

Earth Explorer Mission CFI Software

EXPLORER_LIB SOFTWARE USER MANUAL

Code: CS-MA-DMS-GS-0003
Issue: 3.0.1
Date: 13/08/04

	Name	Function	Signature
Prepared by:	Mariano Sánchez-Nogales Fabrizio Pirondini Juan José Borrego	Project Engineer Project Engineer Project Engineer	
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
28760 Tres Cantos (Madrid), SPAIN
Tel.: +34 91 806 34 50
Fax: +34 91 806 34 51
E-mail: deimos@deimos-space.com

© DEIMOS Space S.L., 2004

All Rights Reserved. No part of this document may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of DEIMOS Space S.L. or ESA.

Document Information

Contract Data		Classification	
Contract Number:	15583/01/NL/GS	Internal	<input type="checkbox"/>
		Public	<input type="checkbox"/>
Contract Issuer:	ESA / ESTEC	Industry	<input checked="" type="checkbox"/>
		Confidential	<input type="checkbox"/>

External Distribution		
Name	Organisation	Copies

Electronic handling	
Word Processor:	Adobe Framemaker 6.0
Archive Code:	P/SUM/DMS/01/026-008
Electronic file name:	cs-ma-dms-gs-0003-21

Document Status Log

Issue	Change Description	Date	Approval
1.0	New document	08/11/01	
1.1	Updated Time Transformation functions	04/02/02	
1.2	Updated the following functions: xl_change_cart_cs, xl_geod_to_cart, xl_cart_to_geod, xl_kepl_to_cart, xl_cart_to_kepl, xl_sun, xl_moon, xl_planet, xl_star_radec, xl_geod_distance, xl_time_ref_init_file, xl_time_ref_close. The xl_attitude_cs function has been removed and replaced by xp_attitude in the EXPLORER_POINTING library.	15/04/02	
1.3	Added xl_time_get_leap_second_info	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	Added xl_default_sat_init function.	30/09/03	
3.0	New initialisation strategy and interfaces	21/07/04	

Table of Contents

1. SCOPE	17
2. ACRONYMS AND NOMENCLATURE	18
2.1. Acronyms	18
2.2. Nomenclature	18
3. APPLICABLE AND REFERENCE DOCUMENTS	19
3.1. Applicable Documents	19
3.2. Reference Documents	19
4. INTRODUCTION	20
4.1. Functions Overview	20
4.1.1. Time Computations	20
4.1.2. Coordinate Systems Transformations.....	21
4.1.3. Other Basic Computations.....	22
4.2. Time Reference Transformations Calling Sequence.....	22
5. LIBRARY INSTALLATION	24
6. LIBRARY USAGE	25
6.1. Usage hints	27
6.2. General Enumerations	28
7. CFI FUNCTIONS DESCRIPTION	34
7.1. xl_time_ref_init_file.....	35
7.1.1. Overview	35
7.1.2. Calling interface	35
7.1.3. Input parameters	36
7.1.4. Output parameters	37
7.1.5. Warnings and errors	38
7.1.6. Runtime performances.....	39
7.2. xl_time_ref_init.....	40
7.2.1. Overview	40
7.2.2. Calling interface	40
7.2.3. Input parameters	41
7.2.4. Output parameters	41
7.2.5. Warnings and errors	42
7.2.6. Runtime performances.....	42
7.3. xl_time_close	43
7.3.1. Overview	43
7.3.2. Calling interface	43
7.3.3. Input parameters	44

7.3.4. Output parameters	44
7.3.5. Warnings and errors	44
7.3.6. Runtime performances.....	45
7.4. xl_run_init	46
7.4.1. Overview	46
7.4.2. Calling interface	46
7.4.3. Input parameters	47
7.4.4. Output parameters	47
7.4.5. Warnings and errors	47
7.4.6. Runtime performances.....	48
7.5. xl_run_get_ids	49
7.5.1. Overview	49
7.5.2. Calling interface	49
7.5.3. Input parameters	50
7.5.4. Output parameters	50
7.5.5. Warnings and errors	50
7.5.6. Runtime performances.....	50
7.6. xl_run_close	51
7.6.1. Overview	51
7.6.2. Calling interface	51
7.6.3. Input parameters	52
7.6.4. Output parameters	52
7.6.5. Warnings and errors	52
7.6.6. Runtime performances.....	52
7.7. xl_time_ascii_to_ascii	53
7.7.1. Overview	53
7.7.2. Calling Interface	53
7.7.3. Input Parameters	54
7.7.4. Output Parameters	55
7.7.5. Warnings and Errors	55
7.7.6. Runtime Performances	57
7.8. xl_time_ascii_to_processing	58
7.8.1. Overview	58
7.8.2. Calling Interface	58
7.8.3. Input Parameters	59
7.8.4. Output Parameters	60
7.8.5. Warnings and Errors	60
7.8.6. Runtime Performances	61
7.9. xl_time_ascii_to_transport	62
7.9.1. Overview	62
7.9.2. Calling Interface	62
7.9.3. Input Parameters	63
7.9.4. Output Parameters	64
7.9.5. Warnings and Errors	64
7.9.6. Runtime Performances	65

7.10. xl_time_processing_to_ascii	66
7.10.1. Overview	66
7.10.2. Calling Interface	66
7.10.3. Input Parameters	67
7.10.4. Output Parameters	68
7.10.5. Warnings and Errors	68
7.10.6. Runtime Performances	69
7.11. xl_time_processing_to_processing	70
7.11.1. Overview	70
7.11.2. Calling Interface	70
7.11.3. Input Parameters	71
7.11.4. Output Parameters	72
7.11.5. Warnings and Errors	72
7.11.6. Runtime Performances	73
7.12. xl_time_processing_to_transport	74
7.12.1. Overview	74
7.12.2. Calling Interface	74
7.12.3. Input Parameters	75
7.12.4. Output Parameters	76
7.12.5. Warnings and Errors	76
7.12.6. Runtime Performances	77
7.13. xl_time_transport_to_ascii	78
7.13.1. Overview	78
7.13.2. Calling Interface	78
7.13.3. Input Parameters	79
7.13.4. Output Parameters	80
7.13.5. Warnings and Errors	80
7.13.6. Runtime Performances	81
7.14. xl_time_transport_to_processing	82
7.14.1. Overview	82
7.14.2. Calling Interface	82
7.14.3. Input Parameters	83
7.14.4. Output Parameters	84
7.14.5. Warnings and Errors	84
7.14.6. Runtime Performances	85
7.15. xl_time_transport_to_transport	86
7.15.1. Overview	86
7.15.2. Calling Interface	86
7.15.3. Input Parameters	87
7.15.4. Output Parameters	88
7.15.5. Warnings and Errors	88
7.15.6. Runtime Performances	89
7.16. xl_time_add	90
7.16.1. Overview	90
7.16.2. Calling interface	90

7.16.3. Input parameters	91
7.16.4. Output parameters	91
7.16.5. Warnings and errors	91
7.16.6. Runtime performances.....	92
7.17. xl_time_diff.....	93
7.17.1. Overview	93
7.17.2. Calling interface	93
7.17.3. Input parameters	94
7.17.4. Output parameters	94
7.17.5. Warnings and errors	94
7.17.6. Runtime performances.....	95
7.18. xl_time_obt_to_time	96
7.18.1. Overview	96
7.18.2. Calling interface	97
7.18.3. Input parameters	98
7.18.4. Output parameters	99
7.18.5. Warnings and errors	100
7.18.6. Runtime performances.....	100
7.19. xl_time_time_to_obt	101
7.19.1. Overview	101
7.19.2. Calling interface	102
7.19.3. Input parameters	103
7.19.4. Output parameters	104
7.19.5. Warnings and errors	104
7.19.6. Runtime performances.....	105
7.20. xl_change_cart_cs	106
7.20.1. Overview	106
7.20.2. Calling interface	106
7.20.3. Input parameters	107
7.20.4. Output parameters	108
7.20.5. Warnings and errors	108
7.20.6. Runtime performances.....	109
7.21. xl_geod_to_cart.....	110
7.21.1. Overview	110
7.21.2. Calling interface	110
7.21.3. Input parameters	111
7.21.4. Output parameters	111
7.21.5. Warnings and errors	112
7.21.6. Runtime performances.....	112
7.22. xl_cart_to_geod.....	113
7.22.1. Overview	113
7.22.2. Calling interface	113
7.22.3. Input parameters	114
7.22.4. Output parameters	114
7.22.5. Warnings and errors	115

7.22.6. Runtime performances.....	116
7.23. xl_kepl_to_cart.....	117
7.23.1. Overview	117
7.23.2. Calling interface	117
7.23.3. Input parameters	118
7.23.4. Output parameters	118
7.23.5. Warnings and errors	118
7.23.6. Runtime performances.....	119
7.24. xl_cart_to_kepl.....	120
7.24.1. Overview	120
7.24.2. Calling interface	120
7.24.3. Input parameters	121
7.24.4. Output parameters	121
7.24.5. Warnings and errors	122
7.24.6. Runtime performances.....	122
7.25. xl_sun	123
7.25.1. Overview	123
7.25.2. Calling interface	123
7.25.3. Input parameters	124
7.25.4. Output parameters	124
7.25.5. Warnings and errors	124
7.25.6. Runtime performances.....	125
7.26. xl_moon.....	126
7.26.1. Overview	126
7.26.2. Calling interface	126
7.26.3. Input parameters	127
7.26.4. Output parameters	127
7.26.5. Warnings and errors	127
7.26.6. Runtime performances.....	128
7.27. xl_planet.....	129
7.27.1. Overview	129
7.27.2. Calling interface	129
7.27.3. Input parameters	130
7.27.4. Output parameters	130
7.27.5. Warnings and errors	130
7.27.6. Runtime performances.....	131
7.28. xl_star_radec	132
7.28.1. Overview	132
7.28.2. Calling interface	132
7.28.3. Input parameters	133
7.28.4. Output parameters	134
7.28.5. Warnings and errors	134
7.28.6. Runtime performances.....	135
7.29. xl_geod_distance.....	136
7.29.1. Overview	136

7.29.2. Calling interface	136
7.29.3. Input parameters	137
7.29.4. IOutput parameters	137
7.29.5. Warnings and errors	138
7.29.6. Runtime performances.....	138
7.30. xl_time_get_leap_second_info.....	139
7.30.1. Overview	139
7.30.2. Calling interface	139
7.30.3. Input parameters	141
7.30.4. Output parameters	141
7.30.5. Warnings and errors	142
7.30.6. Runtime performances.....	142
7.31. xl_euler_to_matrix	143
7.32. xl_matrix_to_euler	144
7.33. xl_position_on_orbit	145
7.33.1. Overview	145
7.33.2. Calling interface	145
7.33.3. Input parameters	146
7.33.4. Output parameters	147
7.33.5. Warnings and errors	147
7.33.6. Runtime performances.....	147
7.34. xl_get_rotation_angles	148
7.34.1. Overview	148
7.34.2. Calling interface	148
7.34.3. Input parameters	149
7.34.4. Output parameters	149
7.34.5. Warnings and errors	150
7.34.6. Runtime performances.....	150
7.35. xl_get_rotated_vectors	151
7.35.1. Overview	151
7.35.2. Calling interface	151
7.35.3. Input parameters	152
7.35.4. Output parameters	152
7.35.5. Warnings and errors	153
7.35.6. Runtime performances.....	153
7.36. xl_quaternions_to_vectors	154
7.36.1. Overview	154
7.36.2. Calling interface	154
7.36.3. Input parameters	155
7.36.4. Output parameters	155
7.36.5. Warnings and errors	155
7.36.6. Runtime performances.....	156
7.37. xl_vectors_to_quaternions	157
7.37.1. Overview	157
7.37.2. Calling interface	157

7.37.3. Input parameters	158
7.37.4. Output parameters	158
7.37.5. Warnings and errors	158
7.37.6. Runtime performances.....	159
7.38. xl_default_sat_init.....	160
7.38.1. Overview	160
7.38.2. Calling interface	160
7.38.3. Input parameters	161
7.38.4. Output parameters	161
7.38.5. Warnings and errors	161
7.38.6. Runtime performances.....	162
8. LIBRARY PRECAUTIONS.....	163
9. KNOWN PROBLEMS.....	164
10. Appendix: Default Satellite Configuration File.....	165

List of Tables

Table 1:	CFI functions included within EXPLORER_LIB library.....	25
Table 2:	Enumerations within EXPLORER_LIB library	28
Table 3:	Transport time formats.....	30
Table 4:	Basic ASCII time formats.....	31
Table 5:	Derived ASCII time formats.....	31
Table 6:	Definition of BOM and EOM for basic ASCII time formats	33
Table 7:	Definition of BOM and EOM for derived ASCII time formats	33
Table 8:	Time reference correlations from reference files.....	35
Table 9:	Input parameters of xl_time_ref_init_file function	36
Table 10:	Output parameters of xl_time_ref_init_file function.....	37
Table 11:	Error messages of xl_time_ref_init_file function.....	38
Table 12:	Runtime performances of xl_time_ref_init_file function.....	39
Table 13:	Input parameters of xl_time_ref_init function.....	41
Table 14:	Output parameters of xl_time_ref_init function	41
Table 15:	Error messages of xl_time_ref_init function	42
Table 16:	Runtime performances of xl_time_ref_init function	42
Table 17:	Input parameters of xl_time_close function	44
Table 18:	Output parameters of xl_time_close function.....	44
Table 19:	Error messages of xl_time_close function.....	44
Table 20:	Runtime performances of xl_time_close function.....	45
Table 21:	Input parameters of xl_run_init function	47
Table 22:	Output parameters of xl_run_init function	47
Table 23:	Error messages of xl_run_init function	47
Table 24:	Runtime performances of xl_run_init function	48
Table 25:	Input parameters of xl_run_get_ids function.....	50
Table 26:	Output parameters of xl_run_get_ids function	50
Table 27:	Runtime performances of xl_run_get_ids function	50
Table 28:	Input parameters of xl_run_close function	52
Table 29:	Output parameters of xl_run_close function	52
Table 30:	Runtime performances of xl_run_close function.....	52
Table 31:	Input parameters of xl_time_ascii_to_ascii function.....	54
Table 32:	Output parameters of xl_time_ascii_to_ascii	55
Table 33:	Error messages of xl_time_ascii_to_ascii function	55
Table 34:	Runtime performances of xl_time_ascii_to_ascii.....	57
Table 35:	Input parameters of xl_time_ascii_to_processing function.....	59
Table 36:	Output parameters of xl_time_ascii_to_processing.....	60
Table 37:	Error messages of xl_time_ascii_to_processing function	60

Table 38:	Runtime performances of xl_time_ascii_to_processing.....	61
Table 39:	Input parameters of xl_time_ascii_to_transport function.....	63
Table 40:	Output parameters of xl_time_ascii_to_transport.....	64
Table 41:	Error messages of xl_time_ascii_to_transport function	64
Table 42:	Runtime performances of xl_time_ascii_to_transport.....	65
Table 43:	Input parameters of xl_time_processing_to_ascii function	67
Table 44:	Output parameters of xl_time_processing_to_ascii.....	68
Table 45:	Error messages of xl_time_processing_to_ascii function	68
Table 46:	Runtime performances of xl_time_processing_to_ascii.....	69
Table 47:	Input parameters of xl_time_processing_to_processing function	71
Table 48:	Output parameters of xl_time_processing_to_processing.....	72
Table 49:	Error messages of xl_time_processing_to_processing function.....	72
Table 50:	Runtime performances of xl_time_processing_to_processing	73
Table 51:	Input parameters of xl_time_processing_to_transport function	75
Table 52:	Output parameters of xl_time_processing_to_transport.....	76
Table 53:	Error messages of xl_time_processing_to_transport function.....	76
Table 54:	Runtime performances of xl_time_processing_to_transport.....	77
Table 55:	Input parameters of xl_time_transport_to_ascii function.....	79
Table 56:	Output parameters of xl_time_transport_to_ascii.....	80
Table 57:	Error messages of xl_time_transport_to_ascii function	80
Table 58:	Runtime performances of xl_time_transport_to_ascii.....	81
Table 59:	Input parameters of xl_time_transport_to_processing function	83
Table 60:	Output parameters of xl_time_transport_to_processing.....	84
Table 61:	Error messages of xl_time_transport_to_processing function.....	84
Table 62:	Runtime performances of xl_time_transport_to_processing.....	85
Table 63:	Input parameters of xl_time_transport_to_transport function	87
Table 64:	Output parameters of xl_time_transport_to_transport.....	88
Table 65:	Error messages of xl_time_transport_to_transport function	88
Table 66:	Runtime performances of xl_time_transport_to_transport.....	89
Table 67:	Input parameters of xl_time_add function.....	91
Table 68:	Output parameters of xl_time_add function	91
Table 69:	Error messages of xl_time_add function	92
Table 70:	Runtime performances of xl_time_add function	92
Table 71:	Input parameters of xl_time_diff function.....	94
Table 72:	Output parameters of xl_time_diff function	94
Table 73:	Error messages of xl_time_diff function	95
Table 74:	Runtime performances of xl_time_diff function	95
Table 75:	Input parameters of xl_time_obt_to_time function	98
Table 76:	Input parameters of xl_envisat_obt_param structure.....	98
Table 77:	Input parameters of xl_envisat_obt_value structure.....	98

Table 78:	Input parameters of xl_goce_obt_param structure	99
Table 79:	Input parameters of xl_goce_obt_value structure.....	99
Table 80:	Output parameters of xl_time_obt_to_time function.....	99
Table 81:	Error messages of xl_time_obt_to_time function.....	100
Table 82:	Runtime performances of xl_time_obt_to_time function.....	100
Table 83:	Input parameters of xl_time_obt_to_time function	103
Table 84:	Input parameters of xl_envisat_obt_param structure.....	103
Table 85:	Input parameters of xl_goce_obt_param structure	103
Table 86:	Output parameters of xl_time_time_to_obt function.....	104
Table 87:	Output parameters of xl_envisat_obt_value structure	104
Table 88:	Output parameters of xl_goce_obt_value structure	104
Table 89:	Error messages of xl_time_time_to_obt function.....	105
Table 90:	Runtime performances of xl_time_time_to_obt function.....	105
Table 91:	Input parameters of xl_xhange_cart_cs function.....	107
Table 92:	Output parameters of xl_change_cart_cs function	108
Table 93:	Error messages of xl_change_cart_cs function	108
Table 94:	Runtime performances of xl_change_cart_cs function	109
Table 95:	Input parameters of xl_geod_to_cart function.....	111
Table 96:	Output parameters of xl_geod_to_cart function	111
Table 97:	Error messages of xl_geod_to_cart function	112
Table 98:	Runtime performances of xl_geod_to_cart function	112
Table 99:	Input parameters of xl_cart_to_geod function.....	114
Table 100:	Output parameters of xl_cart_to_geod function	114
Table 101:	Error messages of xl_cart_to_geod function	115
Table 102:	Runtime performances of xl_cart_to_geod function	116
Table 103:	Input parameters of xl_kepl_to_cart function.....	118
Table 104:	Output parameters of xl_kepl_to_cart function	118
Table 105:	Error messages of xl_kepl_to_cart function	119
Table 106:	Runtime performances of xl_kepl_to_cart function	119
Table 107:	Input parameters of xl_cart_to_kepl function.....	121
Table 108:	Output parameters of xl_cart_to_kepl function.....	121
Table 109:	Error messages of xl_cart_to_kepl function	122
Table 110:	Runtime performances of xl_cart_to_kepl function	122
Table 111:	Input parameters of xl_sun function	124
Table 112:	Output parameters of xl_sun function	124
Table 113:	Error messages of xl_sun function.....	125
Table 114:	Runtime performances of xl_sun function.....	125
Table 115:	Input parameters of xl_moon function.....	127
Table 116:	Output parameters of xl_moon function.....	127
Table 117:	Error messages of xl_moon function.....	128

Table 118:	Runtime performances of xl_moon function	128
Table 119:	Input parameters of xl_planet function	130
Table 120:	Output parameters of xl_planet function	130
Table 121:	Error messages of xl_planet function	131
Table 122:	Runtime performances of xl_planet function	131
Table 123:	Input parameters of xl_star_radec function	133
Table 124:	Output parameters of xl_star_radec function.....	134
Table 125:	Error messages of xl_star_radec function.....	134
Table 126:	Runtime performances of xl_star_radec function.....	135
Table 127:	Input parameters of xl_geod_distance function.....	137
Table 128:	Output parameters of xl_geod_distance function	137
Table 129:	Error messages of xl_geod_distance function	138
Table 130:	Runtime performances of xl_geod_distance function	138
Table 131:	Input parameters of xl_time_get_leap_second_info function	141
Table 132:	Output parameters of xl_time_get_leap_second_info function.....	141
Table 133:	Error messages of xl_time_get_leap_second_info function.....	142
Table 134:	Runtime performances of xl_time_get_leap_second_info function	142
Table 135:	Input parameters of xl_position_on_orbit function	146
Table 136:	Output parameters of xl_position_on_orbit function.....	147
Table 137:	Error messages of xl_position_on_orbit function.....	147
Table 138:	Runtime performances of xl_position_on_orbit function.....	147
Table 139:	Input parameters of xl_get_rotation_angles function	149
Table 140:	Output parameters of xl_get_rotation_angles function	149
Table 141:	Error messages of xl_get_rotation_angles function.....	150
Table 142:	Runtime performances of xl_get_rotation_angles function.....	150
Table 143:	Input parameters of xl_get_rotated_vectors function	152
Table 144:	Output parameters of xl_get_rotated_vectors function	152
Table 145:	Error messages of xl_get_rotated_vectors function.....	153
Table 146:	Runtime performances of xl_get_rotated_vectors function.....	153
Table 147:	Input parameters of xl_quaternions_to_vectors function	155
Table 148:	Output parameters of xl_quaternions_to_vectors function.....	155
Table 149:	Error messages of xl_quaternions_to_vectors function.....	155
Table 150:	Runtime performances of xl_quaternions_to_vectors function.....	156
Table 151:	Input parameters of xl_vectors_to_quaternions function	158
Table 152:	Output parameters of xl_vectors_to_quaternions function.....	158
Table 153:	Error messages of xl_vectors_to_quaternions function.....	158
Table 154:	Runtime performances of xl_vectors_to_quaternions function.....	159
Table 155:	Input parameters of xl_default_sat_init function.....	161
Table 156:	Output parameters of xl_default_sat_init function	161
Table 157:	Error messages of xl_default_sat_init function	162

Table 158:	Runtime performances of xl_default_sat_init function	162
Table 159:	Known problems	164

List of Figures

- Figure1: Time reference transformations sequence 22
Figure2: Azimuth figures returned by xl_geod_distance function 136

1 SCOPE

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

2 ACRONYMS AND NOMENCLATURE

2.1 Acronyms

ANX	Ascending Node Crossing
AOCS	Attitude and Orbit Control Subsystem
ASCII	American Standard Code for Information Interchange
BOM	Beginning Of Mission
CFI	Customer Furnished Item
EOM	End Of Mission
ESA	European Space Agency
ESTEC	European Space Technology and Research Centre
GPL	GNU Public License
GPS	Global Positioning System
IERS	International Earth Rotation Service
I/F	Interface
LS	Leap Second
OBT	On-board Binary Time
OSF	Orbit Scenario File
SRAR	Satellite Relative Actual Reference
SUM	Software User Manual
TAI	International Atomic Time
UTC	Coordinated Universal Time
UT1	Universal Time UT1
WGS[84]	World Geodetic System 1984

2.2 Nomenclature

<i>CFI</i>	A group of CFI functions, and related software and documentation. that will be distributed by ESA to the users as an independent unit
<i>CFI function</i>	A single function within a CFI that can be called by the user
<i>Library</i>	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 3.0. 21/07/04

3.2 Reference Documents

[MCD] Earth Explorer Mission CFI Software. Mission Conventions Document. CS-MA-DMS-GS-0001. Issue 1.4. 21/07/04.

[F_H_SUM] Earth Explorer Mission CFI Software. EXPLORER_FILE_HANDLING Software User Manual. CS-MA-DMS-GS-0008. Issue 3.0. 21/07/04.

[G_F_SUM] Earth Explorer Mission CFI Software. EXPLORER_GEN_FILES Software User Manual. CS-MA-DMS-GS-0007. Issue 3.0. 21/07/04.

[IERS] <http://www.iers.org/iers/publications/bulletins/>

4 INTRODUCTION

4.1 Functions Overview

This software library contains all low-level generic routines, supporting all the other CFI functions. The following CFI functions are included:

4.1.1 Time Computations

All time time computations are performed internally using the continuous TAI time reference. Therefore the input and output parameters are converted internally to the adequate time reference.

4.1.1.1 Time Reference Transformations Initialization

- **xl_time_ref_init_file**: initializes time correlations between TAI, UTC, UT1 and GPS times from reference data files.
- **xl_time_ref_init**: initializes time correlations between TAI, UTC, UT1 and GPS times from input reference times.
- **xl_time_close**: cleans up any memory allocation performed by the initialization functions.
- **xl_time_get_leap_second_info**: retrieves the leap second location (if any) in the initialised time range.

4.1.1.2 Time Format and Reference Transformations

- **xl_time_ascii_to_ascii**: transforms a time expressed in a given ASCII format and reference (TAI, UTC, UT1 or GPS) into a time in a different ASCII format and/or reference (TAI, UTC, UT1 or GPS).
- **xl_time_ascii_to_transport**: transforms a time expressed in a given ASCII format and reference (TAI, UTC, UT1 or GPS) into a time in a Transport format, performing a reference transformation if necessary (to TAI, UTC, UT1 or GPS).
- **xl_time_ascii_to_processing**: transforms a time expressed in a given ASCII format and reference (TAI, UTC, UT1 or GPS) into a time in Processing format, performing a reference transformation if necessary (to TAI, UTC, UT1 or GPS).
- **xl_time_processing_to_ascii**: transforms a time expressed in Processing format and a given reference (TAI, UTC, UT1 or GPS) into a time in an ASCII format, performing a reference transformation if necessary (to TAI, UTC, UT1 or GPS).
- **xl_time_processing_to_transport**: transforms a time expressed in Processing format and a given reference (TAI, UTC, UT1 or GPS) into a time in a Transport format, performing a reference transformation if necessary (to TAI, UTC, UT1 or GPS).
- **xl_time_processing_to_processing**: transforms a time expressed in Processing format and a given reference (TAI, UTC, UT1 or GPS) into a time in Processing format with a different reference (TAI, UTC, UT1 or GPS).
- **xl_time_transport_to_ascii**: transforms a time expressed in a given Transport format and reference (TAI, UTC, UT1 or GPS) into a time in an ASCII format, performing a reference transformation if necessary (to TAI, UTC, UT1 or GPS).
- **xl_time_transport_to_transport**: transforms a time expressed in a given Transport format and reference (TAI, UTC, UT1 or GPS) into a time in a different Transport format and/or reference (TAI, UTC, UT1 or GPS).

