Earth Explorer
Mission CFI Software

EXPLORER_POINTING
SOFTWARE USER MANUAL

Code:	CS-MA-DMS-GS-0005
Issue:	3.0.1
Date:	13/08/04

	Name	Function	Signature
Prepared by:	Fabrizio Pirondini José Antonio González Abeytua	Project Engineer Project Manager	
Checked by:	José Antonio González Abeytua	Project Manager	
Approved by:	José Antonio González Abeytua	Project Manager	

DEIMOS Space S.L.
Ronda de Poniente, 19
Edificio Fiteni VI, Portal 2, 2ª Planta
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	
		Public	
Contract Issuer:	ESA / ESTEC	Industry	X
		Confidential	

External Distribution		
Name	Organisation	Copies

Electronic handling	
Word Processor:	Adobe Framemaker 6.0
Archive Code:	P/SUM/DMS/01/026-024
Electronic file name:	cs-ma-dms-gs-0005-21

Document Status Log

Issue	Change Description	Date	Approval
0.1	First draft version	23/05/02	
1.0	First release	19/07/02	
2.0	Second release	29/11/02	
2.1	Maintenance release	13/05/03	
2.2	Added the following functions: • xp_target_extra_aux • xp_target_extra_target_to_sun • xp_target_extra_ef_to_sat • xp_converter	30/09/03	
3.0	Completely new initialisation strategy and attitude functions	21/07/04	

Table of Contents

1. SCOPE 20

2. ACRONYMS AND NOMENCLATURE 21

2.1. Acronyms 21
2.2. Nomenclature 22

3. APPLICABLE AND REFERENCE DOCUMENTS 23

3.1. Applicable Documents 23
3.2. Reference Documents 23

4. INTRODUCTION 24

4.1. Functions Overview 24
4.1.1. Attitude Data Flow 25
4.1.2. Geolocation Routines Data Flow 27
4.1.2.1. LOS targets 31
4.1.2.2. DEM targets 32

5. LIBRARY INSTALLATION 33

6. LIBRARY USAGE 34

6.1. Usage hints 37
6.2. General Enumerations 37

7. CFI FUNCTIONS DESCRIPTION 44

7.1. xp_sat_nominal_att_init 45
7.1.1. Overview 45
7.1.2. Calling Interface 45
7.1.3. Input Parameters 46
7.1.4. Output Parameters 46
7.1.5. Warnings and Errors 46
7.1.6. Runtime Performances 46
7.2. xp_sat_nominal_att_init_model 48
7.2.1. Overview 48
7.2.2. Calling Interface 48
7.2.3. Input Parameters 49
7.2.3.1. Generic Model description 49
7.2.4. Output Parameters 51
7.2.5. Warnings and Errors 51
7.2.6. Runtime Performances 51
7.3. xp_sat_nominal_att_init_harmonic 52
7.3.1. Overview 52
7.3.2. Calling Interface 52

7.3.3. Input Parameters 54
7.3.4. Output Parameters 55
7.3.5. Example 55
7.3.6. Warnings and Errors 56
7.3.7. Runtime Performances 56
7.4. xp_sat_nominal_att_init_file 57
7.4.1. Overview 57
7.4.2. Calling Interface 57
7.4.3. Input Parameters 58
7.4.4. Output Parameters 59
7.4.5. Warnings and Errors 59
7.4.6. Runtime Performances 60
7.5. xp_sat_nominal_att_close 61
7.5.1. Overview 61
7.5.2. Calling Interface 61
7.5.3. Input Parameters 62
7.5.4. Output Parameters 62
7.5.5. Warnings and Errors 62
7.5.6. Runtime Performances 63
7.6. xp_sat_att_angle_init 64
7.6.1. Overview 64
7.6.2. Calling Interface 64
7.6.3. Input Parameters 65
7.6.4. Output Parameters 65
7.6.5. Warnings and Errors 65
7.6.6. Runtime Performances 66
7.7. xp_sat_att_matrix_init 67
7.7.1. Overview 67
7.7.2. Calling Interface 67
7.7.3. Input Parameters 68
7.7.4. Output Parameters 68
7.7.5. Example 68
7.7.6. Warnings and Errors 68
7.7.7. Runtime Performances 69
7.8. xp_sat_att_init_harmonic 70
7.8.1. Overview 70
7.8.2. Calling Interface 70
7.8.3. Input Parameters 72
7.8.4. Output Parameters 73
7.8.5. Warnings and Errors 73
7.8.6. Runtime Performances 74
7.9. xp_sat_att_init_file 75
7.9.1. Overview 75
7.9.2. Calling Interface 75
7.9.3. Input Parameters 76

7.9.4. Output Parameters 77
7.9.5. Warnings and Errors 77
7.9.6. Runtime Performances 78
7.10. xp_sat_att_close 79
7.10.1. Overview 79
7.10.2. Calling Interface 79
7.10.3. Input Parameters 80
7.10.4. Output Parameters 80
7.10.5. Warnings and Errors 80
7.10.6. Runtime Performances 81
7.11. xp_instr_att_angle_init 82
7.11.1. Overview 82
7.11.2. Calling Interface 82
7.11.3. Input Parameters 83
7.11.4. Output Parameters 83
7.11.5. Warnings and Errors 83
7.11.6. Runtime Performances 84
7.12. xp_instr_att_matrix_init 85
7.12.1. Overview 85
7.12.2. Calling Interface 85
7.12.3. Input Parameters 86
7.12.4. Output Parameters 86
7.12.5. Example 86
7.12.6. Warnings and Errors 86
7.12.7. Runtime Performances 87
7.13. xp_instr_att_init_harmonic 88
7.13.1. Overview 88
7.13.2. Calling Interface 88
7.13.3. Input Parameters 90
7.13.4. Output Parameters 91
7.13.5. Warnings and Errors 91
7.13.6. Runtime Performances 92
7.14. xp_instr_att_init_file 93
7.14.1. Overview 93
7.14.2. Calling Interface 93
7.14.3. Input Parameters 94
7.14.4. Output Parameters 94
7.14.5. Warnings and Errors 95
7.14.6. Runtime Performances 95
7.15. xp_instr_att_close 96
7.15.1. Overview 96
7.15.2. Calling Interface 96
7.15.3. Input Parameters 97
7.15.4. Output Parameters 97
7.15.5. Warnings and Errors 97

7.15.6. Runtime Performances 97
7.16. xp_run_init 99
7.16.1. Overview 99
7.16.2. Calling interface 99
7.16.3. Input parameters 100
7.16.4. Output parameters 100
7.16.5. Warnings and errors 100
7.16.6. Runtime performances 101
7.17. xp_run_get_ids 102
7.17.1. Overview 102
7.17.2. Calling interface 102
7.17.3. Input parameters 103
7.17.4. Output parameters 103
7.17.5. Warnings and errors 103
7.17.6. Runtime performances 103
7.18. xp_run_close 104
7.18.1. Overview 104
7.18.2. Calling interface 104
7.18.3. Input parameters 105
7.18.4. Output parameters 105
7.18.5. Warnings and errors 105
7.18.6. Runtime performances 105
7.19. xp_attitude_init 106
7.19.1. Overview 106
7.19.2. Calling Interface 106
7.19.3. Input Parameters 107
7.19.4. Output Parameters 107
7.19.5. Warnings and Errors 107
7.19.6. Runtime Performances 107
7.20. xp_attitude_compute 108
7.20.1. Overview 108
7.20.2. Calling interface 108
7.20.3. Input parameters 109
7.20.4. Output parameters 110
7.20.5. Warnings and errors 110
7.20.6. Runtime performances 111
7.21. xp_attitude_user_set 112
7.21.1. Overview 112
7.21.2. Calling interface 112
7.21.3. Input parameters 113
7.21.4. Output parameters 114
7.21.5. Warnings and errors 114
7.21.6. Runtime performances 115
7.22. xp_attitude_close 116
7.22.1. Overview 116

7.22.2. Calling Interface 116
7.22.3. Input Parameters 117
7.22.4. Output Parameters 117
7.22.5. Warnings and Errors 117
7.22.6. Runtime Performances 118
7.23. xp_change_frame 119
7.23.1. Overview 119
7.23.2. Calling interface 119
7.23.3. Input parameters 121
7.23.4. Output parameters 122
7.23.5. Warnings and errors 122
7.23.6. Runtime performances 123
7.24. xp_atmos_init 124
7.24.1. Overview 124
7.24.2. Calling Interface 124
7.24.3. Input Parameters 125
7.24.4. Output Parameters 125
7.24.5. Warnings and Errors 125
7.24.6. Runtime Performances 126
7.25. xp_atmos_close 127
7.25.1. Overview 127
7.25.2. Calling Interface 127
7.25.3. Input Parameters 128
7.25.4. Output Parameters 128
7.25.5. Warnings and Errors 128
7.25.6. Runtime Performances 129
7.26. xp_dem_init 130
7.26.1. Overview 130
7.26.2. Calling Interface 130
7.26.3. Input Parameters 131
7.26.4. Output Parameters 131
7.26.5. Warnings and Errors 131
7.26.6. Runtime Performances 132
7.27. xp_dem_close 133
7.27.1. Overview 133
7.27.2. Calling Interface 133
7.27.3. Input Parameters 134
7.27.4. Output Parameters 134
7.27.5. Warnings and Errors 134
7.27.6. Runtime Performances 135
7.28. xp_target_inter 136
7.28.1. Overview 136
7.28.2. Calling Interface 136
7.28.3. Input Parameters 138
7.28.4. Output Parameters 139

7.28.5. Warnings and Errors 139
7.28.6. Runtime Performances 140
7.29. xp_target_ground_range 141
7.29.1. Overview 141
7.29.2. Calling Interface 141
7.29.3. Input Parameters 143
7.29.4. Output Parameters 144
7.29.5. Warnings and Errors 144
7.29.6. Runtime Performances 145
7.30. xp_target_incidence_angle 146
7.30.1. Overview 146
7.30.2. Calling Interface 146
7.30.3. Input Parameters 148
7.30.4. Output Parameters 149
7.30.5. Warnings and Errors 149
7.30.6. Runtime Performances 150
7.31. xp_target_range 151
7.31.1. Overview 151
7.31.2. Calling Interface 151
7.31.3. Input Parameters 153
7.31.4. Output Parameters 154
7.31.5. Warnings and Errors 154
7.31.6. Runtime Performances 155
7.32. xp_target_range_rate 156
7.32.1. Overview 156
7.32.2. Calling Interface 156
7.32.3. Input Parameters 158
7.32.4. Output Parameters 159
7.32.5. Warnings and Errors 159
7.32.6. Runtime Performances 160
7.33. xp_target_tangent 161
7.33.1. Overview 161
7.33.2. Calling Interface 161
7.33.3. Input Parameters 163
7.33.4. Output Parameters 164
7.33.5. Warnings and Errors 164
7.33.6. Runtime Performances 165
7.34. xp_target_altitude 166
7.34.1. Overview 166
7.34.2. Calling Interface 166
7.34.3. Input Parameters 168
7.34.4. Output Parameters 169
7.34.5. Warnings and Errors 169
7.34.6. Runtime Performances 170
7.35. xp_target_star 171

7.35.1. Overview 171
7.35.2. Calling Interface 171
7.35.3. Input Parameters 173
7.35.4. Output Parameters 174
7.35.5. Warnings and Errors 174
7.35.6. Runtime Performances 175
7.36. xp_target_station 176
7.36.1. Overview 176
7.36.2. Calling Interface 176
7.36.3. Input Parameters 178
7.36.4. Output Parameters 179
7.36.5. Warnings and Errors 179
7.36.6. Runtime Performances 180
7.37. xp_target_drs 181
7.37.1. Overview 181
7.37.2. Calling Interface 181
7.37.3. Input Parameters 182
7.37.4. Output Parameters 183
7.37.5. Warnings and Errors 183
7.37.6. Runtime Performances 184
7.38. xp_target_generic 185
7.38.1. Overview 185
7.38.2. Calling Interface 185
7.38.3. Input Parameters 186
7.38.4. Output Parameters 187
7.38.5. Warnings and Errors 187
7.38.6. Runtime Performances 188
7.39. xp_target_travel_time 189
7.39.1. Overview 189
7.39.2. Calling Interface 189
7.39.3. Input Parameters 191
7.39.4. Output Parameters 192
7.39.5. Warnings and Errors 192
7.39.6. Runtime Performances 192
7.40. xp_target_tangent_sun 193
7.40.1. Overview 193
7.40.2. Calling Interface 193
7.40.3. Input Parameters 195
7.40.4. Output Parameters 195
7.40.5. Warnings and Errors 196
7.40.6. Runtime Performances 197
7.41. xp_target_tangent_moon 198
7.41.1. Overview 198
7.41.2. Calling Interface 198
7.41.3. Input Parameters 199

