Earth Explorer
Mission CFI Software

EXPLORER_ORBIT
SOFTWARE USER MANUAL

Code:	CS-MA-DMS-GS-0004
Issue:	3.2
Date:	15/11/04

	Name	Function	Signature
Prepared by:	Mariano Sánchez Nogales	Project Engineer	
	Sara Cuenda Cuenda	Project Engineer	
	José Antonio González Abeytua	Project Manager	
	Juan José Borrego Bote	Project Manager	
Checked by:	José Antonio González Abeytua	Project Manager	
Approved by:	José Antonio González Abeytua	Project Manager	

DEIMOS Space S.L.
Ronda de Poniente, 19,
Edificio Fiteni VI, Portal 2, 2ª Planta, Tres Cantos
28760 Madrid, SPAIN
Tel.: +34 91 806 34 50
Fax: +34 91 806 34 51
E-mail: deimos@deimos-space.com

Document Information

Contract Data		Classification	
Contract Number:	15583/01/NL/GS	Internal	
		Public	
Contract Issuer:	ESA / ESTEC	Industry	X
		Confidential	

External Distribution		
Name	Organisation	Copies

Electronic handling	
Word Processor:	Framemaker 6.0
Archive Code:	P/SUM/DMS/01/026-011
Electronic file name:	cs-ma-dms-gs-0004-21

Document Status Log

Issue	Change Description	Date	Approval
1.0	New document	08/11/01	
1.1	• New xo_orbit_to_time, xo_time_to_orbit and xo_free_osf_records functions. • xo_cart_extra removal	23/05/02	
1.2 Draft	• Cosmetic changes • Updated error handling	19/07/02	
2.0	• Maintenance release. See change bars.	29/11/02	
2.1	• Maintenance release. See change bars.	13/05/03	
2.2	• New options for xo_propag_init_file and xo_interpol_init_file functions. • Maintenance release. See change bars.	30/09/03	
2.2.2	• Option to use a simplified algorithm to initialise using xo_propag_init_def • Absolute orbit and time since ANX calculated within xo_propag_extra and xo_interpol_extra functions. • Nodal period calculated by xo_orbit_info_from_<source> functions • Use of enumerations to size extra results arrays	26/04/04	
3.0	• New initialisation strategy for orbit calculations, propagation and interpolation. • New interfaces	21/07/04	
3.1	• Maintenance release	13/10/04	
3.2	• Maintenance release	15/11/04	

Table of Contents

List of Tables

List of Figures

1 SCOPE

The EXPLORER_ORBIT Software User Manual provides a detailed description of usage of the CFI functions included within the EXPLORER_ORBIT CFI software library.

2 ACRONYMS AND NOMENCLATURE

2.1 Acronyms

ANX	Ascending Node Crossing
AOCS	Attitude and Orbit Control Subsystem
CFI	Customer Furnished Item
EF	Earth Fixed reference frame
ESA	European Space Agency
ESTEC	European Space Technology and Research Centre
FOS	Flight Operations Segment
GS	Ground Station
OBT	On-board Binary Time
SSP	Sub-Satellite Point
SRAR	Satellite Relative Actual Reference
SUM	Software User Manual
TOD	True of Date reference frame
UTC	Universal Time Coordinated
UT1	Universal Time UT1
WGS[84]	World Geodetic System 1984

2.2 Nomenclature

CFI	A group of CFI functions, and related software and documentation. that will be distributed by ESA to the users as an independent unit
CFI function	A single function within a CFI that can be called by the user
Library	A software library containing all the CFI functions included within a CFI plus the supporting functions used by those CFI functions (transparently to the user)

3 APPLICABLE AND REFERENCE DOCUMENTS

3.1 Applicable documents

[GEN_SUM] Earth Explorer Mission CFI Software. General Software User Manual. CS-MA-DMS-GS-0002. Issue 3.2. 15/011/2004

3.2 Reference documents

[MCD] Earth Explorer Mission CFI Software. Mission Conventions Document. CS-MA-DMS-GS-0001. Issue 1.4 21/07/2004.

[F_H_SUM] Earth Explorer Mission CFI Software. EXPLORER_FILE_HANDLING Software User Manual. CS-MA-DMS-GS-0008. Issue 3.2. 15/11/2004

[LIB_SUM] Earth Explorer Mission CFI Software. EXPLORER_LIB Software User Manual. CS-MA-DMS-GS-0003. Issue 3.2. 15/11/2004.

[FORMATS] Earth Explorer File Format Guidelines. CS-TN-ESA-GS-0148.

4 INTRODUCTION

4.1 Functions Overview

This software library contains all the CFI functions allowing accurate computation of orbit state vectors, either at ascending node or (by propagation) at any point in the orbit of any Earth Explorer satellite.

The orbit propagation may be performed based on different propagation models. The initial set of models supported are:

- ***Mean Keplerian model***
- ***Spot model***

It includes an interpolator, orbit propagator and several routines used to feed the propagator with either simulated, predicted or restituted initial state vectors, i.e. the following CFI functions:

4.1.1 Orbit Initialisation

Before doing any orbit calculation, the orbit should be initialized using one of the following functions:

- ***xo_orbit_init_def***: this software generates a cartesian state vector around the true ascending node crossings as a function of the date (processing time), the longitude of the ascending node, the satellite Repeat Cycle Length, the mean local solar time and either the drift in mean local solar time or the inclination. For the Spot model, the routine generates the Spot elements.
- ***xo_orbit_cart_init***: This software initializes the orbit using as input a cartesian orbit state vector.
- ***xo_orbit_init_file***: For the Mean Keplerian model, this software reads Cartesian State Vectors. For the Spot model, this routine generates the Spot elements. The following input file types are accepted:
  - Flight Dynamics predicted ascending node state vectors.
  - DORIS Navigator Data
  - FOS Restituted Orbit Files
  - DORIS Preliminary Orbit
  - DORIS Precise Orbit
  - Ascending node state vectors from the Orbit Scenario File
  - State vectors from Spot orbit files.

In all cases a variable of the type xo_orbit_id (*Orbit ID.*) is returned. This variable is a CFI Identifier of the type described in [GEN_SUM]. This variable keeps internally a list of orbit state vectors that will be used in further calculations.

4.1.2 Orbit Propagation

- ***xo_propag_init***: This software initializes the propagation using a cartesian orbit state vector selected from the input Orbit ID.
- ***xo_propag***: This software is a propagator which allows accurate prediction of osculating Cartesian state vectors for user requested time segments.

4.1.3 Orbit Interpolation

- ***xo_interpol_init***: This software initializes the interpolation process using the cartesian state vectors selected from the input Orbit ID. The initialization provides ***xo_interpol*** with a set of orbit state vectors within a margin defined by the user.
- ***xo_interpol***: This software generates Extended Cartesian State Vectors based on the interpolation of orbit restituted state vector. The user defines the time for which an interpolated state vector has to be generated.

4.1.4 Ancillary Results Computation

- ***xo_propag_extra***: This software returns ancillary results, i.e. mean and osculating Keplerian orbit state vectors, satellite osculating true latitude, latitude rate and latitude rate-rate, Sun zenith angle and many more.
- ***xo_interpol_extra***: This software returns ancillary results, i.e. cartesian orbit state vectors, cartesian orbit state vector acceleration, mean and osculating Keplerian orbit state vectors, satellite osculating true latitude, latitude rate and latitude rate-rate and Sun zenith angle.

4.1.5 Time/Orbit Transformation

- ***xo_time_to_orbit***: This software calculates the absolute orbit, number of seconds and number of microseconds since ascending node that corresponds to a given time in processing format.
- ***xo_orbit_to_time***: This software calculates the time, in processing format, that corresponds to a given absolute orbit, number of seconds and number of microseconds since ascending node.

4.1.6 Orbit Information Parameters

- ***xo_orbit_rel_from_abs***: This software calculates the relative orbit, the phase number giving as input an absolute orbit number.
- ***xo_orbit_abs_from_rel***: This software calculates the absolute orbit number giving as input a relative orbit number and its cycle number.
- ***xo_orbit_abs_from_phase***: This software calculates the absolute orbit number, the relative orbit, the phase number giving as input a phase number.
- ***xo_orbit_info***: This software calculates orbit related parameters providing as input the absolute orbit number.

4.1.7 Clean-up Memory

- ***xo_orbit_close***: This software frees the memory allocated by the orbit initialization routines. It closes the xo_orbit_id, so that it cannot be used for further computations.
- ***xo_propag_close***: This software frees the memory allocated by the **xo_propag_init** routine. It closes the xo_propag_id, so that it cannot be used for further computations.
- ***xo_interpol_close***: This software frees the memory allocated by the **xo_interpol_init** routine. It closes the xo_interpol_id, so that it cannot be used for further computations.

4.2 Orbit Propagation Calling Sequence

A complete propagation sequence consists of:

- A call to any of the initialization routines for orbit, ***xo_orbit_init_def***, ***xo_orbit_init_file*** or ***xo_orbit_cart_init***, to generate the internal data necessary for whatever calculation involving orbits.
- A call to the ***xo_propag_init*** function for generate the internal data necessary for the propagation routines.
- An optional call to ***xo_propag_extra*** to calculate any desired ancillary result related to the initializing state vector.
- After initialization, the ***xo_propag*** routine should be called to perform the orbit propagation, taking into account the validity times computed during initialization.
- To obtain some ancillary results, the user might call the ***xo_propag_extra*** function.
- At the end of a sequence is mandatory to call ***xo_propag_close*** to free the memory allocated.

The possible propagation sequences of calls allowing to produce an orbit state vector are shown in figure 1.

4.3 Orbit Interpolation Calling Sequence

A complete interpolation sequence consists of:

- A call to any of the initialization routines for orbit, ***xo_orbit_init_def***, ***xo_orbit_init_file*** or ***xo_orbit_cart_init***, to generate the internal data necessary for whatever calculation involving orbits.
- A call to the ***xo_interpol_init*** routine, to generate the orbit state vector for the interpolation.
- ***xo_interpol*** function utilises the data generated during the initialisation to perform the interpolation.
- To obtain extra ancillary results, the user might call the ***xo_interpol_extra*** function.
- At the end of a sequence is mandatory to call ***xo_interpol_close*** to free the memory allocated.

The possible interpolation sequences of calls allowing to produce an orbit state vector are shown in figure 1.

4.4 Time/Orbit Transformation and Orbit Information Parameters Calling Sequence

A complete time/orbit transformation and orbit information parameters sequence consists of:

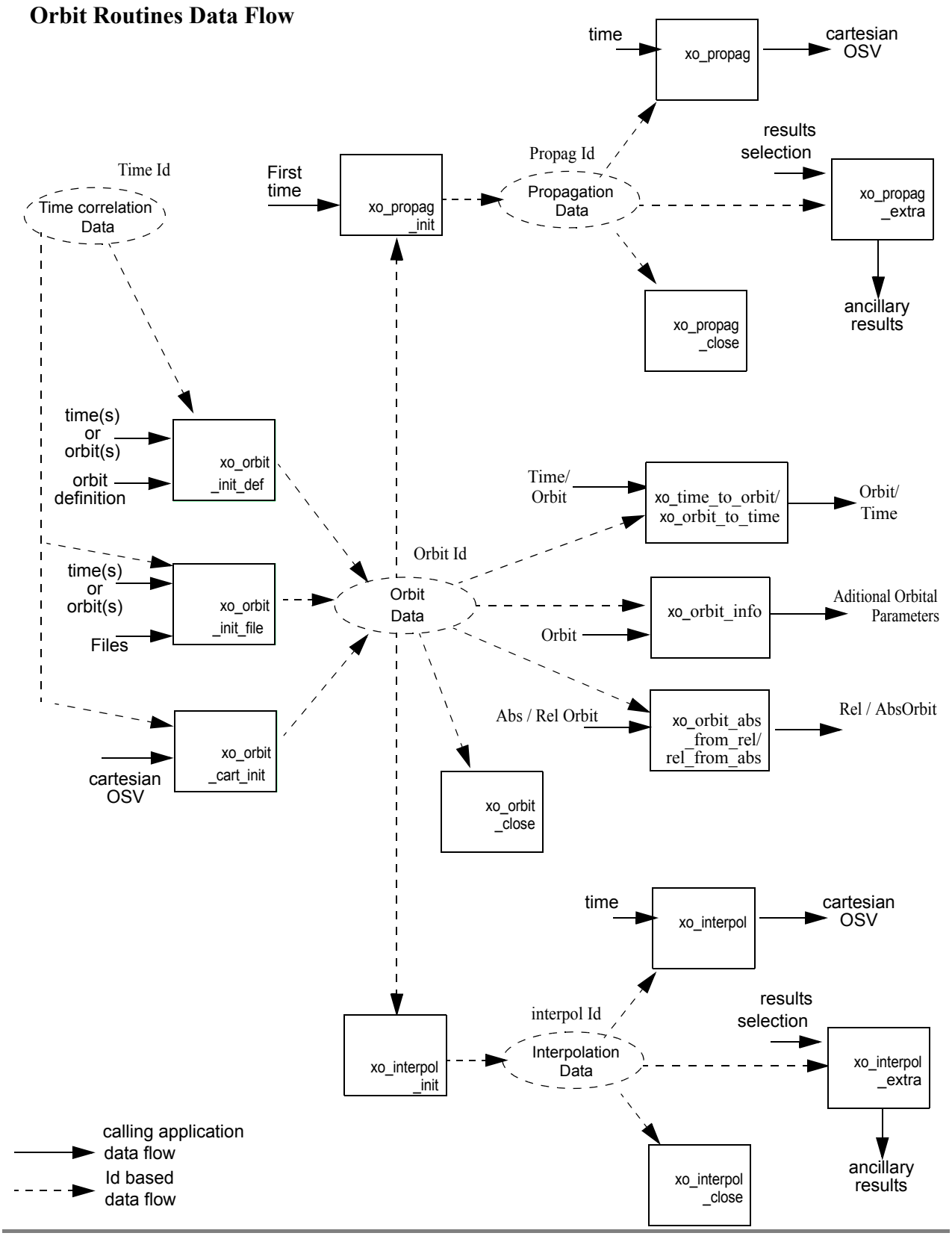
- A call to any of the initialization routines for orbit, ***xo_orbit_init_def***, ***xo_orbit_init_file*** or ***xo_orbit_cart_init***, to generate the internal data necessary for whatever calculation involving orbits. Note that time to orbit transformations cannot be computed if the orbit was initialised with ***xo_orbit_cart_init***.
- A call to a ***time/orbit transformation*** or an ***orbit information parameters*** routine.
- When no more ***time/orbit transformations*** and ***orbit information parameters*** routines are going to be used, call to ***xo_orbit_close*** to free the memory allocated.

The possible time/orbit transformation and orbit information parameters sequences of calls allowing to produce an orbit state vector are shown in figure 1.

A detailed description of each function is provided in section 7. Please refer also to:

- [MCD] for a detailed description of the time references and formats, reference frames, parameters and models used in this document.
- [GEN_SUM] for a complete overview of the CFI, and in particular the detailed description of the *Id* concept and the error handling functions.

Figure 1: Orbit Calling Sequence



5 LIBRARY INSTALLATION

For a detailed description of the installation of any CFI library, please refer to [GEN_SUM].

Note that example data files are provided with this CFI:

- Orbit files to be used with ***xo_orbit_init_file***

These files are orbit file examples.

6 LIBRARY USAGE

Note that to use the EXPLORER_ORBIT software library, the following other CFI software libraries are required:

- `EXPLORER_FILE_HANDLING` (See [F_H_SUM]).
- `EXPLORER_LIB` (See [LIB_SUM]).

It is also needed to have properly installed in the system the following external GPL libary:

- `LIBXML2` (see [GEN_SUM]).