- **xl_time_transport_to_processing**: transforms a time expressed in a given Transport format and reference (TAI, UTC, UT1 or GPS) into a time in Processing format, performing a reference transformation if necessary (to TAI, UTC, UT1 or GPS).

4.1.1.3 Operation between Dates

- **xl_time_add**: adds a duration to a TAI, UTC, UT1 or GPS time expressed in Processing format.
- **xl_time_diff**: subtracts two TAI, UTC, UT1 or GPS times expressed in Processing format.

4.1.1.4 Transformations from/to On-board Times

- **xl_time_obt_to_time**: transforms an On-board Time (OBT) into a TAI, UTC, UT1 or GPS time in Processing format.
- **xl_time_time_to_obt**: transforms a TAI, UTC, UT1 or GPS time expressed in Processing format into an On-board Time (OBT).

4.1.2 Coordinate Systems Transformations

4.1.2.1 Reference Frames Transformations

- **xl_change_cart_cs**: transforms a state vector between different coordinate systems.

4.1.2.2 Attitude-related Computations

- **xl_euler_to_matrix**: computes the elements of the coordinate transformation matrix with respect to the Satellite Relative Reference System given the corresponding Euler rotation vector in the roll, pitch and yaw sequence.
- **xl_matrix_to_euler**: derives the Euler rotation vector with respect to the Satellite Relative Reference System in the roll, pitch and yaw sequence given the corresponding coordinate transformation matrix.
- **xl_get_rotation_angles**: calculates the rotation angles between two sets of orthonormal right-handed unit vectors expressed wrt an identical coordinate frame.
- **xl_get_rotated_vectors**: calculates the rotated unit vectors given a set of unit vectors and the rotation angles expressed wrt an identical coordinate frame.
- **xl_quaternions_to_vectors**: calculates the orthonormal unit vectors from a given set of quaternions.
- **xl_vectors_to_quaternions**: calculates the set of quaternions that correspond to a set of orthonormal unit vectors.

4.1.2.3 Coordinates Transformations

- **xl_geod_to_cart**: transforms from Geodetic to Cartesian coordinates.
- **xl_cart_to_geod**: transforms from Cartesian to Geodetic coordinates.

4.1.2.4 State Vector Transformations

- **xl_kepl_to_cart**: transforms from Keplerian to Cartesian coordinates.
- **xl_cart_to_kepl**: transforms from Cartesian to Keplerian coordinates.

4.1.2.5 Position on orbit calculations

- **xl_position_on_orbit**: calculates a value describing the position of the satellite within the orbit, using as input a Cartesian orbit state vector.

4.1.3 Other Basic Computations

- **xl_sun**: calculates the position and velocity of the Sun in the True of Date coordinate system
- **xl_moon**: calculates the Moon position and velocity in the True of Date coordinate system
- **xl_planet**: calculates the position and velocity of a selected planet in the Heliocentric Mean of 2000.0 coordinate system
- **xl_star_radec**: calculates the right ascension and declination of a star in the True of Date coordinate system.
- **xl_geod_distance**: calculates the geodesic distance between two points that lay on the same ellipsoid, and the azimuth of the related geodesic line at both points.

4.2 Time Reference Transformations Calling Sequence

Time reference transformations, and other functions with time as input, requires the user to initialise correlations between the different allowed time references, i.e. TAI, UTC, UT1 and GPS time. In order to accomplish such correlations, two possible strategies can be used:

- Initialisation from a single or multiple orbit files (**xl_time_ref_init_file**).
- Initialisation from a given set of time references (**xl_time_ref_init**).

The correlations are stored in a data structure, and the software returns a pointer to it, in addition to the validity range of the initialisation. This structure is referred to as the *timeId*.

Once the initialisation has been performed, the user is able to transform any date expressed in one of the allowed time references to another, through the Time Format / Reference Transformation functions. The *timeId* has to be provided to each of these functions. The process can be repeated as needed without initialising the time correlations each time.

After finalising the transformations, the *timeId* must be freed (**xl_time_close**).

A complete view of the time reference transformations sequence is presented in figure 1.

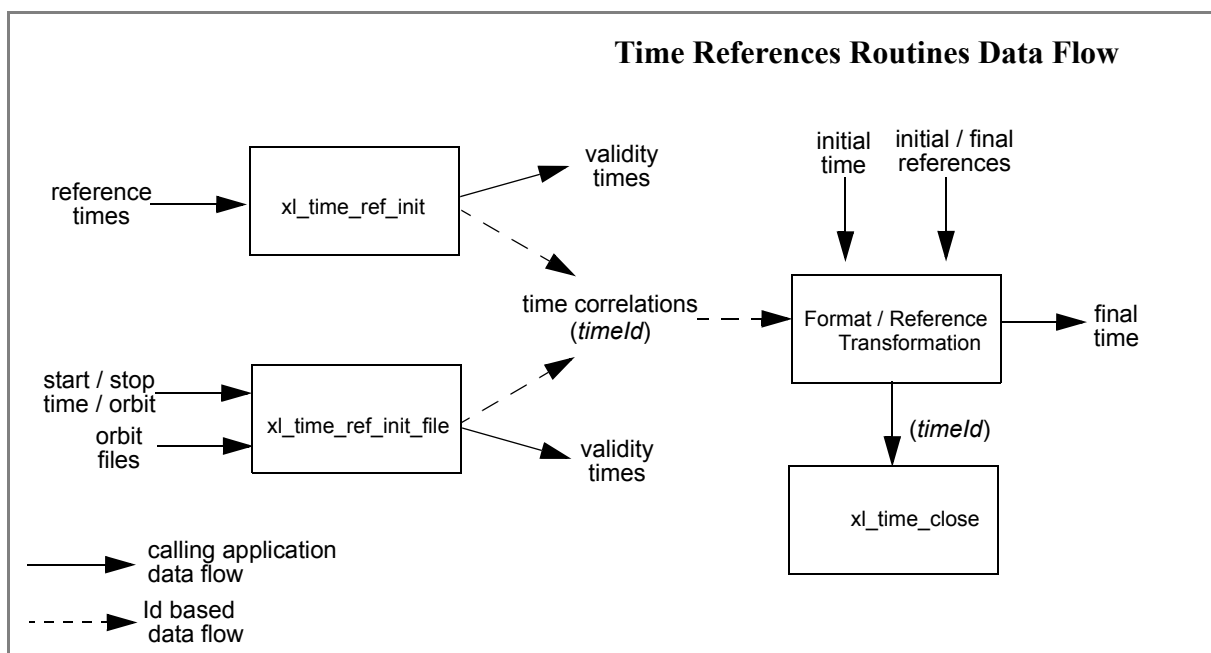


Figure 1: Time reference transformations sequence

There is a second way of calling the functions that require a *timeId* as input.

Similar initialisation functions exist in other CFI libraries, resulting in various *Ids* being generated. It is possible to group different *Ids* into a single entity called *runId*. Using this method, a single *runId* can be passed to all functions across the different libraries, instead of passing several *Ids* through the interface.

In this case, the first step would be to create the *timeId*. Then, a *runId* can be generated using as input the *timeId*. This *runId* is then passed through the interface to equivalent functions to those described before (ending in “_run”).

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, coordinate systems, 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.

5 LIBRARY INSTALLATION

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

6 LIBRARY USAGE

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

- EXPLORER_FILE_HANDLING (See [F_H_SUM]).

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

- LIBXML2 (see [GEN_SUM]).

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

- explorer_lib.h (for a C application)
- explorer_lib.inc (for a ForTran application under SOLARIS)
- explorer_lib_win.inc (for a ForTran application under Windows 95/NT)

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_lib
-lexplorer_file_handling -lxml2
```

- WINDOWS:

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

- MacOS:

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

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

To avoid problems in linking a user application with the EXPLORER_LIB 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 `XL_` or `xl_`.

It is possible to call the following CFI functions from a user application.

Table 1: CFI functions included within EXPLORER_LIB library

Function Name	Enumeration value	Long
Main CFI Functions		
xl_time_transport_to_ascii	XL_TIME_TRANSPORT_TO_ASCII_ID	0
xl_time_transport_to_transport	XL_TIME_TRANSPORT_TO_TRANSPORT_ID	1
xl_time_transport_to_processing	XL_TIME_TRANSPORT_TO_PROCESSING_ID	2
xl_time_processing_to_ascii	XL_TIME_PROCESSING_TO_ASCII_ID	3

Function Name	Enumeration value	Long
xl_time_processing_to_transport	XL_TIME_PROCESSING_TO_TRANSPORT_ID	4
xl_time_processing_to_processing	XL_TIME_PROCESSING_TO_PROCESSING_ID	5
xl_time_ascii_to_ascii	XL_TIME_ASCII_TO_ASCII_ID	6
xl_time_ascii_to_transport	XL_TIME_ASCII_TO_TRANSPORT_ID	7
xl_time_ascii_to_processing	XL_TIME_ASCII_TO_PROCESSING_ID	8
xl_time_add	XL_TIME_ADD_ID	9
xl_time_diff	XL_TIME_DIFF_ID	10
xl_time_obt_to_time	XL_TIME_OBT_TO_TIME_ID	11
xl_time_time_to_obt	XL_TIME_TIME_TO_OBT_ID	12
xl_time_ref_init_file	XL_TIME_REF_INIT_FILE_ID	13
xl_time_ref_init	XL_TIME_REF_INIT_ID	14
xl_time_ref_close	XL_TIME_CLOSE_ID	15
xl_change_cart_cs	XL_CHANGE_CART_CS_ID	16
xl_geod_to_cart	XL_GEOD_TO_CART_ID	17
xl_cart_to_geod	XL_CART_TO_GEOD_ID	18
xl_kepl_to_cart	XL_KEPL_TO_CART_ID	19
xl_cart_to_kepl	XL_CART_TO_KEPL_ID	20
xl_sun	XL_SUN_ID	21
xl_moon	XL_MOON_ID	22
xl_planet	XL_PLANET_ID	23
xl_star_radec	XL_STAR_RADEC_ID	24
xl_geod_distance	XL_GEOD_DISTANCE_ID	25
xl_time_get_leap_second_info	XL_TIME_GET_LEAP_SECOND_INFO_ID	26
xl_default_sat_init	XL_DEFAULT_SAT_INIT_ID	27
xl_instr_name_to_index	XL_INSTR_NAME_TO_INDEX_ID	28
xl_run_init	XL_RUN_INIT_ID	29
xl_quaternions_to_vectors	XL_QUATERNIONS_TO_VEC_ID	30
xl_vectors_to_quaternions	XL_VEC_TO_QUATERNIONS_ID	31
Error Handling Functions		

Function Name	Enumeration value	Long
xl_verbose	not applicable	
xl_silent		
xl_get_code		
xl_get_msg		
xl_print_msg		

Notes about the table:

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

Whenever available **it is strongly recommended to use enumeration values rather than integer values.**

6.1 Usage hints

The runtime performances of few of the CFI functions are improved to a large extent if they are called two consecutive times keeping constant some of their inputs:

- xl_change_cart_cs:time (reference and value).
- xl_planet: time (reference and value).
- xl_star_radec: time (reference and value).

In fact, the time, position, velocity and acceleration vectors do not need to keep exactly constant as long as the difference between two consecutive calls lays within the following thresholds:

- TAI/UT1/UTC time: 0.0864 microseconds
- GPS time: TBD microseconds

Furthermore, the same runtime improvement is achieved in other CFI functions that, although the user may not need to call two consecutive times with the same inputs, are called internally by other higher level CFI functions in those conditions, and thus improving the runtime performances of the latter. This is the case of the following CFI functions:

- xl_cart_to_geod: position, velocity and acceleration vectors
- xl_sun: time (reference and value)
- xl_moon: time (reference and value)

With the following thresholds:

- Position vector: 0.6e-3 m
- Velocity vector: 0.6e-6 m/s
- Acceleration vector: 0.6e-9 m/s²

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 (XL_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_LIB routines, as shown in the table below. The enumerations presented in [GEN_SUM] are also applicable.

Table 2: Enumerations within EXPLORER_LIB library

Input	Description	Enumeration value	Long
Time reference	Undefined	XL_TIME_UNDEF	-1
	TAI	XL_TIME_TAI	0
	UTC	XL_TIME_UTC	1
	UT1	XL_TIME_UT1	2
	GPS	XL_TIME_GPS	3
Processing format	Standard	XL_PROC	0
Transport time format	Standard	XL_TRANS_STD	0
	Envisat Ground Segment	XL_TRANS_ENVI_GS	11
	CryoSat Ground Segment	XL_TRANS_CRYO_GS	21
	CryoSat General Telemetry	XL_TRANS_CRYO_TM	22
	CryoSat SIRAL Telemetry	XL_TRANS_CRYO_TM_SIRAL	23
ASCII time format	Undefined	XL_ASCII_UNDEF	-1
	Standard	XL_ASCII_STD	11
	Standard with reference	XL_ASCII_STD_REF	12
	Standard with microseconds	XL_ASCII_STD_MICROSEC	13
	Standard with reference and microseconds	XL_ASCII_STD_REF_MICROSEC	14
	Compact	XL_ASCII_COMPACT	21
	Compact with reference	XL_ASCII_COMPACT_REF	22
	Compact with microseconds	XL_ASCII_COMPACT_MICROSEC	23
	Compact with reference and microseconds	XL_ASCII_COMPACT_REF_MICROSEC	24
	Envisat	XL_ASCII_ENVI	31
	Envisat with reference	XL_ASCII_ENVI_REF	32
	Envisat with microseconds	XL_ASCII_ENVI_MICROSEC	33
	Envisat with reference and microseconds	XL_ASCII_ENVI_REF_MICROSEC	34
	CCSDS-A	XL_ASCII_CCSDSA	41
	CCSDS-A with reference	XL_ASCII_CCSDSA_REF	42
	CCSDS-A with microseconds	XL_ASCII_CCSDSA_MICROSEC	43
	CCSDS-A with reference and microseconds	XL_ASCII_CCSDSA_REF_MICROSEC	44
	CCSDS-A compact	XL_ASCII_CCSDSA_COMPACT	51

Table 2: Enumerations within EXPLORER_LIB library

Input	Description	Enumeration value	Long
	CCSDS-A compact with reference	XL_ASCII_CCSDSA_COMPACT_REF	52
	CCSDS-A compact with microseconds	XL_ASCII_CCSDSA_COMPACT_MICROSEC	53
	CCSDS-A compact with reference and microseconds	XL_ASCII_CCSDSA_COMPACT_REF_MICROSEC	54
Time Initialization Mode	Initialization from file (data-driven)	XL_SEL_FILE	0
	Initialization within a time range	XL_SEL_TIME	1
	Initialization within a range of orbits	XL_SEL_ORBIT	2
	(not used in LIB)	XL_SEL_DEFAULT	3
Time Initialization Model	User defined	XL_TIMEMOD_USER	-1
	None	XL_TIMEMOD_NONE	0
	IERS Bulletin B - Table 1 (Predicted)	XL_TIMEMOD_IERS_B_PREDICTED	1
	IERS Bulletin B - Table 2 (Restituted)	XL_TIMEMOD_IERS_B_RESTITUTED	2
	FOS Predicted Orbit File	XL_TIMEMOD_FOS_PREDICTED	3
	FOS Restituted Orbit File	XL_TIMEMOD_FOS_RESTITUTED	4
	DORIS Preliminary Orbit	XL_TIMEMOD_DORIS_PRELIMINARY	5
	DORIS Precise Orbit	XL_TIMEMOD_DORIS_PRECISE	6
	DORIS Navigator	XL_TIMEMOD_DORIS_NAVIGATOR	7
Reference frame	Barycentric Mean of 2000	XL_BM2000	1
	Heliocentric Mean of 2000	XL_HM2000	2
	Geocentric Mean of 2000	XL_GM2000	3
	Mean of Date	XL_MOD	4
	True of Date	XL_TOD	5
	Earth Fixed	XL_EF	6
Kepler OSV mode	Mean Kepler State Vector	XL_KEPLER_MEAN	1
	Osculating Kepler State Vector	XL_KEPLER_OSC	2
Planet ID	Mercury	XL_MERCURY	1
	Venus	XL_VENUS	2
	Earth-Moon barycenter	XL_EM_BAR	3
	Mars	XL_MARS	4
	Jupiter	XL_JUPITER	5
	Saturn	XL_SATURN	6
	Uranus	XL_URANUS	7
	Neptune	XL_NEPTUNE	8
Calculation mode	Position	XL_CALC_POS	1
	Position and velocity	XL_CALC_POS_VEL	2
	Position, velocity and acceleration	XL_CALC_POS_VEL_ACC	3

Table 2: Enumerations within EXPLORER_LIB library

Input	Description	Enumeration value	Long
AOCS mode	Default Cx, Cy, Cz values	XL_AOCS_DEFAULT	0
	User defined Cx, Cy, Cz values	XL_AOCS_USER	1
	Geocentric pointing	XL_AOCS_GPM	2
	Local normal pointing	XL_AOCS_LNP	3
	Yaw steering + local normal pointing	XL_AOCS_YSM	4
Angle Type	True Latitude (TOD)	XL_ANGLE_TYPE_TRUE_LAT_TOD	1
	Mean Latitude (TOD)	XL_ANGLE_TYPE_MEAN_LAT_TOD	2
Derivatives	No derivative	XL_NO_DER	0
	First derivative is also calculated	XL_DER_1ST	1
	First and second derivative.	XL_DER_2ND	2
Type of <i>lds</i>	Unknown	XL_INIT_UNKNOWN	0
	<i>runld</i>	XL_INIT_RUN	1
	<i>timeld</i>	XL_INIT_TIME	2
	<i>orbitld</i> (not used in LIB)	XO_INIT_ORBIT	3
	<i>propagld</i> (not used in LIB)	XO_INIT_PROPAG	4
	<i>interpold</i> (not used in LIB)	XO_INIT_INTERPOL	5
	<i>sat_nom_att_ld</i> (not used in LIB)	XP_INIT_SAT_NOM_ATT	6
	<i>sat_att_ld</i> (not used in LIB)	XP_INIT_SAT_ATT	7
	<i>instr_att_ld</i> (not used in LIB)	XP_INIT_INSTR_ATT	8
	<i>attitudeld</i> (not used in LIB)	XP_INIT_ATTITUDE	9
	<i>atmosld</i> (not used in LIB)	XP_INIT_ATMOS	10
	<i>demld</i> (not used in LIB)	XP_INIT_DEM	11
	<i>targetld</i> (not used in LIB)	XP_INIT_TARGET	12

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**.

The meanings and units of the different array elements from the Transport time strongly depend upon the selected Transport format (by means of the Transport format ID). The table below shows the choices:

Table 3: Transport time formats

Input	Array Element	Unit (Format)	Allowed Range
XL_TRANS_STD	[0]	Integer days	[-18262,36524]
	[1]	Integer seconds	[0,86399]
	[2]	Integer microseconds	[0,999999]

Table 3: Transport time formats

Input	Array Element	Unit (Format)	Allowed Range
XL_TRANS_ENVI_GS	[0]	Integer days	[-18262,36524]
	[1]	Integer seconds	[0,86399]
	[2]	Integer microseconds	[0,999999]
XL_TRANS_CRYO_GS	[0]	Integer days	[-18262,36524]
	[1]	Integer seconds	[0,86399]
	[2]	Integer microseconds	[0,999999]
XL_TRANS_CRYO_TM	[0]	Integer days	[-18262,36524]
	[1]	Integer milliseconds	[0,86399999]
	[2]	Integer microseconds	[0,999]
XL_TRANS_CRYO_TM_SIRAL	[0]	Integer days	[-18262,36524]
	[1]	Integer milliseconds	[0,86399999]
	[2]	Integer microseconds	[0,999]
	[3]	SIRAL extra counter	[0,484848]

The string characteristics of the ASCII time formats depends strongly upon the selected ASCII format (by means of the ASCII format ID). The tables below show the available choices:

Note that the value of 86400 for seconds (and 86400000 for milliseconds) is accepted only for UTC in case a leap second is being introduced. This may happen only at 23:59 minutes and only on four days of the year (31/03, 30/06, 30/09, 31/12). The decision to introduce a leap second in UTC is the responsibility of the International Earth Rotation Service (IERS). See [IERS] for further details.

For further details on the SIRAL extra counter for the Cryosat mission please see [MCD].

Table 4: Basic ASCII time formats

Input	String format
XL_ASCII_UNDEF	-
XL_ASCII_STD	"yyyy-mm-dd_hh:nn:ss"
XL_ASCII_COMPACT	"yyyymmdd_hhnnss"
XL_ASCII_ENVI	"dd-mmm-yyyy hh:nn:ss"
XL_ASCII_CCSDSA	"yyyy-mm-ddThh:nn:ss"
XL_ASCII_CCSDSA_COMPACT	"yyyymmddThhnnss"

Table 5: Derived ASCII time formats

Input	String format
XL_ASCII_STD_REF	"RRR=yyyy-mm-dd_hh:nn:ss"
XL_ASCII_STD_MICROSEC	"yyyy-mm-dd_hh:nn:ss.uuuuuu"
XL_ASCII_STD_REF_MICROSEC	"RRR=yyyy-mm-dd_hh:nn:ss.uuuuuu"
XL_ASCII_COMPACT_REF	"RRR=yyyymmdd_hhnnss"

Table 5: Derived ASCII time formats

Input	String format
XL_ASCII_COMPACT_MICROSEC	"yyyymmdd_hhnnssuuuuuu"
XL_ASCII_COMPACT_REF_MICROSEC	"RRR=yyyymmdd_hhnnssuuuuuu"
XL_ASCII_ENVI_REF	"RRR=dd-mmm-yyyy hh:nn:ss"
XL_ASCII_ENVI_MICROSEC	"dd-mmm-yyyy hh:nn:ss.uuuuuu"
XL_ASCII_ENVI_REF_MICROSEC	"RRR=dd-mmm-yyyy hh:nn:ss.uuuuuu"
XL_ASCII_CCSDSA_REF	"RRR=yyyy-mm-ddThh:nn:ss"
XL_ASCII_CCSDSA_MICROSEC	"yyyy-mm-ddThh:nn:ss.uuuuuu"
XL_ASCII_CCSDSA_REF_MICROSEC	"RRR=yyyy-mm-ddThh:nn:ss.uuuuuu"
XL_ASCII_CCSDSA_COMPACT_REF	"RRR=yyyymmddThhnnss"
XL_ASCII_CCSDSA_COMPACT_MICROSEC	"yyyymmddThhnnssuuuuuu"
XL_ASCII_CCSDSA_COMPACT_REF_MICROSEC	"RRR=yyyymmddThhnnssuuuuuu"

where:

- *yyyy* stands for the year
- *mm* stands for the month expressed as a numerical count, i.e. 01 for January, etc
- *mmm* stands for the month expressed in abbreviations, i.e. JAN, MAR, etc
- *dd* stands for the day of month
- *ddd* stands for the day of the year
- *hh* stands for the hour in the day
- *nn* stands for the minutes within a hour
- *ss* stands for the seconds within a minute
- *uuuuuu* stands for the microseconds within a second
- *RRR* stands for the time reference (TAI, UTC, UT1 or GPS)

In ASCII formats two values are defined, by convention, as Beginning of Mission (BOM) and End of Mission (EOM). These values are listed, for the various ASCII time formats, in table 6 and table 7.

Usually a date with all zeros is seen as EOM, and a date with all nines is considered EOM. The only exception are the ENVISAT-specific formats, which use as EOM the date December 31st, 2078 at 23:59:59.999999.

Format transformations of BOM and EOM between ASCII format is allowed.

Time reference is not considered in BOM or EOM, thus any time reference is accepted (TAI, UTC, UT1 or GPS) for the values in table 6 and table 7.

BOM and EOM do not have an equivalent in Processing or Transport formats, so if the user tries to convert them from ASCII to another non-ASCII format an error will occur.

Table 6: Definition of BOM and EOM for basic ASCII time formats

ASCII format	Beginning of Mission	End of Mission
XL_ASCII_UNDEF	-	-
XL_ASCII_STD	"0000-00-00_00:00:00"	"9999-99-99_99:99:99"
XL_ASCII_COMPACT	"00000000_000000"	"99999999_999999"
XL_ASCII_ENVI	"00-000-0000_00:00:00"	"31-DEC-2078 23:59:59"
XL_ASCII_CCSDSA	"0000-00-00T00:00:00"	"9999-99-99T99:99:99"
XL_ASCII_CCSDSA_COMPACT	"00000000T000000"	"99999999T999999"

Table 7: Definition of BOM and EOM for derived ASCII time formats

ASCII format	Beginning of Mission	End of Mission
XL_ASCII_STD_REF	"RRR=0000-00-00_00:00:00"	"RRR=9999-99-99_99:99:99"
XL_ASCII_STD_MICROSEC	"0000-00-00_00:00:00.000000"	"9999-99-99_99:99:99.999999"
XL_ASCII_STD_REF_MICROSEC	"RRR=0000-00-00_00:00:00.000000"	"RRR=9999-99-99_99:99:99.999999"
XL_ASCII_COMPACT_REF	"RRR=00000000_000000"	"RRR=99999999_999999"
XL_ASCII_COMPACT_MICROSEC	"00000000_000000000000"	"99999999_999999999999"
XL_ASCII_COMPACT_REF_MICROSEC	"RRR=00000000_000000000000"	"RRR=99999999_999999999999"
XL_ASCII_ENVI_REF	"RRR=00-000-0000_00:00:00"	"RRR=31-DEC-2078 23:59:59"
XL_ASCII_ENVI_MICROSEC	"00-000-0000_00:00:00.000000"	"31-DEC-2078 23:59:59.999999"
XL_ASCII_ENVI_REF_MICROSEC	"RRR=00-000-0000_00:00:00.000000"	"RRR=31-DEC-2078 23:59:59.999999"
XL_ASCII_CCSDSA_REF	"RRR=0000-00-00T00:00:00"	"RRR=9999-99-99T99:99:99"
XL_ASCII_CCSDSA_MICROSEC	"0000-00-00T00:00:00.000000"	"9999-99-99T99:99:99.999999"
XL_ASCII_CCSDSA_REF_MICROSEC	"RRR=0000-00-00T00:00:00.000000"	"RRR=9999-99-99T99:99:99.999999"
XL_ASCII_CCSDSA_COMPACT_REF	"RRR=00000000T000000"	"RRR=99999999T999999"
XL_ASCII_CCSDSA_COMPACT_MICROSEC	"00000000T000000000000"	"99999999T999999999999"
XL_ASCII_CCSDSA_COMPACT_REF_MICROSEC	"RRR=00000000T000000000000"	"RRR=99999999T999999999999"

where:

- *RRR* stands for the time reference (TAI, UTC, UT1 or GPS)

7 CFI FUNCTIONS DESCRIPTION

The following sections describe each CFI function.

The calling interfaces are described for both C 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 \Leftrightarrow INTEGER*4, double \Leftrightarrow REAL*8)
- Array sizes of N elements (e.g. param[N] \Leftrightarrow param (N))
- Array element M (e.g. [M] \Leftrightarrow (M+1))

7.1 xl_time_ref_init_file

7.1.1 Overview

The `xl_time_ref_init_file` CFI function initializes time correlations between TAI, UTC, UT1 and GPS times from reference data files. The correlations provided by the different input files can be found in the following table.

Table 8: Time reference correlations from reference files

	TAI	UTC	UT1	GPS	orbit
FOS Predicted Orbit File	X ^a	X	X	(x)	X
FOS Restituted Orbit File	X ^a	X	X	(x)	X
DORIS Preliminary Orbit	X ^a	X	X	(x)	X
DORIS Precise Orbit	X ^a	X	X	(x)	X
IERS Bulletin B	X	X	X	(x)	

a. Not for Envisat operational files

Normally a single Predicted or DORIS Orbit file is sufficient to have all correlations needed (the (x) mark indicates that the GPS time correlation, although is not present within the file, can be simulated since it is always a fixed delta from TAI). The last updated IERS Bulletin B can be downloaded from IERS bulletins web page ([IERS]).

All other input files are ESA-provided. These initialization files could even be generated by the users by means of `EXPLORER_FILE_HANDLING` and `EXPLORER_GEN_FILES` CFI libraries.

A complete calling sequence of the time reference computations is presented in section 4.2.

7.1.2 Calling interface

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