7.41.4. Output Parameters 200
7.41.5. Warnings and Errors 200
7.41.6. Runtime Performances 201
7.42. xp_multi_target_inter 202
7.42.1. Overview 202
7.42.2. Calling Interface 202
7.42.3. Input Parameters 204
7.42.4. Output Parameters 205
7.42.5. Warnings and Errors 205
7.42.6. Runtime Performances 207
7.43. xp_multi_target_travel_time 208
7.43.1. Overview 208
7.43.2. Calling Interface 208
7.43.3. Input Parameters 210
7.43.4. Output Parameters 211
7.43.5. Warnings and Errors 211
7.43.6. Runtime Performances 211
7.44. xp_target_extra_vector 212
7.44.1. Overview 212
7.44.2. Calling Interface 212
7.44.3. Input Parameters 213
7.44.4. Output Parameters 214
7.44.5. Warnings and Errors 215
7.44.6. Runtime Performances 216
7.45. xp_target_extra_main 217
7.45.1. Overview 217
7.45.2. Calling Interface 217
7.45.3. Input Parameters 218
7.45.4. Output Parameters 219
7.45.5. Warnings and Errors 221
7.45.6. Runtime Performances 222
7.46. xp_target_extra_aux 223
7.46.1. Overview 223
7.46.2. Calling Interface 223
7.46.3. Input Parameters 224
7.46.4. Output Parameters 225
7.46.5. Warnings and Errors 227
7.46.6. Runtime Performances 228
7.47. xp_target_extra_ef_target 229
7.47.1. Overview 229
7.47.2. Calling Interface 229
7.47.3. Input Parameters 230
7.47.4. Output Parameters 231
7.47.5. Warnings and Errors 232
7.47.6. Runtime Performances 233

7.48. xp_target_extra_target_to_sun 234
7.48.1. Overview 234
7.48.2. Calling Interface 234
7.48.3. Input Parameters 235
7.48.4. Output Parameters 236
7.48.5. Warnings and Errors 237
7.48.6. Runtime Performances 238
7.49. xp_target_close 239
7.49.1. Overview 239
7.49.2. Calling Interface 239
7.49.3. Input Parameters 240
7.49.4. Output Parameters 240
7.49.5. Warnings and Errors 240
7.49.6. Runtime Performances 240
7.50. xp_converter 241
7.50.1. Overview 241
7.50.2. Calling interface 242
7.50.3. Input parameters 242
7.50.4. Output 243
8. LIBRARY PRECAUTIONS 245
9. KNOWN PROBLEMS 246

List of Tables

Table 1: xp_target functions....28
Table 2: CFI functions included within EXPLORER_POINTING library (TO BE UPDATED) 35
Table 3: Enumerations within EXPLORER_POINTING library....37
Table 4: Input parameters of xp_sat_nominal_att_init function....46
Table 5: Output parameters of xp_sat_nominal_att_init....46
Table 6: Error messages of xp_sat_nominal_att_init function....46
Table 7: Runtime performances of xp_sat_nominal_att_init....47
Table 8: Input parameters of xp_sat_nominal_att_init_model function....49
Table 9: Model parameters depending on the attitude model....49
Table 10: Output parameters of xp_sat_nominal_att_init_model....51
Table 11: Error messages of xp_sat_nominal_att_init_model function....51
Table 12: Runtime performances of xp_sat_nominal_att_init_model....51
Table 13: Input parameters of xp_sat_nominal_att_init_harmonic function....54
Table 14: Output parameters of xp_sat_nominal_att_init_harmonic....55
Table 15: Error messages of xp_sat_nominal_att_init_harmonic function....56
Table 16: Runtime performances of xp_sat_att_nominal_init_harmonic....56
Table 17: Input parameters of xp_sat_nominal_att_init_file function....58
Table 18: Output parameters of xp_sat_nominal_att_init_file....59
Table 19: Error messages of xp_sat_nominal_att_init_file function....59
Table 20: Runtime performances of xp_sat_nominal_att_init_file....60
Table 21: Input parameters of xp_sat_nominal_att_close function....62
Table 22: Output parameters of xp_sat_nominal_att_close....62
Table 23: Error messages of xp_sat_nominal_att_close function....62
Table 24: Runtime performances of xp_sat_nominal_att_close....63
Table 25: Input parameters of xp_sat_att_angle_init function....65
Table 26: Output parameters of xp_sat_att_angle_init....65
Table 27: Error messages of xp_sat_att_angle_init function....65
Table 28: Runtime performances of xp_sat_att_angle_init....66
Table 29: Input parameters of xp_sat_att_matrix_init function....68
Table 30: Output parameters of xp_sat_att_matrix_init....68
Table 31: Error messages of xp_sat_att_matrix_init function....68
Table 32: Runtime performances of xp_sat_att_matrix_init....69
Table 33: Input parameters of xp_sat_att_init_harmonic function....72
Table 34: Output parameters of xp_sat_att_init_harmonic....73
Table 35: Error messages of xp_sat_att_init_harmonic function....74
Table 36: Runtime performances of xp_sat_att_init_harmonic....74

Table 37: Input parameters of xp_sat_att_init_file function 76
Table 38: Output parameters of xp_sat_att_init_file 77
Table 39: Error messages of xp_sat_att_init_file function 77
Table 40: Runtime performances of xp_sat_att_init_file 78
Table 41: Input parameters of xp_sat_att_close function 80
Table 42: Output parameters of xp_sat_att_close 80
Table 43: Error messages of xp_sat_att_close function 80
Table 44: Runtime performances of xp_sat_att_close 81
Table 45: Input parameters of xp_instr_att_angle_init function 83
Table 46: Output parameters of xp_instr_att_angle_init 83
Table 47: Error messages of xp_instr_att_angle_init function 84
Table 48: Runtime performances of xp_instr_att_angle_init 84
Table 49: Input parameters of xp_instr_att_matrix_init function 86
Table 50: Output parameters of xp_instr_att_matrix_init 86
Table 51: Error messages of xp_instr_att_matrix_init function 87
Table 52: Runtime performances of xp_instr_att_matrix_init 87
Table 53: Input parameters of xp_instr_att_init_harmonic function 90
Table 54: Output parameters of xp_instr_att_init_harmonic 91
Table 55: Error messages of xp_instr_att_init_harmonic function 92
Table 56: Runtime performances of xp_instr_att_init_harmonic 92
Table 57: Input parameters of xp_instr_att_init_file function 94
Table 58: Output parameters of xp_instr_att_init_file 94
Table 59: Error messages of xp_instr_att_init_file function 95
Table 60: Runtime performances of xp_instr_att_init_file 95
Table 61: Input parameters of xp_instr_att_close function 97
Table 62: Output parameters of xp_instr_att_close 97
Table 63: Error messages of xp_instr_att_close function 97
Table 64: Runtime performances of xp_instr_att_close 98
Table 65: Input parameters of xp_run_init function 100
Table 66: Output parameters of xp_run_init function 100
Table 67: Error messages of xl_run_init function 101
Table 68: Runtime performances of xp_run_init function 101
Table 69: Input parameters of xp_run_get_ids function 103
Table 70: Output parameters of xp_run_get_ids function 103
Table 71: Runtime performances of xp_run_get_ids function 103
Table 72: Input parameters of xp_run_close function 105
Table 73: Output parameters of xp_run_close function 105
Table 74: Runtime performances of xp_run_close function 105
Table 75: Output parameters of xp_attitude_init 107
Table 76: Error messages of xp_attitude_init function 107

Table 77: Runtime performances of xp_attitude_init 107
Table 78: Input parameters of xp_attitude_compute function 109
Table 79: Output parameters of xp_attitude_compute function 110
Table 80: Error messages of xp_attitude_compute function 110
Table 81: Runtime performances of xp_attitude_compute function 111
Table 82: Input parameters of xp_attitude_user_set function 113
Table 83: Output parameters of xp_attitude_user_set function 114
Table 84: Error messages of xp_attitude_user_set function 114
Table 85: Runtime performances of xp_attitude_user_set function 115
Table 86: Input parameters of xp_attitude_close function 117
Table 87: Output parameters of xp_attitude_close 117
Table 88: Error messages of xp_attitude_close function 117
Table 89: Runtime performances of xp_attitude_close 118
Table 90: Input parameters of xp_change_frame function 121
Table 91: Output parameters of xp_change_frame function 122
Table 92: Error messages of xp_change_frame function 122
Table 93: Runtime performances of xp_change_frame function 123
Table 94: Input parameters of xp_atmos_init function 125
Table 95: Output parameters of xp_atmos_init 125
Table 96: Error messages of xp_atmos_init function 126
Table 97: Runtime performances of xp_atmos_init 126
Table 98: Input parameters of xp_atmos_close function 128
Table 99: Output parameters of xp_atmos_close 128
Table 100: Error messages of xp_atmos_close function 128
Table 101: Runtime performances of xp_atmos_close 129
Table 102: Input parameters of xp_dem_init function 131
Table 103: Output parameters of xp_dem_init 131
Table 104: Error messages of xp_dem_init function 132
Table 105: Runtime performances of xp_dem_init 132
Table 106: Input parameters of xp_dem_close function 134
Table 107: Output parameters of xp_dem_close 134
Table 108: Error messages of xp_dem_close function 134
Table 109: Runtime performances of xp_dem_close 135
Table 110: Input parameters of xp_target_inter function 138
Table 111: Output parameters of xp_target_inter 139
Table 112: Error messages of xp_target_inter function 139
Table 113: Runtime performances of xp_target_inter 140
Table 114: Input parameters of xp_target_ground_range function 143
Table 115: Output parameters of xp_target_ground_range 144
Table 116: Error messages of xp_target_ground_range function 144

Table 117: Runtime performances of xp_target_ground_range 145
Table 118: Input parameters of xp_target_incidence_angle function....148
Table 119: Output parameters of xp_target_incidence_angle 149
Table 120: Error messages of xp_target_incidence_angle function 149
Table 121: Runtime performances of xp_target_incidence_angle....150
Table 122: Input parameters of xp_target_range function 153
Table 123: Output parameters of xp_target_range....154
Table 124: Error messages of xp_target_range function 154
Table 125: Runtime performances of xp_target_range....155
Table 126: Input parameters of xp_target_range_rate function 158
Table 127: Output parameters of xp_target_range_rate....159
Table 128: Error messages of xp_target_range_rate function 159
Table 129: Runtime performances of xp_target_range_rate....160
Table 130: Input parameters of xp_target_tangent function 163
Table 131: Output parameters of xp_target_tangent....164
Table 132: Error messages of xp_target_tangent function 164
Table 133: Runtime performances of xp_target_tangent....165
Table 134: Input parameters of xp_target_altitude function....168
Table 135: Output parameters of xp_target_altitude 169
Table 136: Error messages of xp_target_altitude function 169
Table 137: Runtime performances of xp_target_altitude....170
Table 138: Input parameters of xp_target_star function 173
Table 139: Output parameters of xp_target_star....174
Table 140: Error messages of xp_target_star function 174
Table 141: Runtime performances of xp_target_star....175
Table 142: Input parameters of xp_target_station function 178
Table 143: Output parameters of xp_target_station....179
Table 144: Error messages of xp_target_station function....179
Table 145: Runtime performances of xp_target_station....180
Table 146: Input parameters of xp_target_drs function 182
Table 147: Output parameters of xp_target_drs....183
Table 148: Error messages of xp_target_drs function 183
Table 149: Runtime performances of xp_target_drs....184
Table 150: Input parameters of xp_target_generic function 186
Table 151: Output parameters of xp_target_generic....187
Table 152: Error messages of xp_target_generic function 187
Table 153: Runtime performances of xp_target_generic....188
Table 154: Input parameters of xp_target_travel_time function....191
Table 155: Output parameters of xp_target_travel_time 192
Table 156: Error messages of xp_target_travel_time function 192

Table 157: Runtime performances of xp_target_travel_time 192
Table 158: Input parameters of xp_target_tangent_sun function 195
Table 159: Output parameters of xp_target_tangent_sun 195
Table 160: Error messages of xp_target_tangent_sun function 196
Table 161: Runtime performances of xp_target_tangent_sun 197
Table 162: Input parameters of xp_tangent_target_moon function 199
Table 163: Output parameters of xp_tangent_target_moon....200
Table 164: Error messages of xp_target_tangent_moon function 200
Table 165: Runtime performances of xp_target_tangent_moon....201
Table 166: Input parameters of xp_multi_target_inter function 204
Table 167: Output parameters of xp_multi_target_inter....205
Table 168: Error messages of xp_multi_target_inter function 205
Table 169: Runtime performances of xp_multi_target_inter....207
Table 170: Input parameters of xp_multi_target_travel_time function 210
Table 171: Output parameters of xp_multi_target_travel_time....211
Table 172: Error messages of xp_multi_target_travel_time function....211
Table 173: Runtime performances of xp_multi_target_travel_time....211
Table 174: Input parameters of xp_target_extra_vector function....213
Table 175: Output parameters of xp_target_extra_vector 214
Table 176: Error messages of xp_target_extra_vector function 215
Table 177: Runtime performances of xp_target_extra_vector....216
Table 178: Input parameters of xp_target_extra_main function....218
Table 179: Output parameters of xp_target_extra_main 219
Table 180: Error messages of xp_target_extra_main function 222
Table 181: Runtime performances of xp_target_extra_main....222
Table 182: Output parameters of xp_target_extra_aux....225
Table 183: Error messages of xp_target_extra_aux function 227
Table 184: Runtime performances of xp_target_extra_aux....228
Table 185: Input parameters of xp_target_extra_ef_target function....230
Table 186: Output parameters of xp_target_extra_ef_target 231
Table 187: Error messages of xp_target_extra_ef_target function 232
Table 188: Runtime performances of xp_target_extra_ef_target 233
Table 189: Input parameters of xp_target_extra_target_to_sun function 235
Table 190: Output parameters of xp_target_extra_target_to_sun....236
Table 191: Error messages of xp_target_extra_target_to_sun function....237
Table 192: Runtime performances of xp_target_extra_target_to_sun....238
Table 193: Input parameters of xp_target_close function....240
Table 194: Output parameters of xp_target_close 240
Table 195: Error messages of xp_target_close function 240
Table 196: Runtime performances of xp_target_close 240

Table 197: Input parameters for xp_converter....242
Table 198: iray input vs corrective function. 243
Table 199: Known problems....246

List of Figures

Figure1: Attitude Initialisation Overview 25
Figure2: Satellite Nominal Initialisation 26
Figure3: Satellite Initialisation 26
Figure4: Instrument Initialisation 26
Figure5: Geolocation Routines Calling Sequence 27

1 SCOPE

The EXPLORER_POINTING Software User Manual provides a detailed description of usage of the CFI functions included within the EXPLORER_POINTING 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
CFI	Customer Furnished Item
CS	Coordinate System
DRS	Data Relay Satellite
ESA	European Space Agency
ESTEC	European Space Technology and Research Centre
GPL	GNU Public Library
GPS	Global Positioning System
GS	Ground Station
H/W	Hardware
IERS	International Earth Rotation Service
I/F	Interface
LOS	Line Of Sight
LUT	Look-Up Table
OBT	On-board Binary Time
OSF	Orbit Scenario File
RAM	Random Access Memory
SBT	Satellite Binary Time
SRAR	Satellite Relative Actual Reference
SSP	Sub Satellite Point
SUM	Software User Manual
S/W	Software
TAI	International Atomic Time
UTC	Coordinated Universal Time
UT1	Universal Time UT1
WGS[84]	World Geodetic System 1984

2.2 Nomenclature

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

3 APPLICABLE AND REFERENCE DOCUMENTS

3.1 Applicable Documents

[GEN_SUM] Earth Explorer Mission CFI Software. General Software User Manual. CS-MA-DMS-GS-0002. Issue 3.0. 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.