To use the EXPLORER_ORBIT software library in a user application, that application must include in its source code either:

- `explorer_orbit.h` (for a C application)
- `explorer_orbit.inc` (for a ForTran application under SOLARIS/Linux)
- `explorer_orbit_win.inc` (for a ForTran application under Windows 95/98/NT/2000)

To link correctly this application, the user must include in his linking command flags like (assuming *cfi_lib_dir* and *cfi_include_dir* are the directories where respectively all CFI libraries and include files have been installed, see [GEN_SUM] for installation procedures):

- SOLARIS/LINUX:

```
-Icfi_include_dir -Lcfi_lib_dir -lexplorer_orbit -lexplorer_lib
                                -lexplorer_file_handling -lxml2
```

- WINDOWS:

```
/I "cfi_include_dir"  /libpath:"cfi_lib_dir"
                                libexplorer_orbit.lib
                                libexplorer_lib.lib
                                libexplorer_file_handling.lib
                                libxml2.lib
```

- MacOS:

```
-Icfi_include_dir -Lcfi_lib_dir -lexplorer_orbit -lexplorer_lib
                                -lexplorer_file_handling
                                -framework libxml
                                -framework libiconv
```

All functions described in this document have a name starting with the prefix `xo_`.

To avoid problems in linking a user application with the `EXPLORER_ORBIT` software library due to the existence of names multiple defined, the user application should avoid naming any global software item beginning with either the prefix `XO_` or `xo_`.

This is summarized in table 1.

Table 1: CFI functions included within EXPLORER_ORBIT library

Function Name	Enumeration value	long
Main CFI Functions		
xo_orbit_init_def	XO_ORBIT_INIT_DEF_ID	0
xo_orbit_cart_init	XO_ORBIT_CART_INIT_ID	1
xo_orbit_init_file	XO_ORBIT_INIT_FILE_ID	2
xo_orbit_close	XO_ORBIT_CLOSE_ID	3
xo_propag_init	XO_PROPAG_INIT_ID	4
xo_propag	XO_PROPAG_ID	5
xo_propag_extra	XO_PROPAG_EXTRA_ID	6
xo_propag_close	XO_PROPAG_CLOSE_ID	7
xo_interpol_init	XO_INTERPOL_INIT_ID	8
xo_interpol	XO_INTERPOL_ID	9
xo_interpol_extra	XO_INTERPOL_EXTRA_ID	10
xo_interpol_close	XO_INTERPOL_CLOSE_ID	11
xo_orbit_to_time	XO_ORBIT_TO_TIME_ID	12
xo_time_to_orbit	XO_TIME_TO_ORBIT_ID	13
xo_orbit_abs_from_rel	XO_ORBIT_ABS_FROM_REL_ID	14
xo_orbit_rel_from_abs	XO_ORBIT_REL_FROM_ABS_ID	15
xo_orbit_abs_from_phase	XO_ORBIT_ABS_FROM_PHASE_ID	16
xo_orbit_info	XO_ORBIT_INFO_ID	17
xo_run_init	XO_RUN_INIT_ID	18
Error Handling Functions		
xo_verbose	not applicable	
xo_silent		
xo_get_code		
xo_get_msg		
xo_print_msg		

Notes about the table:

- To transform the status vector returned by a CFI function to either a list of error codes or list of error messages, the enumeration value (or the corresponding integer value) described in the table must be used.
- The error handling functions have no enumerated value.

6.1 Usage hints

Every CFI function has a different length of the Error Vector, used in the calling I/F examples of this SUM and defined at the beginning of the library header file. In order to provide the user with a single value that could be used as Error Vector length for every function, a generic value has been defined (XO_ERR_VECTOR_MAX_LENGTH) as the maximum of all the Error Vector lengths. This value can therefore be safely used for every call of functions of this library.

6.2 General enumerations

The aim of the current section is to present the enumeration values that can be used rather than integer parameters for some of the input parameters of the EXPLORER_ORBIT routines, as shown in the table below. The enumerations presented in [GEN_SUM] are also applicable.

Table 2: Some enumerations within EXPLORER_ORBIT library

Input	Description	Enumeration value	Long
Propagation model	Mean Kepler elements model	XO_PROPAG_MODEL_MEAN_KEPL	0
	SPOT elements model	XO_PROPAG_MODEL_SPOT	1
	Auto initialization mode	XO_PROPAG_MODEL_AUTO	10
	Double initialization mode	XO_PROPAG_MODEL_DOUBLE	100
Non Sun-synchronous orbit characterisation	MLST drift	XO_NOSUNSYNC_DRIFT	0
	Inclination	XO_NOSUNSYNC_INCLINATION	1
	Selection of simplified algorithm (additive value)	XO_NOSUNSYNC_USE_SIM_MODEL	10
Time inputs selection	File	XO_SEL_FILE	0
	Time	XO_SEL_TIME	1
	Orbit	XO_SEL_ORBIT	2
	Default value	XO_SEL_DEFAULT	3
Orbit_info vector results calculation switch	Orbit_info vector results not calculated	XO_ORBIT_INFO_EXTRA_OFF	0
	Orbit_info vector results calculated	XO_ORBIT_INFO_EXTRA_ON	1
Interpolation model	Default	XO_INTERPOL_MODEL_DEFAULT	0
Orbit Init Model	Unknown mode	XO_ORBIT_INIT_UNKNOWN_MODE	-1
	Automatic detection of file	XO_ORBIT_INIT_AUTO	0
	Orbit Change mode	XO_ORBIT_INIT_ORBIT_CHANGE_MODE	1
	State Vector mode	XO_ORBIT_INIT_STATE_VECTOR_MODE	2
	Orbit Scenario File mode	XO_ORBIT_INIT_OSF_MODE	3
	Predicted Orbit File mode	XO_ORBIT_INIT_POF_MODE	4
	DORIS mode	XO_ORBIT_INIT_DORIS_MODE	5
	POF refined with DORIS mode	XO_ORBIT_INIT_POF_N_DORIS_MODE	6
	OSF part of the OEF mode	XO_ORBIT_INIT_OEF_OSF_MODE	7
	POF part of the OEF mode	XO_ORBIT_INIT_OEF_POF_MODE	8
	Maximum value of enumeration	XO_ORBIT_INIT_MAX_VALUE	9

The use of the previous enumeration values could be restricted by the particular usage within the different CFI functions. The actual range to be used is indicated within a dedicated reference named **allowed range**. When there are not restrictions to be mentioned, the allowed range column is populated with the label **complete**.

7 CFI FUNCTIONS DESCRIPTION

The following sections describe each CFI function.

The calling interfaces are described both for C users and ForTran users.

Input and output parameters of each CFI function are described in tables, where C programming language syntax is used to specify:

- Parameter types (e.g. long, double)
- Array sizes of N elements (e.g. param[N])
- Array element M (e.g. [M])

ForTran users should adapt the tables using ForTran syntax equivalent terms:

- Parameter types (e.g. long <=> INTEGER*4, double <=>REAL*8)
- Array sizes of N elements (e.g. param[N] <=> param (N))
- Array element M (e.g. [M] <=> (M+1))

7.1 xo_orbit_init_def

7.1.1 Overview

The **xo_orbit_init_def** routine generates a Cartesian orbit state vector around the true ascending node crossings. The result is stored and returned through the xo_orbit_id variable so that can fed other routines involving orbit calculations. The data generated by the **xo_orbit_init_def** function is based on:

- Date (processing time),
- Longitude of the ascending node,
- Satellite Repeat Cycle and Cycle Length
- Mean local solar time at ascending node
- Drift of mean local solar time or the inclination

The user should take into account that **xo_orbit_init_def** only retrieve and stores internal data for one orbit.

The validity start and stop times of the initialization (***val_time0*** and ***val_time1*** output parameters) represents the allowed time window for orbit calculations. If the **xo_orbit_init_def** function is called, this time window starts at 01/01/1950 00:00:00 and ends at 31/12/2099 23:59:59.

Before calling this function it is required to initialise the time correlations, using either **xl_time_ref_init** or **xl_time_ref_init_file** EXPLORER LIB functions (see [LIB_SUM]).

Warning: The algorithm used in this function is only valid for satellites with a finite valid range for the inclination and the semi-major axis of the orbit. In CRYOSAT, for example, as there are no minimum and maximum values defined of these two orbital elements, there are defined provisional ranges of the same size as the ones defined in ENVISAT until new requirements are defined. The nominal values have been taken from the [MCD]. There is not available any other nominal orbital element for any other satellite, so this routine is only valid (at this moment) for both CRYOSAT and ENVISAT.

A complete calling sequence of the orbit calculations procedure is presented in section 4.2.

7.1.2 Calling interface

The calling interface of the **xo_orbit_init_def** CFI function is the following (input parameters are underlined):

```
#include <explorer_orbit.h>
{
      long sat_id, propag_model, time_ref, time_init_mode;
      xl_time_id time_id = {NULL};
      xo_orbit_id orbit_id = {NULL};
      long drift_mode, irep, icyc;
      long orbit0, orbit;
      double time0, time, val_time0, val_time1;
      double ascmlst_drift, inclination, rlong, ascmlst;
      long status, ierr[XO_NUM_ERR_PROPAG_INIT_DEF];
      status = xo_orbit_init_def (&sat_id, &time_id,
                                  &time_ref, &time0, &orbit0,
                                  &drift_mode,
                                  &ascmlst_drift,  &inclination,
```

```
                                        &irep, &icyc, &rlong, &ascmlst,
                                        &val_time0, &val_time1,
                                        &orbit_id, ierr);
}
```

For ForTran programs, the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
#include <explorer_orbit.h>

      INTEGER*4 SAT_ID, PROPAG_MODEL, TIME_REF, TIME_INIT_MODE
      INTEGER*4 DRIFT_MODE, IREP, ICYC
      INTEGER*4 ORBIT0, ORBIT
      REAL*8 TIME0, TIME, VAL_TIME0, VAL_TIME1
      REAL*8 ASCMLST_DRIFT, INCLINATION, RLONG, ASCMLST
      INTEGER*4 STATUS, IERR(XO_NUM_ERR_PROPAG_INIT_DEF)

      STATUS = XO_ORBIT_INIT_DEF (SAT_ID, TIME_ID, PROPAG_MODEL,
     &                            TIME_REF, TIME0, ORBIT0,
     &                            TIME_INIT_MODE, TIME, ORBIT,
     &                            DRIFT_MODE, ASCMLST_DRIFT,
     &                            INCLINATION, IREP, ICYC, RLONG,
     &                            ASCMLST,
     &                            VAL_TIME0, VAL_TIME1,
     &                            ORBIT_ID, IERR)
```

7.1.3 Input parameters

The **xo_orbit_init_def** CFI function has the following input parameters:

Table 3: Input parameters of xo_orbit_init_def function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
time_id	xl_time_id*	-	Structure that contains the time correllations	-	-
time_ref	long*	-	Time reference ID	-	Complete
time0	double*	-	Reference time	Decimal days (Processing format)	[-18262.0,36524.0]
orbit0	long*	-	Absolute orbit number of the reference orbit	-	>= 0
drift_mode	long*	-	Flag to select between drift in mean local solar time and inclination as input characterization of the reference orbit. Note: When initializing a Sun-synchronous orbit, the selected drift mode must be *XO_NOSUNSYNC_DRIFT* and the ascmlst_drift parameter must be set to zero. Note 2: Add *XO_NOSUNSYNC_USE_SIM_MODEL* to the drift mode to select the simplified model in the algorithm.	-	XO_NOSUNSYNC_DRIFT, XO_NOSUNSYNC_INCLINATION, XO_NOSUNSYNC_DRIFT + XO_NOSUNSYNC_USE_SIM_MODEL, XO_NOSUNSYNC_INCLINATION + XO_NOSUNSYNC_USE_SIM_MODEL
ascmlst_drift	double*	-	If *drift_mode* = *XO_NOSUNSYNC_DRIFT* Drift in mean local solar time of the reference orbit: · MLST[N+1]=MLST[N]+MLSTdrift See drift_mode entry in this table.	seconds/day	TBD
inclination	double*	-	If *drift_mode* = *XO_NOSUNSYNC_INCLINATION* Inclination of the reference orbit	deg	[0,180]
irep	long *	-	Repeat cycle of the reference orbit The actual repeat cycle is calculated as per definition included in [MCD].	days	> 0
icyc	long *	-	Cycle length of the reference orbit	orbits	> 0

Table 3: Input parameters of xo_orbit_init_def function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
rlong	double*	-	Geocentric longitude of the [Earth fixed] ascending node (Earth fixed CS)	deg	[0,360)
ascmlst	double*	-	Mean local solar time at ascending node	Decimal days	[0, 24)

It is possible to use enumeration values rather than integer values for some of the input arguments:

- Satellite ID: sat_id. See [GEN_SUM]..
- Time reference ID: time_ref. See [GEN_SUM].
- Time initialisation mode: time_init_mode. See [GEN_SUM].
- Drift mode: drift_mode. Current document, section 6.2.

7.1.4 Output parameters

The output parameters of the **xo_propag_init_def** CFI function are:

Table 4: Output parameters of xo_propag_init_def function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_propag_init_def	long	-	Main status flag	-	-1, 0, +1
val_time0	double*	-	Validity start time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
val_time1	double*	-	Validity stop time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
orbit_id	xo_orbit_id*	-	Structure that contains the orbit initialization.	-	-
ierr[XO_NUM_ERR_ORBIT_INIT_DEF]	long	all	Status vector	-	-

7.1.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_orbit_init_def** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (`WARN`) or an error (`ERR`), the cause of such a message and the impact on the performed calculation, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xo_orbit_init_def** CFI function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 5: Error messages of xo_orbit_init_def function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Wrong satellite flag	No calculation performed	XO_CFI_ORBIT_INIT_DEF_SAT_ERR	0
ERR	Wrong input flag	No calculation performed	XO_CFI_ORBIT_INIT_DEF_FLAG_ERR	1
ERR	Could not perform a time transformation	No calculation performed	XO_CFI_ORBIT_INIT_DEF_TIME_CHANGE_ERR	
ERR	Input out of range	No calculation performed	XO_CFI_PROPAG_INIT_DEF_INPUTS_ERR	2
ERR	An error ocurred in the gen-state routine	No calculation performed	XO_CFI_PROPAG_INIT_DEF_GENSTATE_ERR	4
ERR	Memory Error	No calculation performed	XO_CFI_ORBIT_INIT_DEF_MEMORY_ERR	5

7.1.6 Runtime performances

The following runtime performance has been measured.

Table 6: Runtime performances of xo_orbit_init_def function

Ultra Sparc II-400[ms]
TBD

7.2 xo_orbit_cart_init

7.2.1 Overview

This software initializes the orbit data using as input a Cartesian orbit state vector.

The validity start and stop times of the initialization (***val_time0*** and ***val_time1*** output parameters) represents the allowed time window for orbit calculations. If the **xo_orbit_cart_init** function is called, this time window starts at 01/01/1950 00:00:00 and ends at 31/12/2099 23:59:59.

Before calling this function it is required to initialise the time correlations, using either **xl_time_ref_init** or **xl_time_ref_init_file** EXPLORER LIB functions (see [LIB_SUM]).

A complete calling sequence of the orbit calculations procedure is presented in section 4.2.

7.2.2 Calling interface

The calling interface of the **xo_orbit_cart_init** CFI function is the following (input parameters are underlined):

```
#include <explorer_orbit.h>
{
      xl_time_id time_id = {NULL};
      xo_orbit_id orbit_id = {NULL};
      long sat_id, time_ref, abs_orbit;
      double time, pos[3], vel[3], val_time0, val_time1;
      long status, ierr[XO_NUM_ERR_PROPAG_CART_INIT];
      status = xo_orbit_cart_init(&sat_id, &time_id,
                                  &time_ref, &time,
                                  pos, vel, &abs_orbit,
                                  &val_time0, &val_time1,
                                  &orbit_id, ierr);
}
```

For ForTran programs, the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
#include <explorer_orbit.inc>

      INTEGER*4 SAT_ID, PROPAG_MODEL, TIME_REF
      REAL*8 TIME, POS(3), VEL(3), VAL_TIME0, VAL_TIME1
      INTEGER*4 STATUS, IERR(XO_NUM_ERR_PROPAG_CART_INIT)

      STATUS = XO_ORBIT_CART_INIT(SAT_ID, PROPAG_MODEL, TIME_REF,
     &                            TIME, POS, VEL, VAL_TIME0,
     &                            VAL_TIME1, IERR)
```

7.2.3 Input parameters

The **xo_orbit_cart_init** CFI function has the following input parameters:

Table 7: Input parameters of xo_orbit_cart_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
time_id	xl_time_id*	-	Structure that contains the time correllations	-	-
time_ref	long*	-	Time reference ID	-	Complete
time	double*	-	Reference time	Decimal days (Processing format)	[-18262.0,36524.0]
pos	double[3]	all	Initial osculating position vector (X, Y, Z) (EF reference frame)	m	-
vel	double[3]	all	Initial osculating velocity vector (X, Y, Z) (EF reference frame)	m/s	-
abs_orbit	long*	-	Orbit of the state vector	-	> 0

It is possible to use enumeration values rather than integer values for some of the input arguments:

- Satellite ID: sat_id. See [GEN_SUM].
- Time reference ID: time_ref. See [GEN_SUM].

