voltage (1.0V 1.1V or 1.2V) or an ON state. The different frequency states are obtained by configuring the periodic mode. High frequency is used when the GNSS Library is active, the low frequency is used when the GNSS Library is inactive. During standby state, only the backup LDO is ON.

Check chapter 0 for further configuration details.

Bits	Values	Description	Default
B0-B1	0,1	Enable/disable the stop mode functionality of the backup LDO during High frequency periods. If stop mode functionality is enabled, the power consumption in standby mode is reduced. 0 = stop mode disabled 1= stop mode enabled	1
B2-B3	0,1	Enable/disable the stop mode functionality of the backup LDO during Low frequency periods. If stop mode functionality is enabled, the power consumption in standby mode is reduced. 0 = stop mode disabled 1= stop mode enabled	1
B4-B5	0,1,2,3	LDO1 status during High frequency mode 0 = OFF, 1 = 1.0V, 2 = 1.1V, 3 = 1.2V. If the LDO1 is configured in 1.8V, any value different from 0 means ON.	3

B6-B7	0,1,2,3	LDO1 status during Low frequency mode 0 = OFF, 1 = 1.0V, 2 = 1.1V, 3 = 1.2V. If the LDO1 is configured in 1.8V, any value different from 0 means ON.	3
B8-B9	0,1,2,3	LDO2 status during High frequency mode 0 = OFF, 1 = 1.0V, 2 = 1.1V, 3 = 1.2V.	3
B10-B11	0,1,2,3	LDO2 status during Low frequency mode 0 = OFF, 1 = 1.0V, 2 = 1.1V, 3 = 1.2V.	0
B8-B9	0,1,2,3	SMPS status during High frequency mode 0 = OFF, 1 = 1.0V, 2 = 1.1V, 3 = 1.2V.	2
B10-B11	0,1,2,3	SMPS status during Low frequency mode 0 = OFF, 1 = 1.0V, 2 = 1.1V, 3 = 1.2V.	2

5.3 Low power - APIs Overview for SDK

The purpose of this section is to provide an overview of the specific APIs available to control the low power algorithms.

5.3.1 GNSSAPP API

5.3.1.1 gnssapp_low_power_setup

Configure the low power algorithm implemented in the GNSS library and in the POWER service. The following function can only be called during the initialization in the gnssapp.c file.

Synopsis:

```
#include "gnss_api.h"

static void gnssapp_low_power_setup( gnss_app_lowpow_setup_type_t,
gnss_app_lowpow_standby_type_t, gnss_low_power_cyclic_mode_t *,
gnss_low_power_periodic_mode_t *);
```

Arguments:

`gnss_app_lowpow_setup_type_t`	INIT or UPDATE operation
`gnss_app_lowpow_standby_type_t`	Allow Standby mode or not
`gnss_low_power_cyclic_mode_t *`	pointer to cyclic configuration data structure
`gnss_low_power_periodic_mode_t *`	pointer to periodic configuration data structure

Results:

Errors:

None

Description:

This function setup the low power configuration towards the GNSS Lib and the services. The INIT operation is reserved for early initialization phase, and usually done in the gnssapp.c. For an UPDATE operation, usually called in an application (NMEA or a customer task), it is better to call the *gnssapp_low_power_setup_update* described below.

5.3.1.2 gnssapp_low_power_setup_update

Update the configuration of the low power algorithm implemented in the GNSS library and in the POWER service.

Synopsis:

```
#include "gnss_api.h"
#include "gnssapp.h"

void gnssapp_low_power_setup_update( gnss_app_lowpow_standby_type_t,
gnss_low_power_cyclic_mode_t *, gnss_low_power_periodic_mode_t *);
```

Arguments:

`gnss_app_lowpow_standby_type_t`	Allow Standby mode or not
`gnss_low_power_cyclic_mode_t *`	pointer to cyclic configuration data structure
`gnss_low_power_periodic_mode_t *`	pointer to periodic configuration data structure

Results:

Errors:

None

Description:

This function update the low power configuration towards the GNSS Lib and the services.

5.3.2 GNSS API

5.3.2.1 gnss_low_power_get_config_params

Return current configuration of low power algorithm implemented in the GNSS library

Synopsis:

```
#include "gnss_api.h"

void gnss_low_power_get_config_params(gnss_low_power_config_t *);
```

Arguments:

`gnss_low_power_config_t *` pointer to low power config data structure

Results:

Current configuration is returned in the input parameter

Errors:

None

Description:

Return the current configuration of the low power management.

5.3.2.2 gnss_low_power_get_status

Return the ON/OFF status of the low power management in the GNSS library

Synopsis:

```
#include "gnss_api.h"

boolean_t gnss_low_power_get_status(void);
```

Arguments:

None.

Results:

boolean_t: ON/OFF status (0=OFF 1=ON)

Errors:

None

Description:

Return the Enable/Disable status of the low power management algorithm.

5.3.2.3 gnss_low_power_get_data

Return data from low power management algorithm

Synopsis:

```
#include "gnss_api.h"

void  gnss_low_power_get_data(gnss_low_power_data_t *);
```

Arguments:

gnss_low_power_data_t *: pointer to the low power algorithm data structure

Results:

Current data from low power algorithm is returned in the input structure

Errors:

None

Description:

Return current status data from the low power algorithm.

5.3.2.4 gnss_low_power_get_nvm_sat_valid

Return the number of valid satellites stored by low power in NVM

Synopsis:

```
#include "gnss_api.h"

void  gnss_low_power_get_nvm_sat_valid (tUInt *);
```

Arguments:

`tUInt *:` pointer to a variable

Results:

Number of valid satellites stored in NVM

Errors:

None

Description:

The low power algorithm compute the number of valid satellites in selected constellations (cf. gnss_set_constellation_mask in GNSS API document). It is used to know how much downloaded data (ephemeris and almanac) are expected from the constellation. This number is stored in NVM, and can is returned in the pointer on the function call.

5.3.2.5 gnss_low_power_set_long_eph_refresh_timer

Set the low power ephemeris refresh period interval.