```
#include <explorer_lib.h>
{
    long time_model, n_files, time_init_mode, time_ref;
    long orbit0, orbit1;
    char **time_file;
    double time0, time1, val_time0, val_time1;
    xl_time_id time_id;
    long ierr[XL_NUM_ERR_TIME_REF_INIT_FILE], status;

    status = xl_time_ref_init_file (&time_model, &n_files,
                                   time_file, time_init_mode,
                                   time_ref, time0, time1,
                                   orbit0, orbit1,
                                   val_time0, val_time1,
                                   &time_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_lib.inc>

    INTEGER*4 SAT_ID, TIME_MODEL, N_FILES, TIME_INIT_MODE,
&          TIME_REF, ORBIT0, ORBIT1
    CHARACTER*LENGTH_NAME TIME_FILE(NUM_FILES)
    REAL*8 TIME0, TIME1, VAL_TIME0, VAL_TIME1
    INTEGER*4 IERR(XL_NUM_ERR_TIME_REF_INIT_FILE), STATUS

    STATUS = XL_TIME_REF_INIT_FILE (SAT_ID, TIME_MODEL, N_FILES,
&          TIME_FILE, TIME_INIT_MODE,
&          TIME_REF, TIME0, TIME1,
&          ORBIT0, ORBIT1, VAL_TIME0,
&          VAL_TIME1, IERR)
```

Note that NUM_FILES must be set to the number of input files, i.e. N_FILES with a maximum value of 16 (TBD), whereas LENGTH_NAME must be set to the maximum string length of the filenames (maximum possible value of 512). All strings in ForTran must end in ‘\0’ (for compatibility with C programs).

7.1.3 Input parameters

The `xl_time_ref_init_file` CFI function has the following input parameters:

Table 9: Input parameters of `xl_time_ref_init_file` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_model	long *	-	Time model ID	-	Complete
n_files	long *	-	Number of reference data files	-	> 0
time_file	char**	-	Filenames of the reference data files	-	-
time_init_mode	long *	-	Flag for selecting the time range of the initialisation.	-	Select either: · XL_SEL_ORBIT · XL_SEL_TIME
time_ref	long *	-	Time reference ID	-	Complete
time0	double*	-	If: <i>time_init_mode</i> =XL_SEL_TIME Start of the time range defined by [time0,time1]	Decimal days (Processing format)	[-18262.0,36524.0]

Table 9: Input parameters of xl_time_ref_init_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time1	double*	-	If: <i>time_init_mode</i> = <i>XL_SEL_TIME</i> End of the time range defined by [time0,time1]	Decimal days (Processing format)	[-18262.0,36524.0] > time0
orbit0	long*	-	If: <i>time_init_mode</i> = <i>XL_SEL_ORBIT</i> Absolute orbit number corresponding to the start of the time range defined by [ANX _{orbit0} , ANX _{orbit1+1}]	-	>= 0
orbit1	long*	-	If: <i>time_init_mode</i> = <i>XL_SEL_ORBIT</i> Absolute orbit number corresponding to the end of the time range defined by [ANX _{orbit0} , ANX _{orbit1+1}]	-	>orbit0

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

- Time model ID: *time_model*. See [GEN_SUM].
- Time reference ID: *time_ref*. See [GEN_SUM].
- Time range initialisation flag: *time_init_mode*. See current document, section 6.2.

7.1.4 Output parameters

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

Table 10: Output parameters of xl_time_ref_init_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<i>xl_time_ref_init_file</i>	long	-	Status flag	-	-
<i>val_time0</i>	double*	-	Validity start time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
<i>val_time1</i>	double*	-	Validity end time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
<i>time_id</i>	<i>xl_time_id</i> *	-	Structure that contains the time correlations.	-	-
<i>ierr</i>	long	-	Error vector	-	-

Note that *val_time0* and *val_time1* can define a validity range different to that requested by the user. This range gives the maximum coverage provided by the input files within the margins selected by the user.

It has to be remarked that if the input time is outside the range of initialization, transformations are performed anyway, using the closest correlation data. However a warning is returned, since there is no guarantee that the correlation is correct.

7.1.5 Warnings and errors

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

Table 11: Error messages of xl_time_ref_init_file function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_TIME_REF_INIT_F ILE_SAT_ERR	0
ERR	Time model ID is not correct	No calculation performed	XL_CFI_TIME_REF_INIT_F ILE_TIME_MODEL_ERR	1
ERR	Non-positive number of data files	No calculation performed	XL_CFI_TIME_REF_INIT_F ILE_N_FILES_ERR	2
ERR	Incorrect file names	No calculation performed	XL_CFI_TIME_REF_INIT_F ILE_FILE_NAMES_ERR	3
ERR	Time init mode ID is not correct	No calculation performed	XL_CFI_TIME_REF_INIT_F ILE_INIT_MODE_ERR	4
ERR	Time reference ID is not correct	No calculation performed	XL_CFI_TIME_REF_INIT_F ILE_TIME_ERR	5
ERR	Reference start time out of limits	No calculation performed	XL_CFI_TIME_REF_INIT_F ILE_DAY_0_ERR	6
ERR	Reference end time out of limits	No calculation performed	XL_CFI_TIME_REF_INIT_F ILE_DAY_1_ERR	7
ERR	Wrong reference time range	No calculation performed	XL_CFI_TIME_REF_INIT_F ILE_DAY_RANGE_ERR	8
ERR	Reference start orbit is negative	No calculation performed	XL_CFI_TIME_REF_INIT_F ILE_ORB_0_ERR	9
ERR	Reference end orbit is negative	No calculation performed	XL_CFI_TIME_REF_INIT_F ILE_ORB_1_ERR	10
ERR	Wrong reference orbit range	No calculation performed	XL_CFI_TIME_REF_INIT_F ILE_ORB_RANGE_ERR	11
ERR	File does not exist	No calculation performed	XL_CFI_TIME_REF_INIT_F ILE_FILE_ERR	12
ERR	Time table is empty or has wrong format	No calculation performed	XL_CFI_TIME_REF_INIT_F ILE_TABLE_ERR	13

Table 11: Error messages of xl_time_ref_init_file function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Time range from file is outside input range	No calculation performed	XL_CFI_TIME_REF_INIT_FILE_TIME_OUTSIDE_RANGE_ERR	14
ERR	Orbit range from file is outside input range	No calculation performed	XL_CFI_TIME_REF_INIT_FILE_ORB_OUTSIDE_RANGE_ERR	15
ERR	Memory allocation error	No calculation performed	XL_CFI_TIME_REF_INIT_FILE_MEMORY_ERR	16
ERR	Error in reading file	No calculation performed	XL_CFI_TIME_REF_INIT_FILE_READ_FILE_ERR	17
ERR	Time reference ID is already initialized	No calculation performed	XL_CFI_TIME_REF_INIT_FILE_STATUS_ERR	18

7.1.6 Runtime performances

The following runtime performances have been measured (reading an IERS Bulletin B).

Table 12: Runtime performances of xl_time_ref_init_file function

Ultra Sparc II-400 [ms]
18.900

7.2 xl_time_ref_init

7.2.1 Overview

The `xl_time_ref_init` CFI function initializes time correlations between TAI, UTC, UT1 and GPS times from input reference times for time ranges from -18262.0 and +36524.0 decimal days.

A complete calling sequence of the time reference computations is presented in section 4.2.

7.2.2 Calling interface

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

```
#include <explorer_lib.h>
{
    long orbit_num;
    double time[4], anx_time, orbit_duration;
    xl_time_id time_id;
    long ierr[XL_NUM_ERR_TIME_REF_INIT], status;

    status = xl_time_ref_init (time, &orbit_num, &anx_time,
                             &orbit_duration, &time_id, ierr);
}
```

Note that input time vector must be indexed using the existing enumeration for time references.

The `XL_NUM_ERR_TIME_REF_INIT` constant is defined in the file `explorer_lib.h`.

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_lib.inc>

INTEGER*4 SAT_ID, ORBIT_NUM
REAL*8 TIME(4), ANX_TIME, ORBIT_DURATION
INTEGER*4 IERR(XL_NUM_ERR_TIME_REF_INIT), STATUS

STATUS = XL_TIME_REF_INIT (SAT_ID, TIME, ORBIT_NUM, ANX_TIME,
&                          ORBIT_DURATION, IERR)
```


7.2.3 Input parameters

The `xl_time_ref_init` CFI function has the following input parameters:

Table 13: Input parameters of `xl_time_ref_init` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time	double[4]	[0]	TAI input time	Decimal days (Processing format)	[-18262.0,36524.0]
		[1]	UTC input time	Decimal days (Processing format)	[-18262.0,36524.0]
		[2]	UT1 input time	Decimal days (Processing format)	[-18262.0,36524.0]
		[3]	GPS input time	Decimal days (Processing format)	[-18262.0,36524.0]
orbit_num	long*	-	Absolute orbit number at the reference time	-	>=0
anx_time	double*	-	Time since Ascending node crossing at the reference time	Seconds	[0,orbit_duration]
orbit_duration	double*	-	Duration of the orbit containing the reference time	Seconds	>0

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

- Time vector can be accessible by means of enumeration values, as defined in [GEN_SUM].

7.2.4 Output parameters

The output parameters of the `xl_time_ref_init` CFI function are:

Table 14: Output parameters of `xl_time_ref_init` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xl_time_ref_init	long	-	Status flag	-	-
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
ierr	long	-	Error vector	-	-

7.2.5 Warnings and errors

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

Table 15: Error messages of xl_time_ref_init function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_TIME_REF_INIT_SAT_ERR	0
ERR	TAI time is out of range	No calculation performed	XL_CFI_TIME_REF_INIT_TAI_ERR	1
ERR	UTC time is out of range	No calculation performed	XL_CFI_TIME_REF_INIT_UTC_ERR	2
ERR	UT1 time is out of range	No calculation performed	XL_CFI_TIME_REF_INIT_UT1_ERR	3
ERR	GPS time is out of range	No calculation performed	XL_CFI_TIME_REF_INIT_GPS_ERR	4
ERR	Absolute orbit number is negative	No calculation performed	XL_CFI_TIME_REF_INIT_ORBNUM_ERR	5
ERR	Elapsed time since ANX is negative	No calculation performed	XL_CFI_TIME_REF_INIT_ANXTIME_ERR	6
ERR	Orbit duration is negative	No calculation performed	XL_CFI_TIME_REF_INIT_ORBDUR_ERR	7
ERR	ANX time is bigger than orbit duration	No calculation performed	XL_CFI_TIME_REF_INIT_COMP_ERR	8
ERR	Memory allocation error	No calculation performed	XL_CFI_TIME_REF_INIT_MEMORY_ERR	9
ERR	Time reference ID is already initialized	No calculation performed	XL_CFI_TIME_REF_INIT_STATUS_ERR	9

7.2.6 Runtime performances

The following runtime performances have been measured.

Table 16: Runtime performances of xl_time_ref_init function

Ultra Sparc II-400 [ms]
0.0020

7.3 xl_time_close

7.3.1 Overview

The **xl_time_close** CFI function cleans up any memory allocation performed by the initialization functions. A complete calling sequence of the time reference computations is presented in section 4.2.

7.3.2 Calling interface

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

```
#include <explorer_lib.h>
{
    xl_time_id time_id;
    long ierr[XL_NUM_ERR_TIME_CLOSE], status;
    status = xl_time_close (&time_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):

```
#include <explorer_lib.inc>

INTEGER*4 SAT_ID
STATUS = XL_TIME_CLOSE (SAT_ID)
```

7.3.3 Input parameters

The `xl_time_close` CFI function has the following input parameters:

Table 17: Input parameters of `xl_time_close` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-

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

- Satellite ID: sat_id. See [GEN_SUM].

7.3.4 Output parameters

The output parameters of the `xl_time_close` CFI function are:

Table 18: Output parameters of `xl_time_close` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xl_time_close	long	-	Status flag	-	-
ierr	long	-	Error vector	-	-

7.3.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_time_close` CFI function after translating the returned extended status flag into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 `xl_time_close` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 19: Error messages of `xl_time_close` function

Error type	Error message	Cause and impact	Error code	Error No
ERR	The Time Id is not initialized or it could be in use by another Id.	No calculation performed	XL_CFI_TIME_CLOSE_WRONG_ID_ERR	0

7.3.6 Runtime performances

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

Table 20: Runtime performances of xl_time_close function

Ultra Sparc II-400 [ms]
0.0020

7.4 xl_run_init

7.4.1 Overview

The **xl_run_init** CFI function groups into a single *id* the *satellite Id* and the *time Id*, creating a *run Id*.

7.4.2 Calling interface

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

```
#include <explorer_lib.h>
{
    long sat_id, run_id;
    xl_time_id time_id;
    long ierr[XL_NUM_ERR_RUN_INIT], status;
    status = xl_run_init (&sat_id, &time_id, &run_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

7.4.3 Input parameters

The `xl_run_init` CFI function has the following input parameters:

Table 21: Input parameters of `xl_run_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 correlations.	-	-

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

- Satellite ID: `sat_id`. See [GEN_SUM].

7.4.4 Output parameters

The output parameters of the `xl_run_init` CFI function are:

Table 22: Output parameters of `xl_run_init` function

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

7.4.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_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_LIB software library `xl_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 `xl_run_init` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM])

Table 23: Error messages of `xl_run_init` function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Maximum number of initializations reached	No calculation performed	XL_CFI_RUN_INIT_MAX_INIT_ERR	0

Table 23: Error messages of xl_run_init function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_RUN_INIT_SAT_ERR	1
ERR	Time ID is not initialized	No calculation performed	XL_CFI_RUN_INIT_TIME_INIT_ERR	2
ERR	Memory allocation error	No calculation performed	XL_CFI_RUN_INIT_MEMORY_ERR	3
ERR	Inconsistency between Ids within the run_id	No calculation performed	XL_CFI_RUN_INIT_INCONSISTENCY_ERR	4

7.4.6 Runtime performances

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

Table 24: Runtime performances of xl_run_init function

Ultra Sparc II-400 [ms]
TBD

7.5 xl_run_get_ids

7.5.1 Overview

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

7.5.2 Calling interface

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

```
#include <explorer_lib.h>
{
    long sat_id, run_id;
    xl_time_id time_id;
    long status;
    status = xl_run_get_ids (&run_id, &time_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

7.5.3 Input parameters

The `xl_run_get_ids` CFI function has the following input parameters:

Table 25: Input parameters of `xl_run_get_ids` function

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

7.5.4 Output parameters

The output parameters of the `xl_run_close` CFI function are:

Table 26: Output parameters of `xl_run_get_ids` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xl_run_get_ids	long	-	Status flag	-	-
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-

7.5.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_run_get_ids` CFI function after translating the returned extended status flag into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 `xl_run_get_ids` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

TBW

7.5.6 Runtime performances

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

Table 27: Runtime performances of `xl_run_get_ids` function

Ultra Sparc II-400 [ms]
TBD

7.6 xl_run_close

7.6.1 Overview

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

7.6.2 Calling interface

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

```
#include <explorer_lib.h>
{
    long run_id;
    status = xl_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_lib.inc>

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

7.6.3 Input parameters

The `xl_run_close` CFI function has the following input parameters:

Table 28: Input parameters of `xl_run_close` function

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

7.6.4 Output parameters

The output parameters of the `xl_run_close` CFI function are:

Table 29: Output parameters of `xl_run_close` function

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

7.6.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_run_close` CFI function after translating the returned extended status flag into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 `xl_run_close` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM])

TBW

7.6.6 Runtime performances

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

Table 30: Runtime performances of `xl_run_close` function

Ultra Sparc II-400 [ms]
TBD

7.7 xl_time_ascii_to_ascii

7.7.1 Overview

The **xl_time_ascii_to_ascii** CFI function transforms a time expressed in a given ASCII format and reference (TAI, UTC, UT1 or GPS) into a time in a different ASCII format and/or reference (TAI, UTC, UT1 or GPS).

7.7.2 Calling Interface

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

```
#include <explorer_lib.h>
{
    long ascii_id_in, ascii_id_out;
    long time_ref_in, time_ref_out;
    char ascii_in[XL_TIME_ASCII_DIM_MAX];
    char ascii_out[XL_TIME_ASCII_DIM_MAX];
    xl_time_id time_id;
    long ierr[XL_NUM_ERR_ASCII_ASCII], status;

    status = xl_time_ascii_to_ascii(&time_id, &ascii_id_in,
                                   &time_ref_in, &ascii_in, &ascii_id_out,
                                   &time_ref_out, ascii_out, ierr);

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

    status = xl_time_ascii_to_ascii_run(&run_id, &ascii_id_in,
                                       &time_ref_in, &ascii_in, &ascii_id_out,
                                       &time_ref_out, ascii_out, ierr);
}
```

The **XL_TIME_ASCII_DIM_MAX** and **XL_NUM_ERR_ASCII_ASCII** constants are defined in the file *explorer_lib.h*.

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_lib.inc>
    INTEGER*4 SAT_ID, ASCII_ID_IN, ASCII_ID_OUT
    INTEGER*4 TIME_REF_IN, TIME_REF_OUT
```

```
CHARACTER ASCII_IN(XL_TIME_ASCII_DIM_MAX)
CHARACTER ASCII_OUT(XL_TIME_ASCII_DIM_MAX)
INTEGER*4 IERR(XL_NUM_ERR_ASCII_ASCII), STATUS
```

```
STATUS = XL_TIME_ASCII_TO_ASCII(SAT_ID, ASCII_ID_IN, TIME_REF_IN,
& ASCII_IN, ASCII_ID_OUT, TIME_REF_OUT,
& ASCII_OUT, IERR)
```

7.7.3 Input Parameters

The `xl_time_ascii_to_ascii` CFI function has the following input parameters:

Table 31: Input parameters of `xl_time_ascii_to_ascii` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
ascii_id_in	long *	-	ASCII format ID	-	Complete
time_ref_in	long *	-	Time reference ID	-	Complete
ascii_in	char	See table 4 and table 5	Time in ASCII format	See table 4 and table 5	See table 4 and table 5
ascii_id_out	long *	-	ASCII format ID	-	Complete
time_ref_out	long *	-	Time reference ID	-	Any except XL_TIME_UNDEF

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

- ASCII format ID: `ascii_id_in` and `ascii_id_out`. Current document, section 6.2.
- Time reference ID: `time_ref_in` and `time_ref_out`. See [GEN_SUM].

It is important to point out the usage of the **time_ref_in** parameter in the frame of the current function:

- If **time_ref_in** input parameter is defined, it shall be used by the function.
- If **time_ref_in** input parameter is undefined, it shall be used the time reference part from the `ascii` format string. In case this is omitted, an error shall be returned.

Note that for the function to work correctly, the time references should be properly initialised before calling the function (see section 4.2 for details), unless `time_ref_in = time_ref_out`.

7.7.4 Output Parameters

The output parameters of the `xl_time_ascii_to_ascii` CFI function are:

Table 32: Output parameters of `xl_time_ascii_to_ascii`

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xl_time_ascii_to_ascii</code>	long	-	Status flag	-	-
<code>ascii_out</code>	char	See table 4 and table 5	Time in ASCII format	See table 4 and table 5	See table 4 and table 5
<code>ierr</code>	long	-	Error vector	-	-

7.7.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the `xl_time_ascii_to_ascii` CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 by translating the error vector returned by the `xl_time_ascii_to_ascii` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 33: Error messages of `xl_time_ascii_to_ascii` function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_TIME_ASCII_ASCII_SAT_ERR	0
ERR	Input ascii format ID is not correct	No calculation performed	XL_CFI_TIME_ASCII_ASCII_IN_ERR	1
ERR	Input time reference ID is not correct	No calculation performed	XL_CFI_TIME_ASCII_ASCII_TIME_IN_ERR	2
ERR	Satellite ID and input format ID are not compatible	No calculation performed	XL_CFI_TIME_ASCII_ASCII_COMP_IN_ERR	3
ERR	Input ascii format is not correct	No calculation performed	XL_CFI_TIME_ASCII_ASCII_FORMAT_IN_ERR	4
ERR	Output ascii format ID is not correct	No calculation performed	XL_CFI_TIME_ASCII_ASCII_OUT_ERR	5
ERR	Output time reference ID is not correct	No calculation performed	XL_CFI_TIME_ASCII_ASCII_TIME_OUT_ERR	6

Table 33: Error messages of xl_time_ascii_to_ascii function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID and output format ID are not compatible	No calculation performed	XL_CFI_TIME_ASCII_ASCII_COMP_OUT_ERR	7
ERR	Input ascii year is out of range	No calculation performed	XL_CFI_TIME_ASCII_ASCII_YEAR_IN_ERR	8
ERR	Input ascii month is out of range	No calculation performed	XL_CFI_TIME_ASCII_ASCII_MONTH_IN_ERR	9
ERR	Input ascii day is out of range	No calculation performed	XL_CFI_TIME_ASCII_ASCII_DAY_IN_ERR	10
ERR	Input ascii hour is out of range	No calculation performed	XL_CFI_TIME_ASCII_ASCII_HOUR_IN_ERR	11
ERR	Input ascii minutes are out of range	No calculation performed	XL_CFI_TIME_ASCII_ASCII_MIN_IN_ERR	12
ERR	Input ascii seconds are out of range	No calculation performed	XL_CFI_TIME_ASCII_ASCII_SEC_IN_ERR	13
ERR	Input ascii microseconds are out of range	No calculation performed	XL_CFI_TIME_ASCII_ASCII_MICROSEC_IN_ERR	14
ERR	Internal error: Input Gregorian date to MJD transformation failed	No calculation performed	XL_CFI_TIME_ASCII_ASCII_MJD_IN_ERR	15
ERR	Internal error: Output ascii MJD is out of range	No calculation performed	XL_CFI_TIME_ASCII_ASCII_MJD_OUT_ERR	16
ERR	Internal error: Output ascii year is out of range	No calculation performed	XL_CFI_TIME_ASCII_ASCII_YEAR_OUT_ERR	17
ERR	Internal error: Output ascii month is out of range	No calculation performed	XL_CFI_TIME_ASCII_ASCII_MONTH_OUT_ERR	18
ERR	Internal error: Output ascii day is out of range	No calculation performed	XL_CFI_TIME_ASCII_ASCII_DAY_OUT_ERR	19
ERR	Internal error: Output ascii hour is out of range	No calculation performed	XL_CFI_TIME_ASCII_ASCII_HOUR_OUT_ERR	20
ERR	Internal error: Output ascii minutes are out of range	No calculation performed	XL_CFI_TIME_ASCII_ASCII_MIN_OUT_ERR	21
ERR	Internal error: Output ascii seconds are out of range	No calculation performed	XL_CFI_TIME_ASCII_ASCII_SEC_OUT_ERR	22
ERR	Internal error: Output ascii microseconds are out of range	No calculation performed	XL_CFI_TIME_ASCII_ASCII_MICROSEC_OUT_ERR	23
ERR	Internal error: Output ascii format is not correct	No calculation performed	XL_CFI_TIME_ASCII_ASCII_FORMAT_OUT_ERR	24
ERR	Time reference not initialised	No calculation performed	XL_CFI_TIME_ASCII_ASCII_REF_INIT_ERR	25

Table 33: Error messages of xl_time_ascii_to_ascii function

Error type	Error message	Cause and impact	Error code	Error No
WARN	Time out of initialization range	Calculation performed. A message informs the user.	XL_CFI_TIME_ASCII_ASCII_REF_INIT_WARN	26

7.7.6 Runtime Performances

The following runtime performances have been measured.

Table 34: Runtime performances of xl_time_ascii_to_ascii

Ultra Sparc II-400 [ms]
0.0470

7.8 xl_time_ascii_to_processing

7.8.1 Overview

The `xl_time_ascii_to_processing` CFI function transforms a time expressed in a given ASCII format and reference (TAI, UTC, UT1 or GPS) into a time in Processing format, performing a reference transformation if necessary (to TAI, UTC, UT1 or GPS).

User should be aware that the use of UTC in Processing format is not encouraged, due to the discontinuity that is caused by the introduction of leap seconds. See [IERS] for further details.

7.8.2 Calling Interface

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

```
#include <explorer_lib.h>
{
    long ascii_id_in, proc_id_out;
    long time_ref_in, time_ref_out;
    char ascii_in[XL_TIME_ASCII_DIM_MAX];
    double processing_out;
    xl_time_id time_id;
    long ierr[XL_NUM_ERR_ASCII_PROC], status;

    status = xl_time_ascii_to_processing(&time_id, &ascii_id_in,
                                       &time_ref_in, ascii_in, &proc_id_out,
                                       &time_ref_out, &processing_out, ierr);

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

    status = xl_time_ascii_to_processing_run(&run_id, &ascii_id_in,
                                           &time_ref_in, ascii_in, &proc_id_out,
                                           &time_ref_out, &processing_out, ierr);
}
```

The `XL_TIME_ASCII_DIM_MAX` and `XL_NUM_ERR_ASCII_PROC` constants are defined in the file `explorer_lib.h`.

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_lib.inc>
    INTEGER*4 SAT_ID, ASCII_ID_IN, PROC_ID_OUT
```

```

INTEGER*4 TIME_REF_IN, TIME_REF_OUT
CHARACTER ASCII_IN(XL_TIME_ASCII_DIM_MAX)
REAL*8 PROCESSING_OUT
INTEGER*4 IERR(XL_NUM_ERR_ASCII_PROC), STATUS
STATUS = XL_TIME_ASCII_TO_PROCESSING(SAT_ID, ASCII_ID_IN,
&                                     TIME_REF_IN, ASCII_IN, PROC_ID_OUT,
&                                     TIME_REF_OUT, PROCESSING_OUT, IERR)
    
```

7.8.3 Input Parameters

The `xl_time_ascii_to_processing` CFI function has the following input parameters:

Table 35: Input parameters of `xl_time_ascii_to_processing` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>time_id</code>	<code>xl_time_id*</code>	-	Structure that contains the time correlations.	-	-
<code>ascii_id_in</code>	<code>long *</code>	-	ASCII format ID	-	Complete
<code>time_ref_in</code>	<code>long *</code>	-	Time reference ID	-	Complete
<code>ascii_in</code>	<code>char</code>	See table 4 and table 5	Time in ASCII format	See table 4 and table 5	See table 4 and table 5
<code>proc_id_out</code>	<code>long *</code>	-	Processing format ID	-	Complete
<code>time_ref_out</code>	<code>long *</code>	-	Time reference ID	-	Any except XL_TIME_UNDEF

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

- ASCII format ID: `ascii_id_in`. Current document, section 6.2.
- Time reference ID: `time_ref_in` and `time_ref_out`. See [GEN_SUM].
- Processing format ID: `proc_id_out`. Current document, section 6.2

It is important to point out the usage of the **`time_ref_in`** parameter in the frame of the current function:

- If **`time_ref_in`** input parameter is defined, it shall be used by the function.
- If **`time_ref_in`** input parameter is undefined, it shall be used the time reference part from the `ascii` format string. In case this is omitted, an error shall be returned.