[LIB_SUM] Earth Explorer Mission CFI Software. EXPLORER_LIB Software User Manual. CS-MA-DMS-GS-0003. Issue 3.0. 21/07/04.

4 INTRODUCTION

4.1 Functions Overview

This software library contains the CFI functions required to perform accurate computation of pointing parameters from and to a satellite for various types of targets.

It includes a set of functions to initialize the attitude of the platform and the instruments. The values provided by these functions are later used by all the other functions of the library.

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 usage and the error handling functions

4.1.1 Attitude Data Flow

The following figure shows the tipical data flow for the attitude functions. First, the different transformations between the various reference frames are initialised. Then, given the spacecraft position, the attitude is calculated:

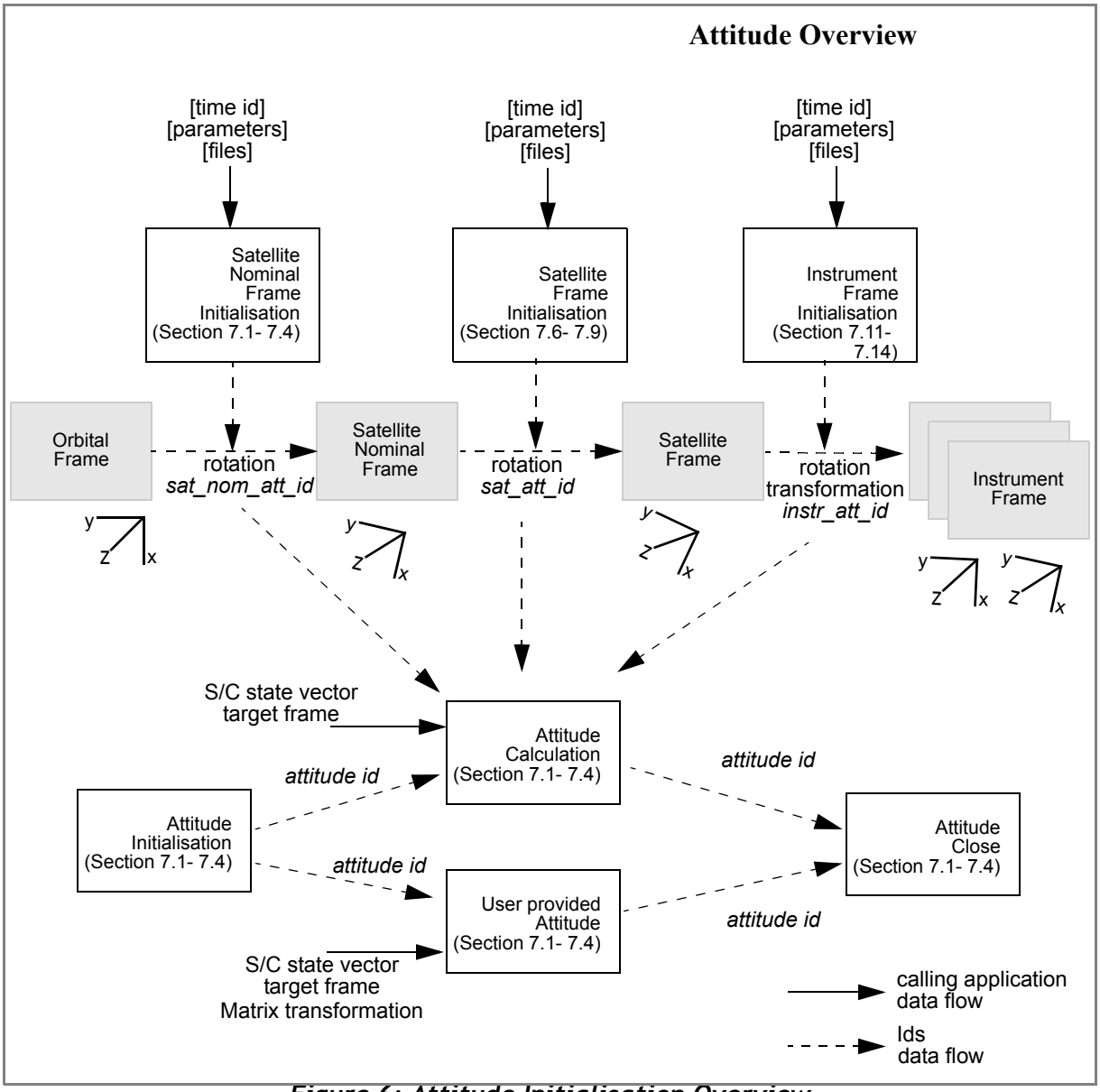


Figure 6: Attitude Initialisation Overview

Each different transformation can be initialised with different models:

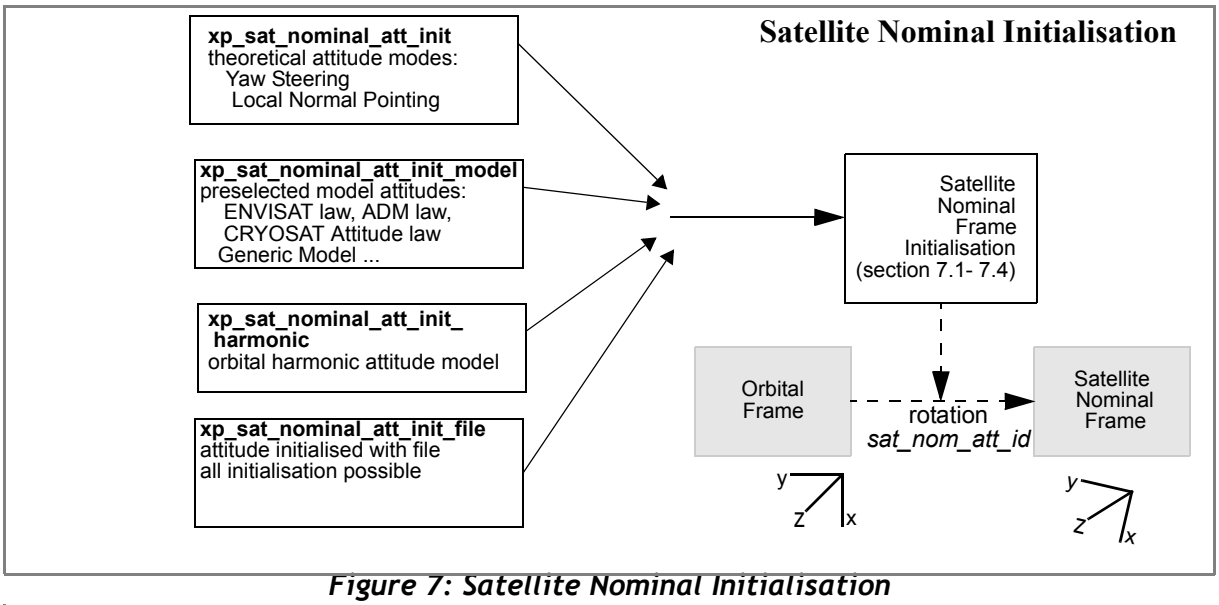


Figure 7: Satellite Nominal Initialisation

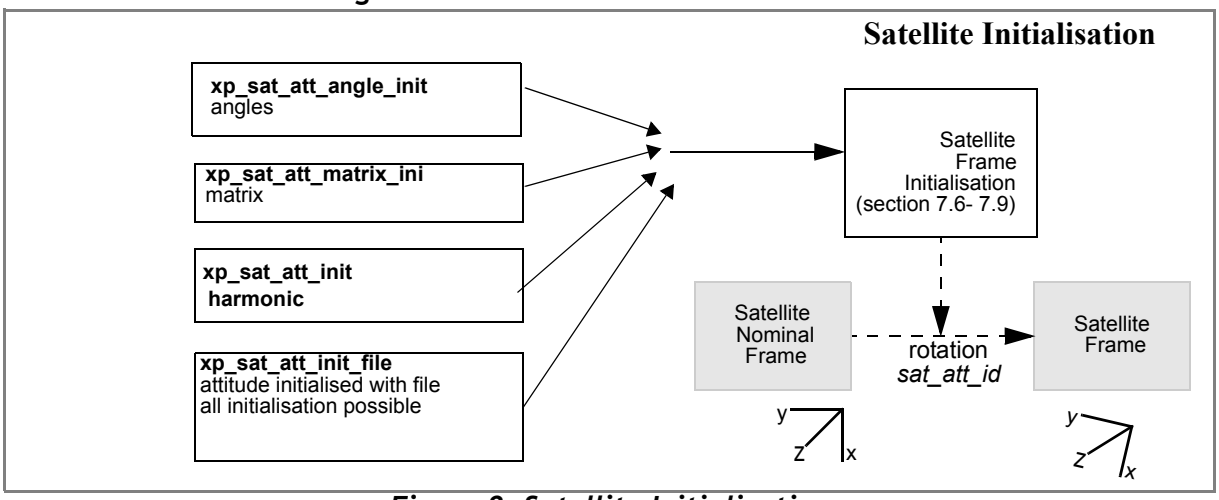


Figure 8: Satellite Initialisation

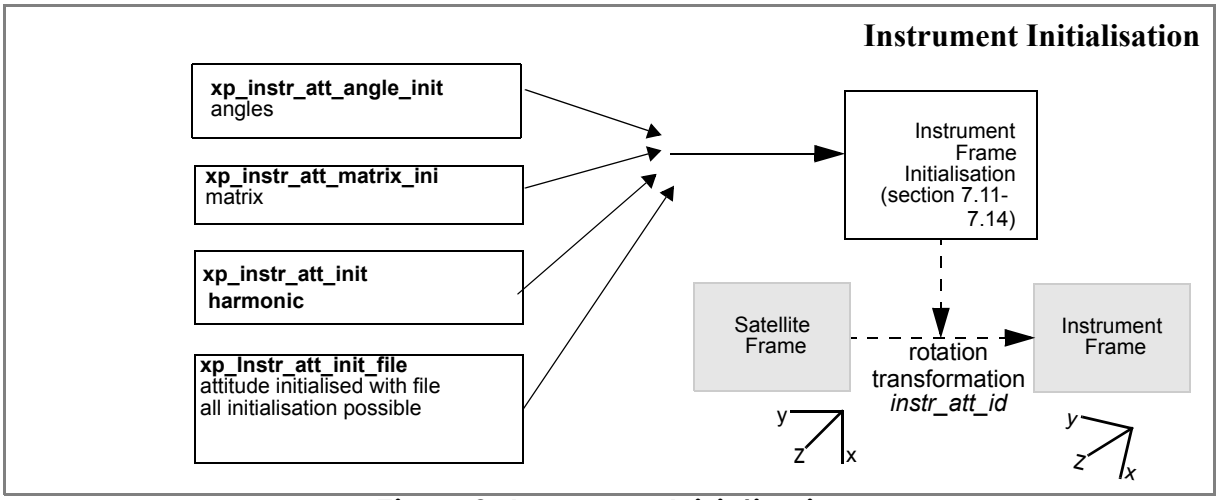


Figure 9: Instrument Initialisation

4.1.2 Geolocation Routines Data Flow

The following figure shows the tipical data flow for the geolocation routines functions. First, the attitude should be calculated, and, if needed, the refraction and Digital Elevation Models initialised.