Synopsis:

```
#include "gnss_api.h"

void  gnss_low_power_set_long_eph_refresh_timer (boolean_t);
```

Arguments:

`boolean_t`: Short ephemeris refresh interval FALSE, Long ephemeris refresh interval TRUE

Results:

None

Errors:

None

Description:

Set the interval between ephemeris refresh intervals. Short is approximately 30 minutes, while long is about 10 hours. It is used by the STAGPS™ feature as soon as at least one ephemeris is available for each visible satellite of the constellations. With one ephemeris on a GPS satellite, STAPGS™ is able to make ephemeris predictions.

5.3.2.6 Low Power types

5.3.2.6.1 gnss_low_power_cyclic_mode_t

Type	Structure Member	Description
tU8	ehpe_threshold	Set the EHPE threshold for automatic activation of adaptive and cyclic mode
tU8	N_sats_reduced	Number of maximum satellites of the adaptive algorithm
gnss_sat_type_mask_t	const_mask_init	Set the constellation mask restored when EHPE is below the threshold
boolean_t	reduced_type	Enable/Disable adaptive algorithm
boolean_t	duty_cycle_on_off	Enable/Disable cyclic mode
tShort	ms_off	Duration of OFF period of the cyclic mode (milliseconds)

5.3.2.6.2 gnss_low_power_periodic_mode_t

Type	Structure Member	Description
tU32	fix_period	Distance between fix activities (seconds)
tU8	fix_on_time	Number of fix reported for each fix activities (number of fix)
tU8	EPH_refresh	Enable/Disable ephemeris refresh
tU8	RTC_refresh	Enable/Disable RTC calibration
tU8	NoFixTimeout	Time to declare fix loss (number of fix attempt)
tU16	NoFixOffTime	OFF duration time after No Fix Timeout occurred (seconds)
boolean_t	periodic_mode	Enable/Disable periodic mode

5.3.2.6.3 gnss_low_power_data_t

Type	Structure Member	Description
boolean_t	steady_state	If TRUE, the GNSS engine has completed the satellite acquisition phase
boolean_t	no_fix	Reserved
boolean_t	glonass_tow_referesh	Reserved
tChar	counter_glonass_eph_ON	Reserved
gnss_sat_type_mask_t	eph_const_mask	Reserved
tUInt	eph_interval	Reserved
boolean_t	eph_long_refresh_timer	Reserved
gnss_low_power_periodic_data_t	cyclic	Adaptive and Cyclic data
gnss_low_power_periodic_data_t	periodic	Periodic data

5.3.2.6.4 gnss_low_power_cyclic_data_t

Type	Structure Member	Description
gnss_low_power_state_t	state	Low Power algorithm status
tDouble	ehpe	Current Estimation of the Horizontal Position Error
tDouble	average_ehpe	EHPE Average
tDouble	average_ehpe_nmea	Reserved
tUChar	counter_ehpe_IN	Reserved
tUChar	counter_ehpe_OUT	Reserved
tUChar	counter_NO_FIX_IN	Reserved
tUChar	counter_NO_FIX_OUT	Reserved
tShort	average_NO_FIX	Reserved
tShort	sats_used	Reserved
boolean_t	reduced_type	Reserved
boolean_t	duty_cycle_on_off	Reserved
tInt	ms_off	Reserved
boolean_t	duty_cycle_state	Reserved

5.3.2.6.5 gnss_low_power_periodic_data_t

Type	Structure Member	Description
gpOS_clock_t	SavedTimerTicks	Reserved
tU16	NoFixCnt	Reserved
tU8	FixOnCnt	Reserved
boolean_t	NoFixCntSelector	Reserved
boolean_t	RTCTrimRequired	Reserved

5.3.3 Wakelock API

Each software task can register and acquire or release a wakelock. A wakelock can prevent the system to go in low frequency mode or in standby mode when it is acquired. Refer to document [OS20_Spec] for further details.

5.3.4 PWR service API

The purpose of this API is to collect the platform clocking resources needed by the software, control the low power hardware state and provide the OS tasks services to recover from a standby period.

5.3.4.1 Low power mode control

5.3.4.1.1 svc_pwr_set_lowpower_allowed

Synopsis:

```
#include "svc_pwr.h"

gpOS_error_t svc_pwr_set_lowpower_allowed ( boolean_t status)
```

Arguments:

`boolean_t status`: Indicates if the low power state is allowed when all wakelocks are released. The kind of low power is given by calling the function svc_pwr_set_standby_allowed.

Results:

None

Errors:

`gpOS_error_t`: Returns if an error occurred.

Description:

This API indicates if the low power hardware state can be entered when all wakelocks are released. It can be used jointly with `svc_pwr_set_standby_allowed` to define if the system will just enter a low frequency state or a STANDBY state.

5.3.4.1.2 svc_pwr_set_standby_allowed

Synopsis:

```
#include "svc_pwr.h"

gpOS_error_t svc_pwr_set_standby_allowed ( boolean_t status)
```

Arguments:

`boolean_t status:` Indicates if the STANDBY state is allowed.

Results:

None

Errors:

`gpOS_error_t:` Returns if an error occurred.

Description:

This API indicates if the low power hardware state entered when all wakelocks and peripherallocks are released is the STANDBY state. Low power mode must be previously activated to have effect.

5.3.4.1.3 svc_pwr_get_standby_allowed

Synopsis:

```
#include "svc_pwr.h"

boolean_t svc_pwr_get_standby_allowed ( void )
```

Arguments:

None

Results:

`boolean_t :` Indicates if the STANDBY state is allowed.

Errors:

None

Description:

This API returns if STANDBY state is allowed.

5.3.4.1.4 svc_pwr_get_standby_status

Synopsis:

```
#include "svc_pwr.h"

gpOS_error_t svc_pwr_get_standby_status ( boolean_t * status)
```

Arguments:

`boolean_t * status:` Pointer on STANDBY state.

Results:

None

Errors:

`gpOS_error_t`: Returns if an error occurred

Description:

This API returns if conditions are matching with STANDBY state entrance.