Note that for the function to work correctly, the time references should be properly initialised before calling the function (see section 4.2 for details), unless `time_ref_in = time_ref_out`.

7.8.4 Output Parameters

The output parameters of the `xl_time_ascii_to_processing` CFI function are:

Table 36: Output parameters of `xl_time_ascii_to_processing`

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xl_time_ascii_to_processing</code>	long	-	Status flag	-	-
<code>processing_out</code>	double*	-	Time in Processing Format	Decimal days, MJD2000 (Processing)	[-18262.0,36524.0]
<code>ierr</code>	long	-	Error vector	-	-

7.8.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the `xl_time_ascii_to_processing` CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 by translating the error vector returned by the `xl_time_ascii_to_processing` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 37: Error messages of `xl_time_ascii_to_processing` function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	<code>XL_CFI_TIME_ASCII_PRO C_SAT_ERR</code>	0
ERR	Input ascii format ID is not correct	No calculation performed	<code>XL_CFI_TIME_ASCII_PRO C_ASCII_IN_ERR</code>	1
ERR	Input time reference ID is not correct	No calculation performed	<code>XL_CFI_TIME_ASCII_PRO C_TIME_IN_ERR</code>	2
ERR	Satellite ID and input format ID are not compatible	No calculation performed	<code>XL_CFI_TIME_ASCII_PRO C_COMP_IN_ERR</code>	3
ERR	Input format is not correct	No calculation performed	<code>XL_CFI_TIME_ASCII_PRO C_FORMAT_IN_ERR</code>	4
ERR	Output processing format ID is not correct	No calculation performed	<code>XL_CFI_TIME_ASCII_PRO C_PROC_OUT_ERR</code>	5
ERR	Output time reference ID is not correct	No calculation performed	<code>XL_CFI_TIME_ASCII_PRO C_TIME_OUT_ERR</code>	6
ERR	Year is out of range	No calculation performed	<code>XL_CFI_TIME_ASCII_PRO C_YEAR_ERR</code>	7

Table 37: Error messages of xl_time_ascii_to_processing function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Month is out of range	No calculation performed	XL_CFI_TIME_ASCII_PRO C_MONTH_ERR	8
ERR	Day is out of range	No calculation performed	XL_CFI_TIME_ASCII_PRO C_DAY_ERR	9
ERR	Hour is out of range	No calculation performed	XL_CFI_TIME_ASCII_PRO C_HOUR_ERR	10
ERR	Minutes are out of range	No calculation performed	XL_CFI_TIME_ASCII_PRO C_MIN_ERR	11
ERR	Seconds are out of range	No calculation performed	XL_CFI_TIME_ASCII_PRO C_SEC_ERR	12
ERR	Microseconds are out of range	No calculation performed	XL_CFI_TIME_ASCII_PRO C_MICROSEC_ERR	13
ERR	Internal Error: Input Gregorian date to MJD transformation failed	No calculation performed	XL_CFI_TIME_ASCII_PRO C_MJD_ERR	14
ERR	Time reference not initialised	No calculation performed	XL_CFI_TIME_ASCII_PRO C_REF_INIT_ERR	15
WARN	Time out of initialization range	Calculation performed. A message informs the user.	XL_CFI_TIME_ASCII_PRO C_REF_INIT_WARN	16

7.8.6 Runtime Performances

The following runtime performances have been measured.

Table 38: Runtime performances of xl_time_ascii_to_processing

Ultra Sparc II-400 [ms]
0.0200

7.9 xl_time_ascii_to_transport

7.9.1 Overview

The **xl_time_ascii_to_transport** CFI function transforms a time expressed in a given ASCII format and reference (TAI, UTC, UT1 or GPS) into a time in a Transport format, performing a reference transformation if necessary (to TAI, UTC, UT1 or GPS).

7.9.2 Calling Interface

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

```
#include <explorer_lib.h>
{
    long ascii_id_in, trans_id_out;
    long time_ref_in, time_ref_out;
    char ascii_in[XL_TIME_ASCII_DIM_MAX];
    long transport_out[XL_TIME_TRANS_DIM_MAX];
    xl_time_id time_id;
    long ierr[XL_NUM_ERR_ASCII_TRANS], status;

    status = xl_time_ascii_to_transport(&time_id, &ascii_id_in,
                                       &time_ref_in, &ascii_in, &trans_id_out,
                                       &time_ref_out, transport_out, ierr);

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

    status = xl_time_ascii_to_transport_run(&run_id, &ascii_id_in,
                                           &time_ref_in, &ascii_in, &trans_id_out,
                                           &time_ref_out, transport_out, ierr);
}
```

The XL_TIME_TRANS_DIM_MAX, XL_TIME_ASCII_DIM_MAX, XL_NUM_ERR_ASCII_TRANS constants are defined in the file *explorer_lib.h*.

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_lib.inc>
    INTEGER*4 SAT_ID, TRANS_ID_IN, TRANS_ID_OUT
```

```

INTEGER*4 TIME_REF_IN, TIME_REF_OUT
CHARACTER ASCII_IN(XL_TIME_ASCII_DIM_MAX)
INTEGER*4 TRANSPORT_OUT(XL_TIME_TRANS_DIM_MAX)
INTEGER*4 IERR(XL_NUM_ERR_ASCII_TRANS), STATUS

STATUS = XL_TIME_ASCII_TO_TRANSPORT(SAT_ID, ASCII_ID_IN,
&                                     TIME_REF_IN, ASCII_IN, TRANS_ID_OUT,
&                                     TIME_REF_OUT, TRANSPORT_OUT, IERR)
  
```

7.9.3 Input Parameters

The `xl_time_ascii_to_transport` CFI function has the following input parameters:

Table 39: Input parameters of `xl_time_ascii_to_transport` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
ascii_id_in	long *	-	ASCII format ID	-	Complete
time_ref_in	long *	-	Time reference ID	-	Complete
ascii_in	char	See table 4 and table 5	Time in ASCII format	See table 4 and table 5	See table 4 and table 5
trans_id_out	long *	-	Transport format ID	-	Complete
time_ref_out	long *	-	Time reference ID	-	Any except XL_TIME_UNDEF

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

- Satellite ID: `sat_id`. See [GEN_SUM].
- ASCII format ID: `trans_id_in`. Current document, section 6.2.
- Time reference ID: `time_ref_in` and `time_ref_out`. See [GEN_SUM].
- Transport format ID: `trans_id_out`. Current document, section 6.2.

It is important to point out the usage of the **time_ref_in** parameter in the frame of the current function:

- If **time_ref_in** input parameter is defined, it shall be used by the function.
- If **time_ref_in** input parameter is undefined, it shall be used the time reference part from the ascii format string. In case this is omitted, an error shall be returned.

Note that for the function to work correctly, the time references should be properly initialised before calling the function (see section 4.2 for details), unless `time_ref_in = time_ref_out`.

7.9.4 Output Parameters

The output parameters of the `xl_time_ascii_to_transport` CFI function are:

Table 40: Output parameters of `xl_time_ascii_to_transport`

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xl_time_ascii_to_transport</code>	long	-	Status flag	-	-
<code>transport_out[dim]</code>	long	See table 3	Time in Transport format	See table 3	See table 3
<code>ierr</code>	long	-	Error vector	-	-

7.9.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the `xl_time_ascii_to_transport` CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 by translating the error vector returned by the `xl_time_ascii_to_transport` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 41: Error messages of `xl_time_ascii_to_transport` function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_TIME_ASCII_TRANS_SAT_ERR	0
ERR	Input ascii format ID is not correct	No calculation performed	XL_CFI_TIME_ASCII_TRANS_ASCII_IN_ERR	1
ERR	Input time reference ID is not correct	No calculation performed	XL_CFI_TIME_ASCII_TRANS_TIME_IN_ERR	2
ERR	Satellite ID and input format ID are not compatible	No calculation performed	XL_CFI_TIME_ASCII_TRANS_COMP_IN_ERR	3
ERR	Input format is not correct	No calculation performed	XL_CFI_TIME_ASCII_TRANS_FORMAT_IN_ERR	4
ERR	Output transport format ID is not correct	No calculation performed	XL_CFI_TIME_ASCII_TRANS_TRANS_OUT_ERR	5
ERR	Output time reference ID is not correct	No calculation performed	XL_CFI_TIME_ASCII_TRANS_TIME_OUT_ERR	6
ERR	Satellite ID and output format ID are not compatible	No calculation performed	XL_CFI_TIME_ASCII_TRANS_COMP_OUT_ERR	7

Table 41: Error messages of *xl_time_ascii_to_transport* function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Year is out of range	No calculation performed	XL_CFI_TIME_ASCII_TRANS_YEAR_ERR	8
ERR	Month is out of range	No calculation performed	XL_CFI_TIME_ASCII_TRANS_MONTH_ERR	9
ERR	Day is out of range	No calculation performed	XL_CFI_TIME_ASCII_TRANS_DAY_ERR	10
ERR	Hour is out of range	No calculation performed	XL_CFI_TIME_ASCII_TRANS_HOUR_ERR	11
ERR	Minutes are out of range	No calculation performed	XL_CFI_TIME_ASCII_TRANS_MIN_ERR	12
ERR	Seconds are out of range	No calculation performed	XL_CFI_TIME_ASCII_TRANS_SEC_ERR	13
ERR	Microseconds are out of range	No calculation performed	XL_CFI_TIME_ASCII_TRANS_MICROSEC_ERR	14
ERR	Internal Error: Input Gregorian date to MJD transformation failed	No calculation performed	XL_CFI_TIME_ASCII_TRANS_MJD_ERR	15
ERR	Time reference not initialised	No calculation performed	XL_CFI_TIME_ASCII_TRANS_REF_INIT_ERR	16
WARN	Time out of initialization range	Calculation performed. A message informs the user.	XL_CFI_TIME_ASCII_TRANS_REF_INIT_WARN	17

7.9.6 Runtime Performances

The following runtime performances have been measured.

Table 42: Runtime performances of *xl_time_ascii_to_transport*

Ultra Sparc II-400 [ms]
0.0200

7.10 xl_time_processing_to_ascii

7.10.1 Overview

The `xl_time_processing_to_ascii` CFI function transforms a time expressed in Processing format and a given reference (TAI, UTC, UT1 or GPS) into a time in an ASCII format, performing a reference transformation if necessary (to TAI, UTC, UT1 or GPS).

User should be aware that the use of UTC in Processing format is not encouraged, due to the discontinuity that is caused by the introduction of leap seconds. See [IERS] for further details.

7.10.2 Calling Interface

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

```
#include <explorer_lib.h>
{
    long proc_id_in, ascii_id_out;
    long time_ref_in, time_ref_out;
    double processing_in;
    char ascii_out[XL_TIME_ASCII_DIM_MAX];
    xl_time_id time_id;
    long ierr[XL_NUM_ERR_PROC_ASCII], status;

    status = xl_time_processing_to_ascii(&time_id, &proc_id_in,
                                       &time_ref_in, &processing_in, &ascii_id_out,
                                       &time_ref_out, ascii_out, ierr);

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

    status = xl_time_processing_to_ascii_run(&run_id, &proc_id_in,
                                           &time_ref_in, &processing_in, &ascii_id_out,
                                           &time_ref_out, ascii_out, ierr);
}
```

The `XL_TIME_ASCII_DIM_MAX` and `XL_NUM_ERR_PROC_ASCII` constants are defined in the file `explorer_lib.h`.

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_lib.inc>
    INTEGER*4 SAT_ID, PROC_ID_IN, ASCII_ID_OUT
    INTEGER*4 TIME_REF_IN, TIME_REF_OUT
```

```

REAL*8 PROCESSING_IN
CHARACTER ASCII_OUT(XL_TIME_ASCII_DIM_MAX)
INTEGER*4 IERR(XL_NUM_ERR_PROC_ASCII), STATUS
STATUS = XL_TIME_PROCESSING_TO_ASCII(SAT_ID, PROC_ID_IN,
&                                     TIME_REF_IN, PROCESSING_IN, ASCII_ID_OUT,
&                                     TIME_REF_OUT, ASCII_OUT, IERR)
  
```

7.10.3 Input Parameters

The `xl_time_processing_to_ascii` CFI function has the following input parameters:

Table 43: Input parameters of `xl_time_processing_to_ascii` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
proc_id_in	long *	-	Processing format ID	-	Complete
time_ref_in	long *	-	Time reference ID	-	Any except XL_TIME_UNDEF
processing_in	double*	-	Time in Processing Format	Decimal days, MJD2000 (Processing)	[-18262.0,36524.0]
ascii_id_out	long *	-	ASCII format ID	-	Complete
time_ref_out	long *	-	Time reference ID	-	Any except XL_TIME_UNDEF

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

- Processing format ID: `proc_id_in`. Current document, section 6.2.
- Time reference ID: `time_ref_in` and `time_ref_out`. See [GEN_SUM].
- ASCII format ID: `ascii_id_out`. Current document, section 6.2.

Note that for the function to work correctly, the time references should be properly initialised before calling the function (see section 4.2 for details), unless `time_ref_in = time_ref_out`.

7.10.4 Output Parameters

The output parameters of the `xl_time_processing_to_ascii` CFI function are:

Table 44: Output parameters of `xl_time_processing_to_ascii`

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xl_time_processing_to_ascii</code>	long	-	Status flag	-	-
<code>ascii_out</code>	char	See table 4 and table 5	Time in ASCII format	See table 4 and table 5	See table 4 and table 5
<code>ierr</code>	long	-	Error vector	-	-

7.10.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the `xl_time_processing_to_ascii` CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 by translating the error vector returned by the `xl_time_processing_to_ascii` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 45: Error messages of `xl_time_processing_to_ascii` function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	<code>XL_CFI_TIME_PROC_ASCII_SAT_ERR</code>	0
ERR	Input processing format ID is not correct	No calculation performed	<code>XL_CFI_TIME_PROC_ASCII_PROC_IN_ERR</code>	1
ERR	Input time reference ID is not correct	No calculation performed	<code>XL_CFI_TIME_PROC_ASCII_TIME_IN_ERR</code>	2
ERR	Input days out of range	No calculation performed	<code>XL_CFI_TIME_PROC_ASCII_DAY_ERR</code>	3
ERR	Output ascii format ID is not correct	No calculation performed	<code>XL_CFI_TIME_PROC_ASCII_ASCII_OUT_ERR</code>	4
ERR	Output time reference ID is not correct	No calculation performed	<code>XL_CFI_TIME_PROC_ASCII_TIME_OUT_ERR</code>	5
ERR	Satellite ID and output format ID are not compatible	No calculation performed	<code>XL_CFI_TIME_PROC_ASCII_COMP_OUT_ERR</code>	6

Table 45: Error messages of xl_time_processing_to_ascii function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Internal error: Output ascii MJD is out of range	No calculation performed	XL_CFI_TIME_PROC_ASCII_ASCII_MJD_ERR	7
ERR	Internal error: Output ascii year is out of range	No calculation performed	XL_CFI_TIME_PROC_ASCII_ASCII_YEAR_ERR	8
ERR	Internal error: Output ascii month is out of range	No calculation performed	XL_CFI_TIME_PROC_ASCII_ASCII_MONTH_ERR	9
ERR	Internal error: Output ascii day is out of range	No calculation performed	XL_CFI_TIME_PROC_ASCII_ASCII_DAY_ERR	10
ERR	Internal error: Output ascii hour is out of range	No calculation performed	XL_CFI_TIME_PROC_ASCII_ASCII_HOUR_ERR	11
ERR	Internal error: Output ascii minutes are out of range	No calculation performed	XL_CFI_TIME_PROC_ASCII_ASCII_MIN_ERR	12
ERR	Internal error: Output ascii seconds are out of range	No calculation performed	XL_CFI_TIME_PROC_ASCII_ASCII_SEC_ERR	13
ERR	Internal error: Output ascii microseconds are out of range	No calculation performed	XL_CFI_TIME_PROC_ASCII_ASCII_MICROSEC_ERR	14
ERR	Internal error: Output ascii format is not correct	No calculation performed	XL_CFI_TIME_PROC_ASCII_FORMAT_OUT_ERR	15
ERR	Time reference not initialised	No calculation performed	XL_CFI_TIME_PROC_ASCII_REF_INIT_ERR	16
WARN	Time out of initialization range	Calculation performed. A message informs the user.	XL_CFI_TIME_PROC_ASCII_REF_INIT_WARN	17

7.10.6 Runtime Performances

The following runtime performances have been measured.

Table 46: Runtime performances of xl_time_processing_to_ascii

Ultra Sparc II-400 [ms]
0.0270

7.11 xl_time_processing_to_processing

7.11.1 Overview

The **xl_time_processing_to_processing** CFI function transforms a time expressed in Processing format and a given reference (TAI, UTC, UT1 or GPS) into a time in Processing format with a different reference (TAI, UTC, UT1 or GPS).

User should be aware that the use of UTC in Processing format is not encouraged, due to the discontinuity that is caused by the introduction of leap seconds. See [IERS] for further details.

7.11.2 Calling Interface

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

```
#include <explorer_lib.h>
{
    long proc_id_in, proc_id_out;
    long time_ref_in, time_ref_out;
    double processing_in, processing_out;
    xl_time_id time_id;
    long ierr[XL_NUM_ERR_PROC_PROC], status;

    status = xl_time_processing_to_processing(&time_id, &proc_id_in,
                                             &time_ref_in, &processing_in, &proc_id_out,
                                             &time_ref_out, &processing_out, ierr);

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

    status = xl_time_processing_to_processing_run(&run_id, &proc_id_in,
                                                &time_ref_in, &processing_in, &proc_id_out,
                                                &time_ref_out, &processing_out, ierr);
}
```

The XL_NUM_ERR_PROC_PROC constant is defined in the file *explorer_lib.h*.

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_lib.inc>
    INTEGER*4 SAT_ID, PROC_ID_IN, PROC_ID_OUT
    INTEGER*4 TIME_REF_IN, TIME_REF_OUT
```

```
REAL*8 PROCESSING_IN, PROCESSING_OUT
INTEGER*4 IERR (XL_NUM_ERR_PROC_PROC), STATUS
```

```
STATUS = XL_TIME_PROCESSING_TO_PROCESSING (SAT_ID, PROC_ID_IN,
& TIME_REF_IN, PROCESSING_IN, PROC_ID_OUT,
& TIME_REF_OUT, PROCESSING_OUT, IERR)
```

7.11.3 Input Parameters

The `xl_time_processing_to_processing` CFI function has the following input parameters:

Table 47: Input parameters of `xl_time_processing_to_processing` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
proc_id_in	long *	-	Processing format ID	-	Complete
time_ref_in	long *	-	Time reference ID	-	Any except XL_TIME_UNDEF
processing_in	double*	-	Time in Processing Format	Decimal days, MJD2000 (Processing)	[-18262.0,36524.0]
proc_id_out	long *	-	Processing format ID	-	Complete
time_ref_out	long *	-	Time reference ID	-	Any except XL_TIME_UNDEF

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

- Processing format ID: `proc_id_in` and `proc_id_out`. Current document, section 6.2.
- Time reference ID: `time_ref_in` and `time_ref_out`. See [GEN_SUM].

Note that for the function to work correctly, the time references should be properly initialised before calling the function (see section 4.2 for details), unless `time_ref_in = time_ref_out`.

7.11.4 Output Parameters

The output parameters of the `xl_time_processing_to_processing` CFI function are:

Table 48: Output parameters of `xl_time_processing_to_processing`

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xl_time_processing_to_processing</code>	long	-	Status flag	-	-
<code>processing_out</code>	double*	-	Time in Processing Format	Decimal days, MJD2000 (Processing)	[-18262.0,36524.0]
<code>ierr</code>	long	-	Error vector	-	-

7.11.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the `xl_time_processing_to_processing` CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 by translating the error vector returned by the `xl_time_processing_to_processing` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 49: Error messages of `xl_time_processing_to_processing` function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_TIME_PROC_PRO C_SAT_ERR	0
ERR	Input processing format ID is not correct	No calculation performed	XL_CFI_TIME_PROC_PRO C_PROC_IN_ERR	1
ERR	Input time reference ID is not correct	No calculation performed	XL_CFI_TIME_PROC_PRO C_TIME_IN_ERR	2
ERR	Output processing format ID is not correct	No calculation performed	XL_CFI_TIME_PROC_PRO C_PROC_OUT_ERR	3
ERR	Output time reference ID is not correct	No calculation performed	XL_CFI_TIME_PROC_PRO C_TIME_OUT_ERR	4
ERR	Number of days out of range	No calculation performed	XL_CFI_TIME_PROC_PRO C_DAY_ERR	5
ERR	Time reference not initialised	No calculation performed	XL_CFI_TIME_PROC_PRO C_REF_INIT_ERR	6

Table 49: Error messages of xl_time_processing_to_processing function

Error type	Error message	Cause and impact	Error code	Error No
WARN	Time out of initialization range	Calculation performed. A message informs the user.	XL_CFI_TIME_PROC_PRO C_REF_INIT_WARN	7

7.11.6 Runtime Performances

The following runtime performances have been measured.

Table 50: Runtime performances of xl_time_processing_to_processing

Ultra Sparc II-400 [ms]
0.0040

7.12 xl_time_processing_to_transport

7.12.1 Overview

The **xl_time_processing_to_transport** CFI function transforms a time expressed in Processing format and a given reference (TAI, UTC, UT1 or GPS) into a time in a Transport format, performing a reference transformation if necessary (to TAI, UTC, UT1 or GPS).

User should be aware that the use of UTC in Processing format is not encouraged, due to the discontinuity that is caused by the introduction of leap seconds. See [IERS] for further details.

7.12.2 Calling Interface

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

```
#include <explorer_lib.h>
{
    long proc_id_in, trans_id_out;
    long time_ref_in, time_ref_out;
    double processing_in;
    long transport_out[XL_TIME_TRANS_DIM_MAX];
    xl_time_id time_id;
    long ierr[XL_NUM_ERR_PROC_TRANS], status;

    status = xl_time_processing_to_transport(&time_id, &proc_id_in,
                                           &time_ref_in, &processing_in, &trans_id_out,
                                           &time_ref_out, transport_out, ierr);

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

    status = xl_time_processing_to_transport_run(&run_id, &proc_id_in,
                                               &time_ref_in, &processing_in, &trans_id_out,
                                               &time_ref_out, transport_out, ierr);
}
```

The `XL_TIME_TRANS_DIM_MAX` and `XL_NUM_ERR_PROC_TRANS` constants are defined in the file *explorer_lib.h*.

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_lib.inc>
    INTEGER*4 SAT_ID, PROC_ID_IN, TRANS_ID_OUT
    INTEGER*4 TIME_REF_IN, TIME_REF_OUT
```

```

REAL*8 PROCESSING_IN
INTEGER*4 TRANSPORT_OUT(XL_TIME_TRANS_DIM_MAX)
INTEGER*4 IERR(XL_NUM_ERR_PROC_TRANS), STATUS
STATUS = XL_TIME_PROCESSING_TO_TRANSPORT(SAT_ID, PROC_ID_IN,
&                                     TIME_REF_IN, PROCESSING_IN, TRANS_ID_OUT,
&                                     TIME_REF_OUT, TRANSPORT_OUT, IERR)
    
```

7.12.3 Input Parameters

The `xl_time_processing_to_transport` CFI function has the following input parameters:

Table 51: Input parameters of `xl_time_processing_to_transport` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
proc_id_in	long *	-	Processing format ID	-	Complete
time_ref_in	long *	-	Time reference ID	-	Any except XL_TIME_UNDEF
processing_in	double*	-	Time in Processing Format	Decimal days, MJD2000 (Processing)	[-18262.0,36524.0]
trans_id_out	long *	-	Transport format ID	-	Complete
time_ref_out	long *	-	Time reference ID	-	Any except XL_TIME_UNDEF

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

- Processing format ID: `proc_id_in`. Current document, section 6.2.
- Time reference ID: `time_ref_in` and `time_ref_out`. See [GEN_SUM].
- Transport format ID: `trans_id_out`. Current document, section 6.2.

Note that for the function to work correctly, the time references should be properly initialised before calling the function (see section 4.2 for details), unless `time_ref_in = time_ref_out`.