7.2.4 Output parameters

The output parameters of the **xo_orbit_cart_init** CFI function are:

Table 8: Output parameters of xo_orbit_cart_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_propag_cart_init	long	-	Main status flag	-	-1, 0, +1
val_time0	double*	-	Validity start time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
val_time1	double*	-	Validity stop time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
orbit_id	xo_orbit_id*	-	Structure that contains the orbit initialization.	-	-
ierr[XO_NUM_ERR_ORBIT_CART_INIT]	long	all	Status vector	-	-

7.2.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_orbit_cart_init** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xo_orbit_cart_init** CFI function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 9: Error messages of xo_orbit_cart_init function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Wrong Satellite Id.	No calculation performed	XO_CFI_ORBIT_CART_INIT_SAT_ERR	0
ERR	Wrong input flag	No calculation performed	XO_CFI_ORBIT_CART_INIT_FLAG_ERR	1
ERR	Input Time Id. is not initialized.	No calculation performed	XO_CFI_ORBIT_CART_INIT_TIME_STATUS_ERR	2
ERR	Orbit Id is already initialized.	No calculation performed	XO_CFI_ORBIT_CART_INIT_STATUS_ERR	3
ERR	Time conversion error.	No calculation performed	XO_CFI_ORBIT_CART_INIT_TIME_TRANSFORMING_ERR	4
ERR	Time out of limits.	No calculation performed	XO_CFI_ORBIT_CART_INIT_TIME_RANGE_ERR	5
ERR	Memory allocation error.	No calculation performed	XO_CFI_ORBIT_CART_INIT_MEMORY_ERR	6

7.2.6 Runtime performances

The following runtime performance has been measured.

Table 10: Runtime performances of xo_orbit_cart_init function

Ultra Sparc II-400[ms]
TBD

7.3 xo_orbit_init_file

7.3.1 Overview

The **xo_orbit_init_file** function is used for initializing the orbit calculations using one of these orbit files:

- One or more FOS Predicted ascending node cartesian state vectors file. In case multiple files are used, the files should be time ordered and the gap between them (i.e. time difference between the last vector of nth file and the first vector of the nth+1 file) should be less than two orbital periods.
- One FOS Predicted Orbit File plus a DORIS Navigator unconsolidated level-0 products file.
- One Orbit Scenario File providing orbital changes.
- One or more Orbit Event files.
- One or more FOS Restituted orbit files.
- One or more DORIS Navigator files.
- One or more DORIS Predicted files.
- One or more DORIS Preliminary files.
- State vectors from Spot orbit files.

The format of these files is described in [FORMATS].

Before calling this function it is required to initialise the time correlations, using either **xl_time_ref_init** or **xl_time_ref_init_file** EXPLORER LIB functions (see [LIB_SUM]).

The user can select the time interval to be used from the input file(s) using three different ways:

Table 11: User requested time range in xo_orbit_init_file

time_mode (see 7.8.3)	input parameter	requested start time (t_req_start)	requested stop time (t_req_stop)
XL_SEL_TIME	time0 / time1	time0	time1
XL_SEL_ORBIT	orbit0 / orbit1	$t_{ANX(orbit0)}$	$t_{ANX(orbit1)}$
XL_SEL_FILE	-	first state vector in the file(s)	last state vector in the file(s)

The validity start and stop times of the initialization (**val_time0** and **val_time1** output parameters) represents the allowed time window for orbit calculation. The following table shows the validity time interval for the different input files:

Table 12: Validity periods for xo_orbit_init_file

Input file type	val_time0	val_time1
Orbit file providing Orbit changes	ANX Time of the first orbital change	Infinity
Orbit files providing a list of orbital state vectors	time of the first state vector	Time of the last state vector

A complete calling sequence of the orbit calculation procedure is presented in section 4.2.

7.3.2 Calling interface

The calling interface of the **xo_orbit_init_file** CFI function is the following (input parameters are underlined):

```
#include <explorer_orbit.h>
{
       xl_time_id time_id = {NULL};
       xo_orbit_id orbit_id = {NULL};
       long sat_id, propag_model, n_files, time_mode;
       long time_ref, orbit0, orbit1;
       char **input_files;
       double time0, time1, val_time0, val_time1;
       long status, ierr[XO_NUM_ERR_PROPAG_INIT_FILE];

       status = xo_orbit_init_file (&sat_id, &time_id,
                                    &orbit_file_mode, &n_files,
                                    input_files,
                                    &time_mode, &time_ref,
                                    &time0, &time1, &orbit0, &orbit1,
                                    &val_time0, &val_time1,
                                    &orbit_id, ierr);
}
```

For ForTran programs, the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
#include <explorer_orbit.inc>

       INTEGER*4 SAT_ID, PROPAG_MODEL, N_FILES
       CHARACTER*LENGTH_NAME ORBIT_FILE(N_FILES)
       INTEGER*4 TIME_INIT_MODE, TIME_REF, ORBIT0, ORBIT1
       REAL*8 TIME0, TIME1, VAL_TIME0, VAL_TIME1
       INTEGER*4 STATUS, IERR(XO_NUM_ERR_PROPAG_INIT_FILE)

       STATUS = XO_ORBIT_INIT_FILE (SAT_ID, PROPAG_MODEL,N_FILES,
      &                             ORBIT_FILE, TIME_INIT_MODE,
      &                             TIME_REF, TIME0, TIME1, ORBIT0,
      &                             ORBIT1, VAL_TIME0, VAL_TIME1, IERR)
```

Note that `N_FILES` must be set to the number of input files of that type, with a maximum value of 16, whereas `LENGTH_NAME` must be set to the maximum string length of the filenames of that type . All strings in ForTran must end in "\0" (for compatibility with C programs).

7.3.3 Input parameters

The **xo_orbit_init_file** CFI function has the following input parameters:

Table 13: Input parameters of xo_orbit_init_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
time_id	xl_time_id*	-	Structure that contains the time correlations	-	-
orbit_file_mode	long*	-	Flag that indicates the type of the input orbit file. There exists the posibility of detecting automatically the type of the files using the value XO_ORBIT_INIT_AUTO. This value cannot be used for ENVI-SAT. The Orbit Event files are used as Orbit Scenario files if the AUTO mode is selected. In case they want to be used as Predicted orbit files, the option XO_ORBIT_INIT_OEF_POF_MODE should be chosen.	-	XO_ORBIT_INIT_AUTO XO_ORBIT_INIT_OSF_MODE XO_ORBIT_INIT_POF_MODE XO_ORBIT_INIT_ROF_MODE XO_ORBIT_INIT_DORIS_MODE XO_ORBIT_INIT_POF_N_DORIS_MODE XO_ORBIT_INIT_OEF_OSF_MODE XO_ORBIT_INIT_OEF_POF_MODE
n_files	long	-	Number of input files	-	>=1
input_files	char**	-	Vector of orbit files	-	-
time_init_mode	long*	-	Flag for selecting the time range of the initialisation.	-	Select either: · XO_SEL_FILE · XO_SEL_ORBIT · XO_SEL_TIME
time_ref	long*	-	Time reference ID	-	Complete
time0	double*	-	Start time. See section 7.8.1. Used only if: · *time_init_mode=XO_SEL_TIME*	Decimal days (Processing format)	[-18262.0,36524.0]
time1	double*	-	Stop time. Used only if: · *time_init_mode=XO_SEL_TIME*	Decimal days (Processing format)	[-18262.0,36524.0]
orbit0	long*	-	Absolute orbit number of the start orbit. Used only if: · *time_init_mode=XO_SEL_ORBIT*	-	-
orbit1	long*		Absolute orbit number of the stop orbit. Used only if: · *time_init_mode=XO_SEL_ORBIT*	-	-

It is possible to use enumeration values rather than integer values for some of the input arguments:

- Satellite ID: sat_id. See [GEN_SUM].

- Orbit init mode: orbit_init_mode. Current document, section 6.2.
- Time mode: time_init_mode. See [GEN_SUM].
- Time reference ID: time_ref. See [GEN_SUM].

7.3.4 Output parameters

The output parameters of the **xo_orbit_init_file** CFI function are:

Table 14: Output parameters of xo_orbit_init_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_propag_init_file	long	-	Main status flag	-	-1, 0, +1
val_time0	double*	-	Validity start time of the initialization	Decimal days (Processing format)	see 7.8.1
val_time1	double*	-	Validity stop time of the initialization	Decimal days (Processing format)	see 7.8.1
orbit_id	xo_orbit_id*	-	Structure that contains the orbit initialization data	-	-
ierr[XO_NUM_ERR_ORBIT_INIT_FILE]	long	all	Status vector	-	-

7.3.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_orbit_init_file** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xo_orbit_init_file** CFI function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 15: Error messages of xo_orbit_init_file function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Wrong satellite flag.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_SAT_ERR	0
ERR	Wrong input flag.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_FLAG_ERR	1
ERR	The Time Id was not initialized.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_TIME_STATUS_ERR	2
ERR	The Orbit Id is already initialized.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_ORBIT_STATUS_ERR	3
ERR	Memory allocation error.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_MEMORY_ERR	4
ERR	Could not detect input files.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_INPUT_FILES_ERR	5
ERR	Error reading OSF.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_WRONG_OSF_FILE_FORMAT_ERR	6
ERR	Wrong time on input.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_TIME_INPUT_INCORR_ERR	7
ERR	Error while processing DORIS file.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_DORIS_INIT_ERR	8
ERR	Time Conversion Error.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_TIME_CONVERSION_ERR	9
ERR	Error reading input files.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_READ_FILES_ERR	10
ERR	No data read within the input range.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_NO_ENOUGH_DATA_ERR	11
ERR	Error while computing ANX data for the state vectors	No calculation performed	XO_CFI_ORBIT_INIT_FILE_INTERPOL_INIT_ANX_ERR	12

Table 15: Error messages of xo_orbit_init_file function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error computing the orbit number for every state vector	No calculation performed	XO_CFI_ORBIT_INIT_FILE_CALC_ORBIT_ERR	13
WARN	Warnings while computing ANX data	Calculation performed.	XO_CFI_ORBIT_INIT_FILE_INTERPOL_INIT_ANX_WARN	14
WARN	Warnings during DORIS initialization	Calculation performed.	XO_CFI_ORBIT_INIT_FILE_DORIS_INIT_WARN	15
WARN	Warnings while reading the input file list	Calculation performed.	XO_CFI_ORBIT_INIT_FILE_READ_FILES_WARN	16

7.3.6 Runtime performances

The following runtime performances have been measured:

Table 16: Runtime performances of xo_orbit_init_file function

Ultra Sparc II-400[msec]
TBD

7.4 xo_orbit_close

7.4.1 Overview

The **xo_orbit_close** function is used to free the memory allocated by the other orbit initialization routines, and it must be called after using them.

A complete calling sequence of the propagation procedure is presented in section 4.2.

7.4.2 Calling interface

The calling interface of the **xo_orbit_close** CFI function is the following (input parameters are underlined):

```
#include <explorer_orbit.h>
{
       xo_orbit_id orbit_id = {NULL};
       long ierr[XO_NUM_ERR_ORBIT_CLOSE]
       long status;

       status = xo_orbit_close (&orbit_id, ierr);
}
```

For ForTran programs, the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
#include <explorer_orbit.inc>

       INTEGER*4 SAT_ID,
       INTEGER*4 STATUS

       STATUS = XO_ORBIT_CLOSE (SAT_ID)
```

7.4.3 Input parameters

The **xo_orbit_close** CFI function has the following input parameters:

Table 17: Input parameters of xo_orbit_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id *	-	Structure that contains the orbit initialization	-	-

7.4.4 Output parameters

The output parameters of the **xo_orbit_close** CFI function are:

Table 18: Output parameters of xo_orbit_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
ierr[XO_NUM_ERR_ORBIT_CLOE]	long	all	Status vector	-	-
xo_orbit_close	long	-	Main status flag	-	-1, 0, +1

7.4.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_orbit_close** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (`WARN`) or an error (`ERR`), the cause of such a message and the impact on the performed calculation, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xo_orbit_close** CFI function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 19: Error messages of xo_orbit_close function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Could not close the Orbit Id.	The Orbit Id. was not closed.	XO_CFI_ORBIT_CLOSE_WRONG_ID_ERR	0

7.4.6 Runtime performances

The following runtime performances have been measured:

Table 20: Runtime performances of xo_orbit_close function

Ultra Sparc II-400[msec]
TBD

7.5 xo_run_init

7.5.1 Overview

The **xo_run_init** CFI function adds to the *run Id* the *orbit id,* and, optionally, the *propag Id* or the *interpol Id.*

It is not possible to assign the same *run Id* to both a *propag Id* and an *interpol Id*.

7.5.2 Calling interface

The calling interface of the **xo_run_init** CFI function is the following:

```
#include <explorer_orbit.h>
{
       long run_id;
       xo_orbit_id orbit_id = {NULL};
       xo_propag_id propag_id = {NULL};
       xo_interpol_id interpol_id = {NULL};
       long ierr[XO_NUM_ERR_RUN_INIT], status;
       status = xo_run_init (&run_id, &orbit_id,
                             &propag_id, &interpol_id,
                             ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (note that the C preprocessor must be used because of the presence of the `#include` statement):

```
TBD
```

7.5.3 Input parameters

The **xo_run_init** CFI function has the following input parameters:

Table 21: Input parameters of xo_run_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
run_id	long *	-	Run ID	-	>=0
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
propag_id	xo_propag_id*	-	Structure that contains the propagator data	-	-
interpol_id	xo_interpol_id*	-	Structure that contains the interpolator data	-	-

7.5.4 Output parameters

The output parameters of the **xo_run_init** CFI function are:

Table 22: Output parameters of xo_run_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_run_init	long	-	Status flag	-	-
run_id	long *	-	Run ID	-	>=0
ierr	long	-	Error vector	-	-

7.5.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_run_init** CFI function after translating the returned extended status flag into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (`WARN`) or an error (`ERR`), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the extended status flag returned by the **xo_run_init** function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM])

Table 23: Error messages of xo_run_init function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Inputs Id no initialized or incompatible.	No calculation performed	XO_CFI_RUN_INIT_STATUS_ERR	0
ERR	Memory allocation error.	No calculation performed	XO_CFI_RUN_INIT_MEMORY_ERR	1
ERR	Input Ids incompatible with the run_id.	No calculation performed	XO_CFI_RUN_INIT_INCONSISTENCY_ERR	2

7.5.6 Runtime performances

The following runtime performances have been extimated (runtime is smaller than CPU clock and it is not possible to perfom loops for measuring it).

Table 24: Runtime performances of xo_run_init function

Ultra Sparc II-400 [ms]
TBD

7.6 xo_run_get_ids

7.6.1 Overview

The **xo_run_get_ids** CFI function returns the *ids* being used..

7.6.2 Calling interface

The calling interface of the **xo_run_get_ids** CFI function is the following:

```
#include <explorer_orbit.h>
{
       long run_id;
       xo_orbit_id orbit_id = {NULL};
       xo_propag_id propag_id = {NULL};
       xo_interpol_id interpol_id = {NULL};
       long status;
       status = xo_run_get_ids (&run_id,
                                &orbit_id,
                                &propag_id,
                                &interpol_id);
}
```

For ForTran programs the declaration and calling procedure is as follows (note that the C preprocessor must be used because of the presence of the `#include` statement):

```
TBD
```

7.6.3 Input parameters

The **xo_run_get_ids** CFI function has the following input parameters:

Table 25: Input parameters of xo_run_get_ids function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
run_id	long *	-	Run ID	-	>=0

7.6.4 Output parameters

The output parameters of the **xo_run_get_ids** CFI function are:

Table 26: Output parameters of xo_run_get_ids function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_run_get_ids	long	-	Status flag	-	-
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
propag_id	xo_propag_id*	-	Structure that contains the propagator data	-	-
interpol_id	xo_interpol_id*	-	Structure that contains the interpolator data	-	-

7.6.5 Warnings and errors

TBW

7.6.6 Runtime performances

The following runtime performances have been extimated (runtime is smaller than CPU clock and it is not possible to perfom loops for measuring it).

Table 27: Runtime performances of xo_run_get_ids function

Ultra Sparc II-400 [ms]
TBD

7.7 xo_run_close

7.7.1 Overview

The **xo_run_close** CFI function cleans up any memory allocation performed by the initialization functions.

7.7.2 Calling interface

The calling interface of the **xo_run_close** CFI function is the following:

```
#include <explorer_orbit.h>
{

        long run_id;
        status = xo_run_close (&run_id);
}
```

For ForTran programs the declaration and calling procedure is as follows (note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_orbit.inc>

       INTEGER*4 RUN_ID
       STATUS = xo_RUN_CLOSE (RUN_ID)
```

7.7.3 Input parameters

The **xo_run_close** CFI function has the following input parameters:

Table 28: Input parameters of xo_run_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
run_id	long *	-	Run ID	-	>=0

7.7.4 Output parameters

The output parameters of the **xo_run_close** CFI function are:

Table 29: Output parameters of xo_run_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xl_run_close	long	-	Status flag	-	-

7.7.5 Warnings and errors

TBW

7.7.6 Runtime performances

The following runtime performances have been extimated (runtime is smaller than CPU clock and it is not possible to perfom loops for measuring it).