5.3.4.2 Start-up services

5.3.4.2.1 svc_pwr_StartupMode

Synopsis:

```
#include "svc_pwr.h"

svc_pwr_startup_mode_t svc_pwr_StartupMode ( void )
```

Arguments:

None

Results:

`svc_pwr_startup_mode_t`: Indicates the startup cause of the system divided in 3 categories:

1. SVC_PWR_STARTUP_POWER_ON : classic power on
2. SVC_PWR_STARTUP_WAKEUP_RTC : system has left standby state, wakeup source is RTC alarm
3. SVC_PWR_STARTUP_WAKEUP_PIN : system has left standby state, wakeup source is GPIO

Errors:

None

Description:

This API returns the startup cause.

5.3.4.2.2 svc_pwr_get_timer_adjustment

Synopsis:

```
#include "svc_pwr.h"

boolean_t svc_pwr_get_timer_adjustment( gpOS_clock_t *
StandbyDuration_rtcbase, gpOS_clock_t * VirtualPreviousOsTime );
```

Arguments:

`gpOS_clock_t * StandbyDuration_rtcbase`: Pointer on OS clock duration

`gpOS_clock_t * VirtualPreviousOsTime`: Pointer on Virtual previous OS time

Results:

`boolean_t` : Returns if time adjustment is available.

Errors:

None

Description:

This API returns STANDBY duration, computed with RTC inputs, and the virtual value of OS timer if standby duration did not happen. The both outputs are returned as OS clock base.

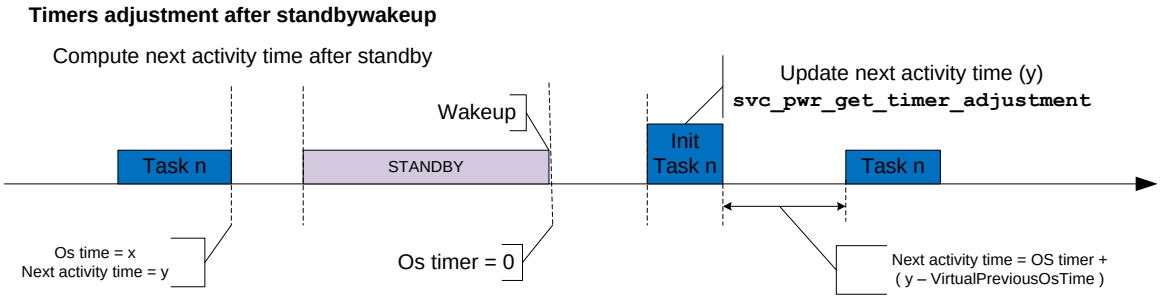


Figure 2 : Update next activity time after STANDBY wakeup

5.3.5 Examples

5.3.5.1 Initialization

This example can be found in file gnssapp.c function “gnssapp_gnss_lib_start”.

```
#include "svc_pwr.h"

    if( (svc_pwr_StartupMode() != SVC_PWR_STARTUP_POWER_ON) )
    {
      GPS_DEBUG_MSG(( "[gnssapp pwr] low power mode reload backup\r\n"));
      gnssapp_low_power_setup(GNSSAPP_LOW_POWER_INIT, gnssapp_lowpow.Standby,
&gnssapp_lowpow.cyclic, &gnssapp_lowpow.periodic );
    }
    else
    {
      if(sw_config_get_software_switch_status( LOW_POWER_ON_OFF_SWITCH
)!=FALSE)
      {
```

```
        gnss_low_power_cyclic_mode_t cyclic;
        gnss_low_power_periodic_mode_t periodic;
        gnss_app_lowpow_standby_type_t Standby;

        tUInt param1,param4,param5;

        GPS_DEBUG_MSG(( "[gnssapp pwr] low power mode reload NVM\r\n"));

        sw_config_get_param( CURRENT_CONFIG_DATA, LOW_POWER_CFG_PARAMS_1_ID,
&param1);
        sw_config_get_param( CURRENT_CONFIG_DATA, LOW_POWER_CFG_PARAMS_4_ID,
&param4);
        sw_config_get_param( CURRENT_CONFIG_DATA, LOW_POWER_CFG_PARAMS_5_ID,
&param5);

        /* Adaptive and Cyclic config bits */
        cyclic.reduced_type       = (boolean_t)((param1      ) & 0x01U);
        cyclic.duty_cycle_on_off  = (boolean_t)((param1 >> 1U) & 0x01U);
        /* 2 bits left */

        /* Adaptive and Cyclic settings */
        cyclic.ehpe_threshold     = (tU8)   ((param1 >> 4U) & 0xFFU);
        cyclic.N_sats_reduced     = (tU8)   ((param1 >> 12U) & 0xFFU);
        cyclic.ms_off             = (tShort)((param1 >> 20U) & 0xFFFU);

        /* ConsteLLation applicable */
        cyclic.const_mask_init = constellation_usage_selection;

        /* Periodic config bits */
        periodic.NoFixTimeout      = (tU8)      ((param5      ) & 0xFFU);
        periodic.NoFixOffTime      = (tU16)     ((param5 >> 8U) & 0xFFFU);
        periodic.periodic_mode     = (boolean_t)((param4      ) & 0x01U);

        periodic.EPH_refresh      = (tU8)((param4 >> 2U) & 0x01U);
        periodic.RTC_refresh      = (tU8)((param4 >> 3U) & 0x01U);
        /* 4 bits left */

        /* Periodic settings */
        periodic.fix_period       = (tU32)((param4 >> 8U) & 0x1FFFFU);
        periodic.fix_on_time      = (tU8) ((param4 >> 25U) & 0x7FU);

        if(((param4 >> 1U) & 0x01U) == 0x01U)
        {
          Standby = GNSSAPP_LOW_POWER_STANDBY_ENABLE;
        }
        else
        {
          Standby = GNSSAPP_LOW_POWER_STANDBY_DISABLE;
        }
```

```
        gnssapp_low_power_setup(GNSSAPP_LOW_POWER_INIT, Standby, &cyclic,
&periodic );

        if( svc_pwr_get_standby_allowed() == TRUE )
        {
          GPS_DEBUG_MSG(( "[gnssapp pwr] NVM standby mode activated\r\n"));
        }
      }
      else
      {
        gnssapp_low_power_setup(GNSSAPP_LOW_POWER_INIT,
GNSSAPP_LOW_POWER_STANDBY_DISABLE, NULL, NULL );
      }
    }
```