7.12.4 Output Parameters

The output parameters of the `xl_time_processing_to_transport` CFI function are:

Table 52: Output parameters of `xl_time_processing_to_transport`

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xl_time_processing_to_transport</code>	long	-	Status flag	-	-
<code>transport_out[dim]</code>	long	See table 3	Time in Transport format	See table 3	See table 3
<code>ierr</code>	long	-	Error vector	-	-

7.12.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the `xl_time_processing_to_transport` CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 error vector returned by the `xl_time_processing_to_transport` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 53: Error messages of `xl_time_processing_to_transport` function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_TIME_PROC_TRANS_SAT_ERR	0
ERR	Input processing format ID is not correct	No calculation performed	XL_CFI_TIME_PROC_TRANS_PROC_IN_ERR	1
ERR	Input time reference ID is not correct	No calculation performed	XL_CFI_TIME_PROC_TRANS_TIME_IN_ERR	2
ERR	Output transport format ID is not correct	No calculation performed	XL_CFI_TIME_PROC_TRANS_TRANS_OUT_ERR	3
ERR	Output time reference ID is not correct	No calculation performed	XL_CFI_TIME_PROC_TRANS_TIME_OUT_ERR	4
ERR	Satellite ID and output format ID are not compatible	No calculation performed	XL_CFI_TIME_PROC_TRANS_COMP_OUT_ERR	5
ERR	Number of days out of range	No calculation performed	XL_CFI_TIME_PROC_TRANS_DAY_ERR	6
ERR	Time reference not initialised	No calculation performed	XL_CFI_TIME_PROC_TRANS_REF_INIT_ERR	7

Table 53: Error messages of xl_time_processing_to_transport function

Error type	Error message	Cause and impact	Error code	Error No
WARN	Time out of initialization range	Calculation performed. A message informs the user.	XL_CFI_TIME_PROC_TRAN NS_REF_INIT_WARN	8

7.12.6 Runtime Performances

The following runtime performances have been measured.

Table 54: Runtime performances of xl_time_processing_to_transport

Ultra Sparc II-400 [ms]
0.0040

7.13 xl_time_transport_to_ascii

7.13.1 Overview

The **xl_time_transport_to_ascii** CFI function transforms a time expressed in a given Transport format and reference (TAI, UTC, UT1 or GPS) into a time in an ASCII format, performing a reference transformation if necessary (to TAI, UTC, UT1 or GPS).

7.13.2 Calling Interface

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

```
#include <explorer_lib.h>
{
    long trans_id_in, ascii_id_out;
    long time_ref_in, time_ref_out;
    long transport_in[XL_TIME_TRANS_DIM_MAX];
    char ascii_out[XL_TIME_ASCII_DIM_MAX];
    xl_time_id time_id;
    long ierr[XL_NUM_ERR_TRANS_ASCII], status;

    status = xl_time_transport_to_ascii(&time_id, &trans_id_in,
                                       &time_ref_in, transport_in, &ascii_id_out,
                                       &time_ref_out, ascii_out, ierr);

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

    status = xl_time_transport_to_ascii_run(&run_id, &trans_id_in,
                                           &time_ref_in, transport_in, &ascii_id_out,
                                           &time_ref_out, ascii_out, ierr);
}
```

The `XL_TIME_TRANS_DIM_MAX`, `XL_TIME_ASCII_DIM_MAX`, `XL_NUM_ERR_TRANS_ASCII` constants are defined in the file *explorer_lib.h*.

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_lib.inc>
    INTEGER*4 SAT_ID, TRANS_ID_IN, ASCII_ID_OUT
    INTEGER*4 TIME_REF_IN, TIME_REF_OUT
```

```
INTEGER*4 TRANSPORT_IN(XL_TIME_TRANS_DIM_MAX)
CHARACTER ASCII_OUT(XL_TIME_ASCII_DIM_MAX)
INTEGER*4 IERR(XL_NUM_ERR_TRANS_ASCII), STATUS
```

```
STATUS = XL_TIME_TRANSPORT_TO_ASCII(SAT_ID, TRANS_ID_IN,
& TIME_REF_IN, TRANSPORT_IN, ASCII_ID_OUT,
& TIME_REF_OUT, ASCII_OUT, IERR)
```

7.13.3 Input Parameters

The `xl_time_transport_to_ascii` CFI function has the following input parameters:

Table 55: Input parameters of `xl_time_transport_to_ascii` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>time_id</code>	<code>xl_time_id*</code>	-	Structure that contains the time correlations.	-	-
<code>trans_id_in</code>	<code>long *</code>	-	Transport format ID	-	Complete
<code>time_ref_in</code>	<code>long *</code>	-	Time reference ID	-	Any except <code>XL_TIME_UNDEF</code>
<code>transport_in[dim]</code>	<code>long</code>	See table 3	Time in Transport format	See table 3	See table 3
<code>ascii_id_out</code>	<code>long *</code>	-	ASCII format ID	-	Complete
<code>time_ref_out</code>	<code>long *</code>	-	Time reference ID	-	Any except <code>XL_TIME_UNDEF</code>

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

- Transport format ID: `trans_id_in`. Current document, section 6.2.
- Time reference ID: `time_ref_in` and `time_ref_out`. See [GEN_SUM].
- ASCII format ID: `ascii_id_out`. Current document, section 6.2.

It is important to point out the usage of the `time_ref_out` parameter within the current function:

- If the time reference flag for the output is undefined, an error shall be returned.

Note that for the function to work correctly, the time references should be properly initialised before calling the function (see section 4.2 for details), unless `time_ref_in = time_ref_out`.

7.13.4 Output Parameters

The output parameters of the `xl_time_transport_to_ascii` CFI function are:

Table 56: Output parameters of `xl_time_transport_to_ascii`

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xl_time_transport_to_ascii</code>	long	-	Status flag	-	-
<code>ascii_out</code>	char	See table 4 and table 5	Time in ASCII format	See table 4 and table 5	See table 4 and table 5
<code>ierr</code>	long	-	Error vector	-	-

7.13.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the `xl_time_transport_to_ascii` CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 by translating the error vector returned by the `xl_time_transport_to_ascii` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 57: Error messages of `xl_time_transport_to_ascii` function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_TIME_TRANS_AS CII_SAT_ERR	0
ERR	Input transport format ID is not correct	No calculation performed	XL_CFI_TIME_TRANS_AS CII_TRANS_IN_ERR	1
ERR	Input time reference ID is not correct	No calculation performed	XL_CFI_TIME_TRANS_AS CII_TIME_IN_ERR	2
ERR	Satellite ID and input format ID are not compatible	No calculation performed	XL_CFI_TIME_TRANS_AS CII_COMP_IN_ERR	3
ERR	Number of days out of range	No calculation performed	XL_CFI_TIME_TRANS_AS CII_DAY_ERR	4
ERR	Number of seconds out of range	No calculation performed	XL_CFI_TIME_TRANS_AS CII_SEC_ERR	5
ERR	Number of milliseconds out of range	No calculation performed	XL_CFI_TIME_TRANS_AS CII_MILLISEC_ERR	6

Table 57: Error messages of xl_time_transport_to_ascii function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Number of microseconds out of range	No calculation performed	XL_CFI_TIME_TRANS_ASCII_MICROSEC_ERR	7
ERR	Number of SIRAL extra counter ticks out of range	No calculation performed	XL_CFI_TIME_TRANS_ASCII_TICK_ERR	8
ERR	Output ascii format ID is not correct	No calculation performed	XL_CFI_TIME_TRANS_ASCII_ASCII_OUT_ERR	9
ERR	Output time reference ID is not correct	No calculation performed	XL_CFI_TIME_TRANS_ASCII_TIME_OUT_ERR	10
ERR	Satellite ID and output format ID are not compatible	No calculation performed	XL_CFI_TIME_TRANS_ASCII_COMP_OUT_ERR	11
ERR	Internal error: Output ascii MJD is out of range	No calculation performed	XL_CFI_TIME_TRANS_ASCII_ASCII_MJD_ERR	12
ERR	Internal error: Output ascii year is out of range	No calculation performed	XL_CFI_TIME_TRANS_ASCII_ASCII_YEAR_ERR	13
ERR	Internal error: Output ascii month is out of range	No calculation performed	XL_CFI_TIME_TRANS_ASCII_ASCII_MONTH_ERR	14
ERR	Internal error: Output ascii day is out of range	No calculation performed	XL_CFI_TIME_TRANS_ASCII_ASCII_DAY_ERR	15
ERR	Internal error: Output ascii hour is out of range	No calculation performed	XL_CFI_TIME_TRANS_ASCII_ASCII_HOUR_ERR	16
ERR	Internal error: Output ascii minutes are out of range	No calculation performed	XL_CFI_TIME_TRANS_ASCII_ASCII_MIN_ERR	17
ERR	Internal error: Output ascii seconds are out of range	No calculation performed	XL_CFI_TIME_TRANS_ASCII_ASCII_SEC_ERR	18
ERR	Internal error: Output ascii microseconds are out of range	No calculation performed	XL_CFI_TIME_TRANS_ASCII_ASCII_MICROSEC_ERR	19
ERR	Internal error: Output ascii format is not correct	No calculation performed	XL_CFI_TIME_TRANS_ASCII_FORMAT_OUT_ERR	20
ERR	Time reference not initialised	No calculation performed	XL_CFI_TIME_TRANS_ASCII_REF_INIT_ERR	21
WARN	Time out of initialization range	Calculation performed. A message informs the user.	XL_CFI_TIME_TRANS_ASCII_REF_INIT_WARN	22

7.13.6 Runtime Performances

The following runtime performances have been measured.

Table 58: Runtime performances of xl_time_transport_to_ascii

Ultra Sparc II-400 [ms]
0.0270

7.14 xl_time_transport_to_processing

7.14.1 Overview

The **xl_time_transport_to_processing** CFI function transforms a time expressed in a given Transport format and reference (TAI, UTC, UT1 or GPS) into a time in Processing format, performing a reference transformation if necessary (to TAI, UTC, UT1 or GPS).

User should be aware that the use of UTC in Processing format is not encouraged, due to the discontinuity that is caused by the introduction of leap seconds. See [IERS] for further details.

7.14.2 Calling Interface

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

```
#include <explorer_lib.h>
{
    long trans_id_in, proc_id_out;
    long time_ref_in, time_ref_out;
    long transport_in[XL_TIME_TRANS_DIM_MAX];
    double processing_out;
    xl_time_id time_id;
    long ierr[XL_NUM_ERR_TRANS_PROC], status;

    status = xl_time_transport_to_processing(&time_id, &trans_id_in,
                                           &time_ref_in, transport_in, &proc_id_out,
                                           &time_ref_out, &processing_out, ierr);

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

    status = xl_time_transport_to_processing_run(&run_id, &trans_id_in,
                                               &time_ref_in, transport_in, &proc_id_out,
                                               &time_ref_out, &processing_out, ierr);
}
```

The `XL_TIME_TRANS_DIM_MAX` and `XL_NUM_ERR_TRANS_PROC` constants are defined in the file `explorer_lib.h`.

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_lib.inc>
    INTEGER*4 SAT_ID, TRANS_ID_IN, PROC_ID_OUT
    INTEGER*4 TIME_REF_IN, TIME_REF_OUT
```

```

INTEGER*4 TRANSPORT_IN(XL_TIME_TRANS_DIM_MAX)
REAL*8 PROCESSING_OUT
INTEGER*4 IERR(XL_NUM_ERR_TRANS_PROC), STATUS
STATUS = XL_TIME_TRANSPORT_TO_PROCESSING(SAT_ID, TRANS_ID_IN,
& TIME_REF_IN, TRANSPORT_IN, PROC_ID_OUT,
& TIME_REF_OUT, PROCESSING_OUT, IERR)
    
```

7.14.3 Input Parameters

The `xl_time_transport_to_processing` CFI function has the following input parameters:

Table 59: Input parameters of `xl_time_transport_to_processing` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>time_id</code>	<code>xl_time_id*</code>	-	Structure that contains the time correlations.	-	-
<code>trans_id_in</code>	<code>long *</code>	-	Transport format ID	-	Complete
<code>time_ref_in</code>	<code>long *</code>	-	Time reference ID	-	Any except <code>XL_TIME_UNDEF</code>
<code>transport_in[dim]</code>	<code>long</code>	See table 3	Time in Transport format	See table 3	See table 3
<code>proc_id_out</code>	<code>long *</code>	-	Processing format ID	-	Complete
<code>time_ref_out</code>	<code>long *</code>	-	Time reference ID	-	Any except <code>XL_TIME_UNDEF</code>

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

- Transport format ID: `trans_id_in`. Current document, section 6.2.
- Time reference ID: `time_ref_in` and `time_ref_out`. See [GEN_SUM].
- Processing format ID: `proc_id_out`. Current document, section 6.2

Note that for the function to work correctly, the time references should be properly initialised before calling the function (see section 4.2 for details), unless `time_ref_in = time_ref_out`.

7.14.4 Output Parameters

The output parameters of the `xl_time_transport_to_processing` CFI function are:

Table 60: Output parameters of `xl_time_transport_to_processing`

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xl_time_transport_to_processing</code>	long	-	Status flag	-	-
<code>processing_out</code>	double*	-	Time in Processing Format	Decimal days, MJD2000 (Processing)	[-18262.0,36524.0]
<code>ierr</code>	long	-	Error vector	-	-

7.14.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the `xl_time_transport_to_processing` CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 by translating the error vector returned by the `xl_time_transport_to_processing` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 61: Error messages of `xl_time_transport_to_processing` function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_TIME_TRANS_PROC_SAT_ERR	0
ERR	Input transport format ID is not correct	No calculation performed	XL_CFI_TIME_TRANS_PROC_TRANS_IN_ERR	1
ERR	Input time reference ID is not correct	No calculation performed	XL_CFI_TIME_TRANS_PROC_TIME_IN_ERR	2
ERR	Satellite ID and input format ID are not compatible	No calculation performed	XL_CFI_TIME_TRANS_PROC_COMP_IN_ERR	3
ERR	Output processing format ID is not correct	No calculation performed	XL_CFI_TIME_TRANS_PROC_PROC_OUT_ERR	4
ERR	Output time reference ID is not correct	No calculation performed	XL_CFI_TIME_TRANS_PROC_TIME_OUT_ERR	5
ERR	Number of days out of range	No calculation performed	XL_CFI_TIME_TRANS_PROC_DAY_ERR	6

Table 61: Error messages of xl_time_transport_to_processing function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Number of seconds out of range	No calculation performed	XL_CFI_TIME_TRANS_P OC_SEC_ERR	7
ERR	Number of milliseconds out of range	No calculation performed	XL_CFI_TIME_TRANS_P OC_MILLISEC_ERR	8
ERR	Number of microseconds out of range	No calculation performed	XL_CFI_TIME_TRANS_P OC_MICROSEC_ERR	9
ERR	Number of SIRAL extra counter ticks out of range	No calculation performed	XL_CFI_TIME_TRANS_P OC_TICK_ERR	10
ERR	Time reference not initialised	No calculation performed	XL_CFI_TIME_TRANS_P OC_REF_INIT_ERR	11
WARN	Time out of initialization range	Calculation performed. A message informs the user.	XL_CFI_TIME_TRANS_P OC_REF_INIT_WARN	12

7.14.6 Runtime Performances

The following runtime performances have been measured.

Table 62: Runtime performances of xl_time_transport_to_processing

Ultra Sparc II-400 [ms]
0.0040

7.15 xl_time_transport_to_transport

7.15.1 Overview

The **xl_time_transport_to_transport** CFI function transforms a time expressed in a given Transport format and reference (TAI, UTC, UT1 or GPS) into a time in a different Transport format and/or reference (TAI, UTC, UT1 or GPS).

7.15.2 Calling Interface

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

```
#include <explorer_lib.h>
{
    long trans_id_in, trans_id_out;
    long time_ref_in, time_ref_out;
    long transport_in[XL_TIME_TRANS_DIM_MAX];
    long transport_out[XL_TIME_TRANS_DIM_MAX];
    xl_time_id time_id;
    long ierr[XL_NUM_ERR_TRANS_TRANS], status;

    status = xl_time_transport_to_transport(&time_id, &trans_id_in,
                                           &time_ref_in, transport_in, &trans_id_out,
                                           &time_ref_out, transport_out, ierr);

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

    status = xl_time_transport_to_transport_run(&run_id, &trans_id_in,
                                               &time_ref_in, transport_in, &trans_id_out,
                                               &time_ref_out, transport_out, ierr);
}
```

The `XL_TIME_TRANS_DIM_MAX` and `XL_NUM_ERR_TRANS_TRANS` constants are defined in the file *explorer_lib.h*.

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_lib.inc>
    INTEGER*4 SAT_ID, TRANS_ID_IN, TRANS_ID_OUT
    INTEGER*4 TIME_REF_IN, TIME_REF_OUT
```

```
INTEGER*4 TRANSPORT_IN(XL_TIME_TRANS_DIM_MAX)
INTEGER*4 TRANSPORT_OUT(XL_TIME_TRANS_DIM_MAX)
INTEGER*4 IERR(XL_NUM_ERR_TRANS_TRANS), STATUS
```

```
STATUS = XL_TIME_TRANSPORT_TO_TRANSPORT(SAT_ID, TRANS_ID_IN,
& TIME_REF_IN, TRANSPORT_IN, TRANS_ID_OUT,
& TIME_REF_OUT, TRANSPORT_OUT, IERR)
```

7.15.3 Input Parameters

The `xl_time_transport_to_transport` CFI function has the following input parameters:

Table 63: Input parameters of `xl_time_transport_to_transport` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
trans_id_in	long *	-	Transport format ID	-	Complete
time_ref_in	long *	-	Time reference ID	-	Any except XL_TIME_UNDEF
transport_in[dim]	long	See table 3	Time in Transport format	See table 3	See table 3
trans_id_out	long *	-	Transport format ID	-	Complete
time_ref_out	long *	-	Time reference ID	-	Any except XL_TIME_UNDEF

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

- Transport format ID: `trans_id_in` and `trans_id_out`. Current document, section 6.2.
- Time reference ID: `time_ref_in` and `time_ref_out`. See [GEN_SUM].

Note that for the function to work correctly, the time references should be properly initialised before calling the function (see section 4.2 for details), unless `time_ref_in = time_ref_out`.

7.15.4 Output Parameters

The output parameters of the `xl_time_transport_to_transport` CFI function are:

Table 64: Output parameters of `xl_time_transport_to_transport`

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xl_time_transport_to_transport</code>	long	-	Status flag	-	-
<code>transport_out[dim]</code>	long	See table 3	Time in Transport format	See table 3	See table 3
<code>ierr</code>	long	-	Error vector	-	-

7.15.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the `xl_time_transport_to_transport` CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 error vector returned by the `xl_time_transport_to_transport` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 65: Error messages of `xl_time_transport_to_transport` function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_TIME_TRANS_TRANS_SAT_ERR	0
ERR	Input transport format ID is not correct	No calculation performed	XL_CFI_TIME_TRANS_TRANS_IN_ERR	1
ERR	Input time reference ID is not correct	No calculation performed	XL_CFI_TIME_TRANS_TRANS_TIME_IN_ERR	2
ERR	Satellite ID and input format ID are not compatible	No calculation performed	XL_CFI_TIME_TRANS_TRANS_COMP_IN_ERR	3
ERR	Output transport format ID is not correct	No calculation performed	XL_CFI_TIME_TRANS_TRANS_OUT_ERR	4
ERR	Output time reference ID is not correct	No calculation performed	XL_CFI_TIME_TRANS_TRANS_TIME_OUT_ERR	5
ERR	Satellite ID and output format ID are not compatible	No calculation performed	XL_CFI_TIME_TRANS_TRANS_COMP_OUT_ERR	6
ERR	Number of days out of range	No calculation performed	XL_CFI_TIME_TRANS_TRANS_DAY_ERR	7

Table 65: Error messages of *xl_time_transport_to_transport* function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Number of seconds out of range	No calculation performed	XL_CFI_TIME_TRANS_TR ANS_SEC_ERR	8
ERR	Number of milliseconds out of range	No calculation performed	XL_CFI_TIME_TRANS_TR ANS_MILLISEC_ERR	9
ERR	Number of microseconds out of range	No calculation performed	XL_CFI_TIME_TRANS_TR ANS_MICROSEC_ERR	10
ERR	Number of SIRAL extra counter ticks out of range	No calculation performed	XL_CFI_TIME_TRANS_TR ANS_TICK_ERR	11
ERR	Time reference not initialised	No calculation performed	XL_CFI_TIME_TRANS_TR ANS_REF_INIT_ERR	12
WARN	Time out of initialization range	Calculation performed. A message informs the user.	XL_CFI_TIME_TRANS_TR ANS_REF_INIT_WARN	13

7.15.6 Runtime Performances

The following runtime performances have been measured.

Table 66: Runtime performances of *xl_time_transport_to_transport*

Ultra Sparc II-400 [ms]
0.0050

7.16 xl_time_add

7.16.1 Overview

The **xl_time_add** CFI function adds a time duration to a TAI, UTC, UT1 or GPS times expressed in Processing format.

User should be aware that the use of UTC in Processing format is not encouraged, due to the discontinuity that is caused by the introduction of leap seconds. See [IERS] for further details.

7.16.2 Calling interface

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

```
#include <explorer_lib.h>
{
|   long proc_id, time_ref;
|   double processing_in, added_duration, processing_out;
|   long ierr[XL_NUM_ERR_TIME_ADD], status;
|
|   status = xl_time_add(&proc_id, &time_ref,
|                       &processing_in, &added_duration,
|                       &processing_out, ierr);
|
| }
}
```

The XL_NUM_ERR_TIME_ADD constant is defined in the file *explorer_lib.h*.

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_lib.inc>
|
|   INTEGER*4 PROC_ID, TIME_REF
|   REAL*8 PROCESSING_IN, ADDED_DURATION, PROCESSING_OUT
|   INTEGER*4 IERR (XL_NUM_ERR_TIME_ADD), STATUS
|
|   STATUS = XL_TIME_ADD (PROC_ID, TIME_REF, PROCESSING_IN,
|   &
|                       &ADDED_DURATION, &PROCESSING_OUT, IERR)
```

7.16.3 Input parameters

The `xl_time_add` CFI function has the following input parameters:

Table 67: Input parameters of `xl_time_add` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
proc_id	long *	-	Processing format ID	-	Complete
time_ref	long *	-	Time reference ID	-	Any except XL_TIME_UNDEF
processing_in	double*	-	Time in Processing Format	Decimal days, MJD2000 (Processing format)	[-18262.0,36524.0]
added_duration	double*	-	Duration to be added	Decimal days (Processing format)	-

It is important to point out that the duration is not a time, but a time interval expressed in decimal days to be added to the original time.

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

- Processing format ID: proc_id. Current document, section 6.2.
- Time reference ID: time_ref. See [GEN_SUM].

7.16.4 Output parameters

The output parameters of the `xl_time_add` CFI function are:

Table 68: Output parameters of `xl_time_add` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xl_time_add	long	-	Status flag	-	-
processing_output	double*	-	Time in Processing Format	Decimal days (Processing format)	[-18262.0,36524.0]
ierr	long	-	Error vector	-	-

7.16.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_time_add` CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 by translating the error vector returned by the `xl_time_add` function by calling the function of the EXPLORER_LIB software

library `xl_get_code` (see [GEN_SUM]).

Table 69: Error messages of `xl_time_add` function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_TIME_ADD_SAT_ERR	0
ERR	Processing format ID is not correct	No calculation performed	XL_CFI_TIME_ADD_PROC_ERR	1
ERR	Time reference ID is not correct	No calculation performed	XL_CFI_TIME_ADD_TIME_ERR	2
ERR	Input processing time is out of range	No calculation performed	XL_CFI_TIME_ADD_DAY_IN_ERR	3
ERR	Output processing time is out of range	No calculation performed	XL_CFI_TIME_ADD_DAY_OUT_ERR	4

7.16.6 Runtime performances

The following runtime performances have been measured.

Table 70: Runtime performances of `xl_time_add` function

Ultra Sparc II-400 [ms]
0.0010

7.17 xl_time_diff

7.17.1 Overview

The **xl_time_diff** CFI function calculates the time difference between two TAI, UTC, UT1 or GPS times expressed in Processing format.

User should be aware that the use of UTC in Processing format is not encouraged, due to the discontinuity that is caused by the introduction of leap seconds. See [IERS] for further details.

7.17.2 Calling interface

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

```
#include <explorer_lib.h>
{
|   long proc_id, time_ref;
|   double processing_in_1, processing_in_2, processing_out;
|   long ierr[XL_NUM_ERR_TIME_DIFF], status;
|
|   status = xl_time_diff(&proc_id, &time_ref,
|                       &processing_in_1, &processing_in_2,
|                       &processing_out, ierr);
|
| }

```

Note that `processing_out` is a duration, not a time itself, so it should not be converted to another reference or format.

The `XL_NUM_ERR_TIME_DIFF` constant is defined in the file *explorer_lib.h*.