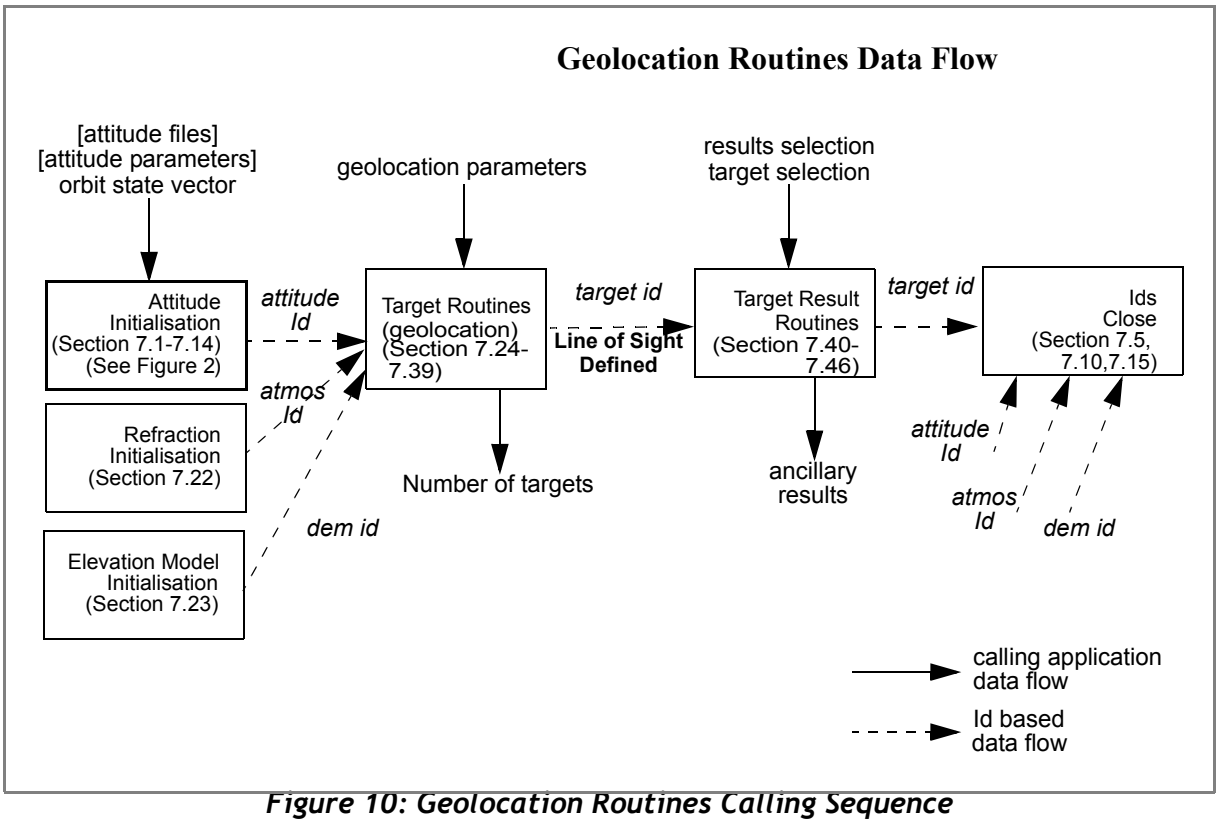
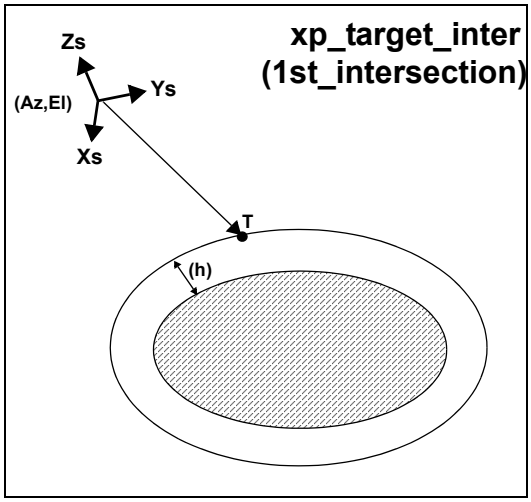


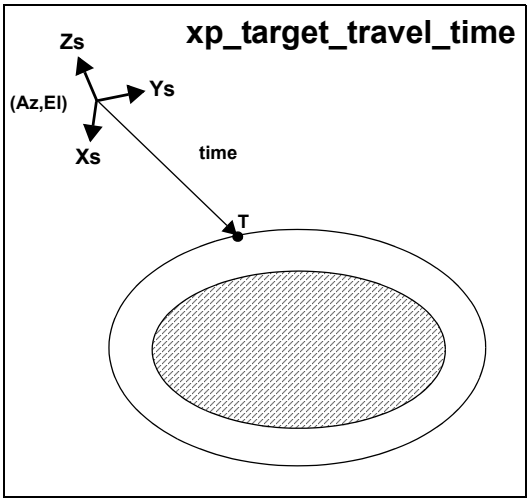
Figure 10: Geolocation Routines Calling Sequence

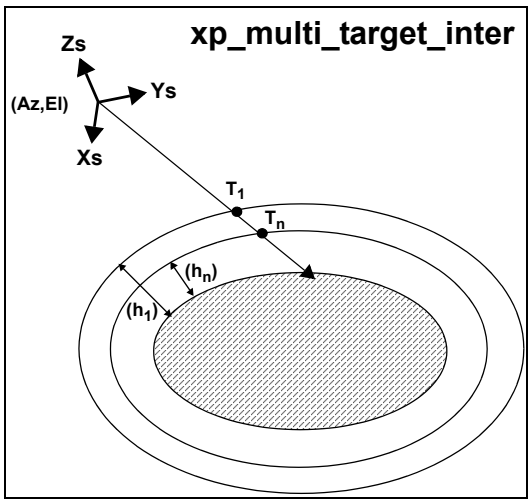
The table below and the diagrams on the next pages describe the various **xp_target_<function>.**

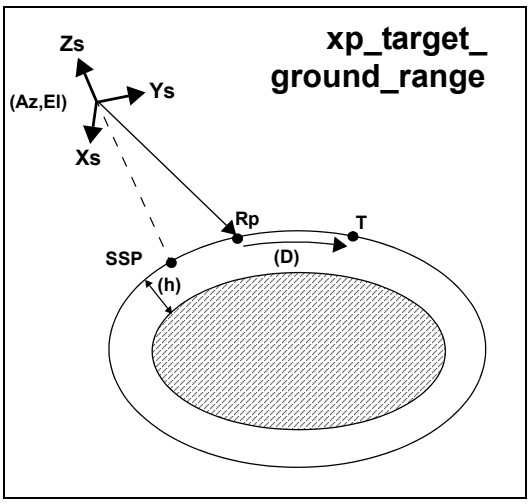
xp_target_ <function>	Description
xp_(multi)_target_ inter	It calculates the intersection point(s) of the line of sight defined by an elevation and an azimuth angle expressed in the input Attitude frame, with a surface(s) located at a certain geodetic altitude(s) over the Earth.
xp_(multi)_target_ travel_time	It calculates the point of the line or sight from the satellite (defined by an elevation and an azimuth angle expressed in the selected Attitude Frame) at a given travel time(s) along the (curved) line of sight.
xp_target_ground _range	It calculates the location of a point that is placed on a surface at a certain geodetic altitude over the Earth, that lays on the plane defined by theS/C position, the nadir and a reference point, and that is at a certain distance or ground range measured along that surface from that reference point. This reference point is calculated being the intersection of the previous surface with the line of sight defined by an elevation and azimuth angle in the input Attitude coordinate system.
xp_target_inciden ce_angle	It calculates the location of a point that is placed on a surface at a certain geodetic altitude over the Earth and that is seen from the S/C on a line of sight that forms a certain azimuth angle in the input Attitude frame and that intersects that surface with a certain incidence angle.
xp_target_range	It calculates the location of a point that is placed on a surface at a certain geodetic altitude over the Earth, that is seen from the S/C on a line of sight that forms a certain azimuth angle in the input Attitude frame, and that is at a certain range or slant-range from the S/C.
xp_target_range_r ate	It calculates the location of a point that is placed on a surface at a certain geodetic altitude over the Earth, that is at a certain range from S/C, and whose associated Earth-fixed target has a certain range-rate value.
xp_target_tangent	It calculates the location of the tangent point over the Earth that is located on the line of sight defined by an elevation and azimuth angles expressed in the input Attitude frame.
xp_target_altitude	It calculates the location of the tangent point over the Earth that is located on a surface at a certain geodetic altitude over the Earth and that is on a line of sight that forms a certain azimuth angle in the input Attitude frame.
xp_target_star	It calculates the location of the tangent point over the Earth that is located on the line of sight that points to a star defined by its right ascension and declination coordinates.
xp_target_generic	The cartesian state vector of the target is taken as an input.
xp_target_tangent _sun	It calculates the location of the tangent point over the Earth that is located on the line of sight that points to the Sun
xp_target_tangent _moon	It calculates the location of the tangent point over the Earth that is located on the line of sight that points to the Moon
xp_target_station	It calculates the most relevant observation parameters of the link between the satellite and a ground station
xp_target_drs	It calculates the most relevant observation parameters of the link between the satellite and a Data Relay Satellite (DRS).

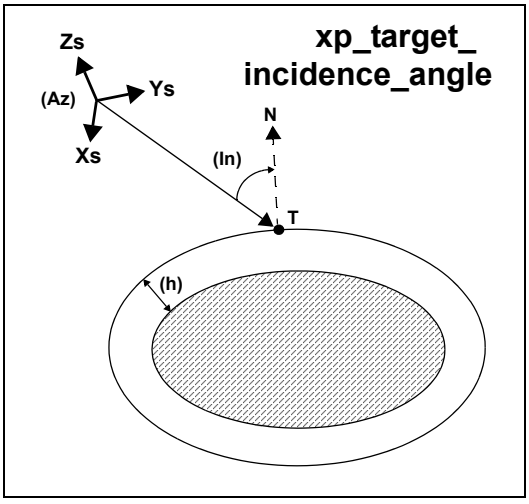
Table 1: xp_target functions.

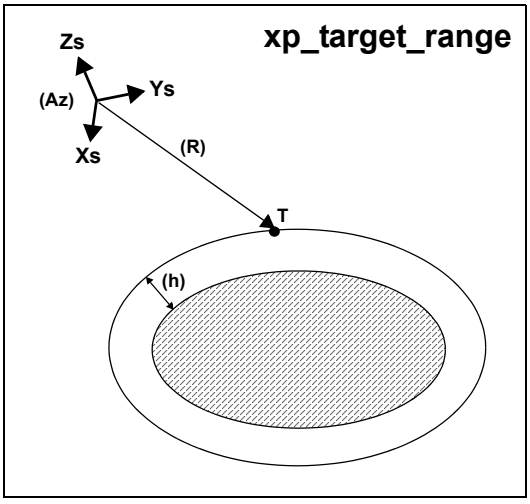

xp_target_inter
(1st_intersection)
Zs
(Az,El)
Ys
Xs
T
(h)

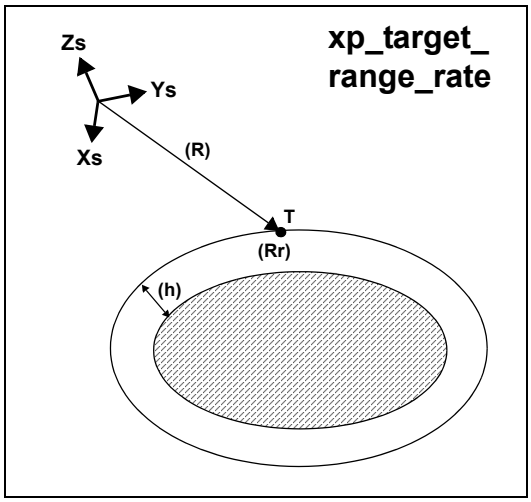

xp_target_travel_time
Zs
(Az,El)
Ys
Xs
time
T

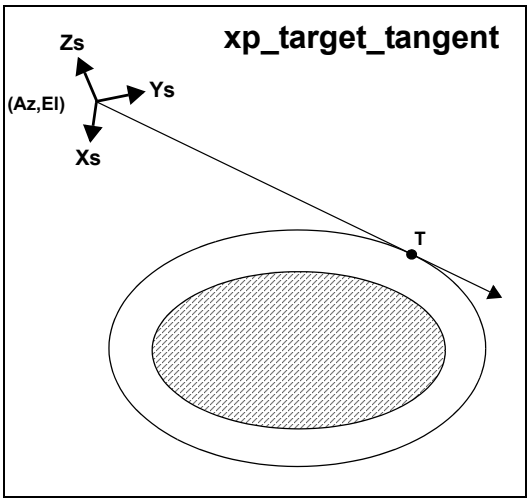

xp_multi_target_inter
Zs
(Az,El)
Ys
Xs
T1
Tn
(hn)
(h1)

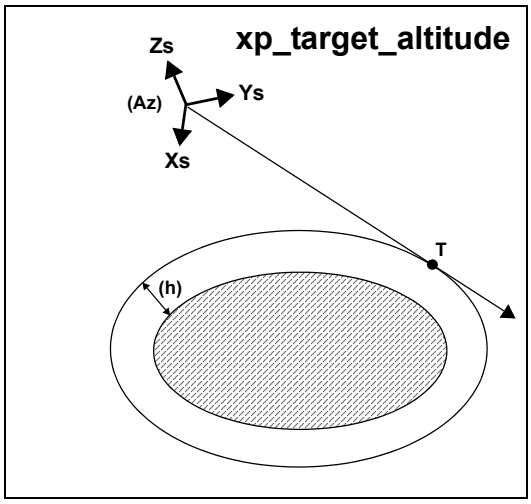

xp_target_
ground_range
Zs
(Az,El)
Ys
Xs
Rp
T
SSP
(D)
(h)

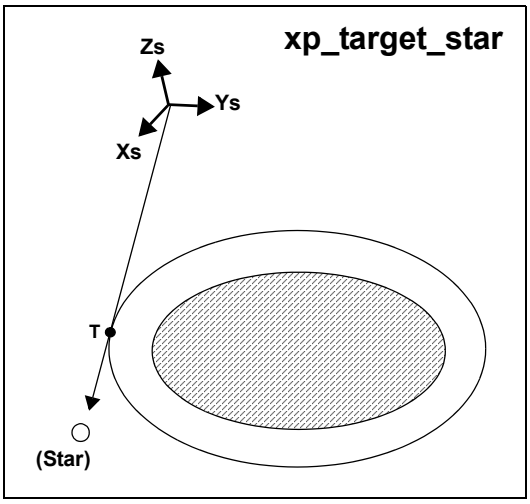

xp_target_
incidence_angle
Zs
(Az)
Ys
Xs
N
(In)
T
(h)