Table 30: Runtime performances of xo_run_close function

Ultra Sparc II-400 [ms]
TBD

7.8 xo_propag_init

7.8.1 Overview

The **xo_propag_init** function is to be used on the ground segment near real time processing chains. This software is used in conjunction with the appropriate propagation initialization routine to initialize the orbit propagator with the necessary internal data.

The propagation initialization routine called depends on the propagation model used (indicated by an input parameter).

Before calling this function it is required to initialise the orbit with one of the following modes:

- XO_ORBIT_INIT_ORBIT_CHANGE_MODE
- XO_ORBIT_INIT_STATE_VECTOR_MODE
- XO_ORBIT_INIT_OSF_MODE
- XO_ORBIT_INIT_POF_MODE
- XO_ORBIT_INIT_POF_N_DORIS_MODE
- XO_ORBIT_INIT_OEF_POF_MODE
- XO_ORBIT_INIT_OEF_OSF_MODE

When using one or more Predicted Orbit files, it is possible to initialize the propagation with these additional modes (not mutually exclusive):

- **Auto mode**: This mode allows to propagate the space craft along the complete initialization range. In the normal **Mean Keplerian** model, the validity range is limited to ±2 orbits. In the **Auto** model, the software automatically re-initialize, transparently to the user, for the closer ANX to the propagation time.
- **Double mode:** the two ANX covering the propagation time are used. When calling **xo_propag**, the propagation is performed from each of the ANX and then a weighted average is done. The weight function is :

$$\cos^2\left(\frac{\pi}{2}\cdot\frac{\Delta t}{T}\right)$$

where $\Delta t = t - t_{ANX}$ and T is the nodal period of the orbit.

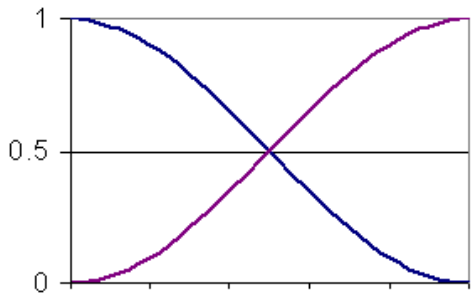


Figure 2: Weight Function for Double Propagation Model

This propagation method removes any discontinuity that may arise when changing the state vector around the true ascending node crossing used to propagate.

The propagation is initialized using a orbital state vector at an ANX. This ANX is chosen in the following way:

- The user have two options:
  - introduce an specific time or orbit (= t_0)
  - ask for a default value. In that case the seleteted time (t_0) is the half value of the time of first state vector (***t_start***) within the input orbit_id plus the time of the last state vector(***t_stop***)
- The ANX used in the initialization depends on the propagation model parameter:
  - **Mean Kepler mode**: ANX of the orbit closer to the t_0 time (within the ***t_start*** / ***t_stop*** range)
  - **Mean Kepler + Double mode:** the closer two ANX covering the t_0.
    If t_0 is less than the ***t_start***, only the first ANX (ANX at ***t_start***)will be chosen and double propagation will no be performed. In the same way, if t_0 is greater than ***t_stop***, only the last ANX (ANX at ***t_stop***) will be chosen and double propagation will not be performed.
  - **Mean Kepler + Auto mode:** the first ANX read from the file(s).
  - **Mean Kepler + Auto + Double mode:** the two firsts ANX read from the file.
    If t_0 is less than ***t_start***, only the first ANX (ANX at ***t_start***) will be chosen and double propagation will no be performed. In the same way, if t_0 is greater than ***t_stop*** (ANX at ***t_stop***), only the last ANX will be chosen and double propagation will no be performed.

The validity start and stop times of the initialization (**val_time0** and **val_time1** output parameters) represents the allowed time window for propagation. The following table shows the validity time interval for the different propagation models. The horizontal line represents the part of the file(s) read (***t_start*** to ***t_stop***), while the tick marks are the ANX times. Square brackets represent the validity period for propagation. When using the auto model, the propagation is re-initialized when the time jumps out of the region in brackets. The red arrow(s) represent the chosen ANX depending on the t_0 value.

Table 31: Validity Time Intervals for Propagation

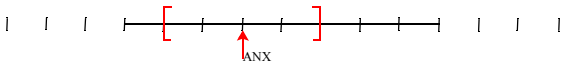
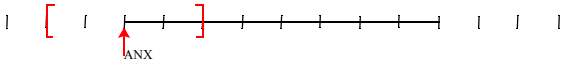
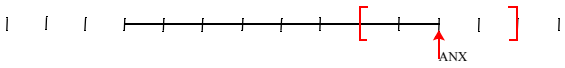
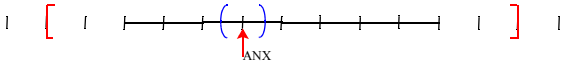
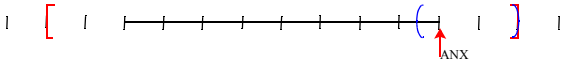
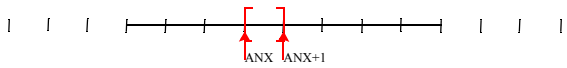
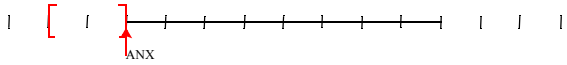
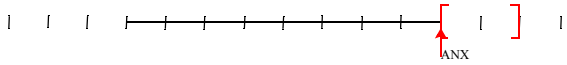
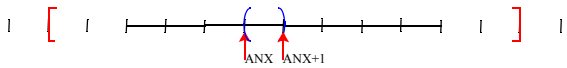
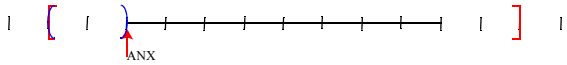
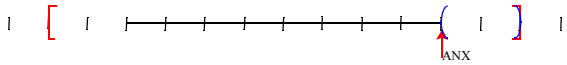
Propag model	t_0 value	Validity time interval
Mean Kepler	t_start < t_0 < t_stop	[ANX - 2 orbits, ANX + 2 orbits] 
	t_0 < t_start	[ANX - 2 orbits, ANX + 2 orbits] 
	t_0 > t_stop	[ANX - 2 orbits, ANX + 2 orbits] 
Mean Kepler + Auto Mode	t_start < t_0 < t_stop	[t_start - 2 orbits, t_stop + 2 orbits]; (ANX - 1/2 orbit, ANX + 1/2 orbit) 
	t_0 < t_start	[t_start - 2 orbits, t_stop + 2 orbits]; (ANX - 2 orbits, ANX + 1/2 orbit) 
	t_0 > t_stop	[t_start - 2 orbits, t_stop + 2 orbits]; (ANX - 1/2 orbit, ANX + 2 orbits) 

Table 31: Validity Time Intervals for Propagation

Propag model	t_0 value	Validity time interval
Mean Kepler + Double Mode	t_start < t_0 < t_stop	[ANX, ANX + 1 orbit] 
	t_0 < t_start	[ANX - 2 orbits, ANX] 
	t_0 > t_stop	[ANX, ANX + 2 orbits] 
Mean Kepler + Auto + Double Mode	t_start < t_0 < t_stop	[t_start - 2 orbits, t_stop + 2 orbits]; (ANX, ANX + 1orbit) 
	t_0 < t_start	[t_start - 2 orbits, t_stop + 2 orbits]; (ANX - 2 orbits, ANX) 
	t_0 > t_stop	[t_start - 2 orbits, t_stop + 2 orbits]; (ANX , ANX + 2 orbits) 

Note that, at the end, with a file it is possible to propag to times that are ± 2 orbits out of the validity time interval of the file.

A complete calling sequence of the propagation procedure is presented in section 4.2.

7.8.2 Calling interface

The calling interface of the **xo_propag_init** CFI function is the following (input parameters are underlined):

```
#include <explorer_orbit.h>
{
       xo_orbit_id *orbit_id = {NULL};
       xo_propag_id *propag_id = {NULL};
       long propag_model, time_mode;
       long time_ref, orbit;
       double time, val_time0, val_time1;
       long status, ierr[XO_NUM_ERR_PROPAG_INIT_FILE];

       status = xo_propag_init (&orbit_id, &propag_model,
                                &time_mode, &time_ref,
                                &time, &orbit,
```

```
                                &val_time0, &val_time1,
                                &propag_id, ierr);
}
```

For ForTran programs, the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
#include <explorer_orbit.inc>

      INTEGER*4 SAT_ID, PROPAG_MODEL, N_FILES
      CHARACTER*LENGTH_NAME ORBIT_FILE(N_FILES)
      INTEGER*4 TIME_INIT_MODE, TIME_REF, ORBIT0, ORBIT1
      REAL*8 TIME0, TIME1, VAL_TIME0, VAL_TIME1
      INTEGER*4 STATUS, IERR(XO_NUM_ERR_PROPAG_INIT_FILE)

      STATUS = XO_PROPAG_INIT (SAT_ID, PROPAG_MODEL,N_FILES,
     &                                 ORBIT_FILE, TIME_INIT_MODE,
     &                                 TIME_REF, TIME0, TIME1, ORBIT0,
     &                                 ORBIT1, VAL_TIME0, VAL_TIME1, IERR)
```

Note that `N_FILES` must be set to the number of input files of that type, with a maximum value of 16, whereas `LENGTH_NAME` must be set to the maximum string length of the filenames of that type . All strings in ForTran must end in "\0" (for compatibility with C programs).

7.8.3 Input parameters

The **xo_propag_init** CFI function has the following input parameters:

Table 32: Input parameters of xo_propag_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that containg the orbit initialization	-	-
propag_model	long*	-	Propagation model ID	-	Complete
time_mode	long*	-	Flag for selecting the time for which the function selects the ANX used to initialize the propagator.	-	Select either: · XO_SEL_ORBIT · XO_SEL_TIME · XO_SEL_DEFAULT
time_ref	long*	-	Time reference ID	-	Complete
time	double*	-	Start time. See section 7.8.1. Used only if: · *time_init_mode=XO_SEL_TIME*	Decimal days (Processing format)	[-18262.0,36524.0]
orbit0	long*	-	Absolute orbit number of the start orbit. Used only if: · *time_init_mode=XO_SEL_ORBIT*	-	-

It is possible to use enumeration values rather than integer values for some of the input arguments:

- Propagation model ID: propag_model. Current document, section 6.2.
- Time mode: time_init_mode. See [GEN_SUM].
- Time reference ID: time_ref. See [GEN_SUM].

7.8.4 Output parameters

The output parameters of the **xo_propag_init** CFI function are:

Table 33: Output parameters of xo_propag_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_propag_init_file	long	-	Main status flag	-	-1, 0, +1
val_time0	double*	-	Validity start time of the initialization	Decimal days (Processing format)	see 7.8.1
val_time1	double*	-	Validity stop time of the initialization	Decimal days (Processing format)	see 7.8.1
propag_id	xo_propag_id*	-	Structure that containg the propagation data	-	-
ierr[XO_NUM_ERR_PROPAG_INIT_FILE]	long	all	Status vector	-	-

7.8.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_propag_init** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xo_propag_init** CFI function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 34: Error messages of xo_propag_init function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Wrong input flag	No calculation performed	XO_CFI_PROPAG_INIT_FLAG_ERR	0
ERR	Auto and/or double model for propagation are incompatible with the input Orbit Id. model	No calculation performed	XO_CFI_PROPAG_INIT_MODEL_INCONSISTENCY_ERR	1
ERR	Orbit Id was not initialized	No calculation performed	XO_CFI_PROPAG_INIT_ORBIT_STATUS_ERR	2
ERR	Error closing input Propag Id. for re-initializing	No calculation performed	XO_CFI_PROPAG_INIT_CLOSE_ERR	3
ERR	Memory allocation error	No calculation performed	XO_CFI_PROPAG_INIT_MEMORY_ERR	4
ERR	Could not initialise	No calculation performed	XO_CFI_PROPAG_INIT_CART_INIT_ERR	5
ERR	Error computing reference state vector	No calculation performed	XO_CFI_PROPAG_INIT_GET_OSV_ERR	6
ERR	Error making a time transformation.	No calculation performed	XO_CFI_PROPAG_INIT_TIME_TRANSFORMATION_ERR	7
WARN	Propagation allowed out of the file boundaries. Propagation could not reach the desired accuracy.	Calculation performed. Warning raised when using **AUTO** and/or **DOUBLE** propagation model and trying to initialize out of the file validity interval.	XO_CFI_PROPAG_INIT_INACCURACY_PROP_WARN	8

7.8.6 Runtime performances

The following runtime performances have been measured:

Table 35: Runtime performances of xo_propag_init function

Ultra Sparc II-400[msec]
TBD

7.9 xo_propag_spot_init

TBW

7.10 xo_propag

7.10.1 Overview

This routine simulates orbit propagations over complete orbits, performing an accurate prediction of osculating Cartesian state vectors for user requested times, which must fall within the validity time interval calculated by the initialization routines.

For the orbit propagation, the user may choose between different propagation models, although for the time being, the initial set of models supported are:

- ***Mean Kepler elements model*** (which is the current model). It implies the use of a formulation for the time rates of change for the different mean Kepler elements as functions of a given initial set of mean Kepler elements. Using the above time rates of change, the mean orbital elements can be propagated forward or backward in time by extrapolating the individual time slopes of the superimposed secular and long-periodic perturbations functions. As the long periodic variations have typically periods on the order of months, a near-linear time slope for prediction intervals of many orbits is warranted.
- ***Spot elements model*** (still TBD). This model is based upon the usage of an extended orbit state vector (originally used for SPOT satellites and currently for MetOp). The calculation of the orbit state vector is made by fitting them using a predicted or restituted orbit file.

The propagation model is set as an input parameter for the initialization routines, and the **xo_propag** routine utilizes that model transparently for the user.

For a general description of the initialization routines and how to use them in conjuction to the **xo_propag** function, see section 4.2.

7.10.2 Calling interface

The calling interface of the **xo_propag** CFI function is the following (input parameters are underlined):

```
#include <explorer_orbit.h>
{
      xo_propag_id propag_id = {NULL};
      long mode, time_ref;
      double time, pos_out[3], vel_out[3], acc_out[3];
      long status, ierr[XO_NUM_ERR_PROPAG];

      status = xo_propag (&propag_id, &mode, &time_ref, &time,
                          pos_out, vel_out, acc_out, ierr);

      /* Or, using the run_id */
      long run_id;

      status = xo_propag_run (&run_id, &mode, &time_ref, &time,
                              pos_out, vel_out, acc_out, ierr);
}
```

For ForTran programs, the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
#include <explorer_orbit.inc>
      INTEGER*4 SAT_ID, MODE, TIME_REF
      REAL*8 TIME, POS_OUT(3), VEL_OUT(3), ACC_OUT(3)
      INTEGER*4 STATUS, IERR(XO_NUM_ERR_PROPAG)
      STATUS = XO_PROPAG (SAT_ID, MODE, TIME_REF, TIME,
     &                    POS_OUT, VEL_OUT, ACC_OUT, IERR)
```

7.10.3 Input parameters

The **xo_propag** CFI function has the following input parameters:

Table 36: Input parameters of xo_propag function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
propag_id	xo_propag_id*	-	Structure that contains the propagator data	-	-
mode	long *	-	Propagation mode. (TBD in Spot model).	-	• XO_PROPAG_MODEL_MEAN_KEPL • XO_PROPAG_MODEL_SPOT
time_ref	long*	-	Time reference ID	-	Complete
time	double*	-	Reference time	Decimal days (Processing format)	[-18262.0,36524.0]

It is possible to use enumeration values rather than integer values for some of the input arguments:

- Time reference ID: time_ref. See [GEN_SUM].