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_lib.inc>
|
|   INTEGER*4 PROC_ID, TIME_REF
|   REAL*8 PROCESSING_IN_1, PROCESSING_IN_2, PROCESSING_OUT
|   INTEGER*4 IERR(XL_NUM_ERR_TIME_DIFF), STATUS
|
|   STATUS = XL_TIME_DIFF (PROC_ID, TIME_REF,
| &                        PROCESSING_IN_1, PROCESSING_IN_2,
| &                        PROCESSING_OUT, IERR)
|

```

7.17.3 Input parameters

The `xl_time_diff` CFI function has the following input parameters:

Table 71: Input parameters of `xl_time_diff` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>proc_id</code>	long *	-	Processing format ID	-	Complete
<code>time_ref</code>	long *	-	Time reference ID	-	Any except <code>XL_TIME_UNDEF</code>
<code>processing_in_1</code>	double*	-	Time in Processing Format	Decimal days, MJD2000 (Processing format)	[-18262.0,36524.0]
<code>processing_in_2</code>	double*	-	Time in Processing Format	Decimal days, MJD2000 (Processing format)	[-18262.0,36524.0]

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

- Processing format ID: `proc_id`. Current document, section 6.2.
- Time reference ID: `time_ref`. See [GEN_SUM].

7.17.4 Output parameters

The output parameters of the `xl_time_diff` CFI function are:

Table 72: Output parameters of `xl_time_diff` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xl_time_diff</code>	long	-	Status flag	-	-
<code>processing_out</code>	double*	-	Time difference between <code>processing_in_1</code> and <code>processing_in_2</code> expressed in decimal days	Decimal days (Processing format)	-
<code>ierr</code>	long	-	Error vector	-	-

7.17.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_time_diff` CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 by translating the error vector returned by the **xl_time_diff** function by calling the function of the EXPLORER_LIB software library **xl_get_code** (see [GEN_SUM]).

Table 73: Error messages of xl_time_diff function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_TIME_DIFF_SAT_ERR	0
ERR	Processing format ID is not correct	No calculation performed	XL_CFI_TIME_DIFF_PROC_ERR	1
ERR	Time reference ID is not correct	No calculation performed	XL_CFI_TIME_DIFF_TIME_ERR	2
ERR	Input processing time #1 is out of range	No calculation performed	XL_CFI_TIME_DIFF_DAY_I_N_1_ERR	3
ERR	Input processing time #2 is out of range	No calculation performed	XL_CFI_TIME_DIFF_DAY_I_N_2_ERR	4

7.17.6 Runtime performances

The following runtime performances have been measured.

Table 74: Runtime performances of xl_time_diff function

Ultra Sparc II-400 [ms]
0.0010

7.18 xl_time_obt_to_time

7.18.1 Overview

The **xl_time_obt_to_time** CFI function transforms from On-board Time (OBT) count to UTC Processing time.

User should be aware that the use of UTC in Processing format is not encouraged, due to the discontinuity that is caused by the introduction of leap seconds. See [IERS] for further details.

See [MCD] for details on time formats and representations, in particular the definition of OBT.

Note that in the Envisat OBT case there is an ambiguity on the UTC to be computed, because a given OBT count corresponds to many possible times. This is due to the wrap-around of the OBT counter, which occurs about every 190 days.

To solve the ambiguity, the chosen time (given as output) is the time nearest to the reference (given as input) and corresponding to the specified OBT (also given as input).

The **xl_time_obt_to_time** CFI function applies to satellites where OBT time is a counter, which needs to be correlated to an actual time reference. Nevertheless, some other satellites, like Cryosat, use an actual time reference on-board. In this case, the on-board time conversions are handled by the **xl_time_processing_to_processing** function.

Due to the different OBT models used by the various spacecraft, specific data structures are used for each of them. To keep a single interface for the function, a void pointer is used to pass the specific structures to the generic function.

The following data structures are defined for ENVISAT:

```
/* Envisat OBT Structure */
typedef struct
{
    long          sat_id;
    double        time0;
    unsigned long obt0[2];
    unsigned long period0;
} xl_envisat_obt_param;

typedef struct
{
    long          sat_id;
    unsigned long obt[2];
} xl_envisat_obt_value;
```

And for GOCE:

```
/* GOCE OBT Structure */
typedef struct
{
    long          sat_id;
    unsigned long utc0_c;
    unsigned int  utc0_f;
    unsigned long obt0_c;
```



```

    unsigned int  obt0_f;
    double        gradient;
    double        offset;
} xl_goce_obt_param;

typedef struct
{
    long          sat_id;
    double        obt;
} xl_goce_obt_value;

```

The `sat_id` parameter within the structure has to be assigned equal to the `sat_id` passed to the function.

7.18.2 Calling interface

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

```

#include <explorer_lib.h>
{
    long sat_id, proc_id;
    xl_envisat_obt_param obt_param; /*example for ENVISAT */
    xl_envisat_obt_value obt_value_in; /*example for ENVISAT */
    double time_out;
    long ierr[XL_NUM_ERR_OBT_TIME], status;

    status = xl_time_obt_to_time (&sat_id,
                                &proc_id,
                                &obt_param,
                                &obt_value_in,
                                &time_out,
                                ierr);

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

    status = xl_time_obt_to_time_run (&run_id,
                                     &proc_id,
                                     &obt_param,
                                     &obt_value_in,
                                     &time_out,
                                     err);
}

```

The XL_NUM_ERR_OBT_TIME constant is defined in the file *explorer_lib.h*.
 This function cannot be called from ForTran (see section 9).

7.18.3 Input parameters

The `xl_time_obt_to_time` CFI function has the following input parameters:

Table 75: Input parameters of `xl_time_obt_to_time` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
proc_id	long *	-	Processing format ID	-	Complete
obt_param	void *	-	Pointer to <code>xl_<satellite>_obt_param</code>	-	-
obt_value_in	void *	-	Pointer to <code>xl_<satellite>_obt_value</code>	-	-

Table 76: Input parameters of `xl_envisat_obt_param` structure

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long	-	Satellite ID	-	XL_SAT_ENVISAT
time0	double	-	Reference time	Decimal days (Processing format)	[-18262.0,36524.0]
obt0	unsigned long[2]	-	Array of counters containing the OBT at the reference time (in the satellite dependant format)	TBD	TBD
period0	unsigned long	-	Actual on-board clock period	TBD	TBD

Table 77: Input parameters of `xl_envisat_obt_value` structure

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long	-	Satellite ID	-	XL_SAT_ENVISAT
obt	unsigned long[2]	-	Array of counters containing the OBT time (in the satellite dependant format)	TBD	TBD

Table 78: Input parameters of xl_goce_obt_param structure

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long	-	Satellite ID	-	XL_SAT_GOCE
utc0_c	unsigned long	-	Coarse UTC0	seconds	>=0
utc0_f	unsigned int	-	Fine UTC0	2 ⁻¹⁶ seconds	>=0
obt0_c	unsigned long	-	Coarse OBT0	seconds	>=0
obt0_f	unsigned int	-	Fine OBT0	2 ⁻¹⁶ seconds	>=0
gradient	double	-	Gradient between the OBT and the UTC	-	-
offset	double	-	Offset between the OBT and the UTC	seconds	-

Table 79: Input parameters of xl_goce_obt_value structure

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long	-	Satellite ID	-	XL_SAT_GOCE
obt	double	-	OBT time	sconds	-

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

- Satellite ID: sat_id. See [GEN_SUM].
- Processing format ID: proc_id. Current document, section 6.2.

7.18.4 Output parameters

The output parameters of the xl_time_obt_to_time CFI function are:

Table 80: Output parameters of xl_time_obt_to_time function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xl_time_obt_to_time	long	-	Status flag	-	-
time_out	double*	-	UTC Time in Processing Format	Decimal days, MJD2000 (Processing format)	[-18262.0,36524.0]
ierr	long	-	Error vector	-	-

7.18.5 Warnings and errors

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

Table 81: Error messages of xl_time_obt_to_time function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_TIME_OBT_TIME_SAT_ERR	0
ERR	Processing format ID is not correct	No calculation performed	XL_CFI_TIME_OBT_TIME_PROC_ERR	1
ERR	Structure inconsistent with Satellite ID	No calculation performed	XL_CFI_TIME_OBT_TIME_INCONSISTENT_STRUCTURE_ERR	2
ERR	Input reference time is out of range	No calculation performed	XL_CFI_TIME_OBT_TIME_DAY_REF_ERR	3
ERR	No OBT defined for this satellite ID	No calculation performed	XL_CFI_TIME_OBT_TIME_OBT_SAT_ERR	4
ERR	OBT at reference time is out of allowed range	No calculation performed	XL_CFI_TIME_OBT_TIME_OBT_ERR	5
ERR	Period of the On-Board clock is null	No calculation performed	XL_CFI_TIME_OBT_TIME_CLOCK_ERR	6
ERR	Output time is out of range	No calculation performed	XL_CFI_TIME_OBT_TIME_DAY_OUT_ERR	7

7.18.6 Runtime performances

The following runtime performances have been measured.

Table 82: Runtime performances of xl_time_obt_to_time function

Ultra Sparc II-400 [ms]
0.0020

7.19 xl_time_time_to_obt

7.19.1 Overview

The **xl_time_time_to_obt** CFI function transforms a UTC Processing time to OBT count.

User should be aware that the use of UTC in Processing format is not encouraged, due to the discontinuity that is caused by the introduction of leap seconds. See [IERS] for further details.

See [MCD] for details on time formats and representations, in particular the definition OBT.

Note that no rounding to any number of significant bits is performed by **xl_time_time_to_obt**. The user application must perform this rounding if necessary. An example of rounding is provided in the example program within the EXPLORER_LIB library.

The **xl_time_time_to_obt** CFI function applies to satellites where OBT time is a counter, which needs to be correlated to an actual time reference. Nevertheless, some other satellites, like Cryosat, use an actual time reference on-board. In this case, the on-board time conversions are handled by the **xl_time_processing_to_processing** function.

Due to the different OBT models used by the various spacecraft, specific data structures are used for each of them. To keep a single interface for the function, a void pointer is used to pass the specific structures to the generic function.

The following data structures are defined for ENVISAT:

```
/* Envisat OBT Structure */
typedef struct
{
    long          sat_id;
    double        time0;
    unsigned long obt0[2];
    unsigned long period0;
} xl_envisat_obt_param;

typedef struct
{
    long          sat_id;
    unsigned long obt[2];
} xl_envisat_obt_value;
```

And for GOCE:

```
/* GOCE OBT Structure */
typedef struct
{
    long          sat_id;
    unsigned long utc0_c;
    unsigned int  utc0_f;
    unsigned long obt0_c;
    unsigned int  obt0_f;
    double        gradient;
    double        offset;
} xl_goce_obt_param;
```

```
typedef struct
{
  long          sat_id;
  double        obt;
} xl_goce_obt_value;
```

The `sat_id` parameter within the structure has to be assigned equal to the `sat_id` passed to the function.

7.19.2 Calling interface

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

```
#include <explorer_lib.h>
{
  long sat_id, proc_id;
  double time_in;
  xl_envisat_obt_param obt_param; /*example for ENVISAT */
  xl_envisat_obt_value obt_value_out; /*example for ENVISAT */
  long ierr[XL_NUM_ERR_TIME_OBT], status;

  status = xl_time_time_to_obt (&sat_id,
                               &proc_id,
                               &obt_param,
                               &time_in,
                               &obt_value_out,
                               ierr);

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

  status = xl_time_time_to_obt_run (&run_id,
                                    &proc_id,
                                    &obt_param,
                                    &time_in,
                                    &obt_value_out,
                                    ierr);
}
```

The `XL_NUM_ERR_TIME_OBT` constant is defined in the file `explorer_lib.h`.

This function cannot be called from ForTran (see section 9).

7.19.3 Input parameters

The `xl_time_time_to_obt` CFI function has the following input parameters:
 wing input parameters:

Table 83: Input parameters of `xl_time_obt_to_time` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
proc_id	long *	-	Processing format ID	-	Complete
obt_param	void *	-	Pointer to <code>xl_<satellite>_obt_param</code>	-	-
time_in	double*	-	UTC Time	Decimal days (Processing format)	[-18262.0,36524.0]

Table 84: Input parameters of `xl_envisat_obt_param` structure

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long	-	Satellite ID	-	XL_SAT_ENVISAT
time0	double	-	Reference time	Decimal days (Processing format)	[-18262.0,36524.0]
obt0	unsigned long[2]	-	Array of counters containing the OBT at the reference time (in the satellite dependant format)	TBD	TBD
period0	unsigned long	-	Actual on-board clock period	TBD	TBD

Table 85: Input parameters of `xl_goce_obt_param` structure

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long	-	Satellite ID	-	XL_SAT_GOCE
utc0_c	unsigned long	-	Coarse UTC0	seconds	>=0
utc0_f	unsigned int	-	Fine UTC0	2 ⁻¹⁶ seconds	>=0
obt0_c	unsigned long	-	Coarse OBT0	seconds	>=0
obt0_f	unsigned int	-	Fine OBT0	2 ⁻¹⁶ seconds	>=0
gradient	double	-	Gradient between the OBT and the UTC	-	-
offset	double	-	Offset between the OBT and the UTC	seconds	-

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].

7.19.4 Output parameters

The output parameters of the `xl_time_time_to_obt` CFI function are:

Table 86: Output parameters of `xl_time_time_to_obt` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xl_time_time_t_o_obt</code>	long	-	Status flag	-	-
<code>obt_value_out</code>	void *	-	Pointer to <code>xl_<satellite>_obt_value</code>	-	-
<code>ierr</code>	long	-	Error vector	-	-

Table 87: Output parameters of `xl_envisat_obt_value` structure

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>sat_id</code>	long	-	Satellite ID	-	XL_SAT_ENVISAT
<code>obt</code>	unsigned long[2]	-	Array of counters containing the OBT time (in the satellite dependant format)	TBD	TBD

Table 88: Output parameters of `xl_goce_obt_value` structure

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>sat_id</code>	long	-	Satellite ID	-	XL_SAT_GOCE
<code>obt</code>	double	-	OBT time	sconds	-

7.19.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_time_time_to_obt` CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 by translating the error vector returned by the `xl_time_time_to_obt` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 89: Error messages of xl_time_time_to_obt function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_TIME_TIME_OBT_SAT_ERR	0
ERR	Processing format ID is not correct	No calculation performed	XL_CFI_TIME_TIME_OBT_PROC_ERR	1
ERR	Structure inconsistent with Satellite ID	No calculation performed	XL_CFI_TIME_TIME_OBT_INCONSISTENT_STRUCT_ERR	2
ERR	Input time is out of range	No calculation performed	XL_CFI_TIME_TIME_OBT_DAY_IN_ERR	3
ERR	Input reference time is out of range	No calculation performed	XL_CFI_TIME_TIME_OBT_DAY_REF_ERR	4
ERR	No OBT defined for this satellite ID	No calculation performed	XL_CFI_TIME_TIME_OBT_OBT_SAT_ERR	5
ERR	OBT at reference time is out of allowed range	No calculation performed	XL_CFI_TIME_TIME_OBT_OBT_ERR	6
ERR	Period of the On-Board clock is null	No calculation performed	XL_CFI_TIME_TIME_OBT_CLOCK_ERR	7

7.19.6 Runtime performances

The following runtime performances have been measured.

Table 90: Runtime performances of xl_time_time_to_obt function

Ultra Sparc II-400 [ms]
0.0022

7.20 xl_change_cart_cs

7.20.1 Overview

The `xl_change_cart_cs` CFI function transforms a cartesian state vector between different reference frames.

7.20.2 Calling interface

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

```
#include <explorer_lib.h>
{
    long mode, cs_in, cs_out, time_ref;
    double time;
    double pos[3], vel[3], acc[3];
    double pos_out[3], vel_out[3], acc_out[3];
    xl_time_id time_id;
    long status;

    status = xl_change_cart_cs (&time_id, &mode, &cs_in, &cs_out,
                               &time_ref, &time, pos, vel, acc,
                               pos_out, vel_out, acc_out);

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

    status = xl_change_cart_cs_run (&run_id, &mode, &cs_in, &cs_out,
                                    &time_ref, &time, pos, vel, acc,
                                    pos_out, vel_out, acc_out);
}
```

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_lib.inc>
    INTEGER*4 SAT_ID, MODE, CS_IN, CS_OUT, TIME_REF
    REAL*8 TIME
    REAL*8 POS(3), VEL(3), ACC(3)
    REAL*8 POS_OUT(3), VEL_OUT(3), ACC_OUT(3)
    INTEGER*4 STATUS
```

```
STATUS = XL_CHANGE_CART_CS (SAT_ID, MODE, CS_IN, CS_OUT, TIME_REF,
& TIME, POS, VEL, ACC, POS_OUT, VEL_OUT,
& ACC_OUT)
```

7.20.3 Input parameters

The `xl_change_cart_cs` CFI function has the following input parameters:

Table 91: Input parameters of `xl_xhange_cart_cs` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
mode	long*	-	Calculation mode selection		Complete
cs_in	long *	-	Initial reference frame ID	-	Complete
cs_out	long *	-	Final reference frame ID	-	Complete
time_ref	long *	-	Time reference ID	-	Any except XL_TIME_UNDEF
time	double*	-	Reference time	Decimal days (Processing format)	[-18262.0,36524.0]
pos[3]	double	all	Input position vector (Initial reference frame)	m	-
vel[3]	double	all	Input velocity vector (Initial reference frame) Dummy if <i>mode</i> is: · XL_CALC_POS	m/s	-
acc[3]	double	all	Input acceleration vector (Initial reference frame) Dummy if <i>mode</i> is either: · XL_CALC_POS · XL_CALC_POS_VEL	m/s ²	-

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

- Calculation mode selection: `mode`. See current document, section 6.2.
- Time reference ID: `time_ref`. See [GEN_SUM].
- Reference frame: `cs_in`, `cs_out`. See current document, section 6.2.

Note that for the function to work correctly, the time references should be properly initialised before calling the function (see section 4.2 for details).

7.20.4 Output parameters

The output parameters of the `xl_change_cart_cs` CFI function are:

Table 92: Output parameters of `xl_change_cart_cs` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xl_change_cart_cs</code>	long	-	Extended status flag	-	-
<code>pos_out[3]</code>	double	all	Output position vector (Final reference frame)	m	-
<code>vel_out[3]</code>	double	all	Output velocity vector (Final reference frame) Returned only if <i>mode</i> is either: · <code>XL_CALC_POS_VEL</code> · <code>XL_CALC_POS_VEL_ACC</code>	m/s	-
<code>acc_out[3]</code>	double	all	Output acceleration vector (Final reference frame) Returned only if <i>mode</i> is: · <code>XL_CALC_POS_VEL_ACC</code>	m/s ²	-

7.20.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_change_cart_cs` CFI function after translating the returned extended status flag into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 `xl_change_cart_cs` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 93: Error messages of `xl_change_cart_cs` function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	<code>XL_CFI_CHANGE_CART_CS_SAT_ERR</code>	0
ERR	Input time reference ID is not correct	No calculation performed	<code>XL_CFI_CHANGE_CART_CS_REF_ERR</code>	1
ERR	Input date is out of range	No calculation performed	<code>XL_CFI_CHANGE_CART_CS_DAY_ERR</code>	2
ERR	Calculation mode ID is not correct	No calculation performed	<code>XL_CFI_CHANGE_CART_CS_MODE_ERR</code>	3
ERR	Input reference frame is not correct	No calculation performed	<code>XL_CFI_CHANGE_CART_CS_INPUT_CS_ERR</code>	4

Table 93: Error messages of xl_change_cart_cs function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Output reference frame is not correct	No calculation performed	XL_CFI_CHANGE_CART_CS_OUTPUT_CS_ERR	5
ERR	Time Reference not initialised	No calculation performed	XL_CFI_CHANGE_CART_CS_REF_INIT_ERR	6

7.20.6 Runtime performances

The following runtime performances have been measured.

Two runtime figures are provided, one with fixed inputs, i.e. the function has been called several times with the same time, but modifying the other input parameters; and a second one with random inputs, i.e all the inputs have been modified from call to call and the average time has been taken.

Table 94: Runtime performances of xl_change_cart_cs function

Ultra Sparc II-400 [ms] RANDOM inputs	Ultra Sparc II-400 [ms] FIXED inputs
0.0855	0.0380

7.21 xl_geod_to_cart

7.21.1 Overview

The `xl_geod_to_cart` CFI function transforms from geodetic to cartesian coordinates.

7.21.2 Calling interface

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

```
#include <explorer_lib.h>
{
    long mode;
    double lon, lat, h, lon_rate, lat_rate, h_rate;
    double pos[3], vel[3];
    long status;

    status = xl_geod_to_cart (&mode, &lon, &lat, &h,
                             &lon_rate, &lat_rate, &h_rate,
                             pos, vel);
}
```

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_lib.inc>

    INTEGER*4 MODE
    REAL*8 LON, LAT, H, LON_RATE, LAT_RATE, H_RATE
    REAL*8 POS(3), VEL(3)
    INTEGER*4 STATUS

    STATUS = XL_GEOD_TO_CART (MODE, LON, LAT, H,
    &                          LON_RATE, LAT_RATE, H_RATE,
    &                          POS, VEL)
```

7.21.3 Input parameters

The `xl_geod_to_cart` CFI function has the following input parameters:

Table 95: Input parameters of `xl_geod_to_cart` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
mode	long*	-	Calculation mode selection	-	Select either: · XL_CALC_POS · XL_CALC_POS_VEL
lon	double *	-	Geocentric longitude (Earth fixed CS)	deg	[0,360)
lat	double *	-	Geodetic latitude (Earth fixed CS)	deg	[-90,90]
h	double *	-	Geodetic altitude (Earth fixed CS)	m	$h \geq -b_{\text{ellipsoid}}$ (sat_id dependent)
lon_rate	double *	-	Geocentric longitude rate (Earth fixed CS) Dummy if <i>mode</i> is: · XL_CALC_POS	deg/s	-
lat_rate	double *	-	Geodetic latitude rate (Earth fixed CS) Dummy if <i>mode</i> is: · XL_CALC_POS	deg/s	-
h_rate	double *	-	Geodetic altitude rate (Earth fixed CS) Dummy if <i>mode</i> is: · XL_CALC_POS	m/s	-

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

- Calculation mode selection: mode. See current document, section 6.2.

7.21.4 Output parameters

The output parameters of the `xl_geod_to_cart` CFI function are:

Table 96: Output parameters of `xl_geod_to_cart` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xl_geod_to_cart	long	-	Extended status flag	-	-
pos[3]	double	all	Cartesian position vector (Earth fixed CS)	m	-

Table 96: Output parameters of `xl_geod_to_cart` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
vel[3]	double	all	Cartesian velocity vector (Earth fixed CS) Returned only if <i>mode</i> is: · XL_CALC_POS_VEL	m/s	-

7.21.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_geod_to_cart` CFI function after translating the returned extended status flag into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 `xl_geod_to_cart` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 97: Error messages of `xl_geod_to_cart` function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_GEOD_CART_SAT_ERR	0
ERR	Wrong geodetic latitude on input (out of range)	No calculation performed	XL_CFI_GEOD_CART_EL_GT_90_ERR	1
WARN	Calculation mode ID is not correct	Calculation performed. A message informs the user.	XL_CFI_GEOD_CART_MODE_WARN	2

The altitude of the geodetic state vector is not checked, so in case it does not satisfy its allowed range it may result in raising an internal error (see section 8).

7.21.6 Runtime performances

The following runtime performances have been measured.

Table 98: Runtime performances of `xl_geod_to_cart` function

Ultra Sparc II-400 [ms]
0.0045

7.22 xl_cart_to_geod

7.22.1 Overview

The `xl_cart_to_geod` CFI function transforms from cartesian to geodetic coordinates.

7.22.2 Calling interface

The calling interface of the `xl_cart_to_geod` function is the following (input parameters are underlined):

```
#include <explorer_lib.h>
{
    long mode;
    double pos[3], vel[3];
    double lon, lat, h, lon_rate, lat_rate, h_rate;
    long status;

    status = xl_cart_to_geod (&mode, pos, vel, &lon, &lat,
                             &h, &lon_rate, &lat_rate, &h_rate);
}
```

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_lib.inc>

    INTEGER*4 MODE
    REAL*8 POS (3), VEL (3)
    REAL*8 LON, LAT, H, LON_RATE, LAT_RATE, H_RATE
    INTEGER*4 STATUS

    STATUS = XL_CART_TO_GEOD (MODE, POS, VEL, LON, LAT, H,
    & LON_RATE, LAT_RATE, H_RATE)
```

7.22.3 Input parameters

The `xl_cart_to_geod` CFI function has the following input parameters:

Table 99: Input parameters of `xl_cart_to_geod` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
mode	long*	-	Calculation mode selection	-	Select either: · XL_CALC_POS · XL_CALC_POS_VEL
pos[3]	double	all	Cartesian position vector (Earth fixed CS)	m	$r > a_{\text{ellipsoid}} - b_{\text{ellipsoid}}$
vel[3]	double	all	Cartesian velocity vector (Earth fixed CS) Dummy if <i>mode</i> is: · XL_CALC_POS	m/s	-

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

- Calculation mode selection: mode. See current document, section 6.2.

7.22.4 Output parameters

The output parameters of the `xl_cart_to_geod` CFI function are:

Table 100: Output parameters of `xl_cart_to_geod` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xl_cart_to_geod	long	-	Extended status flag	-	-
lon	double *	-	Geocentric longitude (Earth fixed CS)	deg	≥ 0 $< +360$
lat	double *	-	Geodetic latitude (Earth fixed CS)	deg	≥ -90 $\leq +90$
h	double *	-	Geodetic altitude (Earth fixed CS)	m	-
lon_rate	double *	-	Geocentric longitude rate (Earth fixed CS) Returned only if <i>mode</i> is: · XL_CALC_POS_VEL	deg/s	-
lat_rate	double *	-	Geodetic latitude rate (Earth fixed CS) Returned only if <i>mode</i> is: · XL_CALC_POS_VEL	deg/s	-

Table 100: Output parameters of `xl_cart_to_geod` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
h_rate	double *	-	Geodetic altitude rate (Earth fixed CS) Returned only if <i>mode</i> is: · XL_CALC_POS_VEL	m/s	-

7.22.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_cart_to_geod` CFI function after translating the returned extended status flag into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 `xl_cart_to_geod` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 101: Error messages of `xl_cart_to_geod` function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_CART_GEOD_SAT_ERR	0
ERR	Internal computation error # 1	No calculation performed	XL_CFI_CART_GEOD_FRAME_ERR	1
ERR	Input vector out of valid range	No calculation performed	XL_CFI_CART_GEOD_VECTOR_ERR	2
WARN	Calculation mode ID is not correct	Calculation performed. A message informs the user.	XL_CFI_CART_GEOD_MODE_WARN	3
WARN	Geocentric longitude set to 0 deg (ambiguous case)	Calculation performed. A message informs the user.	XL_CFI_CART_GEOD_AMBIGUITY_WARN	4
WARN	Internal computation warning # 1	Calculation performed. A message informs the user.	XL_CFI_CART_GEOD_ACCURACY_WARN	5
WARN	Internal computation warning # 2	Calculation performed. A message informs the user.	XL_CFI_CART_GEOD_ITERATIONS_WARN	6
WARN	Internal computation warning # 3	Calculation performed. A message informs the user.	XL_CFI_CART_GEOD_DEFFVAL_WARN	7

7.22.6 Runtime performances

The following runtime performances have been measured.

Table 102: Runtime performances of xl_cart_to_geod function

Ultra Sparc II-400 [ms]
0.0030

7.23 xl_kepl_to_cart

7.23.1 Overview

The `xl_kepl_to_cart` CFI function transforms from keplerian to cartesian coordinates.

7.23.2 Calling interface

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

```
#include <explorer_lib.h>
{
|   long kepl_mode;
|   double kepl_in[6];
|   double pos_out[3], vel_out[3];
|   long ierr[XL_NUM_ERR_KEPL_CART], status;
|
|   status = xl_kepl_to_cart (&kepl_mode, kepl_in, pos_out,
|                           vel_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_lib.inc>
|
|   INTEGER*4 KEPL_MODE
|   REAL*8 KEPL_IN(6)
|   REAL*8 POS_OUT(3), VEL_OUT(3)
|   INTEGER*4 IERR(XL_NUM_ERR_KEPL_CART), STATUS
|
|   STATUS = XL_KEPL_TO_CART (KEPL_MODE, KEPL_IN,
| &                           POS_OUT, VEL_OUT, IERR)
```

7.23.3 Input parameters

The `xl_kepl_to_cart` CFI function has the following input parameters:

Table 103: Input parameters of `xl_kepl_to_cart` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>kepl_mode</code>	<code>long*</code>	-	Flag for selecting: · Mean elements = <code>XL_KEPLER_MEAN</code> · Osculating elements = <code>XL_KEPLER_OSC</code>	-	Complete
<code>kepl_in[6]</code>	double	[0]	Semi-major axis (True of Date CS)	m	≥ 0
		[1]	Eccentricity (True of Date CS)	-	[0,1)
		[2]	Inclination (True of Date CS)	deg	[0,180]
		[3]	Right ascension of the ascending node (True of Date CS)	deg	[0,360)
		[4]	Argument of perigee (True of Date CS)	deg	[0,360)
		[5]	Mean anomaly (True of Date CS)	deg	[0,360)

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

- Kepler state vector model: `kepl_mode`. See section 6.2.