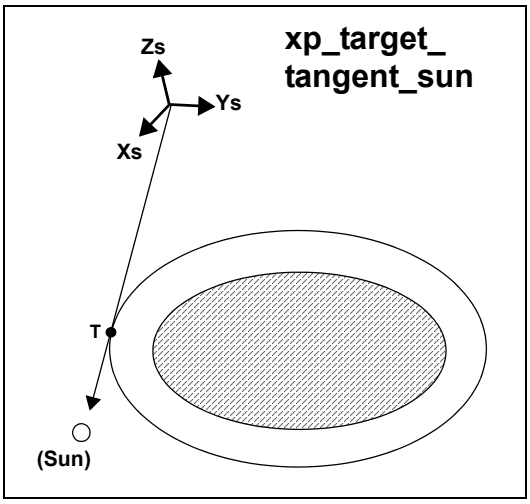

xp_target_range
Zs
(Az)
Ys
Xs
(R)
T
(h)

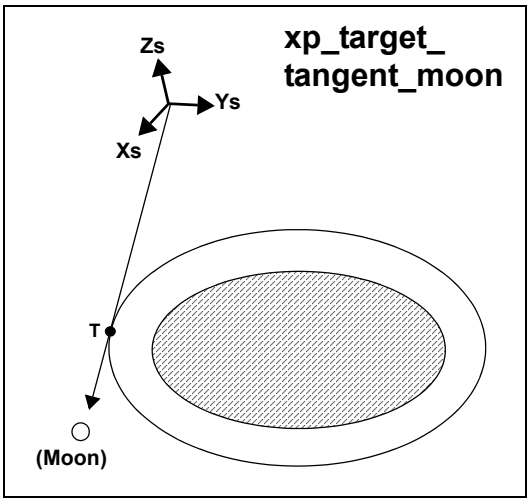

xp_target_
range_rate
Zs
Ys
Xs
(R)
T
(Rr)
(h)

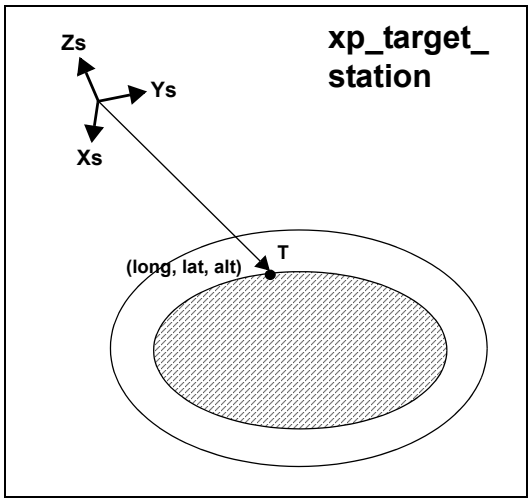

xp_target_tangent
Zs
(Az,El)
Ys
Xs
T

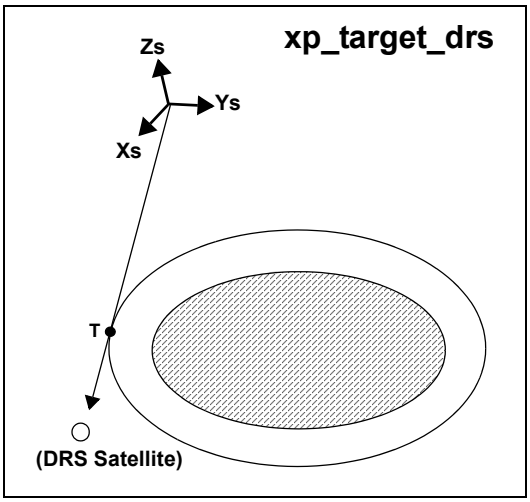

xp_target_altitude
Zs
(Az)
Ys
Xs
T
(h)


xp_target_star
Zs
Ys
Xs
T
(Star)


xp_target_
tangent_sun
Zs
Ys
Xs
T
(Sun)


xp_target_
tangent_moon
Zs
Ys
Xs
T
(Moon)





[Xs,Ys,Zs] = Attitude Frame

() = Input data to the mode

(Az,El) = Azimuth + Elevation of the LOS
(h) = Geodetic altitude of the target
(R) = Range Satellite ⇔ Reference Point/Target
(D) = Distance or Ground range Ref. Point ⇔ Target
(In) = Incidence angle of the LOS
(Rr) = Range-rate of the Earth-fixed target

T = Target
SSP = Sub Satellite Point = Nadir of the satellite
Rp = Reference Point

N = Normal vector to the surface at a geodetic altitude = h

As it can be seen from the list of functions, there are some functions that calculate several targets (xp_multi_target_xxxx). The number of targets found by the functions is returned through the interface.

In addition to these "user" targets, two other categories of targets can be defined, "LOS" targets and "DEM" targets.

4.1.2.1 LOS targets

The idea is to get information about all the ray path points computed by a specific target routine along the Line of Sight (LOS) trajectory.

For every target routine, the output parameter num_los_target will return the number of points in the path.

It applies when the variable "target_type" is equal to XP_LOS_TARGET_TYPE.

1. Start point of LOS

The spacecraft position (Instrument CS) shall be considered as the start point for the LOS path.

2. Stop point of LOS

The stop point for the LOS path will be different depending on the selected target function; nominally it will be the resulting target point.

- xp_target_inter and xp_multi_target_inter: 1st or 2nd intersection point (Point corresponding to the last altitude for the multi_target routine)
- xp_target_ground_range: Target point
- xp_target_incidence_angle: Target point
- xp_target_range: Target point
- xp_target_range_rate: Target point
- p_target_tangent: Two different cases to consider depending on whether refraction is selected or not:
  - No refraction mode: Tangent point
  - Refraction mode:
    - The 2nd intersection point with a surface located at Refraction Model Maximum Height (geodetic altitude) over the Earth if tangent height <= Refraction Model Maximum Height
    - The tangent point if tangent height > Refraction Model Maximum Height
- xp_target_altitude: Point at selected altitude
- xp_target_star: Two different cases to consider depending on whether refraction is selected or not:
  - No refraction mode: Tangent point
  - Refraction mode:
    - The 2nd intersection point with a surface located at Refraction Model Maximum Height (geodetic altitude) over the Earth if tangent height <= Refraction Model Maximum Height
    - The tangent point if tangent height > Refraction Model Maximum Height
- xp_target_station: Ground Station position
- xp_target_drs: DRS position
- xp_target_generic: Target position
- xp_target_travel_time and xp_multi_target_travel_time: Point at selected travel time (Point corresponding to the last travel time for the multi_target routine)
- xp_target_tangent_sun: Tangent point
- xp_target_tangent_moon: Tangent point

4.1.2.2 DEM targets

TBW

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_POINTING software library, the following other CFI software libraries are required:

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

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

- `LIBXML2` (See [GEN_SUM]).

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

- `explorer_pointing.h` (for a C application)
- `explorer_pointing.inc` (for a ForTran application under SOLARIS)
- `explorer_pointing_win.inc` (for a ForTran application under Windows 9X/NT/2000)

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

- SOLARIS/LINUX:

```
-Icfi_include_dir -Lcfi_lib_dir -lexplorer_pointing
                                -lexplorer_lib
                                -lexplorer_file_handling
                                -lxml2
```

- WINDOWS:

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

- MacOS:

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

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

To avoid problems in linking a user application with the `EXPLORER_POINTING` 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 `XP_` or `xp_`.

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

Table 2: CFI functions included within EXPLORER_POINTING library (TO BE UPDATED)

Function Name	Enumeration value	Long
Main CFI Functions		
xp_sat_nominal_att_init	XP_SAT_NOMINAL_ATT_INIT_ID	0
xp_sat_nominal_att_init_model	XP_SAT_NOMINAL_ATT_INIT_MODEL_ID	1
xp_sat_nominal_att_init_harmonic	XP_SAT_NOMINAL_ATT_INIT_HARMONIC_ID	2
xp_sat_nominal_att_init_file	XP_SAT_NOMINAL_ATT_INIT_FILE_ID	3
xp_sat_nominal_att_close	XP_SAT_NOMINAL_ATT_CLOSE_ID	4
xp_sat_att_angle_init	XP_SAT_ATT_ANGLE_INIT_ID	5
xp_sat_att_matrix_init	XP_SAT_ATT_MATRIX_INIT_ID	6
xp_sat_att_init_harmonic	XP_SAT_ATT_INIT_HARMONIC_ID	7
xp_sat_att_init_file	XP_SAT_ATT_INIT_FILE_ID	8
xp_sat_att_close	XP_SAT_ATT_CLOSE_ID	9
xp_instr_att_angle_init	XP_INSTR_ATT_ANGLE_INIT_ID	10
xp_instr_att_matrix_init	XP_INSTR_ATT_MATRIX_INIT_ID	11
xp_instr_att_init_harmonic	XP_INSTR_ATT_INIT_HARMONIC_ID	12
xp_instr_att_init_file	XP_INSTR_ATT_INIT_FILE_ID	13
xp_instr_att_close	XP_INSTR_ATT_CLOSE_ID	14
xp_change_frame	XP_CHANGE_FRAME_ID	15
xp_attitude_init	XP_ATTITUDE_INIT_ID	16
xp_attitude_compute	XP_ATTITUDE_COMPUTE_ID	17
xp_attitude_user_set	XP_ATTITUDE_USER_SET_ID	18
xp_attitude_close	XP_ATTITUDE_CLOSE_ID	19
xp_atmos_init	XP_ATMOS_INIT_ID	20
xp_atmos_close	XP_ATMOS_CLOSE_ID	21
xp_dem_init	XP_DEM_INIT_ID	22
xp_atmos_close	XP_DEM_CLOSE_ID	23
xp_target_inter	XP_TARGET_INTER_ID	24
xp_target_travel_time	XP_TARGET_TRAVEL_TIME_ID	25
xp_target_ground_range	XP_TARGET_GROUND_RANGE_ID	26
xp_target_incidence_angle	XP_TARGET_INCIDENCE_ANGLE_ID	27
xp_target_range	XP_TARGET_RANGE_ID	28

Function Name	Enumeration value	Long
xp_target_range_rate	XP_TARGET_RANGE_RATE_ID	29
xp_target_tangent	XP_TARGET_TANGENT_ID	30
xp_target_altitude	XP_TARGET_ALTITUDE_ID	31
xp_target_star	XP_TARGET_STAR_ID	32
xp_target_station	XP_TARGET_STATION_ID	33
xp_target_drs	XP_TARGET_DRS_ID	34
xp_target_generic	XP_TARGET_GENERIC_ID	35
xp_multi_target_inter	XP_MULTI_TARGET_INTER_ID	36
xp_multi_target_travel_time	XP_MULTI_TARGET_TRAVEL_TIME_ID	37
xp_target_extra_vector	XP_TARGET_EXTRA_VECTOR_ID	38
xp_target_extra_main	XP_TARGET_EXTRA_MAIN_ID	39
xp_target_extra_aux	XP_TARGET_EXTRA_AUX_ID	40
xp_target_extra_ef_target	XP_TARGET_EXTRA_EF_TARGET_ID	41
xp_target_extra_target_to_sun	XP_TARGET_EXTRA_TARGET_TO_SUN_ID	42
xp_target_tangent_sun	XP_TARGET_TANGENT_SUN_ID	43
xp_target_tangent_moon	XP_TARGET_TANGENT_MOON_ID	44
xp_target_close	XP_TARGET_CLOSE_ID	45
xp_run_init	XP_RUN_INIT_ID	46
Error Handling Functions		
xp_verbose	not applicable	
xp_silent		
xp_get_code		
xp_get_msg		
xp_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 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 some of the CFI functions are improved to a large extent if they are called two consecutive times keeping constant some of their inputs.

Nevertheless, although the user may not need to call the CFI functions two consecutive times with the same inputs, there are internal functions that are actually called in those conditions, and thus improving the runtime performances of the former.

Thus, the runtime improvement is achieved with any sequence of calls to those CFI functions, not only with a sequence of calls to the same function.

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

- Time: 0.0864 microsec
- Position vector: 0.6e-3 m
- Velocity vector: 0.6e-6 m/s
- Acceleration vector: 0.6e-9 m/s^2
- AOCS: 5e-9 deg
- Mispointing angles: 5e-9 deg
- Mispointing angles-rate: 5e-12 deg
- Mispointing angles-rate-rate: 5e-15 deg

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 (XP_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_POINTING routines, as shown in the table below. The enumerations presented in [GEN_SUM], [F_H_SUM] and [LIB_SUM] are also applicable.