5.3.5.2 Update

This example can be found in file nmea.c function “nmea_cmdif_exec_setlowpower”.

```
#include "svc_pwr.h"

static void nmea_cmdif_exec_setlowpower( tChar *cmd_par )
{
  tChar out_msg[NMEA_OUT_MSG_BUFFER_SIZE];
  tInt index = 0;
  tInt field_count = 0;
  tInt on_off;
  tInt const_type;
  tInt ehpe_threshold;
  tInt N_sats_reduced;
  tInt reduced_type;
  tInt duty_cycle_on_off;
  tInt ms_off;
  tInt fix_period;
  tInt fix_on_time;
  tInt EPH_refresh;
  tInt RTC_refresh;
  tInt NoFixTimeout;
  tInt NoFixOffTime;
  tInt periodic_mode;
  gnss_app_lowpow_standby_type_t Standby;

  gnss_low_power_cyclic_mode_t cyclic;
  gnss_low_power_periodic_mode_t periodic;

  field_count = _clibs_sscanf( cmd_par,
",%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d", &on_off, &const_type,
&ehpe_threshold, &N_sats_reduced, &reduced_type, &duty_cycle_on_off, &ms_off,
```

```
&periodic_mode, &fix_period, &fix_on_time, &EPH_refresh, &RTC_refresh,
&NoFixTimeout, &NoFixOffTime );

  if ( ( field_count == 14 ) && ( on_off == 1 ) )
  {
    cyclic.ehpe_threshold         = (tU8)ehpe_threshold;
    cyclic.N_sats_reduced         = (tU8)N_sats_reduced;

    periodic.periodic_mode        = (boolean_t)((periodic_mode >> 0) & 0x01);
    periodic.fix_period           = (tU32)fix_period;
    periodic.fix_on_time          = (tU8)fix_on_time;
    periodic.EPH_refresh          = (tU8)EPH_refresh;
    periodic.RTC_refresh          = (tU8)RTC_refresh;
    periodic.NoFixTimeout         = (tU8)NoFixTimeout;
    periodic.NoFixOffTime         = (tU16)NoFixOffTime;

    if ( reduced_type == 0 )
    {
      if ( duty_cycle_on_off == 1 )
      {
        cyclic.reduced_type       = FALSE;
        cyclic.duty_cycle_on_off  = TRUE;
        cyclic.ms_off             = (tShort)ms_off;
      }
      else if ( duty_cycle_on_off == 0 )
      {
        cyclic.reduced_type       = FALSE;
        cyclic.duty_cycle_on_off  = FALSE;
        cyclic.ms_off             = (tShort)ms_off;
      }
    }
    else if ( reduced_type == 1 )
    {
      if ( duty_cycle_on_off == 1 )
      {
        cyclic.reduced_type       = TRUE;
        cyclic.duty_cycle_on_off  = TRUE;
        cyclic.ms_off             = (tShort)ms_off;
      }
      else if ( duty_cycle_on_off == 0 )
      {
        cyclic.reduced_type       = TRUE;
        cyclic.duty_cycle_on_off  = FALSE;
        cyclic.ms_off             = (tShort)ms_off;
      }
    }

    /* load constellation applicable */
    cyclic.const_mask_init = const_type;
```

```
    if(((periodic_mode >> 1) & 0x01) == 0x01)
    {
      Standby = GNSSAPP_LOW_POWER_STANDBY_ENABLE;
    }
    else
    {
      Standby = GNSSAPP_LOW_POWER_STANDBY_DISABLE;
    }
    gnssapp_low_power_setup_update( Standby, &cyclic, &periodic );

  }
  else if ( ( field_count == 2 ) && ( on_off == 0 ) )
  {
    gnssapp_low_power_setup_update( GNSSAPP_LOW_POWER_STANDBY_DISABLE , NULL,
NULL );
  }
}
Timer update after STANDBY wakeup
```

5.3.5.3 Task timer recovery

At STANDBY wake up, it's possible to recover data, included timers, if data are located in BACKUP RAM. The following sample, describes how to test STANDBY wakeup state and how to fix the timers value according to new OS time reference.

```
 BACKUP_DATA gpOS_clock_t NextActivityTime;

void Demo_Init()
{
  ……

  // Verify if come back from standby
  if( svc_pwr_StartupMode() == SVC_PWR_STARTUP_WAKEUP_RTC )
  {
    gpOS_clock_t StandbyDuration_rtcbase;
    gpOS_clock_t VirtualPreviousOsTime;

    // Get previous OS time reference to update next activity time according
to new os time reference
    if( svc_pwr_get_timer_adjustment( &StandbyDuration_rtcbase,
                                      &VirtualPreviousOsTime ) == TRUE )
    {
      // NEW TIMER =
      // NOW + ( PREVIOUS ACTIVITY TIME – PREVIOUS OS TIME REFERENCE)
      NextActivityTime =
gpOS_time_plus(gpOS_time_now(),gpOS_time_minus(NextActivityTime,VirtualPreviou
sOsTime));
    }
  }
}
```

6 Periodic mode application and test results

6.1 Application

6.1.1 Active Periodic mode

In this mode, two different clock settings are used. One configured by CBD-ID 130 (Typically 48f0 or 192f0) used during the FIX activities. Outside FIX activities, the RING oscillator (low power oscillator) at 4MHz is used to clock the ARM and Timers.

Here are limitations induced by this mode:

- PPS, WAAS, RTCM, DR features are not available,
- Timer's exactness is not guaranteed outside the fix activities,
- UART usage is not guaranteed outside the fix activities, waiting for the command acknowledge is advised,

6.1.2 Standby Periodic mode

This mode includes constraints of Active Periodic mode. Other parameters, like peripherals activities, affect the condition to enter standby state. Peripheral locks protect system from standby state during critical operation as flash memory swapping or writing and external peripheral access.

Moreover, wakeup periodicity is conditioned by calibration and navigation settings and all entities registered on wakelocks power services. That's why, each entities has the responsibility to check its next activity according to current time, after STANDBY wakeup.