7.10.4 Output parameters

The output parameters of the **xo_propag** CFI function are:

Table 37: Output parameters of xo_propag function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_propag	long	-	Main status flag	-	-1, 0, +1
pos_out[3]	double	all	Osculating position vector at predicted time (Earth fixed CS)	m	-
vel_out[3]	double	all	Osculating velocity vector at predicted time (Earth fixed CS)	m/s	-
acc_out[3]	double	all	Osculating acceleration vector at predicted time (Earth fixed CS)	m/s^2	-
ierr[XO_NUM_ERR_PROPAG]	long	all	Status vector	-	-

7.10.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_propag** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xo_propag** CFI function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 38: Error messages of xo_propag function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Wrong input flag	No calculation performed	XO_CFI_PROPAG_FLAG_ERR	0
ERR	The internal data were not initialized	No calculation performed	XO_CFI_PROPAG_NOT_INTERNAL_DATA_ERR	1
ERR	An error ocurred in the Mean Keplerian OSV routine	No calculation performed	XO_CFI_PROPAG_MKO_ERR	2

7.10.6 Runtime performances

The following runtime performances have been measured:

Table 39: Runtime performances of xo_propag function

Ultra Sparc II-400[msec]
TBD

7.11 xo_propag_extra

7.11.1 Overview

This software returns ancillary results derived from an orbit state vector obtained from the orbit propagation routines (stored within the *orbit Id*). This state vector depends on wich is the last function called:

- when calling to **xo_propag_extra** after initialising **xo_propag_init** with the *orbit Id* from **xo_orbit_cart_init**, the Cartesian orbit state vector used to calculate the ancillary results is the one given as input in the initialization routine.
- when calling after initialising **xo_propag_init** with the *orbit Id* from **xo_orbit_init_def**, the Cartesian orbit state vector is the one generated internally at the requested ANX in the initialization routine.
- when calling after initialising **xo_propag_init** with the *orbit Id* from **xo_orbit_init_file**, the Cartesian orbit state vector is the one generated internally by the routine around the ANX (in Mean Keplerian model; in Spot model is not defined yet).
- when calling after **xo_propag**, the Cartesian orbit state vector is the one predicted at the requested time in the propagation routine.

A description of the ancillary results may be found in the section 7.11.5.

A complete calling sequence of the propagation procedure is presented in section 4.2.

7.11.2 Calling interface

The calling interface of the **xo_propag_extra** CFI function is the following:

```
#include <explorer_orbit.h>
{
      xo_propag_id propag_id = {NULL};
      long extra_choice;
      double model_out[XO_PROPAG_EXTRA_NUM_DEP_ELEMENTS],
             extra_out[XO_PROPAG_EXTRA_NUM_INDEP_ELEMENTS];
      long status, ierr[XO_NUM_ERR_PROPAG_EXTRA];

      status = xo_propag_extra (&propag_id, &extra_choice,
                                model_out, extra_out, ierr);

      /* Or, using the run_id */
      long run_id;

      status = xo_propag_extra_run (&run_id, &extra_choice,
                                    model_out, extra_out, ierr);
}
```

For ForTran programs, the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
#include <explorer_orbit.inc>
```

```
      INTEGER*4 SAT_ID, EXTRA_CHOICE
      REAL*8 MODEL_OUT(XO_PROPAG_EXTRA_NUM_DEP_ELEMENTS),
             EXTRA_OUT(XO_PROPAG_EXTRA_NUM_INDEP_ELEMENTS)
      INTEGER*4 STATUS, IERR(XO_NUM_ERR_PROPAG_EXTRA)

      STATUS = XO_PROPAG_EXTRA (SAT_ID, EXTRA_CHOICE, MODEL_OUT,
     &                          EXTRA_OUT, IERR)
```

7.11.3 Input parameters

The **xo_propag_extra** CFI function has the following input parameters:

Table 40: Input parameters of xo_propag_extra

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
propag_id	xo_propag_id*	-	Structure that contains the propagator data	-	-
extra_choice	long *	-	Flag to allow an ancillary results choice	-	[0, 4095]

It is possible to use enumeration values rather than integer values for some of the input arguments:

- Flag to select ancillary results: extra_choice. See tables below:

Table 41: Enumeration values of extra_choice input flag

Model independant	Description	Long
XO_PROPAG_EXTRA_NO_RESULTS	No extra results	0
XO_PROPAG_EXTRA_GEOLOCATION	Geolocation results	1
XO_PROPAG_EXTRA_GEOLOCATION_D	Geolocation rate results	2
XO_PROPAG_EXTRA_GEOLOCATION_2D	Geolocation rate-rate results	4
XO_PROPAG_EXTRA_GEOLOCATION_EXTRA	Geolocation extra results	8
XO_PROPAG_EXTRA_EARTH_FIXED_D	Earth fixed velocity results	16
XO_PROPAG_EXTRA_EARTH_FIXED_2D	Earth fixed acceleration results	32
XO_PROPAG_EXTRA_SUN	Sun results	64
XO_PROPAG_EXTRA_MOON	Moon results	128
XO_PROPAG_EXTRA_OSCULATING_KEPLER	Osculating keplerian elements	256
XO_PROPAG_EXTRA_INERTIAL_AUX	Inertial auxiliary results	512
Model dependant (Mean Keplerian model)	**Description**	**Long**
XO_PROPAG_EXTRA_DEP_ANX_TIMING	ANX timing results	1024
XO_PROPAG_EXTRA_DEP_MEAN_KEPLER	Mean keplerian elements	2048

To calculate all results there is an extra enumeration value, defined as the addition of all the enumeration result values:

Enumeration value	Description	Long
XO_PROPAG_EXTRA_ALL_RESULTS	All results	4095

The elements calculated in each case are shown in section 7.6.5. It is possible to select the calculation of different sets of output parameters, or to make any combination of them by adding the results enumeration desired. In order to calculate some elements it might be necessary to calculate elements which have not been explicitly requested. The function identifies internally all the dependencies and those elements are also returned in the result vectors.

7.11.4 Output parameters

The output parameters of the **xo_propag_extra** CFI function are:

Table 42: Output parameters of xo_propag_extra

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_propag_extra	long	-	Main status flag	-	-1, 0, +1
model_out[XO_PROPAG_EXTRA_NUM_DEP_ELEMENTS]	double	all	Vector of model-dependent parameters	-	-
extra_out[XO_PROPAG_EXTRA_NUM_INDEP_ELEMENTS]	double	all	Vector of model-independent parameters. It depends upon extra-choice	-	-
ierr[XO_NUM_ERR_PROPAG_EXTRA]	long	all	Status vector	-	-

7.11.5 Results vectors

The model-dependent parameters vector for the **Mean Keplerian propagation model** is (note that there is an enumeration associated to the elements of the results vectors) in table 43:

Table 43: Ancillary results vector. Model-dependent parameters

Result parameter	Set	Description (Reference)	Unit (Format)	Allowed Range
[0] XO_PROPAG_EXTRA_DEP_NODAL_PERIOD	ANX Timing	Nodal period	s	>= 0
[1] XO_PROPAG_EXTRA_DEP_UTC_CURRENT_ANX		Time of current ANX	decimal days (Processing format)	-
[2] XO_PROPAG_EXTRA_DEP_ORBIT_NUMBER	Position in orbit[a]	Absolute Orbit Number		> 0
[3] XO_PROPAG_EXTRA_DEP_SEC_SINCE_ANX		Time since ANX	s	>= 0 < Nodal Period
[4:9] XO_PROPAG_EXTRA_DEP_MEAN_KEPL_A XO_PROPAG_EXTRA_DEP_MEAN_KEPL_E XO_PROPAG_EXTRA_DEP_MEAN_KEPL_I XO_PROPAG_EXTRA_DEP_MEAN_KEPL_RA XO_PROPAG_EXTRA_DEP_MEAN_KEPL_W XO_PROPAG_EXTRA_DEP_MEAN_KEPL_M	Mean Kepler	Mean Kepler elements of the propagated OSV (True of Date)	-	-

a. These parameters are calculated only when initialising with xo_propag_init_file and xo_propag_init_def

The model-dependent parameters vector for the **Spot propagation model** is TBD.

The model-independent parameters vector is (note that there is an enumeration associated to the elements of the results vectors) in table 44:

Table 44: Ancillary results vector. Model-independent parameters

Result parameter (res element)	Set	Description (Reference)	Unit (Format)	Allowed Range
[0] XO_PROPAG_EXTRA_GEOC_LONG	Geolocation	Geocentric longitude of satellite and SSP (EF frame)	deg	>= 0 < 360
[1] XO_PROPAG_EXTRA_GEOD_LAT		Geodetic latitude of satellite and SSP (EF frame)	deg	>= -90 <= +90
[2] XO_PROPAG_EXTRA_GEOD_ALT		Geodetic altitude of the satellite (EF frame)	m	-

Table 44: Ancillary results vector. Model-independent parameters

Result parameter (res element)	Set	Description (Reference)	Unit (Format)	Allowed Range
[3] XO_PROPAG_EXTRA_GEOC_LONG_D	Geolocation rate	Geocentric longitude rate of satellite and SSP (EF frame)	deg/s	-
[4] XO_PROPAG_EXTRA_GEOD_LAT_D		Geodetic latitude rate of satellite and SSP (EF frame)	deg/s	-
[5] XO_PROPAG_EXTRA_GEOD_ALT_D		Geodetic altitude rate of the satellite (EF frame)	m/s	-
[6] XO_PROPAG_EXTRA_GEOC_LONG_2D	Geolocation rate rate	Geocentric longitude rate-rate of satellite and SSP (EF frame)	deg/s^2	-
[7] XO_PROPAG_EXTRA_GEOD_LAT_2D		Geodetic latitude rate-rate of satellite and SSP (EF frame)	deg/s^2	-
[8] XO_PROPAG_EXTRA_GEOD_ALT_2D		Geodetic altitude rate-rate of the satellite (EF frame)	m/s^2	-
[9] XO_PROPAG_EXTRA_RAD_CUR_PARALLEL_MERIDIAN	Geolocation extra	Radius of curvature parallel to meridian at the SSP (EF frame)	m	>= 0
[10] XO_PROPAG_EXTRA_RAD_CUR_ORTHO_MERIDIAN		Radius of curvature orthogonal to meridian at the SSP (EF frame)	m	>= 0
[11] XO_PROPAG_EXTRA_RAD_CUR_ALONG_GROUNDTRACK		Radius of curvature along groundtrack at the SSP (EF frame)	m	>= 0
[12] XO_PROPAG_EXTRA_NORTH_VEL	Earth-fixed velocity	Northward component of the velocity relative to the Earth of the SSP (Topocentric frame)	m/s	-
[13] XO_PROPAG_EXTRA_EAST_VEL		Eastward component of the velocity relative to the Earth of the SSP (Topocentric frame)	m/s	-
[14] XO_PROPAG_EXTRA_MAG_VEL		Magnitude of the velocity relative to the Earth of the SSP (Topocentric frame)	m/s	>= 0
[15] XO_PROPAG_EXTRA_AZ_VEL		Azimuth of the velocity relative to the Earth of the SSP (Topocentric frame)	deg	>= 0 < 360

Table 44: Ancillary results vector. Model-independent parameters

Result parameter (res element)	Set	Description (Reference)	Unit (Format)	Allowed Range
[16] XO_PROPAG_EXTRA_NORTH_ACC	Earth-fixed acceleration	Northward component of the acceleration relative to the Earth of the SSP (Topocentric frame)	m/s^2	-
[17] XO_PROPAG_EXTRA_EAST_ACC		Eastward component of the acceleration relative to the Earth of the SSP (Topocentric frame)	m/s^2	-
[18] XO_PROPAG_EXTRA_GROUNDTRACK_TANG_ACC		Groundtrack tangential component of the acceleration relative to the Earth of the SSP (Topocentric frame)	m/s^2	-
[19] XO_PROPAG_EXTRA_AZ_ACC		Azimuth of the acceleration relative to the Earth of the SSP (Topocentric frame)	deg	>= 0 < 360
[20] XO_PROPAG_EXTRA_SAT_ECLIPSE_FLAG	Sun	Satellite eclipse flag 0 = No 1 = Yes		0, 1
[21] XO_PROPAG_EXTRA_SZA		Sun Zenith Angle	deg	>= 0 < 360
[22] XO_PROPAG_EXTRA_MSLT		Mean local solar time at the SSP	decimal hour	>= 0 < 24
[23] XO_PROPAG_EXTRA_TLST		True local solar time at the SSP	decimal hour	>= 0 < 24
[24] XO_PROPAG_EXTRA_TRUE_SUN_RA		True Sun's (centre) right ascension (TOD frame)	deg	>= 0 < 360
[25] XO_PROPAG_EXTRA_TRUE_SUN_DEC		True Sun's (centre) declination (TOD frame)	deg	>= -90 <= +90
[26] XO_PROPAG_EXTRA_TRUE_SUN_SEMI_DIAM		True Sun's semi-diameter	deg	>= 0
[27] XO_PROPAG_EXTRA_MOON_RA	Moon	Moon's (centre) right ascension (TOD frame)	deg	>= 0 < 360
[28] XO_PROPAG_EXTRA_MOON_DEC		Moon's (centre) declination (TOD frame)	deg	>= -90 <= +90
[29] XO_PROPAG_EXTRA_MOON_SEMI_DIAM		Moon's semi-diameter	deg	>= 0
[30] XO_PROPAG_EXTRA_MOON_AREA_LIT		Area of Moon lit by Sun		>= 0 <= 1

Table 44: Ancillary results vector. Model-independent parameters

Result parameter (res element)	Set	Description (Reference)	Unit (Format)	Allowed Range
[31:36] XO_PROPAG_EXTRA_OSC_KEPL_A XO_PROPAG_EXTRA_OSC_KEPL_E XO_PROPAG_EXTRA_OSC_KEPL_I XO_PROPAG_EXTRA_OSC_KEPL_RA XO_PROPAG_EXTRA_OSC_KEPL_W XO_PROPAG_EXTRA_OSC_KEPL_M	Osculating Kepler	Osculating Keplerian elements of the OSV (TOD frame)		
[37] XO_PROPAG_EXTRA_ORBIT_RAD	Inertial Aux	Orbit radius (TOD frame)	m	>= 0
[38] XO_PROPAG_EXTRA_RADIAL_ORB_VEL		Radial orbit velocity component (TOD frame)	m/s	-
[39] XO_PROPAG_EXTRA_TRANS_ORB_VEL		Transversal orbit velocity component (TOD frame)	m/s	-
[40] XO_PROPAG_EXTRA_ORB_VEL_MAG		Orbit velocity magnitude (TOD frame)	m/s	>= 0
[41] XO_PROPAG_EXTRA_RA_SAT		Right ascension of the satellite (TOD frame)	deg	>= 0 < 360
[42] XO_PROPAG_EXTRA_DEC_SAT		Declination of the satellite (TOD frame)	deg	>= -90 <= +90
[43] XO_PROPAG_EXTRA_EARTH_ROTATION_ANGLE		Earth rotation angle [H]	deg	>= 0 < 360
[44] XO_PROPAG_EXTRA_RA_SAT_D		Right ascension rate of the satellite (TOD frame)	deg/s	-
[45] XO_PROPAG_EXTRA_RA_SAT_2D		Right ascension rate-rate of the satellite (TOD frame)	deg/s^2	-
[46] XO_PROPAG_EXTRA_OSC_TRUE_LAT		Satellite osculating true latitude (TOD frame)	deg	>= 0 < 360
[47] XO_PROPAG_EXTRA_OSC_TRUE_LAT_D		Satellite osculating true latitude rate (TOD frame)	deg/s	-
[48] XO_PROPAG_EXTRA_OSC_TRUE_LAT_2D		Satellite osculating true latitude rate-rate (TOD frame)	deg/s^2	-

7.11.6 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_propag_extra** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (`WARN`) or an error (`ERR`), the cause of such a message and the impact on the performed calculation, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xo_propag_extra** CFI function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 45: Error messages of xo_propag_extra function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	The internal data were not initialized	No calculation performed	XO_CFI_PROPAG_EXTRA_NOT_INTERNAL_DATA_ERR	0
ERR	Could not create extra results	No calculation performed	XO_CFI_PROPAG_EXTRA_RESULTS_ERR	1

7.11.7 Runtime performances

The following runtime performances have been measured:

Table 46: Runtime performances of xo_propag_extra function

mode	Ultra sparc II-400 [ms]
XO_PROPAG_EXTRA_NO_RESULTS	0.008
XO_PROPAG_EXTRA_GEOLOCATION	0.009
XO_PROPAG_EXTRA_GEOLOCATION_EXTRA	0.047
XO_PROPAG_EXTRA_EARTH_FIXED_D	0.031
XO_PROPAG_EXTRA_SUN	0.241
XO_PROPAG_EXTRA_MOON	0.319
XO_PROPAG_EXTRA_OSCULATING_KEPLER	0.008
XO_PROPAG_EXTRA_INERTIAL_AUX	0.104
XO_PROPAG_EXTRA_DEP_ANX_TIMING	0.013
XO_PROPAG_EXTRA_DEP_MEAN_KEPLER	0.009
XO_PROPAG_EXTRA_DEP_ANX_TIMING + XO_PROPAG_EXTRA_DEP_MEAN_KEPLER	0.012
XO_PROPAG_EXTRA_DEP_ANX_TIMING + XO_PROPAG_EXTRA_DEP_MEAN_KEPLER + XO_PROPAG_EXTRA_INERTIAL_AUX	0.114
XO_PROPAG_EXTRA_ALL_RESULTS	0.446

7.12 xo_propag_close

7.12.1 Overview

The **xo_propag_close** function is used to free the memory allocated by the other propagation routines, ant it must be called after using them.

A complete calling sequence of the propagation procedure is presented in section 4.2.

7.12.2 Calling interface

The calling interface of the **xo_propag_close** CFI function is the following (input parameters are underlined):