Table 3: Enumerations within EXPLORER_POINTING library

Input	Description	Enumeration value	Long
Time Initialization Mode	Initialization from file (data-driven)	XP_SEL_FILE	0
	Initialization within a time range	XP_SEL_TIME	1
	Initialization within a range of orbits	XP_SEL_ORBIT	2
	(not used in POINTING)	XP_SEL_DEFAULT	3
Atmosphere Initialization Mode	User's refraction ray tracing model	XP_USER_INIT	1
	User's predefined refraction LUTs	XP_LUT_INIT	2
	Complex atmospheric model	XP_COMPLEX_INIT	3
Earth Intersection	No intersection with Earth geoid	XP_NO_INTER	0

Table 3: Enumerations within EXPLORER_POINTING library

Input	Description	Enumeration value	Long
Mode	First intersection with Earth geoid	XP_INTER_1ST	1
	Second intersection with Earth geoid	XP_INTER_2ND	2
AOCS mode	Geocentric pointing	XP_AOCS_GPM	0
	Local normal pointing	XP_AOCS_LNP	1
	Yaw steering + local normal pointing	XP_AOCS_YSM	2
Satellite Nominal Attitude Model	Generic model	XP_MODEL_GENERIC	0
	Envisat model	XP_MODEL_ENVISAT	1
	Cryosat model	XP_MODEL_CRYOSAT	2
	ADM model	XP_MODEL_ADM	3
Axis enumeration	X axis	XP_X_AXIS	0
	-X axis	XP_NEG_X_AXIS	1
	Y axis	XP_Y_AXIS	2
	-Y axis	XP_NEG_Y_AXIS	3
	Z axis	XP_Z_AXIS	4
	-Z axis	XP_NEG_Z_AXIS	5
Axis target	Sun pointing	XP_SUN_VEC	0
	Moon pointing	XP_MOON_VEC	1
	Nadir pointing	XP_NADIR_VEC	2
	Inertial velocity pointing	XP_INERTIAL_VEL_VEC	3
	Earth Fixed velocity pointing	XP_EF_VEL_VEC	4
	Inertial target pointing	XP_INERTIAL_TARGET_VEC	5
	Earth Fixed target pointing	XP_EF_TARGET_VEC	6
Angle Type	True Latitude (TOD)	XP_ANGLE_TYPE_TRUE_LAT_TOD	1
	Mean Latitude (TOD)	XP_ANGLE_TYPE_MEAN_LAT_TOD	2
Attittude Frame ID	Satellite Orbital Reference Frame	XP_SAT_ORBITAL_REF	0
	Satellite Nominal Attitude Frame	XP_SAT_NOMINAL_ATT	1
	Satellite Attitude Frame	XP_SAT_ATT	2
	Instrument(s) Attitude Frame(s)	XP_INSTR_ATT	3
Target Type	User Target	XP_USER_TARGET_TYPE	0
	Line of Sight Target	XP_LOS_TARGET_TYPE	1
	DEM Target	XP_DEM_TARGET_TYPE	2

Table 3: Enumerations within EXPLORER_POINTING library

Input	Description	Enumeration value	Long
Ray tracing model		XP_NO_REF	0
		XP_STD_REF	1
		XP_USER_REF	2
		XP_PRED_REF	3
		XP_STD_REF_N	10
		XP_USER_REF_N	20
		XP_PRED_REF_N	30
		XP_US76_REF	300
		XP_TROPIC_REF	301
		XP_MID_SUM_REF	302
		XP_MID_WIN_REF	303
		XP_SUBAR_SUM_REF	304
		XP_SUBAR_WIN_REF	305
		XP_LUT_REF	400
		XP_US76_REF_N	3000
		XP_TROPIC_REF_N	3001
		XP_MID_SUM_REF_N	3002
		XP_MID_WIN_REF_N	3003
		XP_SUBAR_SUM_REF_N	3004
		XP_SUBAR_WIN_REF_N	3005
		XP_LUT_REF_N	4000

Table 3: Enumerations within EXPLORER_POINTING library

Input	Description	Enumeration value	Long
Target extra main results choice	Geocentric longitude and latitude. Geodetic altitude and latitude.	XP_TARG_EXTRA_MAIN_GEO	1
	Geocentric longitude and latitude rates. Geodetic altitude and latitude rates.	XP_TARG_EXTRA_MAIN_GEO_D	2
	Geocentric longitude and latitude rate rates. Geodetic altitude and latitude rate rates.	XP_TARG_EXTRA_MAIN_GEO_2D	4
	Target to satellite azimuth and elevation (Topocentric CS)	XP_TARG_EXTRA_MAIN_TARG2SAT_TOP	8
	Target to satellite azimuth and elevation rates (Topocentric CS)	XP_TARG_EXTRA_MAIN_TARG2SAT_TOP_D	16
	Target to satellite azimuth and elevation rate rates (Topocentric CS)	XP_TARG_EXTRA_MAIN_TARG2SAT_TOP_2D	32
	Satellite to target azimuth and elevation (Topocentric CS)	XP_TARG_EXTRA_MAIN_SAT2TARG_TOP	64
	Satellite to target azimuth and elevation rates (Topocentric CS)	XP_TARG_EXTRA_MAIN_SAT2TARG_TOP_D	128
	Satellite to target azimuth and elevation rate rates (Topocentric CS)	XP_TARG_EXTRA_MAIN_SAT2TARG_TOP_2D	256
	Satellite to target azimuth and elevation (Attitude Frame)	XP_TARG_EXTRA_MAIN_SAT2TARG_ATTITUDE	512
	Satellite to target azimuth and elevation rates (Attitude Frame)	XP_TARG_EXTRA_MAIN_SAT2TARG_ATTITUDE_D	1024
	Satellite to target azimuth and elevation rate rates (Attitude Frame)	XP_TARG_EXTRA_MAIN_SAT2TARG_ATTITUDE_2D	2048
	All parameters	XP_TARG_EXTRA_MAIN_ALL	4095

Table 3: Enumerations within EXPLORER_POINTING library

Input	Description	Enumeration value	Long
Target extra aux results choice	Minimum distance from the nadir of the target to the ground track.	XP_TARG_EXTRA_AUX_DIST_NAD_TARG_GT	1
	Radius of curvature in the look direction at the nadir of the target.	XP_TARG_EXTRA_AUX_RAD_CUR	2
	Minimum distance rate from the nadir of the target to the ground track.	XP_TARG_EXTRA_AUX_DIST_NAD_TARG_GT_D	4
	Minimum distance rate rate from the nadir of the target to the ground track.	XP_TARG_EXTRA_AUX_DIST_NAD_TARG_GT_2D	8
	Radius of curvature rate in the look direction at the nadir of the target.	XP_TARG_EXTRA_AUX_RAD_CUR_D	16
	Radius of curvature rate rate in the look direction at the nadir of the target.	XP_TARG_EXTRA_AUX_RAD_CUR_2D	32
	Target Nadir Velocity relative to the Earth. (Topocentric CS)	XP_TARG_EXTRA_AUX_TARGET_NADIR_VEL	64
	Mean Local Solar Time at target.	XP_TARG_EXTRA_AUX_MLST	128
	True Local Solar Time at target.	XP_TARG_EXTRA_AUX_TLST	256
	Distance from the nadir of the target to the satellite nadir (Earth fixed CS)	XP_TARG_EXTRA_AUX_DIST_NAD_TARG_SAT_NAD	512
	Distance rate from the nadir of the target to the satellite nadir (Earth fixed CS)	XP_TARG_EXTRA_AUX_DIST_NAD_TARG_SAT_NAD_D	1024
	Distance rate rate from the nadir of the target to the satellite nadir (Earth fixed CS)	XP_TARG_EXTRA_AUX_DIST_NAD_TARG_SAT_NAD_2D	2048
	R.A. and declination at which the look direction from the satellite to the target point after crossing the atmosphere.	XP_TARG_EXTRA_AUX_LOOK_DIR	4096
	Distance from the SSP to the point on the ground track nearest to the nadir of the target. (Earth fixed CS)	XP_TARG_EXTRA_AUX_DIST_SSP_MIN_DIST_GT	8192
	Distance rate from the SSP to the point on the ground track nearest to the nadir of the target. (Earth fixed CS)	XP_TARG_EXTRA_AUX_DIST_SSP_MIN_DIST_GT_D	16384
	Distance rate rate from the SSP to the point on the ground track nearest to the nadir of the target. (Earth fixed CS)	XP_TARG_EXTRA_AUX_DIST_SSP_MIN_DIST_GT_2D	32768
	All parameters	XP_TARG_EXTRA_AUX_ALL	65535

Table 3: Enumerations within EXPLORER_POINTING library

Input	Description	Enumeration value	Long
Satellite Nominal Attitude Mode	Satellite Nominal Attitude initialised with AOCS mode	XP_SAT_NOMINAL_ATT_INIT_MODE	0
	Satellite Nominal Attitude initialised with Model	XP_SAT_NOMINAL_ATT_INIT_MODEL_MODE	1
	Satellite Nominal Attitude initialised with Harmonics	XP_SAT_NOMINAL_ATT_INIT_HARMONIC_MODE	2
	Satellite Nominal Attitude initialised with a File	XP_SAT_NOMINAL_ATT_INIT_FILE_MODE	3
Satellite Attitude Mode	Satellite Attitude initialised with angles	XP_SAT_ATT_ANGLE_INIT_MODE	0
	Satellite Attitude initialised with matrices	XP_SAT_ATT_MATRIX_INIT_MODE	1
	Satellite Attitude initialised with Harmonics	XP_SAT_ATT_INIT_HARMONIC_MODE	2
	Satellite Attitude initialised with a File	XP_SAT_ATT_INIT_FILE_MODE	3
Instrument Attitude Mode	Instrument Attitude initialised with angles	XP_INSTR_ATT_ANGLE_INIT_MODE	0
	Instrument Attitude initialised with matrices	XP_INSTR_ATT_MATRIX_INIT_MODE	1
	Instrument Attitude initialised with Harmonics	XP_INSTR_ATT_INIT_HARMONIC_MODE	2
	Instrument Attitude initialised with a File	XP_INSTR_ATT_INIT_FILE_MODE	3
Attitude Mode	Attitude not calculated	XP_ATTITUDE_INIT_NO_DATA_MODE	0
	Attitude calculated	XP_ATTITUDE_COMPUTE_MODE	1
	Attitude defined by the user	XP_ATTITUDE_USER_SET_MODE	2

Table 3: Enumerations within EXPLORER_POINTING library

Input	Description	Enumeration value	Long
Target Mode	Target calculated with Inter (1st) function	XP_TARGET_INTER_1ST_MODE	0
	Target calculated with Inter (2nd) function	XP_TARGET_INTER_2ND_MODE	1
	Target calculated with Travel Time (1st) function	XP_TARGET_TRAVEL_TIME_1ST_MODE	2
	Target calculated with Travel Time (2nd) function	XP_TARGET_TRAVEL_TIME_2ND_MODE	3
	Target calculated with Ground Range function	XP_TARGET_GROUND_RANGE_MODE	4
	Target calculated with Incidence Angle function	XP_TARGET_INCIDENCE_ANGLE_MODE	5
	Target calculated with Range function	XP_TARGET_RANGE_MODE	6
	Target calculated with Range Rate function	XP_TARGET_RANGE_RATE_MODE	7
	Target calculated with Tangent function	XP_TARGET_TANGENT_MODE	8
	Target calculated with Altitude function	XP_TARGET_ALTITUDE_MODE	9
	Target calculated with Star function	XP_TARGET_STAR_MODE	10
	Target calculated with Tangent to Sun function	XP_TARGET_TANGENT_SUN_MODE	11
	Target calculated with Tangent to Moon function	XP_TARGET_TANGENT_MOON_MODE	12
	Target calculated with Station function	XP_TARGET_STATION_MODE	13
	Target calculated with DRS function	XP_TARGET_DRS_MODE	14
	Target calculated with Generic function	XP_TARGET_GENERIC_MODE	15
	Target calculated with Multi Inter (1st) function	XP_MULTI_TARGET_INTER_1ST_MODE	16
	Target calculated with Multi Inter (2nd) function	XP_MULTI_TARGET_INTER_2ND_MODE	17
	Target calculated with Multi Travel Time (1st) function	XP_MULTI_TARGET_TRAVEL_TIME_1ST_MODE	18
	Target calculated with Multi Travel Time (2nd) function	XP_MULTI_TARGET_TRAVEL_TIME_2ND_MODE	19

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

7.1 xp_sat_nominal_att_init

7.1.1 Overview

The **xp_sat_nominal_att_init** CFI function initialises the AOCS mode for a given satellite. The initalised mode will be stored in the *sat_nom_trans_id* output structure.

7.1.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long aocs_mode;
      xp_sat_nom_trans_id sat_nom_trans_id;
      long ierr[XP_NUM_ERR_NOM_ATT_INIT_DEF], status;

      status = xp_sat_nominal_att_init(&aocs_mode,
                                       &sat_nom_trans_id, ierr);
}
```

The `XP_NUM_ERR_SAT_NOM_ATT_INIT` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 AOCS_MODE
      INTEGER*4 IERR(XP_NUM_ERR_SAT_NOM_ATT_INIT), STATUS

      STATUS = XP_SAT_NOMINAL_ATT_INIT(SAT_ID, AOCS_MODE, IERR)
```

7.1.3 Input Parameters

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

Table 4: Input parameters of xp_sat_nominal_att_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
aocs_mode	long *	-	AOCS Mode ID	-	Complete

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

- AOCS Mode ID: aocs_mode. See current document, table 3.

7.1.4 Output Parameters

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

Table 5: Output parameters of xp_sat_nominal_att_init

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_nom_trans_id	xp_sat_nom_trans_id*	-	Structure that contains the Satellite nominal Transformation	-	-
ierr	long	-	Error vector	-	-

7.1.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_sat_nominal_att_init** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_sat_nominal_att_init** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 6: Error messages of xp_sat_nominal_att_init function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Memory allocation error	No calculation performed	XP_CFI_SAT_NOMINAL_ATT_INIT_MEMORY_ERR	0

7.1.6 Runtime Performances

The following runtime performances have been measured.

Table 7: Runtime performances of xp_sat_nominal_att_init

Ultra Sparc II-400 [ms]
TBD

7.2 xp_sat_nominal_att_init_model

7.2.1 Overview

The **xp_sat_nominal_att_init_model** CFI function initialises the satellite nominal attitude model for a given satellite. The initalised model will be stored in the *sat_nom_trans_id* output structure.