7.23.4 Output parameters

The output parameters of the `xl_kepl_to_cart` CFI function are:

Table 104: Output parameters of `xl_kepl_to_cart` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xl_kepl_to_cart</code>	<code>long</code>	-	Status flag	-	-
<code>pos_out[3]</code>	double	all	Cartesian position vector (True of Date CS)	m	-
<code>vel_out[3]</code>	double	all	Cartesian velocity vector (True of Date CS)	m/s	-
<code>ierr</code>	<code>long</code>	-	Error vector	-	-

7.23.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xl_kepl_to_cart** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library **xl_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 error vector returned by the **xl_kepl_to_cart** function by calling the function of the EXPLORER_LIB software library **xl_get_code** (see [GEN_SUM])

Table 105: Error messages of xl_kepl_to_cart function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Input semi-major axis ≤ 0	No calculation performed	XL_CFI_K2C_A_ZERO_ERR	0
ERR	Input eccentricity < 0	No calculation performed	XL_CFI_K2C_E_ZERO_ERR	1
ERR	Input eccentricity > 1	No calculation performed	XL_CFI_K2C_E_ONE_ERR	2
ERR	Internal Error: Error in calling XL_Mean_to_osc	No calculation performed	XL_CFI_K2C_INTERNAL_M2O_ERR	3
ERR	Internal computation error #1	No calculation performed	XL_CFI_K2C_COMPUTATION_ERR	4
WARN	Internal Warning: Warning in calling XL_Mean_to_osc	Calculation performed. A message informs the user.	XL_CFI_K2C_INTERNAL_M2O_WARN	5
WARN	Kepler's equations not converged	Calculation performed. A message informs the user.	XL_CFI_K2C_NO_CONVERGED_WARN	6

7.23.6 Runtime performances

The following runtime performances have been measured.

Table 106: Runtime performances of xl_kepl_to_cart function

Ultra Sparc II-400 [ms]
0.0130

7.24 xl_cart_to_kepl

7.24.1 Overview

The `xl_cart_to_kepl` CFI function transforms from cartesian to keplerian coordinates.

7.24.2 Calling interface

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

```
#include <explorer_lib.h>
{
|   long kepl_mode;
|   double pos_in[3], vel_in[3];
|   double kepl_out[6];
|   long ierr[XL_NUM_ERR_CART_KEPL], status;
|
|   status = xl_cart_to_kepl (pos_in, vel_in, &kepl_mode
|                           &kepl_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_lib.inc>
|
|   INTEGER*4 KEPL_MODE
|   REAL*8 POS_IN(3), VEL_IN(3)
|   REAL*8 KEPL_OUT(6)
|   INTEGER*4 IERR(XL_NUM_ERR_CART_KEPL), STATUS
|
|   STATUS = XL_CART_TO_KEPL (POS_IN, VEL_IN, KEPL_MODE,
|   & KEPL_OUT, IERR)
```


7.24.3 Input parameters

The `xl_cart_to_kepl` CFI function has the following input parameters:

Table 107: Input parameters of `xl_cart_to_kepl` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>pos_in[3]</code>	double	all	Cartesian position vector (True of Date CS)	m	-
<code>vel_in[3]</code>	double	all	Cartesian velocity vector (True of Date CS)	m/s	-
<code>kepl_mode</code>	long*	-	Flag for selecting: · Mean elements = <code>XL_KEPLER_MEAN</code> · Osculating elements = <code>XL_KEPLER_OSC</code>	-	Complete

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

- Kepler state vector model: `kepl_mode`. See section 6.2.

7.24.4 Output parameters

The output parameters of the `xl_cart_to_kepl` CFI function are:

Table 108: Output parameters of `xl_cart_to_kepl` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xl_cart_to_kepl</code>	long	-	Status flag	-	-
<code>kepl_out[6]</code>	double	[0]	Semi-major axis (True of Date CS)	m	≥ 0
		[1]	Eccentricity (True of Date CS)	-	[0,1)
		[2]	Inclination (True of Date CS)	deg	[0,180]
		[3]	Right ascension of the ascending node (True of Date CS)	deg	[0,360)
		[4]	Argument of perigee (True of Date CS)	deg	[0,360)
		[5]	Mean anomaly (True of Date CS)	deg	[0,360)
<code>ierr</code>	long	-	Error vector	-	-

7.24.5 Warnings and errors

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

Table 109: Error messages of xl_cart_to_kepl function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Earth's Mu < 0	No calculation performed	XL_CFI_C2K_MU_ZERO_ERR	0
ERR	Input orbit radius = 0	No calculation performed	XL_CFI_C2K_OR_ZERO_ERR	1
ERR	Input orbit velocity = 0	No calculation performed	XL_CFI_C2K_OV_ZERO_ERR	2
ERR	Semi-major axis undefined	No calculation performed	XL_CFI_C2K_OA_UNDEFINED_ERR	3
ERR	Semi-major axis < 0	No calculation performed	XL_CFI_C2K_OA_ZERO_ERR	4
ERR	Internal computation error #1	No calculation performed	XL_CFI_C2K_COMPUTATION_ERR	5
ERR	Internal Error: Error in calling XL_Osc_to_mean	No calculation performed	XL_CFI_C2K_INTERNAL_O2M_ERR	6
WARN	Inclination = 0 or 180 deg	Calculation performed. A message informs the user.	XL_CFI_C2K_OI_ZERO_WARN	7
WARN	Eccentricity = 0	Calculation performed. A message informs the user.	XL_CFI_C2K_OE_ZERO_WARN	8
WARN	Internal Warning: Warning in calling XL_Osc_to_mean	Calculation performed. A message informs the user.	XL_CFI_C2K_INTERNAL_O2M_WARN	9

7.24.6 Runtime performances

The following runtime performances have been measured.

Table 110: Runtime performances of xl_cart_to_kepl function

Ultra Sparc II-400 [ms]
0.0130

7.25 xl_sun

7.25.1 Overview

The **xl_sun** CFI function calculates the position and velocity vector of the Sun in the True of Date coordinate system.

7.25.2 Calling interface

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

```
#include <explorer_lib.h>
{
    long time_ref;
    double time, sun_pos[3], sun_vel[3];
    xl_time_id time_id;
    long ierr[XL_NUM_ERR_SUN], status;

    status = xl_sun(&time_id, &time_ref, &time, sun_pos, sun_vel,
                   ierr);

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

    status = xl_sun_run(&run_id, &time_ref, &time, sun_pos, sun_vel,
                       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_lib.inc>

INTEGER*4 SAT_ID, TIME_REF
REAL*8 TIME, SUN_POS(3), SUN_VEL(3)
INTEGER*4 IERR(XL_NUM_ERR_SUN), STATUS

STATUS = XL_SUN(SAT_ID, TIME_REF, TIME, SUN_POS, SUN_VEL, IERR)
```

7.25.3 Input parameters

The `xl_sun` CFI function has the following input parameters:

Table 111: Input parameters of `xl_sun` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
time_ref	long *	-	Initial time reference ID	-	Any except XL_TIME_UNDEF
time	double*	-	Input 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].

Note that for the function to work correctly, the time references should be properly initialised before calling the function (see section 4.2 for details).

7.25.4 Output parameters

The output parameters of the `xl_sun` CFI function are:

Table 112: Output parameters of `xl_sun` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xl_sun	long	-	Status flag	-	-
sun_pos[3]	double	all	Position vector of the Sun in the True of Date CS	m	-
sun_vel[3]	double	all	Velocity vector of the Sun in the True of Date CS	m/s	-
ierr	long	-	Error vector	-	-

7.25.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_sun` CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 error vector returned by the `xl_sun` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 113: Error messages of xl_sun function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_SUN_SAT_ERR	0
ERR	Input time reference ID is not correct	No calculation performed	XL_CFI_SUN_REF_ERR	1
ERR	Input date is out of range	No calculation performed	XL_CFI_SUN_DAY_ERR	2
ERR	Time Reference not initialised	No calculation performed	XL_CFI_SUN_REF_INIT_ERR	3
ERR	Error in calling XL_Sun_PosVel	No calculation performed	XL_CFI_SUN_SUN_ERR	4

7.25.6 Runtime performances

The following runtime performances have been measured.

Two runtime figures are provided, one with fixed inputs, i.e. the function has been called several times with the same time, but modifying the other input parameters; and a second one with random inputs, i.e all the inputs have been modified from call to call and the average time has been taken.

Table 114: Runtime performances of xl_sun function

Ultra Sparc II-400 [ms] RANDOM inputs	Ultra Sparc II-400 [ms] FIXED inputs
0.3020	0.0255

7.26 xl_moon

7.26.1 Overview

The **xl_moon** CFI function calculates the position and velocity vector of the Moon in the True of Date coordinate system.

7.26.2 Calling interface

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

```
#include <explorer_lib.h>
{
    long time_ref;
    double time, moon_pos[3], moon_vel[3];
    xl_time_id time_id;
    long ierr[XL_NUM_ERR_MOON], status;

    status = xl_moon(&time_id, &time_ref, &time, moon_pos, moon_vel,
                    ierr);

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

    status = xl_moon_run(&run_id, &time_ref, &time,
                        moon_pos, moon_vel,
                        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_lib.inc>

INTEGER*4 SAT_ID, TIME_REF
REAL*8 TIME, MOON_POS(3), MOON_VEL(3)
INTEGER*4 IERR(XL_NUM_ERR_MOON), STATUS

STATUS = XL_MOON(SAT_ID, TIME_REF, TIME, MOON_POS, MOON_VEL, IERR)
```

7.26.3 Input parameters

The `xl_moon` CFI function has the following input parameters:

Table 115: Input parameters of `xl_moon` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
time_ref	long *	-	Initial time reference ID	-	Any except XL_TIME_UNDEF
time	double*	-	Input 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].

Note that for the function to work correctly, the time references should be properly initialised before calling the function (see section 4.2 for details).

7.26.4 Output parameters

The output parameters of the `xl_moon` CFI function are:

Table 116: Output parameters of `xl_moon` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xl_moon	long	-	Status flag	-	-
moon_pos[3]	double	all	Position vector of the Moon in the True of Date CS	m	-
moon_vel[3]	double	all	Velocity vector of the Moon in the True of Date CS	m/s	-
ierr	long	-	Error vector	-	-

7.26.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_moon` CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 error vector returned by the `xl_moon` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 117: Error messages of xl_moon function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_MOON_SAT_ERR	0
ERR	Input time reference ID is not correct	No calculation performed	XL_CFI_MOON_REF_ERR	1
ERR	Input date is out of range	No calculation performed	XL_CFI_MOON_DAY_ERR	2
ERR	Time Reference not initialised	No calculation performed	XL_CFI_MOON_REF_INIT_ERR	3
ERR	Error in calling XL_Moon_PosVel	No calculation performed	XL_CFI_MOON_MOON_ERR	4

7.26.6 Runtime performances

The following runtime performances have been measured.

Two runtime figures are provided, one with fixed inputs, i.e. the function has been called several times with the same time, but modifying the other input parameters; and a second one with random inputs, i.e all the inputs have been modified from call to call and the average time has been taken.

Table 118: Runtime performances of xl_moon function

Ultra Sparc II-400 [ms] RANDOM inputs	Ultra Sparc II-400 [ms] FIXED inputs
0.2860	0.0255

7.27 xl_planet

7.27.1 Overview

The **xl_planet** CFI function calculates the position and velocity vector of a planet in the Heliocentric Mean of 2000.0 coordinate system.

7.27.2 Calling interface

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

```
#include <explorer_lib.h>
{
    long sat_id, planet, time_ref;
    double time, planet_pos[3], planet_vel[3];
    xl_time_id time_id;
    long ierr[XL_NUM_ERR_PLANET], status;

    status = xl_planet(&time_id, &planet, &time_ref, &time,
                      planet_pos, planet_vel, ierr);

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

    status = xl_planet_run(&run_id, &planet, &time_ref, &time,
                          planet_pos, planet_vel, 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_lib.inc>

INTEGER*4 SAT_ID, PLANET, TIME_REF
REAL*8 TIME, PLANET_POS(3), PLANET_VEL(3)
INTEGER*4 IERR(XL_NUM_ERR_PLANET), STATUS

STATUS = XL_PLANET(SAT_ID, PLANET, TIME_REF, TIME, PLANET_POS,
&                  PLANET_VEL, IERR)
```

7.27.3 Input parameters

The `xl_planet` CFI function has the following input parameters:

Table 119: Input parameters of `xl_planet` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
planet	long *	-	Planet ID	-	Complete
time_ref	long *	-	Initial time reference ID	-	Any except XL_TIME_UNDEF
time	double*	-	Input 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].
- Planet ID: `planet`. Current document, section 6.2.

Note that for the function to work correctly, the time references should be properly initialised before calling the function (see section 4.2 for details).

7.27.4 Output parameters

The output parameters of the `xl_planet` CFI function are:

Table 120: Output parameters of `xl_planet` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xl_planet	long	-	Status flag	-	-
planet_pos[3]	double	all	Position vector of the Planet in the Heliocentric Mean of 2000.0 coordinate system	m	-
planet_vel[3]	double	all	Velocity vector of the Planet in the Heliocentric Mean of 2000.0 coordinate system	m/s	-
ierr	long	-	Error vector	-	-

7.27.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_planet` CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 error vector returned by the **xl_planet** function by calling the function of the EXPLORER_LIB software library **xl_get_code** (see [GEN_SUM]).

Table 121: Error messages of xl_planet function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_PLANET_SAT_ERR	0
ERR	Input time reference ID is not correct	No calculation performed	XL_CFI_PLANET_REF_ERR	1
ERR	Input date is out of range	No calculation performed	XL_CFI_PLANET_DAY_ERR	2
ERR	Time Reference not initialised	No calculation performed	XL_CFI_PLANET_REF_INIT_ERR	3
ERR	Planet code is not correct	No calculation performed	XL_CFI_PLANET_PLANET_ERR	4
WARN	Internal Warning: XL_Planets solution didn't converge	Calculation performed. A message informs the user.	XL_CFI_PLANET_CONV_WARN	5

7.27.6 Runtime performances

The following runtime performances have been measured.

Two runtime figures are provided, one with fixed inputs, i.e. the functions has been called several times with the same time (reference and value); and a second one with random inputs (random time).

Table 122: Runtime performances of xl_planet function

Ultra Sparc II-400 [ms] RANDOM inputs	Ultra Sparc II-400 [ms] FIXED inputs
0.0400	0.0075

7.28 xl_star_radec

7.28.1 Overview

The **xl_star_radec** CFI function calculates the right ascension and declination of a star in the True of Date coordinate system.

7.28.2 Calling interface

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

```
#include <explorer_lib.h>
{
    long time_ref;
    double time, ra0, dec0, mu_ra, mu_dec;
    double rad_vel, par, ra, dec;
    xl_time_id time_id;
    long ierr[XL_NUM_ERR_STAR], status;

    status = xl_star_radec(&time_id, &time_ref, &time, &ra0, &dec0,
                          &mu_ra, &mu_dec, &rad_vel, &par,
                          &ra, &dec, ierr);

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

    status = xl_star_radec_run(&run_id, &time_ref, &time, &ra0, &dec0,
                              &mu_ra, &mu_dec, &rad_vel, &par,
                              &ra, &dec, 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_lib.inc>

INTEGER*4 SAT_ID, TIME_REF
REAL*8 TIME, RA0, DEC0, MU_RA, MU_DEC
REAL*8 RAD_VEL, PAR, RA, DEC
INTEGER*4 IERR(XL_NUM_ERR_STAR), STATUS
```

```
STATUS = XL_STAR_RADEC (SAT_ID, TIME_REF, TIME, RA0, DEC0, MU_RA,
& MU_DEC, RAD_VEL, PAR, RA, DEC, IERR)
```

7.28.3 Input parameters

The `xl_star_radec` CFI function has the following input parameters:

Table 123: Input parameters of `xl_star_radec` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
time_ref	long *	-	Initial time reference ID	-	Any except XL_TIME_UNDEF
time	double*	-	Input time	Decimal days (Processing format)	[-18262.0,36524.0]
ra0	double *	-	Right ascension of the star at J2000.0 (Barycentric Mean of 2000.0 CS)	deg	[0,360)
dec0	double *	-	Declination of the star at J2000.0 (Barycentric Mean of 2000.0 CS)	deg	[-90,90]
mu_ra	double *	-	Proper motion in the right ascension at J2000.0 (Barycentric Mean of 2000.0 CS)	deg/century	-
mu_dec	double *	-	Proper motion in the declination at J2000.0 (Barycentric Mean of 2000.0 CS)	deg/century	-
rad_vel	double *	-	Radial velocity of the star at J2000.0 (Barycentric Mean of 2000.0 CS)	AU/century	-
par	double *	-	Parallax of the star at J2000.0 (Barycentric Mean of 2000.0 CS)	deg	-

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].

Note that for the function to work correctly, the time references should be properly initialised before calling the function (see section 4.2 for details).

7.28.4 Output parameters

The output parameters of the `xl_star_radec` CFI function are:

Table 124: Output parameters of `xl_star_radec` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xl_star_radec</code>	long	-	Status flag	-	-
<code>ra</code>	double *	-	Right ascension of the star at specified time (True of Date CS)	deg	[0,360)
<code>dec</code>	double *	-	Declination of the star at specified time (True of Date CS)	deg	[-90,90]
<code>ierr</code>	long	-	Error vector	-	-

7.28.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_star_radec` CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 error vector returned by the `xl_star_radec` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 125: Error messages of `xl_star_radec` function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_STAR_RADEC_SAT_ERR	0
ERR	Input time reference ID is not correct	No calculation performed	XL_CFI_STAR_RADEC_REF_ERR	1
ERR	Input date is out of range	No calculation performed	XL_CFI_STAR_RADEC_DATE_ERR	2
ERR	Time Reference not initialised	No calculation performed	XL_CFI_STAR_RADEC_REF_INIT_ERR	3
ERR	Error in calling XL_Star	No calculation performed	XL_CFI_STAR_RADEC_STAR_ERR	4
ERR	Error in calling XL_Dir_Pointing	No calculation performed	XL_CFI_STAR_RADEC_DIRPOINT_ERR	5
WARN	Warning in calling XL_Dir_Pointing	Calculation performed. A message informs the user.	XL_CFI_STAR_RADEC_DIRPOINT_WARN	6

The declination is not checked, so in case it does not satisfy its allowed range it may result in raising an internal error (see section 8).

7.28.6 Runtime performances

The following runtime performances have been measured.

Two runtime figures are provided, one with fixed inputs, i.e. the functions has been called several times with the same time (reference and value), but modifying the other input parameters; and a second one with random inputs, i.e all the inputs have been modified from call to call and the average time has been taken.

Table 126: Runtime performances of *xl_star_radec* function

Ultra Sparc II-400 [ms] RANDOM inputs	Ultra Sparc II-400 [ms] FIXED inputs
0.3120	0.0017

7.29 xl_geod_distance

7.29.1 Overview

The **xl_geod_distance** CFI function calculates the geodesic distance between two points that lay on the same ellipsoid, and the azimuth of the related geodesic line at both points. See diagram below.

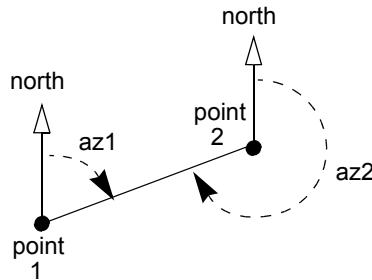


Figure 2: Azimuth figures returned by xl_geod_distance function

7.29.2 Calling interface

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

```
#include <explorer_lib.h>
{
    double lon1, lat1, lon2, lat2, h;
    double distance, az_1_to_2, az_2_to_1;
    long status;
    status = xl_geod_distance (<u>lon1</u>, <u>lat1</u>, <u>lon2</u>, <u>lat2</u>,
                              <u>h</u>, <u>distance</u>, <u>az_1_to_2</u>,
                              <u>az_2_to_1</u>);
}
```

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_lib.inc>

REAL*8 LON1, LAT1, LON2, LAT2, H
REAL*8 DISTANCE, AZ_1_TO_2, AZ_2_TO_1
INTEGER*4 STATUS
STATUS = XL_GEOD_DISTANCE (<u>LON1</u>, <u>LAT1</u>, <u>LON2</u>, <u>LAT2</u>, <u>H</u>, &
                           <u>DISTANCE</u>, <u>AZ_1_TO_2</u>, <u>AZ_2_TO_1</u>)
```


7.29.3 Input parameters

The `xl_geod_distance` CFI function has the following input parameters:

Table 127: Input parameters of `xl_geod_distance` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
lon1	double *	-	Geocentric longitude of the first input point (Earth fixed CS)	deg	[0,360)
lat1	double *	-	Geodetic latitude of the first input point (Earth fixed CS)	deg	[-90,90]
lon2	double *	-	Geocentric longitude of the second input point (Earth fixed CS)	deg	[0,360)
lat2	double *	-	Geodetic latitude of the second input point (Earth fixed CS)	deg	[-90,90]
h	double *	-	Geodetic altitude of both input points (Earth fixed CS)	m	$h \geq -b_{WGS}$ (satellite ID dependent)

7.29.4 Output parameters

The output parameters of the `xl_geod_distance` CFI function are:

Table 128: Output parameters of `xl_geod_distance` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xl_geod_distance</code>	long	-	Extended status flag	-	-
distance	double *	-	Geodesic distance between the two input points (Earth fixed CS)	m	≥ 0
az_1_to_2	double *	-	Azimuth of the geodesic line from point 1 to point 2 (Topocentric CS)	deg	[0,360)
az_2_to_1	double *	-	Azimuth of the geodesic line from point 2 to point 1 (Topocentric CS) Note that $az_2 = az_1 + 180$ approximately	deg	≥ 0 < 360

7.29.5 Warnings and errors

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

Table 129: Error messages of xl_geod_distance function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID not correct	No calculation performed	XL_CFI_GEOD_DIST_SAT_ERR	0
ERR	Different altitudes in the two points	No calculation performed	XL_CFI_GEOD_DIST_ALTITUDE_ERR	1
ERR	Calculation not performed in XL_Geo_Car	No calculation performed	XL_CFI_GEOD_DIST_GEO_CAR_ERR	2
ERR	Calculation not performed in XL_Pt_Dir_Range	No calculation performed	XL_CFI_GEOD_DIST_DIR_RANGE_ERR	3
ERR	No solution returned by XL_Dir_Pointing	No calculation performed	XL_CFI_GEOD_DIST_DIR_POINTING_ERR	4
WARN	Antipodal points. Two possible azimuth values (0 or 180). Selected value is 0.0 deg	Calculation performed. A message informs the user.	XL_CFI_GEOD_DIST_ANTIPODAL_POINTS_WARN	5
WARN	Default values returned by XL_Dir_Pointing	Calculation performed. A message informs the user.	XL_CFI_GEOD_DIST_DIR_POINTING_WARN	6

The altitude of the two points is not checked, so in case it does not satisfy its allowed range it may result in raising an internal error (see section 8).

For antipodal points, a little variation of the input coordinates may lead to incoherent values for the output distance, depending on the point location on the ellipsoid.

7.29.6 Runtime performances

The following runtime performances have been measured.

Table 130: Runtime performances of xl_geod_distance function

Ultra Sparc II-400 [ms]
0.1820

7.30 xl_time_get_leap_second_info

7.30.1 Overview

The `xl_time_get_leap_second_info` CFI function retrieves the leap second location (if any) in the initialised time range.

In order to avoid ambiguities the instant of Leap Second insertion is given both as the instant just before insertion (i.e. when the LS start) and the instant just after insertion (i.e. when the LS ends).

As an example, in the case of the (positive) LS inserted on January 1st, 1999, the function would return (if `ascii_id_out = XL_ASCII_STD_REF_MICROSEC`):

```
leap_flag = 1
ascii_utc_time_before_leap = UTC=1998-12-31_23:59:60.000000
ascii_utc_time_after_leap = UTC=1999-01-01_00:00:00.000000
```

In the case of a negative LS, inserted as an example on January 1st, 2009, the function would return (if `ascii_id_out = XL_ASCII_STD_REF_MICROSEC`):

```
leap_flag = -1
ascii_utc_time_before_leap = UTC=2008-12-31_23:59:58.000000
ascii_utc_time_after_leap = UTC=2009-01-01_00:00:00.000000
```

7.30.2 Calling interface

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

```
#include <explorer_lib.h>
{
|   long ascii_id_out, leap_flag;
|   char ascii_utc_time_before_leap[XL_TIME_ASCII_DIM_MAX];
|   char ascii_utc_time_after_leap[XL_TIME_ASCII_DIM_MAX]
|   xl_time_id time_id;
|   long ierr[XL_NUM_ERR_LEAP_INFO], status;
|
|   status = xl_time_get_leap_second_info(&time_id, &ascii_id_out,
|                                       &leap_flag, ascii_utc_time_before_leap,
|                                       ascii_utc_time_after_leap, ierr);
|
|   /* Or, using the run_id */
|   long run_id;
|
|   status = xl_time_get_leap_second_info_run(&run_id, &ascii_id_out,
|                                           &leap_flag, ascii_utc_time_before_leap,
```

```

    }
        ascii_utc_time_after_leap, ierr);

```

The XL_TIME_ASCII_DIM_MAX and XL_NUM_ERR_LEAP_INFO constants are defined in the file *explorer_lib.h*.

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 #include statement):

```

#include <explorer_lib.inc>

INTEGER*4 SAT_ID, ASCII_ID_OUT, LEAP_FLAG
CHARACTER ASCII.UTC.TIME.BEFORE.LEAP(XL.TIME.ASCII.DIM.MAX)
CHARACTER ASCII.UTC.TIME.AFTER.LEAP(XL.TIME.ASCII.DIM.MAX)
INTEGER*4 IERR(XL.NUM.ERR.LEAP.INFO), STATUS