It's strongly recommended to use Periodic operating mode setting 1 (CDB-ID 257) to store standby status. The GNSS application already implements a mechanism with this setting at start up. Nevertheless, power services offers interfaces to switch STANDBY state without affect setting, but wakelock interface keeps the main vector to forbid/allow STANDBY activity and inform the system of next activity.

Many parameters could affect consumption:

- Navigation fix & satellite refresh periodicity
- RTC Calibration
- Verbose output stream
- Flash swapping

SD card logging is optimized for STANDBY state. DEBUG and/or NMEA streams are logged into two different files, where data are concatenated into previous file. SDLOG will create new file for logging, only if service svc_pwr_StartupMode reports a `SVC_PWR_STARTUP_POWER_ON`.

6.2 Periodic modes measurements

Two hardware configurations are suitable for a power application, SMPS 3.3V and LDO 1.8V. For further implementation details and measurement procedure, please refer to document [HW_PwrStrgOvw].

6.2.1 Active Periodic mode

6.2.1.1 Measurement procedure

All tests are performed with Roof antenna.

Ephemeris and RTC Calibration are activated.

Ephemeris and Fix are asynchronous.

When the library is ON, all NMEA messages are ON too. So during Ephemeris refresh, the fix is sent on NMEA interface.

The following naming rules are used in the Active Periodic mode chapter:

2s_1fix correspond to the command:

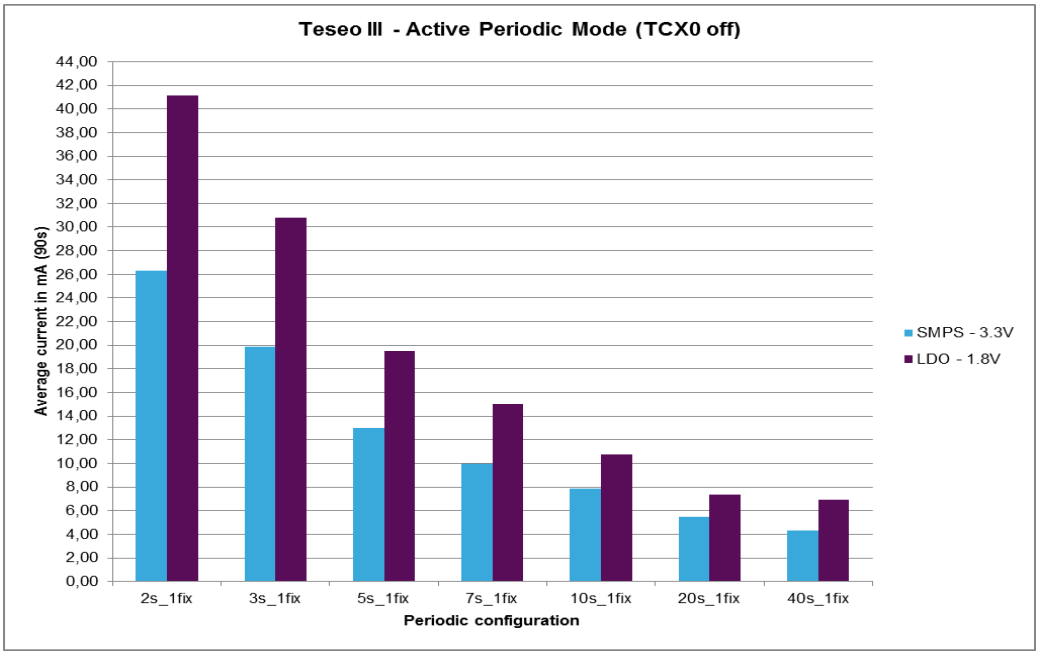
- $PSTMLOWPOWERONOFF,1,129,255,32,0,0,700,1,**2,1**,1,1,15,90

10s_5fix correspond to the command:

- $PSTMLOWPOWERONOFF,1,129,255,32,0,0,700,1,**10,5**,1,1,15,90

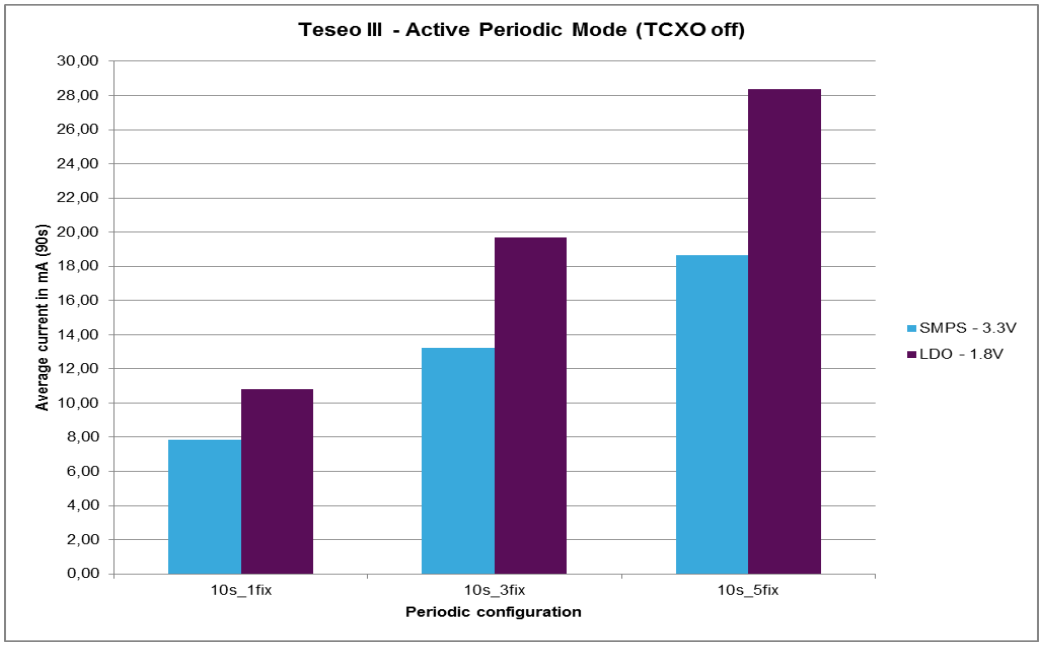
6.2.1.2 Measurements

The figures below summarize the total average for different fix period and 1 fix only as well as for LDO power supply scheme and SMPS power supply scheme (SMPS on baseband and LDO2 on RF).