7.2.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
       long model_enum;
       double model_param[XP_NUM_MODEL_PARAM];
       xp_sat_nom_trans_id sat_nom_trans_id;
       long ierr[XP_NUM_ERR_SAT_NOM_ATT_INIT_MODEL], status;

       status = xp_sat_nominal_att_init_model(&model_enum,
                                              model_param,
                                              &sat_nom_trans_id, ierr);
}
```

The `XP_NUM_ERR_SAT_NOM_ATT_INIT_MODEL` and `XP_NUM_MODEL_PARAM` constants are defined in the file *explorer_pointing.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>
       INTEGER*4 MODEL_ENUM
       REAL*8 MODEL_PARAM(XP_NUM_MODEL_PARAM)
       INTEGER*4 IERR(XP_NUM_ERR_SAT_NOM_ATT_INIT_MODEL), STATUS

       STATUS = XP_SAT_NOMINAL_ATT_INIT_MODEL(SAT_ID, MODEL_ENUM,
&              MODEL_PARAM, IERR)
```

7.2.3 Input Parameters

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

Table 8: Input parameters of xp_sat_nominal_att_init_model function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
model_enum	long *	-	Sat Nom Attitude Model ID	-	Complete
model_param	double	-	Model dependant parameters	-	Complete

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

- Satellite Nominal Attitude Model ID: model_enum. See current document, table 3.
- Model dependant parameters: model_param. See current document, table 9

Table 9: Model parameters depending on the attitude model

Attitude Model	Array Element	Description (Reference)	Unit (Format)
XP_MODEL_GENERIC	[0]	First Axis enumeration value	-
	[1]	First Target enumeration value	-
	[2]	First Vector[0]	- or deg
	[3]	First Vector[1]	- or deg
	[4]	First Vector[2]	- or deg
	[5]	Second Axis enumeration value	
	[6]	Second Target enumeration value	
	[7]	Second Vector[0]	- or deg
	[8]	Second Vector[1]	- or deg
	[9]	Second Vector[2]	- or deg
XP_MODEL_ENVISAT	[0]	AOCS Cx parameter [pitch]	deg
	[1]	AOCS Cy parameter [roll]	deg
	[2]	AOCS Cz parameter [yaw]	deg
XP_MODEL_CRYOSAT	[0]	Local Normal Z Coefficient	-
XP_MODEL_ADM	[0]	Scan Angle	deg

7.2.3.1 Generic Model description

The generic model builds the reference frames from the specified direction vectors.

The model parameters are:

- first_axis: It can be any of {+/-XP_X_AXIS, +/-XP_Y_AXIS, +/-XP_Z_AXIS}
- first_target: It can be any of {XP_SUN_VEC, XP_MOON_VEC, XP_EARTH_VEC, XP_NADIR_VEC, XP_INERTIAL_VEL_VEC, XP_EF_VEL_VEC, XP_INERTIAL_TARGET_VEC, XP_EF_TARGET_VEC}

- first_vector[3]: contains either dummies, [long, lat, alt] (if first_target= XP_EF_TARGET_VEC) or [ra, decl, parallax] (if first_target=XP_INERTIAL_TARGET_VEC})
- second_axis: It can be any of {+/-XP_X_AXIS, +/-XP_Y_AXIS, +/-XP_Z_AXIS}
- second_target: : It can be any of {XP_SUN_VEC, XP_MOON_VEC, XP_EARTH_VEC, XP_NADIR_VEC, XP_INERTIAL_VEL_VEC, XP_EF_VEL_VEC, XP_INERTIAL_TARGET_VEC, XP_EF_TARGET_VEC}
- second_vector[3]: contains either dummies, [long, lat, alt] (if first_target= XP_EF_TARGET_VEC) or [ra, decl, parallax] (if first_target=XP_INERTIAL_TARGET_VEC})

It is necessary to define a convention for each target type (e.g, always from Satellite to XXX):

- XP_SUN_VEC: Unit direction vector from Satellite to Sun
- XP_MOON_VEC: Unit direction vector from Satellite to Moon
- XP_EARTH_VEC: Unit direction vector from Satellite to Earth centre (opposite to Satellite Position Vector)
- XP_NADIR_VEC: Unit direction vector from Satellite to Nadir point
- XP_INERTIAL_VEL_VEC: Inertial Velocity vector (in TOD)
- XP_EF_VEL_VEC: Earth Fixed Velocity vector
- XP_INERTIAL_TARGET_VEC: Unit direction vector from Satellite to a target defined by a given [ra, decl, parallax]. The annual parallax is used in case we are pointing to a close object (for instance, the Moon), in order to get the distance. For stars, parallax=0 shall be used, meaning infinite distance. Units: degrees
- XP_EF_TARGET_VEC: Unit direction vector from Satellite to a target defined by a given [long, lat, alt]

With these parameters, the calculation is done as follows:

- Compute the unit direction vector specified by first_target
  - Assign the calculated first target vector to the first axis vector
- Compute the unit direction vector specified by second_target
  - Cross-product of the first axis vector and the second target vector
  - Assign the resulting vector to the second axis vector
  - Complete the right-handed frame

The following are some examples:

3. Sun-Fixed Reference Frame

- model_param = {XP_X_AXIS, XP_SUN_VEC, 0.0, 0.0, 0.0, XP_Z_AXIS, XP_EARTH_VEC, 0.0, 0.0, 0.0}

  Then:

- X-axis = Unit vector from Satellite to Sun (Sun Vector)
- Z-axis = Unit cross product: X-axis x (Unit vector from Satellite to Earth (Earth Vector))
- Y-axis = Z-axis x X-axis (completing the right-handed frame)

4. Yaw Steering Mode

- model_param={-XP_Z_AXIS, XP_NADIR_VEC, 0.0, 0.0, 0.0, XP_X_AXIS, XP_EF_VEL_VEC, 0.0, 0.0, 0.0}

  Then:

- Z-axis = -(Unit vector from Satellite to Nadir (Nadir Vector))
- X-axis = Unit cross product: Z-axis x (Earth-Fixed Velocity Vector)
- Y-axis = Z-axis x X-axis (completing the right-handed frame)

7.2.4 Output Parameters

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

Table 10: Output parameters of xp_sat_nominal_att_init_model

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_nom_trans_id	xp_sat_nom_trans_id*	-	Structure that contains the Satellite nominal Transformation	-	-
ierr	long	-	Error vector	-	-

7.2.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_sat_nominal_att_init_model** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_sat_nominal_att_init_model** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 11: Error messages of xp_sat_nominal_att_init_model function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Memory allocation error	No calculation performed	XP_CFI_SAT_NOMINAL_ATT_INIT_MODEL_MEMORY_ERR	0

7.2.6 Runtime Performances

The following runtime performances have been measured.

Table 12: Runtime performances of xp_sat_nominal_att_init_model

Ultra Sparc II-400 [ms]
TBD

7.3 xp_sat_nominal_att_init_harmonic

7.3.1 Overview

The **xp_sat_nominal_init_harmonic** CFI function initialises the satellite orbital to satellite nominal attitude mispointing angles for a given satellite with a user-provided set of values. The initalised values will be stored in the *sat_nom_trans_id* output structure.

The xp_attitude and xp_change_frame functions will then compute the values as follows:

$$\begin{aligned} attitudeangle = {} & bias + 1stsincoef \cdot \sin\left(\frac{angle \cdot 2\pi}{360}\right) + 1stcoscoef \cdot \cos\left(\frac{angle \cdot 2\pi}{360}\right) \\ & + 2ndsincoef \cdot \sin\left(\frac{2 \cdot angle \cdot 2\pi}{360}\right) + 2ndcoscoef \cdot \cos\left(\frac{2 \cdot angle \cdot 2\pi}{360}\right) \\ & + 3rdsincoef \cdot \sin\left(\frac{3 \cdot angle \cdot 2\pi}{360}\right) + 3rdcoscoef \cdot \cos\left(\frac{3 \cdot angle \cdot 2\pi}{360}\right) \\ & + \ldots \end{aligned}$$

7.3.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long angle_type, num_terms[3];
      long harmonic_type_pitch[XP_MAX_NUM_HARMONIC],
           harmonic_type_roll[XP_MAX_NUM_HARMONIC],
           harmonic_type_yaw[XP_MAX_NUM_HARMONIC];
      double harmonic_coef_pitch[XP_MAX_NUM_HARMONIC],
          harmonic_coef_roll[XP_MAX_NUM_HARMONIC],
          harmonic_coef_yaw[XP_MAX_NUM_HARMONIC];
      xp_sat_nom_trans_id sat_nom_trans_id;
      long ierr[XP_NUM_ERR_SAT_NOM_ATT_INIT_HARMONIC], status;

      status = xp_sat_nominal_att_init_harmonic(&angle_type,
                                        num_terms,
                                        harmonic_type_pitch,
                                        harmonic_type_roll,
                                        harmonic_type_yaw,
                                        harmonic_coef_pitch,
```

```
                                        harmonic_coef_roll,
                                        harmonic_coef_yaw,
                                        &sat_nom_trans_id,
                                        ierr);
}
```

The XP_NUM_ERR_SAT_NOM_ATT_INIT_HARMONIC and XP_MAX_NUM_HARMONIC constants are defined in the file *explorer_pointing.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>
      INTEGER*4 ANGLE_TYPE, NUM_TERMS(3)
      INTEGER*4 HARMONIC_TYPE_PITCH(XP_MAX_NUM_HARMONIC),
&               HARMONIC_TYPE_ROLL(XP_MAX_NUM_HARMONIC),
&               HARMONIC_TYPE_YAW(XP_MAX_NUM_HARMONIC)
      REAL*8 HARMONIC_COEF_PITCH(XP_MAX_NUM_HARMONIC),
&               HARMONIC_COEF_ROLL(XP_MAX_NUM_HARMONIC),
&               HARMONIC_COEF_YAW(XP_MAX_NUM_HARMONIC)
      INTEGER*4 IERR(XP_NUM_ERR_SAT_NOMINAL_ATT_INIT_HARMONIC),
&          STATUS

      STATUS = XP_SAT_NOMINAL_ATT_INIT_HARMONIC(SAT_ID, ANGLE_TYPE,
&                                NUM_TERMS, HARMONIC_TYPE_PITCH,
&                                HARMONIC_TYPE_ROLL, HARMONIC_TYPE_YAW,
&                                HARMONIC_COEF_PITCH,HARMONIC_COEF_ROLL,
&                                HARMONIC_COEF_YAW, IERR)
```

7.3.3 Input Parameters

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

Table 13: Input parameters of xp_sat_nominal_att_init_harmonic function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
angle_type	long *	-	Type of angle	-	XP_ANGLE_TYPE_TRUE_LAT_TOD XP_ANGLE_TYPE_MEAN_LAT_TOD
num_terms[3]	long	[0]	Number of elements used in vectors harmonic_type_pitch and harmonic_coef_pitch	-	>=0
		[1]	Number of elements used in vectors harmonic_type_roll and harmonic_coef_roll	-	>=0
		[2]	Number of elements used in vectors harmonic_type_yaw and harmonic_coef_yaw	-	>=0
harmonic_type_pitch	long[XP_MAX_NUM_HARMONIC]	all	Type of coeficients: =0 for the bias parameter <0 for the sinus coeficients (-n means that correponds to the sinus coeficient of order n) >0 for the cosinus coeficients (+n means that correponds to the cosinus coeficient of order n)	-	
harmonic_type_roll	long[XP_MAX_NUM_HARMONIC]	all	Type of coeficients: =0 for the bias parameter <0 for the sinus coeficients (-n means that correponds to the sinus coeficient of order n) >0 for the cosinus coeficients (+n means that correponds to the cosinus coeficient of order n)	-	-
harmonic_type_yaw	long[XP_MAX_NUM_HARMONIC]	all	Type of coeficients: =0 for the bias parameter <0 for the sinus coeficients (-n means that correponds to the sinus coeficient of order n) >0 for the cosinus coeficients (+n means that correponds to the cosinus coeficient of order n)	-	-

Table 13: Input parameters of xp_sat_nominal_att_init_harmonic function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
harmonic_coef_pitch	double[XP_MAX_NUM_HARMONIC]	all	Bias, sinus and cosinus coefficients for the pitch angle	deg	-
harmonic_coef_roll	double[XP_MAX_NUM_HARMONIC]	all	Bias, sinus and cosinus coefficients for the roll angle	deg	-
harmonic_coef_yaw	double[XP_MAX_NUM_HARMONIC]	all	Bias, sinus and cosinus coefficients for the yaw angle	deg	-

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

- Angle Type: See current document, table 3.

7.3.4 Output Parameters

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

Table 14: Output parameters of xp_sat_nominal_att_init_harmonic

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_nom_trans_id	xp_sat_nom_trans_id*	-	Structure that contains the Satellite nominal Transformation	-	-
ierr	long	-	Error vector	-	-

7.3.5 Example

For the satellite ERS:
pitch = -0.16725*cos(true_lat)*sin(true_lat)*2=-0.16725*sin(2*true_lat)
num_terms[0]=1
harmonic_type_pitch={-2} harmonic_coef_pitch={-0.16725}

roll = 0.05012*sin(true_lat)
num_terms[1]=1
harmonic_type_roll={-1} harmonic_coef_roll={0.05012}

yaw= 3.9163*cos(true_lat)
num_terms[2]=1
harmonic_type_yaw={+1} harmonic_coef_yaw={3.9163}

7.3.6 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_sat_nominal_att_init_harmonic** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_sat_nominal_att_init_harmonic** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 15: Error messages of xp_sat_nominal_att_init_harmonic function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Memory allocation error	No calculation performed	XP_CFI_SAT_NOMINAL_ATT_INIT_HARMONIC_MEMORY_ERR	0

7.3.7 Runtime Performances

The following runtime performances have been measured:

Table 16: Runtime performances of xp_sat_att_nominal_init_harmonic

Ultra Sparc II-400 [ms]
TBD

7.4 xp_sat_nominal_att_init_file

7.4.1 Overview

The **xp_sat_nominal_att_init_file** CFI function initialises the satellite nominal attitude angles for a given satellite reading values from the attitude file(s). The validity time or orbital range for the attitude angles can be specified by the user. The initalised values will be stored in the *sat_nom_trans_id* output structure.