STATUS = XL.TIME.GET.LEAP.SECOND.INFO (SAT_ID, ASCII_ID_OUT,
&                                     LEAP_FLAG, ASCII.UTC.TIME.BEFORE.LEAP,
&                                     ASCII.UTC.TIME.AFTER.LEAP, IERR);

```

7.30.3 Input parameters

The `xl_time_get_leap_second_info` CFI function has the following input parameters:

Table 131: Input parameters of `xl_time_get_leap_second_info` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>time_id</code>	<code>xl_time_id*</code>	-	Structure that contains the time correlations.	-	-
<code>ascii_id_out</code>	<code>long *</code>	-	ASCII format ID for output	-	Complete

It is possible to use enumeration values rather than integer values for the input argument:

- ASCII format ID: `ascii_id_out`. Current document, section 6.2.

7.30.4 Output parameters

The output parameters of the `xl_time_get_leap_second_info` CFI function are:

Table 132: Output parameters of `xl_time_get_leap_second_info` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xl_time_get_leap_second_info</code>	<code>long</code>	-	Status flag	-	-
<code>leap_flag</code>	<code>long *</code>	-	Flag for leap second presence within time initialization data	-	-1 = Negative Leap Second (a LS has been removed) (very rare case) 0 = No leap second within initialization data +1 = Positive Leap Second (a LS has been added) (usual case)
<code>ascii_utc_time_before_leap</code>	<code>char</code>	See table 4 and table 5	UTC time just before leap second insertion (dummy if <code>leap_flag=0</code>)	See table 4 and table 5	See table 4 and table 5
<code>ascii_utc_time_after_leap</code>	<code>char</code>	See table 4 and table 5	UTC time just after leap second insertion (dummy if <code>leap_flag=0</code>)	See table 4 and table 5	See table 4 and table 5
<code>ierr</code>	<code>long</code>	-	Error vector	-	-

Note that if more than one leap second is contained within the time initialization data for the selected satellite, only the last (most recent) one is returned.

No more than one leap second is likely to be found in the data, unless the range of time initialization span more than one year (a total of 23 leap seconds have been inserted until 2002, since the system was introduced in 1972).

7.30.5 Warnings and errors

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

Table 133: Error messages of xl_time_get_leap_second_info function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Satellite ID is not correct	No calculation performed	XL_CFI_TIME_LEAP_SEC OND_SAT_ERR	0
ERR	Output ascii format ID is not correct	No calculation performed	XL_CFI_TIME_LEAP_SEC OND_ASCII_OUT_ERR	1
ERR	Satellite ID and output format ID are not compatible	No calculation performed	XL_CFI_TIME_LEAP_SEC OND_COMP_OUT_ERR	2
ERR	Error in adding times in Processing format	No calculation performed	XL_CFI_TIME_LEAP_SEC OND_ADD_ERR	3
ERR	Error in converting from Processing to ASCII format	No calculation performed	XL_CFI_TIME_LEAP_SEC OND_P2A_ERR	4
WARN	Time Reference not initialised	No calculation performed A message informs the user.	XL_CFI_TIME_LEAP_SEC OND_TIME_REF_INIT_WA RN	5

7.30.6 Runtime performances

The following runtime performances have been measured.

Table 134: Runtime performances of xl_time_get_leap_second_info function

Ultra Sparc II-400 [ms]
0.0010

7.31 xl_euler_to_matrix

TBW

7.32 xl_matrix_to_euler

TBW

7.33 xl_position_on_orbit

7.33.1 Overview

The **xl_position_on_orbit** CFI function calculates a value describing the position of the satellite within the orbit, using as input a Cartesian orbit state vector.

7.33.2 Calling interface

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

```
#include <explorer_lib.h>
{
    long angle_type, time_ref, deriv;
    double time, pos[3], vel[3], acc[3],
    double angle, angle_rate, angle_rate_rate;
    xl_time_id time_id;
    long status, ierr[XL_NUM_ERR_POSITION_ON_ORBIT];

    status = xl_position_on_orbit(&time_id,
                                &angle_type,
                                &time_ref, &time,
                                pos, vel, acc, &deriv,
                                &angle, &angle_rate,
                                &angle_rate_rate,
                                ierr);

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

    status = xl_position_on_orbit_run(&run_id,
                                     &angle_type,
                                     &time_ref, &time,
                                     pos, vel, acc, &deriv,
                                     &angle, &angle_rate,
                                     &angle_rate_rate,
                                     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_lib.inc>
```

```
INTEGER*4 ANGLE_TYPE, DERIV, TIME_REF
REAL*8 TIME, POS(3), VEL(3), ACC(3), ANGLE, ANGLE_RATE,
& ANGLE_RATE_RATE
INTEGER*4 STATUS, IERR(XL_NUM_ERR_POSITION_ON_ORBIT)

STATUS = XL_POSITION_ON_ORBIT(ANGLE_TYPE, TIME_REF,
& TIME, POS, VEL, ACC, DERIV, ANGLE,
& ANGLE_RATE, ANGLE_RATE_RATE, IERR)
```

7.33.3 Input parameters

The `xl_position_on_orbit` CFI function has the following input parameters:

Table 135: Input parameters of `xl_position_on_orbit` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
angle_type	long*	-	Type of angle	-	XL_ANGLE_TYPE _TRUE_LAT_TOD XL_ANGLE_TYPE _MEAN_LAT_TOD
time_ref	long*	-	Time reference ID	-	Complete
time	double*	-	Reference time	Decimal days (Processing format)	[-18262.0,36524.0]
pos	double[3]	all	Satellite position vector (Earth Fixed CS)	m	-
vel	double[3]	all	Satellite velocity vector (Earth Fixed CS)	m/s	-
acc	double[3]	all	Satellite acceleration vector (Earth Fixed CS)	m/s ²	-
deriv	long *	-	Derivative ID	-	Allowed values: (0) XL_NO_DER (1) XL_DER_1ST (2) XL_DER_2ND

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

- Time reference ID: `time_ref`.

7.33.4 Output parameters

The output parameters of the `xl_position_on_orbit` CFI function are:

Table 136: Output parameters of `xl_position_on_orbit` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>angle</code>	<code>double*</code>	-	Angle describing the position in the orbit	deg	-
<code>angle_rate</code>	<code>double*</code>	-	Angle describing the position in the orbit-rate	deg/s	-
<code>angle_rate_rate</code>	<code>double*</code>	-	Angle describing the position in the orbit-rate-rate	deg/s ²	-
<code>ierr[XL_NUM_ERR_POSITION_ON_ORBIT]</code>	<code>long</code>	all	Status vector	-	-

7.33.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_position_on_orbit` CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 `xl_position_on_orbit` CFI function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 137: Error messages of `xl_position_on_orbit` function

Error type	Error message	Cause and impact	Error code	Error No
TBW				

7.33.6 Runtime performances

The following runtime performance has been measured.

Table 138: Runtime performances of `xl_position_on_orbit` function

Ultra Sparc II-400[ms]
TBD

7.34 xl_get_rotation_angles

7.34.1 Overview

The `xl_get_rotation_angles` CFI function calculates the rotation angles between two sets of orthonormal right-handed unit vectors expressed wrt an identical coordinate frame.

7.34.2 Calling interface

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

```
#include <explorer_lib.h>
{
    double xs_initial[3], ys_initial[3], zs_initial[3];
    double xs_final[3], ys_final[3], zs_final[3];
    double ang[3];
    long ierr[XL_NUM_ERR_GET_ROTATION_ANGLES], status;
    status = xl_get_rotation_angles (xs_initial, ys_initial,
                                   zs_initial, xs_final, ys_final, zs_final,
                                   ang, ierr);
}
```

The `XL_NUM_ERR_GET_ROTATION_ANGLES` constant is defined in the file `explorer_lib.h`.

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_pointing.inc>

REAL*8 XS_INITIAL(3), YS_INITIAL(3), ZS_INITIAL(3)
REAL*8 XS_FINAL(3), YS_FINAL(3), ZS_FINAL(3)
REAL*8 ANG(3)
INTEGER*4 IERR(XL_NUM_ERR_GET_ROTATION_ANGLES), STATUS

STATUS = XL_GET_ROTATION_ANGLES (XS_INITIAL, YS_INITIAL, &
&                                ZS_INITIAL, XS_FINAL, YS_FINAL, ZS_FINAL,
&                                ANG, IERR)
```

7.34.3 Input parameters

The `xl_get_rotation_angles` CFI function has the following input parameters:

Table 139: Input parameters of `xl_get_rotation_angles` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xs_initial[3]</code>	double	all	Unitary direction vector along the X-axes of the initial attitude frame (Coordinate System)	-	-
<code>ys_initial[3]</code>	double	all	Unitary direction vector along the Y-axes of the initial attitude frame (Coordinate System)	-	-
<code>zs_initial[3]</code>	double	all	Unitary direction vector along the Z-axes of the initial attitude frame (Coordinate System)	-	-
<code>xs_final[3]</code>	double	all	Unitary direction vector along the X-axes of the final attitude frame (Coordinate System)	-	-
<code>ys_final[3]</code>	double	all	Unitary direction vector along the Y-axes of the final attitude frame (Coordinate System)	-	-
<code>zs_final[3]</code>	double	all	Unitary direction vector along the Z-axes of the final attitude frame (Coordinate System)	-	-

7.34.4 Output parameters

The output parameters of the `xl_get_rotation_angles` CFI function are:

Table 140: Output parameters of `xl_get_rotation_angles` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>ang[3]</code>	double	[0]	Pitch angle between initial and final Attitude Frames	deg	[-180,180)
		[1]	Roll angle between initial and final Attitude Frames	deg	[-180,180)
		[2]	Yaw angle between initial and final Attitude Frames	deg	[-180,180)

Table 140: Output parameters of `xl_get_rotation_angles` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
ierr	long	-	Error vector	-	-

7.34.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_get_rotation_angles` CFI function after translating the returned extended status flag into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 `xl_get_rotation_angles` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 141: Error messages of `xl_get_rotation_angles` function

Error type	Error message	Cause and impact	Error code	Error No
TBW				

7.34.6 Runtime performances

The following runtime performances have been measured.

Two runtime figures are provided, one with fixed inputs, i.e. the function has been called several times with the same position, velocity and acceleration vectors, but modifying the other input parameters; and a second one with random inputs, i.e. all the inputs have been modified from call to call and the average time has been taken.

Table 142: Runtime performances of `xl_get_rotation_angles` function

Ultra Sparc II-400 [ms] RANDOM inputs	Ultra Sparc II-400 [ms] FIXED inputs
TBD	TBD

7.35 xl_get_rotated_vectors

7.35.1 Overview

The `xl_get_rotated_vectors` CFI function calculates the rotated unit vectors given a set of unit vectors and the rotation angles expressed wrt an identical coordinate frame.

7.35.2 Calling interface

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

```
#include <explorer_lib.h>
{
    double xs_initial[3], ys_initial[3], zs_initial[3];
    double xs_final[3], ys_final[3], zs_final[3];
    double ang[3];
    long ierr[XL_NUM_ERR_GET_ROTATED_VECTORS], status;
    status = xl_get_rotated_vectors (xs_initial, ys_initial,
                                    zs_initial, ang, xs_final, ys_final,
                                    zs_final, ierr);
}
```

The `XL_NUM_ERR_GET_ROTATED_VECTORS` constant is defined in the file `explorer_lib.h`.

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_lib.inc>

REAL*8 XS_INITIAL(3), YS_INITIAL(3), ZS_INITIAL(3)
REAL*8 XS_FINAL(3), YS_FINAL(3), ZS_FINAL(3)
REAL*8 ANG(3)
INTEGER*4 IERR(XL_NUM_ERR_GET_ROTATED_VECTORS), STATUS

STATUS = XL_GET_ROTATED_VECTORS (XS_INITIAL, YS_INITIAL,
&                                ZS_INITIAL, ANG, XS_FINAL, YS_FINAL,
&                                ZS_FINAL, IERR)
```

7.35.3 Input parameters

The `xl_get_rotated_vectors` CFI function has the following input parameters:

Table 143: Input parameters of `xl_get_rotated_vectors` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xs_initial[3]</code>	double	all	Unitary direction vector along the X-axes of the initial attitude frame (Coordinate System)	-	-
<code>ys_initial[3]</code>	double	all	Unitary direction vector along the Y-axes of the initial attitude frame (Coordinate System)	-	-
<code>zs_initial[3]</code>	double	all	Unitary direction vector along the Z-axes of the initial attitude frame (Coordinate System)	-	-
<code>ang[3]</code>	double	[0]	Pitch angle between initial and final Attitude Frames	deg	[-180,180)
		[1]	Roll angle between initial and final Attitude Frames	deg	[-180,180)
		[2]	Yaw angle between initial and final Attitude Frames	deg	[-180,180)

7.35.4 Output parameters

The output parameters of the `xl_get_rotated_vectors` CFI function are:

Table 144: Output parameters of `xl_get_rotated_vectors` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xs_final[3]</code>	double	all	Unitary direction vector along the X-axes of the rotated attitude frame (Coordinate System)	-	-
<code>ys_final[3]</code>	double	all	Unitary direction vector along the Y-axes of the rotated attitude frame (Coordinate System)	-	-
<code>zs_final[3]</code>	double	all	Unitary direction vector along the Z-axes of the rotated attitude frame (Coordinate System)	-	-

Table 144: Output parameters of `xl_get_rotated_vectors` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
ierr	long	-	Error vector	-	-

7.35.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_get_rotated_vectorsCFI` function after translating the returned extended status flag into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 `xl_get_rotated_vectors` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 145: Error messages of `xl_get_rotated_vectors` function

Error type	Error message	Cause and impact	Error code	Error No
TBW				

7.35.6 Runtime performances

The following runtime performances have been measured.

Two runtime figures are provided, one with fixed inputs, i.e. the function has been called several times with the same position, velocity and acceleration vectors, but modifying the other input parameters; and a second one with random inputs, i.e. all the inputs have been modified from call to call and the average time has been taken.

Table 146: Runtime performances of `xl_get_rotated_vectors` function

Ultra Sparc II-400 [ms] RANDOM inputs	Ultra Sparc II-400 [ms] FIXED inputs
TBD	TBD

7.36 xl_quaternions_to_vectors

7.36.1 Overview

The `xl_quaternions_to_vectors` CFI function calculates the orthonormal unit vectors from a given set of quaternions.

7.36.2 Calling interface

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

```
#include <explorer_lib.h>
{
    double quaternions[4];
    double ux_vec[3], uy_vec[3], uz_vec[3];
    long ierr[XL_NUM_ERR_QUATERNIONS_TO_VECTORS], status;

    status = xl_quaternions_to_vectors (quaternions,
                                       ux_vec, uy_vec, uz_vec, ierr);
}
```

The `XL_NUM_ERR_QUATERNIONS_TO_VECTORS` constant is defined in the file `explorer_lib.h`.

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_lib.inc>

REAL*8 QUATERNIONS(4)
REAL*8 UX_VEC(3), UY_VEC(3), UZ_VEC(3)
INTEGER*4 IERR(XL_NUM_ERR_QUATERNIONS_TO_VECTORS), STATUS

STATUS = XL_QUATERNIONS_TO_VECTORS (QUATERNIONS,
&                                     UX_VEC, UY_VEC, UZ_VEC, IERR)
```

7.36.3 Input parameters

The `xl_quaternions_to_vectors` CFI function has the following input parameters:

Table 147: Input parameters of `xl_quaternions_to_vectors` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
quaternions	double[4]	-	Quaternions	-	-

7.36.4 Output parameters

The output parameters of the `xl_quaternions_to_vectors` CFI function are:

Table 148: Output parameters of `xl_quaternions_to_vectors` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
ux_vec[3]	double	all	Unitary direction vector along the X-axes of the coordinate or attitude frame	-	-
uy_vec[3]	double	all	Unitary direction vector along the Y-axes of the coordinate or attitude frame	-	-
uz_vec[3]	double	all	Unitary direction vector along the Z-axes of the coordinate or attitude frame	-	-
ierr	long	-	Error vector	-	-

7.36.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_quaternions_to_vectors` CFI function after translating the returned extended status flag into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 `xl_quaternions_to_vectors` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 149: Error messages of `xl_quaternions_to_vectors` function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Wrong input quaternion. The module is different from 1	No calculation performed	XL_CFI_QUATERNIONS_TO_VEC_WRONG_INPUT_ERROR	0

7.36.6 Runtime performances

The following runtime performances have been measured. Two runtime figures are provided, one with fixed inputs, i.e. the function has been called several times with the same position, velocity and acceleration vectors, but modifying the other input parameters; and a second one with random inputs, i.e. all the inputs have been modified from call to call and the average time has been taken.

Table 150: Runtime performances of *xl_quaternions_to_vectors* function

Ultra Sparc II-400 [ms] RANDOM inputs	Ultra Sparc II-400 [ms] FIXED inputs
TBD	TBD

7.37 xl_vectors_to_quaternions

7.37.1 Overview

The `xl_vectors_to_quaternions` CFI function calculates the set of quaternions that correspond to a set of orthonormal unit vectors.

7.37.2 Calling interface

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

```
#include <explorer_lib.h>
{
    double quaternions[4];
    double ux_vec[3], uy_vec[3], uz_vec[3];
    long ierr[XL_NUM_ERR_VECTORS_TO_QUATERNIONS], status;

    status = xl_vectors_to_quaternions (ux_vec, uy_vec, uz_vec,
                                        quaternions, ierr);
}
```

The `XL_NUM_ERR_VECTORS_TO_QUATERNIONS` constant is defined in the file `explorer_lib.h`.

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_lib.inc>

REAL*8 QUATERNIONS(4)
REAL*8 UX_VEC(3), UY_VEC(3), UZ_VEC(3)
INTEGER*4 IERR(XL_NUM_ERR_VECTORS_TO_QUATERNIONS), STATUS

STATUS = XL_VECTORS_TO_QUATERNIONS (UX_VEC, UY_VEC, UZ_VEC,
&                                     QUATERNIONS, IERR)
```

7.37.3 Input parameters

The `xl_vectors_to_quaternions` CFI function has the following input parameters:

Table 151: Input parameters of `xl_vectors_to_quaternions` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>ux_vec[3]</code>	double	all	Unitary direction vector along the X-axes of the coordinate or attitude frame	-	-
<code>uy_vec[3]</code>	double	all	Unitary direction vector along the Y-axes of the coordinate or attitude frame	-	-
<code>uz_vec[3]</code>	double	all	Unitary direction vector along the Z-axes of the coordinate or attitude frame	-	-

7.37.4 Output parameters

The output parameters of the `xl_vectors_to_quaternions` CFI function are:

Table 152: Output parameters of `xl_vectors_to_quaternions` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>quaternions</code>	double[4]	-	Quaternions	-	-
<code>ierr</code>	long	-	Error vector	-	-

7.37.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_vectors_to_quaternions` CFI function after translating the returned extended status flag into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 `xl_vectors_to_quaternions` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 153: Error messages of `xl_vectors_to_quaternions` function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Wrong input vectors.	No calculation performed	XL_CFI_VEC_TO_QUATERNIONS_WRONG_INPUT_ERROR	0

7.37.6 Runtime performances

The following runtime performances have been measured. Two runtime figures are provided, one with fixed inputs, i.e. the function has been called several times with the same position, velocity and acceleration vectors, but modifying the other input parameters; and a second one with random inputs, i.e. all the inputs have been modified from call to call and the average time has been taken.

Table 154: Runtime performances of *xl_vectors_to_quaternions* function

Ultra Sparc II-400 [ms] RANDOM inputs	Ultra Sparc II-400 [ms] FIXED inputs
TBD	TBD

7.38 xl_default_sat_init

7.38.1 Overview

The `xl_default_sat_init` CFI function initializes a default satellite from a configuration file (see section 10). This operation is needed whenever a default satellite is to be used for the first time, otherwise the satellite will not be recognised.

7.38.2 Calling interface

The calling interface `xl_default_sat_init` function is the following (input parameters are underlined):

```
#include <explorer_lib.h>
{
    long sat_id;
    char *conf_file;
    long ierr[XL_NUM_ERR_DEFAULT_SAT_INIT];
    long status;

    status = xl_default_sat_init(&sat_id, conf_file, 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_lib.inc>

INTEGER*4 SAT_ID
CHARACTER*(*) *CONF_FILE
INTEGER*4 IERR(XL_NUM_ERR_DEFAULT_SAT_INIT), STATUS

STATUS = XL_DEFAULT_SAT_INIT (SAT_ID, CONF_FILE, IERR)
```


7.38.3 Input parameters

The `xl_default_sat_init` function has the following input parameters:

Table 155: Input parameters of `xl_default_sat_init` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long*	-	Satellite ID	-	From XL_SAT_DEFAULT to XL_SAT_DEFAULT9
conf_file	char*	-	Path and name for the Satellite Configuration File (see section 10 for further details about the configuration file).	-	-

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

- Satellite ID: `sat_id`. See [GEN_SUM].

7.38.4 Output parameters

The output parameters of the `xl_default_sat_init` function are:

Table 156: Output parameters of `xl_default_sat_init` function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
ierr	long*	all	Error status flags	-	

7.38.5 Warnings and errors

Next table lists the possible error messages that can be returned by the `xl_default_sat_init` CFI function after translating the returned extended status flag into the equivalent list of error messages by calling the function of the EXPLORER_LIB software library `xl_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 `xl_default_sat_init` function by calling the function of the EXPLORER_LIB software library `xl_get_code` (see [GEN_SUM]).

Table 157: Error messages of xl_default_sat_init function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Default satellite ID is not correct	The satellite identification number does not belong to a default satellite. No computation performed	XL_CFI_DEFAULT_SAT_INIT_SAT_ERR	
ERR	Error while reading satellite configuration file	Wrong configuration file. No computation performed	XL_CFI_DEFAULT_SAT_INIT_READ_FILE_ERR	

7.38.6 Runtime performances

The following runtime performances have been measured.

Table 158: Runtime performances of xl_default_sat_init function

Ultra Sparc II-400 [ms]
TBD

8 LIBRARY PRECAUTIONS

The following precaution shall be taking into account when using EXPLORER_LIB library:

- The functions *xl_time_obt_to_time* and *xl_time_time_to_obt* cannot be called from ForTran (no ForTran equivalence to *unsigned long*).
- When a message like

EXPLORER_LIB >>> ERROR in *xl_function*: Internal computation error # *n*

or

EXPLORER_LIB >>> WARNING in *xl_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 159: Known problems

CFI library	Problem	Work around solution
xl_time_obt_to_time	Cannot be called from ForTran (no ForTran equivalence to unsigned long)	-
xl_time_obt_to_time	Since OBT is presently defined only for Envisat and GOCE, this function only works if: sat_id=XL_SAT_ENVISAT or sat_id=XL_SAT_GOCE	-
xl_time_time_to_obt	Cannot be called from ForTran (no ForTran equivalence to unsigned long)	-
xl_time_time_to_obt	Since OBT is presently defined only for Envisat and GOCE, this function only works if: sat_id=XL_SAT_ENVISAT or sat_id=XL_SAT_GOCE	-
xl_euler_to_matrix	Functionality is not currently available	-
xl_matrix_to_euler	Functionality is not currently available	-
xl_position_on_orbit	Functionality is not currently available	-
xl_get_rotation_angles	Functionality is not currently available	-
xl_get_rotated_vectors	Functionality is not currently available	-
xl_quaternions_to_vectors	Functionality is not currently available	-
xl_vectors_to_quaternions	Functionality is not currently available	-
xl_time_ref_init_file	XL_SEL_FILE is not supported for time inputs selection (i.e. it is required to specify a time or an orbit range).	

10 APPENDIX: DEFAULT SATELLITE CONFIGURATION FILE.

Following there is an example of a default satellite configuration file. This type of file is needed to define some parameter of the default satellite.

The configuration file must be written in **XML** format. The relevant data are written within the data block tag and are divided into three main parts:

- Satellite name.
- Lib init, where the tolerances are defined.
- Orbit init, whose data are needed to run the **explorer_orbit** functions.
- Pointing init, whose data are needed to run the **explorer_pointing** functions, and therefore needed for the **explorer_visibility** and **explorer_gen_files** CFI functions.

Important note: Some CFI functions within **explorer_gen_files** library use swath definition files. The instrument to which a Swath Definition File (see [G_F_SUM]) belongs, is identified with the first three letters from the *Output_File_Type* parameter. For this reason the first three letters of the instrument name should be equal to the first three letters from the *Output_File_Type* parameter in the Swath Definition File.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Earth_Explorer_File>
  <Common_Header/>
  <Data_Block type="xml">
    <Satellite_Name>Cryosat</Satellite_Name>
    <Lib_init>
      <Low_Tolerances>
        <Min_Semi_Major_Axis>1000000.0</Min_Semi_Major_Axis>
        <Max_Semi_Major_Axis>19300000.0</Max_Semi_Major_Axis>
        <Min_Inclination>60.0</Min_Inclination>
        <Max_Inclination>130.0</Max_Inclination>
        <Min_Eccentricity>0.0</Min_Eccentricity>
        <Max_Eccentricity>0.5</Max_Eccentricity>
      </Low_Tolerances>
      <Tight_Tolerances>
        <Min_Semi_Major_Axis>1118050.0</Min_Semi_Major_Axis>
        <Max_Semi_Major_Axis>19194056.0</Max_Semi_Major_Axis>
        <Min_Inclination>60.4475</Min_Inclination>
        <Max_Inclination>130.6226</Max_Inclination>
        <Min_Eccentricity>0.000</Min_Eccentricity>
        <Max_Eccentricity>0.500</Max_Eccentricity>
      </Tight_Tolerances>
    </Lib_init>
    <Orbit_init>
      <Min_Semi_Major_Axis>7055200.0</Min_Semi_Major_Axis>
      <Nom_Semi_Major_Axis>7096643.0</Nom_Semi_Major_Axis>
      <Max_Semi_Major_Axis>7131206.0</Max_Semi_Major_Axis>
      <Min_Inclination>91.8981</Min_Inclination>
      <Nom_Inclination>92.0000</Nom_Inclination>
      <Max_Inclination>92.0732</Max_Inclination>
```

```
<Nom_Eccentricity>0.0012</Nom_Eccentricity>
<Nom_Arg_Perigee>90.0</Nom_Arg_Perigee>
</Orbit_init>
<Pointing_init>
  <List_of_Attitude_Model count="1">
    <Attitude_Model>
      <Default_YSM>
        <AocsCX>-0.167074</AocsCX>
        <AocsCY>0.050233</AocsCY>
        <AocsCZ>3.912987</AocsCZ>
      </Default_YSM>
    </Attitude_Model>
  </List_of_Attitude_Model>
  <List_of_Instruments count="2">
    <Instrument_Name>MERIS</Instrument_Name>
    <Instrument_Name>ASAR</Instrument_Name>
  </List_of_Instruments>
</Pointing_init>
</Data_Block>
</Earth_Explorer_File>
```