Since during low frequency period the current consumption is about 0.5mA lower with LDO compared with SMPS (we recommend to use the LDO mode with fix period above 5 minutes).

The graph below represents the total average current versus the number of fix per period.



It is important to keep in mind that SQI is still working during the low frequency period of Active periodic. Therefore, only TCXO power supply can be switched off.

6.2.1.3 GPS Position accuracy

The position accuracy is measured in Active periodic mode over 1 hour duration with the roof antenna. From Logs are computed CEP (50).

As reference, CEP are also reported in continuous mode (24h).

Periodic configuration	CEP50 (m)
No periodic (24H continuous)	1.4
5s_1fix	2.36
7s_1fix	2.7
10s_1fix	2.37
10s_2fix	1.89
10s_5fix	1.68
20s_1fix	2.3

20s_5fix	2.38
20s_10fix	2.27
60s_1fix	2
60s_5fix	1.75
60s_10fix	1.75

6.2.2 Standby Periodic mode

6.2.2.1 Measurement procedure

In standby periodic mode, the configuration parameters must be stored in non-volatile memory (NVM or Back-up RAM) in order to recover the configuration when leaving stand-by state.

This is not supported with the existing $PSTMLOWPOWERONOFF. Therefore, configuration of Standby periodic mode is done by using $PSTMSETPAR command with the proper ID in order to store the configuration parameters in NVM.

ID257 is used to configure the Low Power Mode:

- Bit 0/1 = 3 (standby periodic mode)
- Bit 2 = 1 (ephemeris update approximately every 30 minutes activated)
- Bit 3 = 1 (RTC calibration activated)
- Bit 8:24: fix period value (s)
- Bit 25:31: number of fix

By default RTC calibration and ephemeris update are activated

Ephemeris and Fix are asynchronous.

When the library is ON, all NMEA messages are ON too. So during Ephemeris refresh, the fix is sent on NMEA interface.

The following naming rules are used in the document:

5s_1fix correspond to the command:

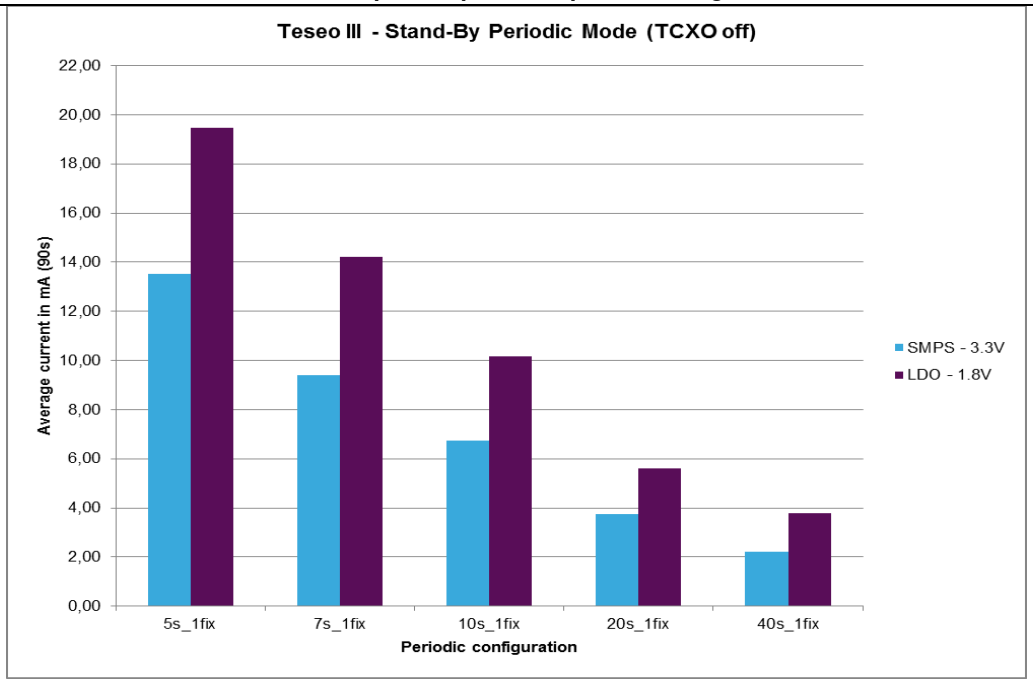
- $ PSTMSETPAR,1257,200050F

10s_5fix correspond to the command:

- $ PSTMSETPAR,1257,A000A0F

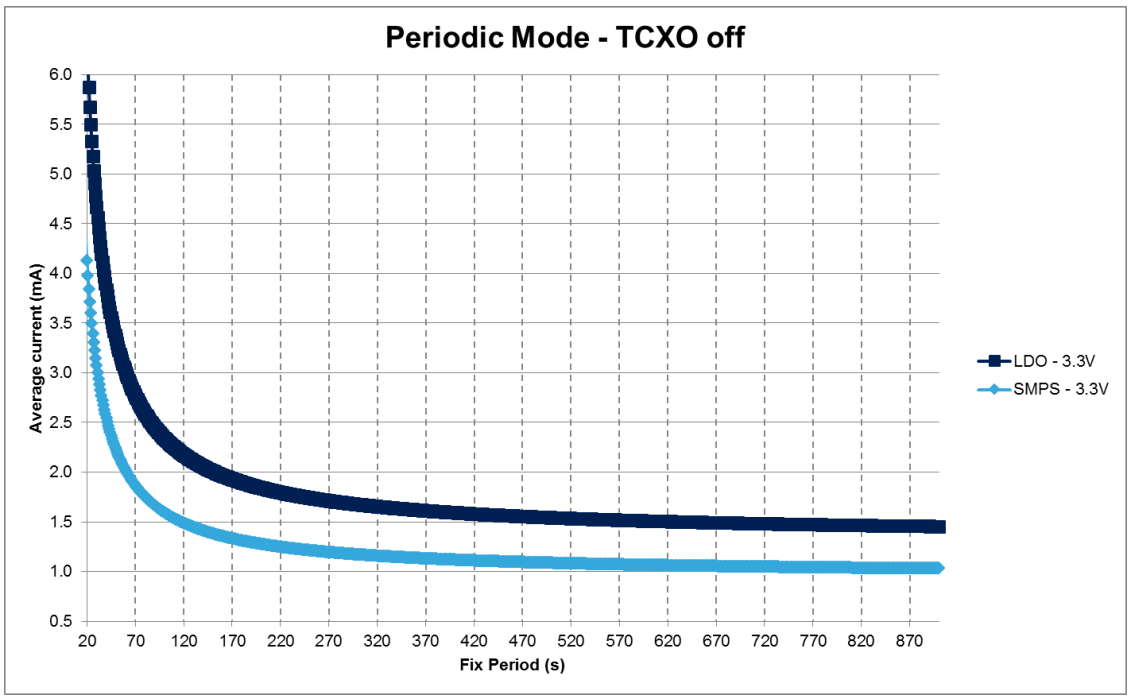
6.2.2.2 Measurements

The figures below summarize the total average current for different fix period and 1 fix only as well as for LDO power supply scheme and SMPS power supply scheme (SMPS on baseband and LDO2 on RF).

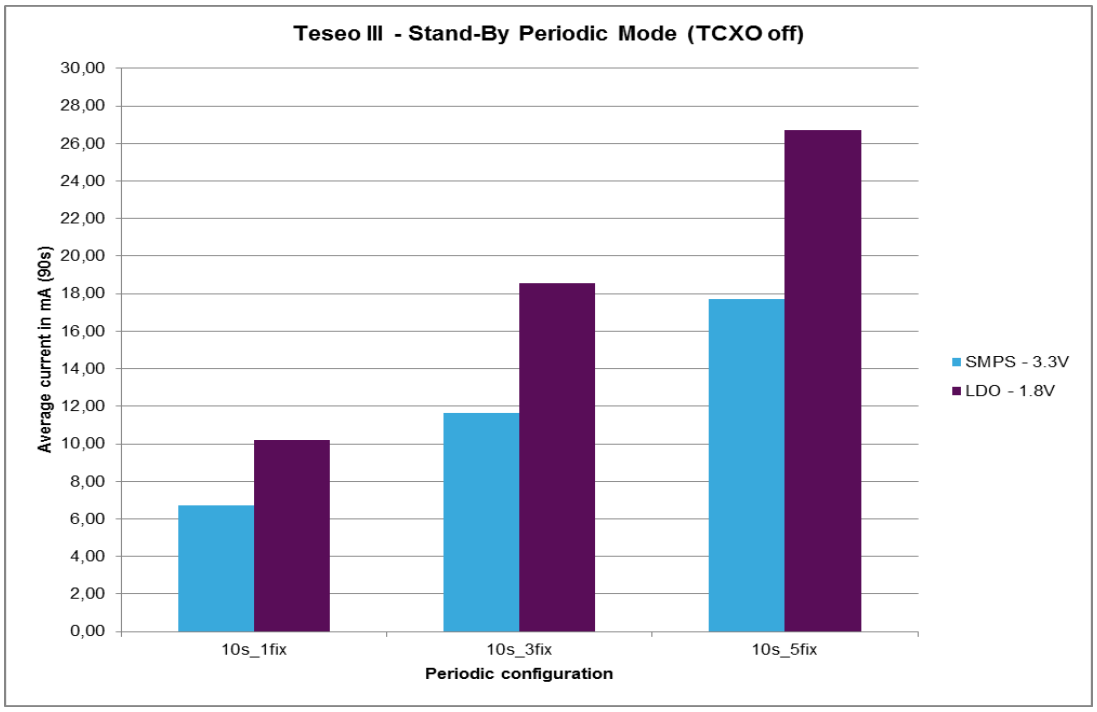


SMPS is the power scheme which gives the lowest current consumption especially for short stand-by period

Based on a computation model, a comparison between LDO and SMPS with Vbat=3.3V shows that delta is about 0.5mA for stand-by period above 230s. The delta is coming from ephemeris update current which is lower in SMPS mode.



The graph below represents the total average current versus the number of fix per period.



By increasing the number of fix per period, SMPS gain is more and more important.

6.2.2.3 GPS Position accuracy

The position accuracy is measured in Standby periodic mode over 1 hour duration with the roof antenna. From Logs are computed CEP (50).

As reference, CEP and SEP are also reported in continuous mode (24h).

Periodic configuration	CEP50 (m)
No periodic (24H continuous)	1.4
5s_1fix	3.01
7s_1fix	2.41
10s_1fix	1.76
10s_2fix	1.67
10s_5fix	1.95
20s_1fix	2.5
20s_5fix	1.90