```
#include <explorer_orbit.h>
{
      xo_propag_id propag_id = {NULL};
      long status;
      long ierr[XO_NUM_ERR_PROPAG_CLOSE];

      status = xo_propag_close (&propag_id, ierr)
}
```

For ForTran programs, the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
#include <explorer_orbit.inc>

      INTEGER*4 SAT_ID,
      INTEGER*4 STATUS, IERR(XO_NUM_ERR_PROPAG_CLOSE)

      STATUS = XO_PROPAG_CLOSE (SAT_ID, IERR)
```

7.12.3 Input parameters

The **xo_propag_close** CFI function has the following input parameters:

Table 47: Input parameters of xo_propag_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
propag_id	xo_propag_id*	-	Structure that contains the propagator data	-	-

7.12.4 Output parameters

The output parameters of the **xo_propag_close** CFI function are:

Table 48: Output parameters of xo_propag_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_propag_close	long	-	Main status flag	-	-1, 0, +1
ierr[XO_NUM_ERR_PROPAG_CLOSE]	long	all	Status vector	-	-

7.12.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_propag_close** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (`WARN`) or an error (`ERR`), the cause of such a message and the impact on the performed calculation, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xo_propag_close** CFI function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 49: Error messages of xo_propag_close function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Wrong Propag Id. It was not initialized or it is in use	The Propag Id. was not closed.	XO_CFI_PROPAG_CLOSE_WRONG_ID_ERR, ERR, NO_PAR	0

7.12.6 Runtime performances

The following runtime performances have been measured:

Table 50: Runtime performances of xo_propag_close function

Ultra Sparc II-400[msec]
TBD

7.13 xo_interpol_init

7.13.1 Overview

The **xo_interpol_init** initializes the interpolation process, i. e., it produces internal data to be used by the **xo_interpol** function to perform the interpolation. The internal data consists of Cartesian orbit state vectors extracted from the *orbit_id*, and validity times giving the allowed time window for interpolation.

Before calling this function it is required to initialise the orbit with one of the following modes:

- XO_ORBIT_INIT_ROF_MODE
- XO_ORBIT_INIT_DORIS_MODE

The validity start and stop times of the initialization (***val_time0*** and ***val_time1*** output parameters) represents the allowed time window for interpolation.

CAUTION: The interpolation is highly accurate (1 mm. accuracy TBC) when it is performed between 4 input file(s) time intervals after start of file(s) and before end of file(s), but it degrades (up to a few cm. TBC) until 1 or 2 time intervals (TBD) before start of file(s) and after end of file(s). figure 3 provides a graphical explanation.

The **xo_interpo**l function allows to extrapolate, that is, compute results for the 1 or 2 (TBC) intervals before start of the input file(s) and after enf of the input file. Anyway, as seen above in the caution statement, extrapolation is not recommended. In this case, the extrapolation window is NOT included in the valid time interval.

When the interpolation is in “degraded” mode, that is, when extrapolation is used, or when there is less than four orbit state vectors available in the input file before or after the requested time, **xo_interpol** function will issue different warnings messages indicating that a degraded interpolation or extrapolation is performed. If the requested time is out the allowed extrapolation range, the function will return an error message.

A complete calling sequence of the interpolation procedure is presented in section 4.3.

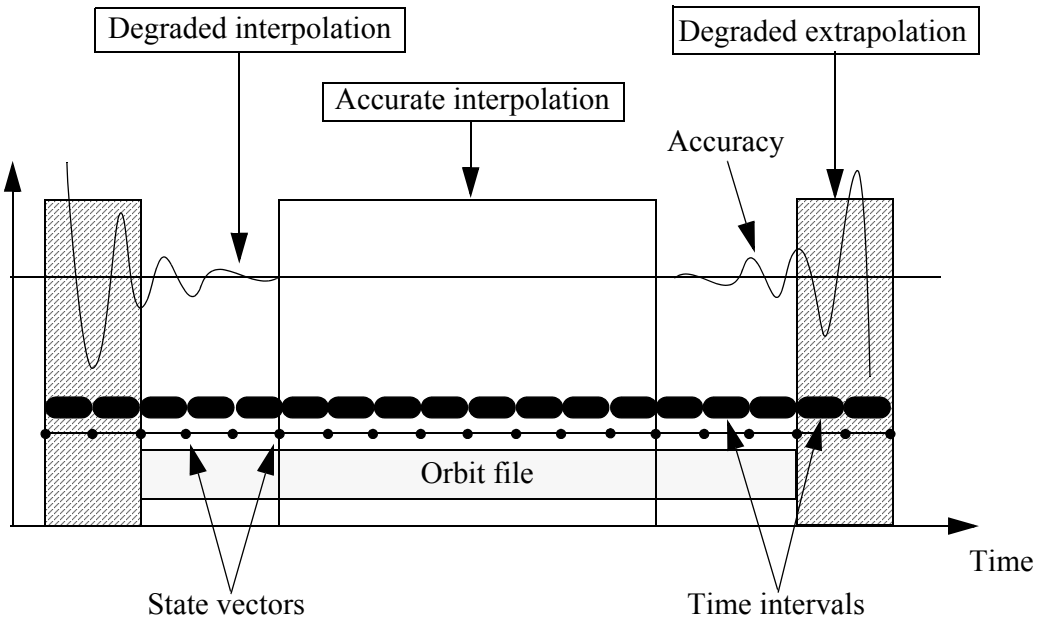


Figure 3: Performances of the interpolation algorithm

7.13.2 Calling interface

The calling interface of the **xo_interpol_init** CFI function is the following (input parameters are underlined):

```
#include <explorer_orbit.h>
{
       xo_orbit_id *orbit_id = {NULL};
       xo_interpol_id *interpol_id = {NULL};
       long interpol_model;
       long time_mode, time_ref;
       double val_time0, val_time1;
       long status, ierr[XO_NUM_ERR_INTERPOL_INIT_FILE];

       status = xo_interpol_init (&orbit_id, &interpol_model,
                                  &time_mode, &time_ref,
                                  &val_time0, &val_time1,
                                  &interpol_id, ierr);
}
```

For ForTran programs, the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
#include <explorer_orbit.inc>

       INTEGER*4 SAT_ID, INTERPOL_MODEL, N_FILES
       CHARACTER*LENGTH_NAME ORBIT_FILE(N_FILES)
       INTEGER*4 TIME_MODE, TIME_REF, ORBIT0, ORBIT1
       REAL*8 TIME0, TIME1, VAL_TIME0, VAL_TIME1
       INTEGER*4 STATUS, IERR(XO_NUM_ERR_INTERPOL_INIT_FILE)

       STATUS = XO_INTERPOL_INIT (SAT_ID, INTERPOL_MODEL, N_FILES,
      &                           ORBIT_FILE, TIME_MODE, TIME_REF,
      &                           TIME0, TIME1, ORBIT0, ORBIT1,
      &                           VAL_TIME0, VAL_TIME1, IERR)
```

Note that `N_FILES` must be set to the number of input files of that type, with a maximum value of 16, whereas `LENGTH_NAME` must be set to the maximum string length of the filenames of that type. All strings in ForTran must end in "\0" (for compatibility with C programs).

7.13.3 Input parameters

The **xo_interpol_init** CFI function has the following input parameters:

Table 51: Input parameters of xo_interpol_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that containg the orbit initialization	-	-
interpol_model	long*	-	Interpolation model ID	-	Complete
time_ref	long*	-	Time reference ID used in the output validity period	-	Complete

It is possible to use enumeration values rather than integer values for some of the input arguments:

- Interpolation model ID: interpol_model. Current document, section 6.2.
- Time reference ID: time_ref. See [GEN_SUM].

7.13.4 Output parameters

The output parameters of the **xo_interpol_init** CFI function are:

Table 52: Output parameters of xo_interpol_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_interpol_init_file	long	-	Main status flag	-	-1, 0, +1
val_time0	double*	-	Validity start time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
val_time1	double*	-	Validity stop time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
interpol_id	xo_interpol_id*	-	Structure that containg the interpolation data	-	-
ierr[XO_NUM_ERR_INTERPOL_INIT]	long	all	Status vector	-	-

7.13.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_interpol_init** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xo_interpol_init** CFI function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 53: Error messages of xo_interpol_init function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Memory allocation error.	No calculation performed	XO_CFI_INTERPOL_INIT_MEMORY_ERR	0
ERR	Orbit Id. is not initialized	No calculation performed	XO_CFI_INTERPOL_INIT_ORBIT_STATUS_ERR	1
ERR	Wrong input orbit Id.	No calculation performed	XO_CFI_INTERPOL_INIT_ORBIT_ID_ERR	2
ERR	Interpol Id is already initialized.	No calculation performed	XO_CFI_INTERPOL_INIT_STATUS_ERR	3
ERR	ANX state vector does not satisfy loose Earth Explorer tolerance requirements.	No calculation performed	XO_CFI_INTERPOL_INIT_LOOSE_TOL_ERR	4
ERR	Problem calculating the Earth Explorer acceleration.	No calculation performed	XO_CFI_INTERPOL_INIT_ACCELERATION_ERR	5
ERR	Error changing time format or reference	No calculation performed	XO_CFI_INTERPOL_INIT_TIME_CORRELATION_ERR	6
WARN	ANX state vector does not satisfy tight Earth Explorer tolerance requirements.	Calculation performed	XO_CFI_INTERPOL_INIT_TIGHT_TOL_WARN	7

7.13.6 Runtime performances

The following runtime performances have been measured:

Table 54: Runtime performances of xo_interpol_init function

Ultra Sparc II-400[msec]
TBD

7.14 xo_interpol

7.14.1 Overview

The **xo_interpol** function is used to compute a Cartesian state vector at a requested time, using the internal data generated by the **xo_interpol_init_file** routine.

To complete the description of the **xo_interpol** function see comments in section 7.13.1.

A complete calling sequence of the interpolation procedure is presented in section 4.3.

7.14.2 Calling interface

The calling interface of the **xo_interpol** CFI function is the following (input parameters are underlined):

```
#include <explorer_orbit.h>
{
      xo_interpol_id interpol_id = {NULL};
      long model, time_ref;
      double time, pos_out[3], vel_out[3], acc_out[3];
      long status, ierr[XO_NUM_ERR_INTERPOL];

      status =xo_interpol(&interpol_id, &model, &time_ref, &time,
                          pos_out, vel_out, acc_out, ierr);

      /* Or, using the run_id */
      long run_id;

      status =xo_interpol_run(&run_id, &model, &time_ref, &time,
                              pos_out, vel_out, acc_out, ierr);
}
```

For ForTran programs, the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
#include <explorer_orbit.inc>

      INTEGER*4 SAT_ID, MODEL, TIME_REF
      REAL*8 TIME, POS_OUT(3), VEL_OUT(3), ACC_OUT(3)
      INTEGER*4 STATUS, IERR(XO_NUM_ERR_INTERPOL)

      STATUS = XO_INTERPOL (SAT_ID, MODEL, TIME_REF, TIME,
     &                      POS_OUT, VEL_OUT, ACC_OUT, IERR)
```

7.14.3 Input parameters

The **xo_interpol** CFI function has the following input parameters:

Table 55: Input parameters of xo_interpol function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
interpol_id	xo_interpol_id*	-	Structure that contains the interpolator data	-	-
model	long *	-	Interpolation model	-	Complete
time_ref	long*	-	Time reference ID	-	Complete
time	double*	-	Reference time	Decimal days (Processing format)	[-18262.0,36524.0]

It is possible to use enumeration values rather than integer values for some of the input arguments:

- Interpolation model: model. Current document, section 6.2.
- Time reference ID: time_ref. See [GEN_SUM].

7.14.4 Output parameters

The output parameters of the **xo_interpol** CFI function are:

Table 56: Output parameters of xo_interpol function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_propag	long	-	Main status flag	-	-1, 0, +1
pos_out[3]	double	all	Osculating position vector at interpolated time (EF reference frame)	m	-
vel_out[3]	double	all	Osculating velocity vector at interpolated time (EF reference frame)	m/s	-
acc_out[3]	double	all	Osculating acceleration vector at interpolated time (EF reference frame)	m/s^2	-
ierr[XO_NUM_ERR_INTERPOL]	long	all	Status vector	-	-

7.14.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_interpol** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xo_interpol** CFI function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 57: Error messages of xo_interpol function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Wrong input flag	No calculation performed	XO_CFI_INTERPOL_FLAG_ERR	0
ERR	Error changing time format or reference	No calculation performed	XO_CFI_INTERPOL_TIME_CORRELATION_ERR	1
ERR	Input time is out of range.	No calculation performed	XO_CFI_INTERPOL_WRONG_TIME2_ERR	2
ERR	The data base has not been previously initialisated.	No calculation performed	XO_CFI_INTERPOL_NOT_INITIALISED_ERR	3
ERR	The requested date is out of the data base.	No calculation performed	XO_CFI_INTERPOL_TIME_OUT_OF_DB_ERR	4
ERR	Fatal error in XO_Interpol	No calculation performed	XO_CFI_INTERPOL_FATAL_ERROR_INTERPOL_ERR	5
ERR	State vector does not satisfy loose Earth Explorer tolerance requirements	No calculation performed	XO_CFI_INTERPOL_LOOSE_TOL_ERR	6
ERR	Problem calculating the Earth Explorer acceleration	No calculation performed	XO_CFI_INTERPOL_ACCELERATION_ERR	7
WARN	State vector does not satisfy tight Earth Explorer tolerance requirements	Calculation performed	XO_CFI_INTERPOL_TIGHT_TOL_WARN	8
WARN	Warning error in XO_Interpol	Calculation performed	XO_CFI_INTERPOL_FATAL_ERROR_INTERPOL_WARN	9
WARN	Time out of range. Computing State Vector with extrapolation algorithm	Calculation performed using an extrapolation algorithm. The results could not reach the desired accuracy.	XO_CFI_INTERPOL_EXTRAPOL_WARN	10
WARN	Less than four Orbit State Vectors to interpolate	Calculation performed Interpolation could not reach the desired accuracy	XO_CFI_INTERPOL_FEW_OSV_WARN	11

7.14.6 Runtime performances

The following runtime performances have been measured:

Table 58: Runtime performances of xo_interpol function

Ultra Sparc II-400[msec]
TBD

7.15 xo_interpol_extra

7.15.1 Overview

This software returns ancillary results derived from an orbit interpolation using the **xo_interpol** function. The ancillary results are similar to the ones produced by the **xo_propag_extra** routine.

A complete calling sequence of the interpolation procedure is presented in section 4.3.

7.15.2 Calling interface

The calling interface of the **xo_interpol_extra** CFI function is the following:

```
#include <explorer_orbit.h>
{
       xo_interpol_id interpol_id = {NULL};
       long extra_choice;
       double model_out[XO_INTERPOL_EXTRA_NUM_DEP_ELEMENTS],
              extra_out[XO_INTERPOL_EXTRA_NUM_INDEP_ELEMENTS];
       long status, ierr[XO_NUM_ERR_INTERPOL_EXTRA];

       status = xo_interpol_extra (&interpol_id,&extra_choice,
                                   model_out, extra_out, ierr);

       /* Or, using the run_id */
       long run_id;

       status = xo_interpol_extra_run (&run_id,&extra_choice,
                                       model_out, extra_out, ierr);
}
```

For ForTran programs, the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
#include <explorer_orbit.inc>

       INTEGER*4 SAT_ID, EXTRA_CHOICE
       REAL*8 MODEL_OUT(XO_INTERPOL_EXTRA_NUM_DEP_ELEMENTS),
              EXTRA_OUT(XO_INTERPOL_EXTRA_NUM_INDEP_ELEMENTS)
       INTEGER*4 STATUS, IERR(XO_NUM_ERR_INTERPOL_EXTRA)

       STATUS = XO_INTERPOL_EXTRA (SAT_ID, EXTRA_CHOICE,MODEL_OUT,
                                   EXTRA_OUT, IERR)
```

7.15.3 Input parameters

The **xo_interpol_extra** CFI function has the following input parameters:

Table 59: Input parameters of xo_interpol_extra function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
interpol_id	xo_interpol_id*	-	Structure that contains the interpolator data	-	-
extra_choice	long *	-	Flag to allow an ancillary results choice	-	[0, 2047]

It is possible to use enumeration values rather than integer values for some of the input arguments:

- Flag to select ancillary results: extra_choice. See tables below:

Table 60: Enumeration values of extra_choice input flag

Model independant	Description	Long
XO_INTERPOL_EXTRA_NO_RESULTS	No extra results	0
XO_INTERPOL_EXTRA_GEOLOCATION	Geolocation results	1
XO_INTERPOL_EXTRA_GEOLOCATION_D	Geolocation rate results	2
XO_INTERPOL_EXTRA_GEOLOCATION_2D	Geolocation rate-rate results	4
XO_INTERPOL_EXTRA_GEOLOCATION_EXTRA	Geolocation extra results	8
XO_INTERPOL_EXTRA_EARTH_FIXED_D	Earth fixed velocity results	16
XO_INTERPOL_EXTRA_EARTH_FIXED_2D	Earth fixed acceleration results	32
XO_INTERPOL_EXTRA_SUN	Sun results	64
XO_INTERPOL_EXTRA_MOON	Moon results	128
XO_INTERPOL_EXTRA_OSCULATING_KEPLER	Osculating keplerian elements	256
XO_INTERPOL_EXTRA_INERTIAL_AUX	Inertial auxiliary results	512
Model dependant (Mean Keplerian model)	**Description**	**Long**
XO_INTERPOL_EXTRA_DEP_ANX_TIMING	ANX timing results	1024

To calculate all results there is an extra enumeration value, defined as the addition of all the enumeration result values:

Enumeration value	Description	Long
XO_INTERPOL_EXTRA_ALL_RESULTS	All results	2047

The elements calculated in each case are shown in sections 7.6.5 and 7.10.5. It is possible to select the calculation of different sets of output parameters, or to make any combination of them by adding the results enumeration desired. In order to calculate some elements it might be necessary to calculate elements which have not been explicitly requested. The function identifies internally all the dependencies and those elements are also returned in the result vectors.

7.15.4 Output parameters

The output parameters of the **xo_interpol_extra** CFI function are:

Table 61: Output parameters of xo_interpol_extra function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_interpol_extra	long	-	Main status flag	-	-1, 0, +1
model_out[XO_INTERPOL_EXTRA_NUM_DEP_ELEMENTS]	double	all	Vector of model-dependant parameters	-	-
extra_out[XO_INTERPOL_EXTRA_NUM_INDEP_ELEMENTS]	double	all	Vector of model-independ-ant parameters	-	-
ierr[XO_NUM_ERR_INTERPOL_EXTRA]	long	all	Status vector	-	-

7.15.5 Results vectors

The model-dependant parameters vector for the **default interpolation model** is (note that there is an enumeration associated to the elements of the results vectors) in the following table:

Table 62: Ancillary results vector. Model-dependent parameters

Result parameter (res element)	Set	Description (Reference)	Unit (Format)	Allowed Range
[0] XO_INTERPOL_EXTRA_DEP_NODAL_PERIOD	ANX Tim-ing	Nodal period	s	>= 0
[1] XO_INTERPOL_EXTRA_DEP_UTC_CURRENT_ANX		UTC of current ANX	decimal days (Process-ing format)	-
[2] XO_INTERPOL_EXTRA_DEP_ORBIT_NUMBER	Position in orbit	Absolute Orbit Number		> 0
[3] XO_INTERPOL_EXTRA_DEP_SEC_SINCE_ANX		Time since ANX	s	>= 0 < Nodal Period

The model-independant results vectors are the same as the **xo_propag_extra** model-independant results vectors (see 7.11.5). The enumeration names are the same, changing PROPAG with INTERPOL (e.g. XO_INTERPOL_EXTRA_ORBIT_RAD).

7.15.6 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_interpol_extra** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xo_interpol_extra** CFI function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 63: Error messages of xo_interpol_extra function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Interpol Id. was not initialized	No calculation performed	XO_CFI_INTERPOL_EXTRA_ID_STATUS_ERR	0
ERR	Wrong input Interpol Id.	No calculation performed	XO_CFI_INTERPOL_EXTRA_FLAG_ERR	1
ERR	Could not perform a time transformation	No calculation performed	XO_CFI_INTERPOL_EXTRA_TIME_ERR	2
ERR	Could not calculate extra results	No calculation performed	XO_CFI_INTERPOL_EXTRA_RESULTS_ERR	3

7.15.7 Runtime performances

The following runtime performances have been measured:

Table 64: Runtime performances of xo_interpol_extra function

mode	Ultra sparc II-400 [ms]
XO_INTERPOL_EXTRA_NO_RESULTS	0.026
XO_INTERPOL_EXTRA_GEOLOCATION	0.030
XO_INTERPOL_EXTRA_GEOLOCATION_EXTRA	0.072
XO_INTERPOL_EXTRA_EARTH_FIXED_D	0.058
XO_INTERPOL_EXTRA_SUN	0.256
XO_INTERPOL_EXTRA_MOON	0.349
XO_INTERPOL_EXTRA_OSCULATING_KEPLER	0.026
XO_INTERPOL_EXTRA_INERTIAL_AUX	0.104
XO_INTERPOL_EXTRA_DEP_ANX_TIMING	0.040

mode	Ultra sparc II-400 [ms]
XO_INTERPOL_EXTRA_DEP_ANX_TIMING + XO_INTERPOL_EXTRA_INERTIAL_AUX	0.154
XO_INTERPOL_EXTRA_ALL_RESULTS	0.517

7.16 xo_interpol_close

7.16.1 Overview

The **xo_interpol_close** function is used to free the memory allocated by the other orbit interpolation routines, and it must be called after using them.

A complete calling sequence of the interpolation procedure is presented in section 4.3.

7.16.2 Calling interface

The calling interface of the **xo_interpol_close** CFI function is the following (input parameters are underlined):

```
#include <explorer_orbit.h>
{
      xo_interpol_id interpol_id = {NULL};
      long status, ierr[XO_NUM_ERR_INTERPOL_CLOSE];

      status = xo_interpol_close (&interpol_id, ierr)
}
```

For ForTran programs, the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
#include <explorer_orbit.inc>

      INTEGER*4 SAT_ID
      INTEGER*4 STATUS, IERR(XO_NUM_ERR_INTERPOL_CLOSE)

      STATUS = XO_INTERPOL_CLOSE (SAT_ID, IERR)
```

7.16.3 Input parameters

The **xo_interpol_close** CFI function has the following input parameters:

Table 65: Input parameters of xo_interpol_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
interpol_id	xo_interpol_id*	-	Structure that contains the interpolator data	-	-

7.16.4 Output parameters

The output parameters of the **xo_interpol_close** CFI function are:

Table 66: Output parameters of xo_interpol_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_interpol_close	long	-	Main status flag	-	-1, 0, +1
ierr[XO_NUM_ERR_INTERPOL_CLOSE]	long	all	Status vector	-	-

7.16.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_interpol_close** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (`WARN`) or an error (`ERR`), the cause of such a message and the impact on the performed calculation, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xo_interpol_close** CFI function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 67: Error messages of xo_interpol_close function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Wrong Interpol Id. It was not initialized or it is in use.	The Interpol Id. was not closed.	XO_CFI_INTERPOL_CLOSE_WRONG_ID_ERR	0

7.16.6 Runtime performances

The following runtime performances have been measured:

Table 68: Runtime performances of xo_interpol_close function

Ultra Sparc II-400[msec]
TBD

7.17 xo_orbit_to_time

7.17.1 Overview

The **xo_orbit_to_time** function converts an orbit-relative time into processing time.

7.17.2 Calling sequence of xo_orbit_to_time:

For C programs, the call to **xo_orbit_to_time** is (input parameters are underlined):

```
#include <explorer_orbit.h>
{
      xo_orbit_id   orbit_id = {NULL};
      long time_ref;
      long orbit, second, microsec;
      long status, ierr[XO_NUM_ERR_ORBIT_TO_TIME];
      double time;

      status = xo_orbit_to_time (&orbit_id,
                                 &orbit, &second, &microsec, &time_ref,
                                 &time, ierr);

      /* Or, using the run_id */
      long run_id;

      status = xo_orbit_to_time_run (&run_id,
                                     &orbit, &second, &microsec,
                                     &time_ref,
                                     &time, ierr);
}
```

For FORTRAN programs **xo_orbit_to_time** has the following calling sequence (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
       INTEGER*4   SAT_ID, TIME_REF,
      &            ORBIT, SECOND, MICROSEC,
      &            IERR(20), STATUS
       REAL*8      TIME
       CHARACTER*(*)   ORBIT_SCENARIO_FILE

#include <explorer_orbit.inc>

       STATUS = XO_ORBIT_TO_TIME ( SAT_ID,ORBIT_SCENARIO_FILE,
      &                    ORBIT, SECOND, MICROSEC, TIME_REF,
```

```
     &                          TIME, IERR)
```

7.17.3 Input parameters

Table 69: Input parameters for xo_orbit_to_time

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
orbit	long*		Orbit number		> 0
second	long*		Seconds since ascending node	s	>= 0 <orbital period
microsec	long*		Micro seconds within second	µs	0 =< =< 999999
time_ref	long*		Time reference ID	-	Complete

7.17.4 Output parameters

Table 70: Output parameters for xo_orbit_to_time

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_to_time	long		Main status flag		-1, 0, 1
time	double*		Resulting time	Dedimal days (processing format)	[-18262.0, +36519.0]
ierr[XO_NUM_ERR_ORBIT_TO_TIME]	long		Error status flags		

7.17.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_orbit_to_time** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xo_orbit_to_time** CFI function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 71: Error messages of xo_orbit_to_time function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Wrong input flag	Computation not performed	XO_CFI_ORBIT_TO_TIME_FLAG_ERR	0
ERR	Input incorrect: negative orbit number	Computation not performed	XO_CFI_ORBIT_TO_TIME_ORB_NUM_1ST_ERR	1
ERR	Orbit Id. is not initialised.	Computation not performed	XO_CFI_ORBIT_TO_TIME_ORBIT_STATUS_ERR	2
ERR	Seconds and microseconds greater than nodal period	Computation not performed	XO_CFI_ORBIT_TO_TIME_SEC_MICROSEC_ERR	3
ERR	Requested orbit less than the first orbital change	Computation not performed	XO_CFI_ORBIT_TO_TIME_ORB_ERR	4
ERR	Input incorrect: negative number of seconds	Computation not performed	XO_CFI_ORBIT_TO_TIME_SEC_ERR	5
ERR	Input incorrect: number of microseconds out of range	Computation not performed	XO_CFI_ORBIT_TO_TIME_MICROSEC_ERR	6
ERR	Error computing time.	Computation not performed	XO_CFI_ORBIT_TO_TIME_COMPUTE_ERR	7

7.17.6 Runtime performances

The following runtime performances have been measured:

Table 72: Runtime performances of xo_orbit_to_time function

Calls	Ultra Sparc II-400[msec]
TBD	TBD

7.18 xo_time_to_orbit

7.18.1 Overview

The **xo_time_to_orbit** function converts an orbit-relative time into processing time.

This CFI function requires access to one file to produce its results the Orbit Scenario File, describing all orbit changes occurring during the corresponding scenario.

7.18.2 Calling sequence of xo_time_to_orbit

For C programs, the call to **xo_time_to_orbit** is (input parameters are underlined):

```
#include <explorer_orbit.h>
{
      xo_orbit_id   orbit_id = {NULL};
      long time_ref;
      long orbit, second, microsec;
      long status, ierr[XO_NUM_ERR_ORBIT_TO_TIME];
      double time;

      status = xo_time_to_orbit ( &orbit_id,
                                  &time_ref, &time,
                                  &orbit, &second, &microsec,
                                  ierr);

      /* Or, using the run_id */
      long run_id;

      status = xo_time_to_orbit_run ( &run_id,
                                      &time_ref, &time,
                                      &orbit, &second, &microsec,
                                      ierr);
}
```

For FORTRAN programs **xo_orbit_to_time** has the following calling sequence (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
       INTEGER*4   SAT_ID, TIME_REF,
      &            ORBIT, SECOND, MICROSEC,
      &            IERR(20), STATUS
       REAL*8      TIME
       CHARACTER*(*)   ORBIT_SCENARIO_FILE

#include <explorer_orbit.inc>
```

```
 STATUS = XO_TIME_TO_ORBIT ( SAT_ID,ORBIT_EVENT_FILE,
&                    TIME_REF, TIME,
&                    ORBIT, SECOND, MICROSEC,
&                    IERR)
```

7.18.3 Input parameters

Table 73: Input parameters for xo_time_to_orbit function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
time_ref	long*		Time reference ID	-	Complete
time	double*		Requested time	Dedimal days (processing format)	[-18262.0, +36519.0]

7.18.4 Output parameters

Table 74: Output parameters for xo_time_to_orbit

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_time_to_orbit	long		Main status flag		-1, 0, 1
orbit	long*		Orbit number		> 0
second	long*		Seconds since ascending node	s	>= 0 <orbital period
microsec	long*		Micro seconds within second	µs	0 =< =< 999999
ierr[XO_NUM_ERR_TIME_TO_ORBIT]	long		Error status flags		

7.18.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_time_to_orbit** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xo_time_to_orbit** CFI function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 75: Error messages of xo_time_to_orbit function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Wrong input flag	Computation not performed	XO_CFI_TIME_TO_ORBIT_FLAG_ERR	0
ERR	Orbit Id. was not initialized.	Computation not performed	XO_CFI_TIME_TO_ORBIT_ORBIT_STATUS_ERR	1
ERR	Input incorrect: time out of range.	Computation not performed	XO_CFI_TIME_TO_ORBIT_TIME_ERR	2
ERR	Input time smaller than the first ANX time.	Computation not performed	XO_CFI_TIME_TO_ORBIT_BEFORE_RANGE_ERR	3
ERR	Could not compute the orbit number.	Computation not performed	XO_CFI_TIME_TO_ORBIT_COMPUTE_ERR	4
ERR	The current orbit initialization does not allow to compute the time.	Computation not performed	XO_CFI_TIME_TO_ORBIT_WRONG_ORBIT_MODE_ERR	5
ERR	Orbit number computed with warnings.	Computation not performed	XO_CFI_TIME_TO_ORBIT_COMPUTE_WARN	6

7.18.6 Runtime performances

The following runtime performances have been measured:

Table 76: Runtime performances of xo_time_to_orbit function

Calls	Ultra Sparc II-400[msec]
TBD	TBD

7.19 xo_orbit_info

7.19.1 Overview

The **xo_orbit_info** function retrieves from the orbit initialisation orbit information related with a certain orbit (specified by means of absolute orbit number).