20s_10fix	2.25
60s_1fix	1.58
60s_5fix	1.59
60s_10fix	1.5

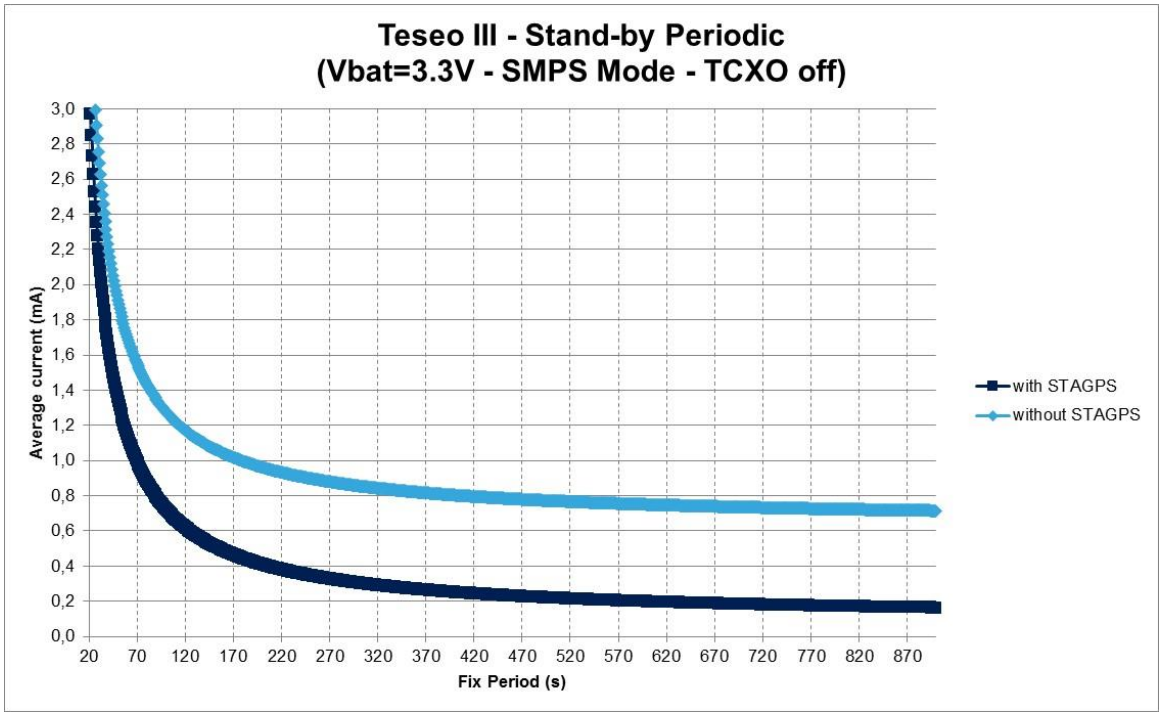
6.2.2.4 STAGPS™

To further reduce current consumption especially for long fix period, the solution is to use the STAGPS™ feature which allows to increase time between two ephemeris refresh.

The average current needed to maintain hot start conditions is then drastically reduced.

The table below compares the two cases:

Average current for ephemeris refresh	LDO	SMPS (Vbat=3.3V)	SMPS (Vbat=1.8V)
Without STAGPS	0.86mA	0.60mA	0.80mA
With STAGPS	0.07mA	0.05mA	0.07mA



7 Appendix A: Current measurement

7.1 Evaluation Board and TeseoIII

On the EVB RevE v1.1, J505 provides measurement on Vcc + IOs, whereas J504 provides measurement on backup domain.

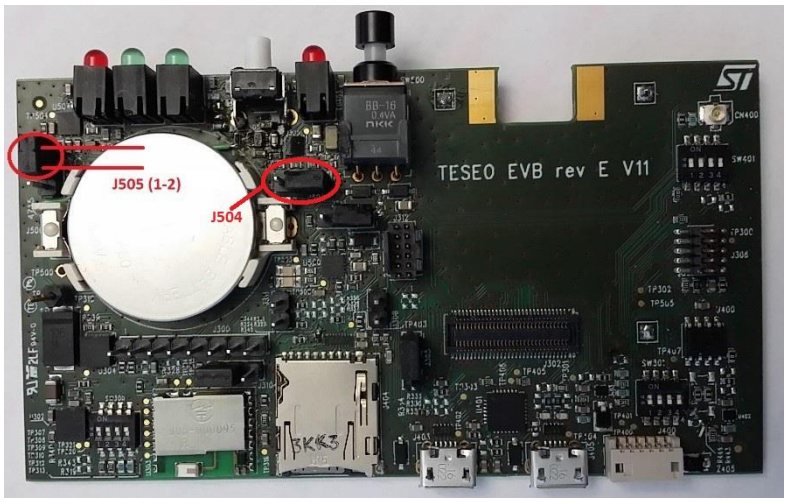


Figure 3 EVB TIII Measurement point

8 Appendix B: HW Settings - CDB-ID 259 Values

Power Supply	Default	LDO 1.8V	SMPS 3.3V
Backup LDO High	0x1	0x1	0x1
Backup LDO Low	0x1	0x1	0x1
LDO1 High	0x3	0x2	0x0
LDO1 Low	0x3	0x1	0x0
LDO2 High	0x3	0x3	0x3
LDO2 Low	0x0	0x0	0x0
SMPS High	0x2	0x0	0x2
SMPS Low	0x2	0x0	0x1
Overall Configuration	0xA3F5	0x0365	0x6305

9 Appendix C: Settings to maximize power saving

The following table summarize the best settings to obtain the maximum power saving:

Parameter	Value	Comment
Code in ITCM	Maximum	ITCM usage shorten code execution, thus shorten wakeup time. A binary standalone platform already maximize its usage. For SoC platform it is recommended to activate GNSSLIB and STAGPS™ FAST options.
CDB-ID 103	1	Debug log set to OFF. Reduce UART activity, shorten wakeup time.
CDB-ID 130	0x00	CPU frequency speed set to 192f0. Fasten position computation. Shorten wakeup time versus low consumption increase during wakeup time.
CDB-ID 200		
WAAS	Bit 2 OFF	Feature not applicable
STAGPS™	Bit 4 ON	After a training period, reduce ephemeris download periods.
Constellation	Bit 16 ON Bit 17 OFF Bit 18 OFF Bit 21 OFF Bit 22 ON Bit 23 OFF	GPS ON GLONASS OFF, QZSS OFF
PPS	Bit 24 OFF	Feature not applicable
Low Power	Bit 31 ON	Activate low power feature
CDB-ID 201, 210, 211, 228, 229, 230	minimum	NMEA message list reduced to its minimum Reduces UART activity, shorten wakeup time.
CDB-ID 227		
Constellation	Bit 7 OFF	Galileo OFF, COMPASS OFF

	Bit 8 OFF Bit 9 OFF Bit 10 OFF	
HW TCXO	48MHz	Speed up initialisation phase. Shorten wakeup time.
CDB-ID 245	0x0A	TCXO Frequency set to 48MHz
CDB-ID 257		
Standby	Bit 0 Bit 1 set to 0x3	Standby periodic mode
Eph required	Bit 2 ON	Ephemeris refresh set to ON. Ensure best compromise low power/accuracy.
RTC Calibration	Bit 3 ON	RTC calibration set to ON. Ensure best compromise low power/accuracy.
CDB-ID 259		Adapted to HW application settings

10 Disclaimer

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. 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 acknowledgement.

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 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.

ST and the ST logo are trademarks of ST. 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.