7.19.2 Calling sequence of xo_orbit_info

For C programs, the call to **xo_orbit_info** is (input parameters are underlined, some may be input or output depending on the calling mode):

```
#include <explorer_orbit.h>
{
      xo_orbit_id    orbit_id = {NULL};
      long           abs_orbit;
      long           ierr[XO_NUM_ERR_ORBIT_INFO_FROM_ABS], status;
      double         result_vector[XO_ORBIT_INFO_EXTRA_NUM_ELEMENTS];

      status = xo_orbit_info (&orbit_id,
                              &abs_orbit,
                              result_vector, ierr);


      /* Or, using the run_id */
      long run_id;

      status = xo_orbit_info_run (&run_id,
                                  &abs_orbit,
                                  result_vector, ierr);
}
```

For FORTRAN programs **xo_orbit_info** has the following calling sequence (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
#include "explorer_orbit.inc"
       INTEGER*4   SAT_ID, REL_SWITCH
       INTEGER*4   ABS_ORBIT, REL_ORBIT, CYCLE, PHASE
       INTEGER*4   REPEAT_CYCLE, CYCLE_LENGT
       INTEGER*4   IERR(XO_NUM_ERR_ORBIT_INFO_FROM_ABS), STATUS
       REAL*8      RESULT_VECTOR(XO_ORBIT_INFO_EXTRA_NUM_ELEMENTS)
       CHARACTER*(*)   ORBIT_SCENARIO_FILE

      STATUS = XO_ORBIT_INFO (SAT_ID, ORBIT_SCENARIO_FILE,
      &                       ABS_ORBIT, REL_ORBIT,
      &                       CYCLE, PHASE, REL_SWITCH,
      &                       RESULT_VECTOR, IERR)
```

7.19.3 Input parameters

Table 77: Input parameters for xo_orbit_info

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
abs_orbit	long *		Absolute orbit number		within orbit_id range

7.19.4 Output parameters

Table 78: Output parameters for xo_orbit_info

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_info	long		Main status flag,		-1, 0, 1
result_vector[XO_ORBIT_INFO_EXTRA_NUM_ELEMENTS]	double	[0]	repeat_cycle	days	>0
		[1]	cycle_length	orbits	>0
		[2]	MLST drift	s/day	
		[3]	MLST	deg	>0 <360
		[4]	phasing	deg	>0 <360
		[5]	UTC time at ascending node	days (processing format)	
		[6-8]	position at ANX	m	
		[9-11]	velocity at ANX	m/s	
		[12-17]	mean keplerian elements at ANX		
		[18-23]	osculating keplerian elements at ANX		
		[24]	Nodal period	s	
ierr[XO_ORBIT_INFO_FROM_ABS]	long	all	Error status flags		

7.19.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_orbit_info** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the pv_utcanx CFI function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 79: Error messages of xo_orbit_info function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error calling genstate	Computation not performed	XO_CFI_ORBIT_INFO_RESULTS_GENSTATE_ERR	0
ERR	Time conversion error.	Computation not performed	XO_CFI_ORBIT_INFO_RESULTS_TIME_ERR	1
ERR	Error while computing the osculating keplerian elements	Computation not performed	XO_CFI_ORBIT_INFO_RESULTS_OSC_KEP_ERR	2

7.19.6 Runtime performances

The following runtime performances have been measured:

Table 80: Runtime performances of xo_orbit_info function

Calls	Ultra Sparc II-400[msec]
TBD	TBD

7.20 xo_orbit_rel_from_abs

7.20.1 Overview

The **xo_orbit_rel_from_abs** function retrieves from an Orbit Scenario File (previously initialised through the *orbit Id*) the relative orbit corresponding to a given absolute orbit number.

7.20.2 Calling sequence of xo_orbit_rel_from_abs

For C programs, the call to **xo_orbit_rel_from_abs** is (input parameters are underlined, some may be input or output depending on the calling mode):

```
#include <explorer_orbit.h>
{
      xo_orbit_id    orbit_id = {NULL};
      long           abs_orbit, rel_orbit, cycle, phase;
      long           ierr[XO_NUM_ERR_ORBIT_REL_FROM_ABS], status;

      status = xo_orbit_rel_from_abs (&orbit_id,
                                      &abs_orbit,
                                      &rel_orbit, &cycle,
                                      &phase, ierr);


      /* Or, using the run_id */
      long run_id;
      status = xo_orbit_rel_from_abs_run (&run_id,
                                          &abs_orbit,
                                          &rel_orbit, &cycle,
                                          &phase, ierr);
}
```

For FORTRAN programs **xo_orbit_rel_from_abs** has the following calling sequence (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
#include "explorer_orbit.inc"
       INTEGER*4   SAT_ID, REL_SWITCH
       INTEGER*4   ABS_ORBIT, REL_ORBIT, CYCLE, PHASE
       INTEGER*4   REPEAT_CYCLE, CYCLE_LENGT
       INTEGER*4   IERR(XO_NUM_ERR_ORBIT_INFO_FROM_ABS), STATUS
       REAL*8      RESULT_VECTOR(XO_ORBIT_INFO_EXTRA_NUM_ELEMENTS)
       CHARACTER*(*)   ORBIT_SCENARIO_FILE

      STATUS = XO_ORBIT_REL_FROM_ABS (SAT_ID, ORBIT_SCENARIO_FILE,
      &                    ABS_ORBIT, REL_ORBIT,
      &                    CYCLE, PHASE, REL_SWITCH,
      &                    RESULT_VECTOR, IERR)
```

7.20.3 Input parameters

Table 81: Input parameters for xo_orbit_rel_from_abs

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
abs_orbit	long *		Absolute orbit number		within orbit_id range

7.20.4 Output parameters

Table 82: Output parameters for xo_orbit_rel_from_abs

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_rel_from_abs	long		Main status flag,		-1, 0, 1
rel_orbit	long *		Relative orbit number		
cycle	long *		Cycle number		
phase	long *		Phase number		
ierr[XO_ORBIT_REL_FROM_ABS]	long	all	Error status flags		

7.20.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_orbit_rel_from_abs** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the pv_utcanx CFI function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 83: Error messages of xo_orbit_rel_from_abs function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Orbit Id. is not initialised.	Computation not performed	XO_CFI_ORBIT_REL_FROM_ABS_ORBIT_INIT_ERR	0
ERR	The relative orbit could not be computed with the current orbit initialization.	Computation not performed	XO_CFI_ORBIT_REL_FROM_ABS_ORBIT_WRONG_MODE_ERR	1
ERR	Wrong input orbit number	Computation not performed	XO_CFI_ORBIT_REL_FROM_ABS_WRONG_ORBIT	2

7.20.6 Runtime performances

The following runtime performances have been measured:

Table 84: Runtime performances of xo_orbit_rel_from_abs function

Calls	Ultra Sparc II-400[msec]
TBD	TBD

7.21 xo_orbit_abs_from_rel

7.21.1 Overview

The **xo_orbit_abs_from_rel** function retrieves from an Orbit Scenario File (previously initialised through the *orbit Id*) the absolute orbit corresponding to a given relative orbit number and cycle.

7.21.2 Calling sequence of xo_orbit_abs_from_rel

For C programs, the call to **xo_orbit_abs_from_rel** is (input parameters are underlined, some may be input or output depending on the calling mode):

```
#include <explorer_orbit.h>
{
      xo_orbit_id    orbit_id = {NULL};
      long           abs_orbit, rel_orbit, cycle, phase;
      long           ierr[XO_NUM_ERR_ORBIT_ABS_FROM_REL], status;

      status = xo_orbit_abs_from_rel (&orbit_id,
                                      &rel_orbit, &cycle,
                                      &abs_orbit, &phase, ierr);

      /* Or, using the run_id */
      long run_id;

      status = xo_orbit_abs_from_rel_run (&run_id,
                                          &rel_orbit, &cycle,
                                          &abs_orbit, &phase, ierr);
}
```

For FORTRAN programs **xo_orbit_abs_from_rel** has the following calling sequence (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
#include "explorer_orbit.inc"
       INTEGER*4   SAT_ID, REL_SWITCH
       INTEGER*4   ABS_ORBIT, REL_ORBIT, CYCLE, PHASE
       INTEGER*4   REPEAT_CYCLE, CYCLE_LENGT
       INTEGER*4   IERR(XO_NUM_ERR_ORBIT_INFO_FROM_ABS), STATUS
       REAL*8      RESULT_VECTOR(XO_ORBIT_INFO_EXTRA_NUM_ELEMENTS)
       CHARACTER*(*)   ORBIT_SCENARIO_FILE

      STATUS = XO_ORBIT_ABS_FROM_REL (SAT_ID, ORBIT_SCENARIO_FILE,
      &                    ABS_ORBIT, REL_ORBIT,
      &                    CYCLE, PHASE, REL_SWITCH,
      &                    RESULT_VECTOR, IERR)
```

7.21.3 Input parameters

Table 85: Input parameters for xo_orbit_abs_from_rel

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
rel_orbit	long *		Relative orbit number		
cycle	long *		Cycle number		

7.21.4 Output parameters

Table 86: Output parameters for xo_orbit_abs_from_rel

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_abs_from_rel	long		Main status flag,		-1, 0, 1
abs_orbit	long *		Absolute orbit number		within orbit_id range
phase	long *		Phase number		
ierr[XO_ORBIT_ABS_FROM_REL]	long	all	Error status flags		

7.21.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_orbit_abs_from_rel** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the pv_utcanx CFI function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 87: Error messages of xo_orbit_abs_from_rel function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Orbit Id. is not initialised.	Computation not performed	XO_CFI_ORBIT_ABS_FROM_REL_ORBIT_INIT_ERR	0
ERR	The orbit numbers could not be computed with the current orbit initialization.	Computation not performed	XO_CFI_ORBIT_ABS_FROM_REL_ORBIT_WRONG_MODE_ERR	1
ERR	Wrong input relative orbit and/or cycle.	Computation not performed	XO_CFI_ORBIT_ABS_FROM_REL_INPUT_PARAMETER_ERR	2

7.21.6 Runtime performances

The following runtime performances have been measured:

Table 88: Runtime performances of xo_orbit_abs_from_rel function

Calls	Ultra Sparc II-400[msec]
TBD	TBD

7.22 xo_orbit_abs_from_phase

7.22.1 Overview

The **xo_orbit_abs_from_phase** function retrieves from an Orbit Scenario File (previously initialised through the *orbit Id*) the absolute orbit corresponding to a given phase.

7.22.2 Calling sequence of xo_orbit_abs_from_phase

For C programs, the call to **xo_orbit_abs_from_phase** is (input parameters are underlined, some may be input or output depending on the calling mode):

```
#include <explorer_orbit.h>
{
      xo_orbit_id    orbit_id = {NULL};
      long           abs_orbit, rel_orbit, cycle, phase;
      long           ierr[XO_NUM_ERR_ORBIT_ABS_FROM_REL], status;

      status = xo_orbit_abs_from_phase (&orbit_id,
                                        &phase,
                                        &abs_orbit,
                                        &rel_orbit, &cycle,
                                        ierr);


      /* Or, using the run_id */
      long run_id;
      status = xo_orbit_abs_from_phase_run (&run_id,
                                            &phase,
                                            &abs_orbit,
                                            &rel_orbit, &cycle,
                                            ierr);
}
```

For FORTRAN programs **xo_orbit_abs_from_phase** has the following calling sequence (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
#include "explorer_orbit.inc"
       INTEGER*4   SAT_ID, REL_SWITCH
       INTEGER*4   ABS_ORBIT, REL_ORBIT, CYCLE, PHASE
       INTEGER*4   REPEAT_CYCLE, CYCLE_LENGT
       INTEGER*4   IERR(XO_NUM_ERR_ORBIT_INFO_FROM_ABS), STATUS
       REAL*8      RESULT_VECTOR(XO_ORBIT_INFO_EXTRA_NUM_ELEMENTS)
       CHARACTER*(*)   ORBIT_SCENARIO_FILE

      STATUS = XO_ORBIT_ABS_FROM_PHASE (SAT_ID, ORBIT_SCENARIO_FILE,
      &                   ABS_ORBIT, REL_ORBIT,
```

```
     &                    CYCLE, PHASE, REL_SWITCH,
     &                    RESULT_VECTOR, IERR)
```

7.22.3 Input parameters

Table 89: Input parameters for xo_orbit_abs_from_phase

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
phase	long *		Phase number		

7.22.4 Output parameters

Table 90: Output parameters for xo_orbit_abs_from_phase

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_abs_from_phase	long		Main status flag,		-1, 0, 1
abs_orbit	long *		Absolute orbit number		within orbit_id range
rel_orbit	long *		Relative orbit number		
cycle	long *		Cycle number		
ierr[XO_ORBIT_ABS_FROM_PHASE]	long	all	Error status flags		

7.22.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_orbit_abs_from_phase** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the pv_utcanx CFI function by calling the function of the EXPLORER_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 91: Error messages of xo_orbit_abs_from_phase function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Orbit Id. is not initialised.	Computation not performed	XO_CFI_ORBIT_ABS_FROM_PHASE_ORBIT_INIT_ERR	0
ERR	The orbit numbers could not be computed with the current orbit initialization.	Computation not performed	XO_CFI_ORBIT_ABS_FROM_PHASE_ORBIT_WRONG_MODE_ERR	1
ERR	Wrong input phase number.	Computation not performed	XO_CFI_ORBIT_ABS_FROM_PHASE_INPUT_PARAMETER_ERR	2

7.22.6 Runtime performances

The following runtime performances have been measured:

Table 92: Runtime performances of xo_orbit_abs_from_phase function

Calls	Ultra Sparc II-400[msec]
TBD	TBD

8 LIBRARY PRECAUTIONS

The following precautions shall be taken into account when using EXPLORER_ORBIT software library:

- When a message like

  EXPLORER_ORBIT >>> ERROR in *xo_function*: Internal computation error # *n*

  or

  EXPLORER_ORBIT >>> WARNING in *xo_function*: Internal computation warning # *n*

  appears, run the program in *verbose* mode for a complete description of warnings and errors, and call for maintenance if necessary.

9 KNOWN PROBLEMS

The following precautions shall be taken into account when using the CFI software libraries:

Table 93: Known problems

CFI library	Problem	Work around solution
Spot model	Functionality is not currently available	-
xo_propag_spot_init	Functionality is not currently available	-
xo_interpol	Extrapolation is only allowed for Doris Navigation Files and ROF files	-