7.4.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      xl_time_id time_id;
      long n_files, time_init_mode, time_ref;
      char **attitude_file;
      double time0, time1;
      double val_time0, val_time1;
      xp_sat_nom_trans_id sat_nom_trans_id;
      long ierr[XP_NUM_ERR_SAT_NOM_ATT_INIT_FILE], status;

      status = xp_sat_nominal_att_init_file(&time_id, &n_files,
          attitude_file, &time_init_mode, &time_ref, &time0, &time1,
          &val_time0, &val_time1, &sat_nom_trans_id, ierr);
}
```

The `XP_NUM_ERR_SAT_NOM_ATT_INIT_FILE` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, N_FILES, TIME_INIT_MODE, TIME_REF
      INTEGER*4 ORBIT0, ORBIT1
      CHARACTER*LENGTH_NAME ATTITUDE_FILE(NUM_FILES)
      REAL*8 TIME0, TIME1
      INTEGER*4 IERR(XP_NUM_ERR_SAT_NOM_ATT_INIT_FILE), STATUS

      STATUS = XP_SAT_NOMINAL_ATT_INIT_FILE(SAT_ID, N_FILES,
&             ATTITUDE_FILE, TIME_INIT_MODE, TIME_REF, TIME0, TIME1,
&             ORBIT0, ORBIT1, IERR)
```

7.4.3 Input Parameters

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

Table 17: Input parameters of xp_sat_nominal_att_init_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
n_files	long *	-	Number of reference data files	-	> 0
attitude_file	char **	-	Filenames of the reference data files. In case multiple files are used, the files should be time ordered. Details TBD.	-	-
time_init_mode	long *	-	Flag for selecting the time range of the initialisation.	-	Select either: · XP_SEL_TIME · XP_SEL_FILE
time_ref	long *	-	Time reference ID	-	Complete
time0	double*	-	If:*time_init_mode=XP_SEL_TIME* S Start of the time range defined by[time0,time1]	Decimal days (Process-ing format)	[-18262.0,36524.0]
time1	double*	-	If: t*ime_init_mode=XP_SEL_TIME* End of the time range defined by [time0,time1]	Decimal days (Process ing format)	[-18262.0,36524.0] > time0

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].
- Time Init Mode ID: time_init_mode. See current document, table 3.
- NOTE (TBC): The supported Attitude File formats will contain:
  - Time vs attitude angles
  - Angle (any of angle_type) vs attitude angles

  where attitude angles will be:
  - pitch, roll and yaw angles
  - quaternions

7.4.4 Output Parameters

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

Table 18: Output parameters of xp_sat_nominal_att_init_file

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
val_time0	double*	-	Validity start time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
val_time1	double*	-	Validity end time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
sat_nom_trans_id	xp_sat_nom_trans_id*	-	Structure that contains the Satellite nominal Transformation	-	-
ierr	long	-	Error vector	-	-

7.4.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_sat_nominal_att_init_file** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_sat_nominal_att_init_file** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 19: Error messages of xp_sat_nominal_att_init_file function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Memory allocation error	No calculation performed	XP_CFI_SAT_NOMINAL_ATT_INIT_FILE_MEMORY_ERR	0
ERR	Wrong input time reference	No calculation performed	XP_CFI_SAT_NOMINAL_ATT_INIT_FILE_WRONG_TIME_REF_ERR	1

7.4.6 Runtime Performances

The following runtime performances have been measured.

Table 20: Runtime performances of xp_sat_nominal_att_init_file

Ultra Sparc II-400 [ms]
TBD

7.5 xp_sat_nominal_att_close

7.5.1 Overview

The **xp_sat_nominal_att_close** CFI function cleans up any memory allocation performed by the satellite nominal attitude initialization functions.

7.5.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      xp_sat_nom_trans_id sat_nom_trans_id;
      long ierr[XP_NUM_ERR_SAT_NOM_ATT_CLOSE], status;

      status = xp_sat_nominal_att_close(&sat_nom_trans_id, ierr);
}
```

The `XP_NUM_ERR_SAT_NOM_ATT_CLOSE` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID
      INTEGER*4 IERR(XP_NUM_ERR_SAT_NOMINAL_ATT_CLOSE), STATUS

      STATUS = XP_SAT_NOMINAL_ATT_CLOSE(SAT_ID, IERR)
```

7.5.3 Input Parameters

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

Table 21: Input parameters of xp_sat_nominal_att_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_nom_trans_id	xp_sat_nom_trans_id*	-	Structure that contains the Satellite Nom. Trans.	-	-

7.5.4 Output Parameters

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

Table 22: Output parameters of xp_sat_nominal_att_close

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

7.5.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_sat_nominal_att_close** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_sat_nominal_att_close** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 23: Error messages of xp_sat_nominal_att_close function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Could not close the Id. as it is not initialized or it is being used	No calculation performed	XP_CFI_SAT_NOMINAL_ATT_CLOSE_WRONG_ID_ERR	0

7.5.6 Runtime Performances

The following runtime performances have been measured.

Table 24: Runtime performances of xp_sat_nominal_att_close

Ultra Sparc II-400 [ms]
TBD

7.6 xp_sat_att_angle_init

7.6.1 Overview

The **xp_sat_att_angle_init** CFI function initialises the satellite nominal attitude to satellite attitude mis-pointing angles for a given satellite with a user-provided set of values. The initalised values will be stored in the *sat_trans_id* output structure.

7.6.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      double ang[3];
      xp_sat_trans_id sat_trans_id;
      long ierr[XP_NUM_ERR_MISP_ANGLE_INIT_DEF], status;

      status = xp_sat_att_angle_init(ang, &sat_trans_id, ierr);
}
```

The `XP_NUM_ERR_SAT_ATT_ANGLE_INIT` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID
      REAL*8 ANG(3)
      INTEGER*4 IERR(XP_NUM_ERR_SAT_ATT_ANGLE_INIT), STATUS

      STATUS = XP_SAT_ATT_ANGLE_INIT(SAT_ID, ANG, IERR)
```

7.6.3 Input Parameters

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

Table 25: Input parameters of xp_sat_att_angle_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
ang	double[3]	[0]	Pitch mispointing angle (Satellite Nominal Attitude Frame)	deg	If no better value, assume 0.0
		[1]	Roll mispointing angle (Satellite Nominal Attitude Frame)	deg	If no better value, assume 0.0
		[2]	Yaw mispointing angle (Satellite Nominal Attitude Frame)	deg	If no better value, assume 0.0

7.6.4 Output Parameters

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

Table 26: Output parameters of xp_sat_att_angle_init

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_trans_id	xp_sat_trans_id*	-	Structure that contains the Satellite Transformation	-	-
ierr	long	-	Error vector	-	-

7.6.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_sat_att_angle_init** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_sat_att_angle_init** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 27: Error messages of xp_sat_att_angle_init function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Memory allocation error	No calculation performed	XP_CFI_SAT_ATT_ANGLE_INIT_MEMORY_ERR	0

7.6.6 Runtime Performances

The following runtime performances have been measured.

Table 28: Runtime performances of xp_sat_att_angle_init

Ultra Sparc II-400 [ms]
TBD

7.7 xp_sat_att_matrix_init

7.7.1 Overview

The **xp_sat_att_matrix_init** CFI function initialises misalignement matrix between the satellite nominal attitude frame and satellite attitude frame with a user-provided matrix. The initalised values will be stored in the *sat_trans_id* output structure.

7.7.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      double att_matrix[3][3];
      xp_sat_trans_id sat_trans_id;
      long ierr[XP_NUM_ERR_SAT_ATT_MATRIX_INIT], status;

      status = xp_sat_att_matrix_init_def(att_matrix,
                                          &sat_trans_id, ierr);
}
```

The `XP_NUM_ERR_SAT_ATT_MATRIX_INIT` constant is defined in the file *explorer_pointing.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 ATT_MATRIX(3,3)
      INTEGER*4 IERR(XP_NUM_ERR_SAT_ATT_MATRIX_INIT), STATUS

      STATUS = XP_SAT_ATT_MATRIX_INIT_DEF(ATT_MATRIX, IERR)
```

`Note`: The matrices are handled differently in C and in ForTran programs. Details TBW.

7.7.3 Input Parameters

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

Table 29: Input parameters of xp_sat_att_matrix_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
att_matrix	double[3][3]	all	Mispointing Matrix	-	-

7.7.4 Output Parameters

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

Table 30: Output parameters of xp_sat_att_matrix_init

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_trans_id	xp_sat_trans_id*	-	Structure that contains the Satellite Transformation	-	-
ierr	long	-	Error vector	-	-

7.7.5 Example

TBD

7.7.6 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_sat_att_matrix_init** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_sat_att_matrix_init** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 31: Error messages of xp_sat_att_matrix_init function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Memory allocation error	No calculation performed	XP_CFI_SAT_ATT_MATRIX_INIT_MEMORY_ERR	0

7.7.7 Runtime Performances

The following runtime performances have been measured.

Table 32: Runtime performances of xp_sat_att_matrix_init

Ultra Sparc II-400 [ms]
TBD

7.8 xp_sat_att_init_harmonic

7.8.1 Overview

The **xp_sat_att_init_harmonic** CFI function initialises the satellite nominal orbital to satellite attitude mispointing angles for a given satellite with a user-provided set of values. The initalised values will be stored in the *sat_trans_id* output structure.

The xp_attitude and xp_change_frame functions will then compute the values as follows:

$$\begin{aligned} attitudeangle = {} & bias + 1stsincoef \cdot \sin\left(\frac{angle \cdot 2\pi}{360}\right) + 1stcoscoef \cdot \cos\left(\frac{angle \cdot 2\pi}{360}\right) \\ & + 2ndsincoef \cdot \sin\left(\frac{2 \cdot angle \cdot 2\pi}{360}\right) + 2ndcoscoef \cdot \cos\left(\frac{2 \cdot angle \cdot 2\pi}{360}\right) \\ & + 3rdsincoef \cdot \sin\left(\frac{3 \cdot angle \cdot 2\pi}{360}\right) + 3rdcoscoef \cdot \cos\left(\frac{3 \cdot angle \cdot 2\pi}{360}\right) \\ & + \ldots \end{aligned}$$

7.8.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long angle_type, num_terms[3];
      long harmonic_type_pitch[XP_MAX_NUM_HARMONIC],
           harmonic_type_roll[XP_MAX_NUM_HARMONIC],
           harmonic_type_yaw[XP_MAX_NUM_HARMONIC];
      double harmonic_coef_pitch[XP_MAX_NUM_HARMONIC],
          harmonic_coef_roll[XP_MAX_NUM_HARMONIC],
          harmonic_coef_yaw[XP_MAX_NUM_HARMONIC];
      xp_sat_trans_id sat_trans_id;
      long ierr[XP_NUM_ERR_SAT_ATT_INIT_HARMONIC], status;

      status = xp_sat_att_init_harmonic(&angle_type, num_terms,
                                        harmonic_type_pitch,
                                        harmonic_type_roll,
                                        harmonic_type_yaw,
                                        harmonic_coef_pitch,
                                        harmonic_coef_roll,
```

```
                                        harmonic_coef_yaw,
                                        &sat_trans_id, ierr);
}
```

The XP_NUM_ERR_SAT_ATT_INIT_HARMONIC and XP_MAX_NUM_HARMONIC constants are defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, ANGLE_TYPE, NUM_TERMS(3)
      INTEGER*4 HARMONIC_TYPE_PITCH(XP_MAX_NUM_HARMONIC),
&               HARMONIC_TYPE_ROLL(XP_MAX_NUM_HARMONIC),
&               HARMONIC_TYPE_YAW(XP_MAX_NUM_HARMONIC)
      REAL*8 HARMONIC_COEF_PITCH(XP_MAX_NUM_HARMONIC),
&               HARMONIC_COEF_ROLL(XP_MAX_NUM_HARMONIC),
&               HARMONIC_COEF_YAW(XP_MAX_NUM_HARMONIC)
      INTEGER*4 IERR(XP_NUM_ERR_SAT_ATT_INIT_HARMONIC),
&          STATUS

      STATUS = XP_SAT_ATT_INIT_HARMONIC(SAT_ID, ANGLE_TYPE,
&                                NUM_TERMS, HARMONIC_TYPE_PITCH,
&                                HARMONIC_TYPE_ROLL, HARMONIC_TYPE_YAW,
&                                HARMONIC_COEF_PITCH,HARMONIC_COEF_ROLL,
&                                HARMONIC_COEF_YAW, IERR)
```

7.8.3 Input Parameters

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

Table 33: Input parameters of xp_sat_att_init_harmonic function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
angle_type	long *	-	Type of angle	-	XP_ANGLE_TYPE_TRUE_LAT_TOD XP_ANGLE_TYPE_MEAN_LAT_TOD
num_terms[3]	long	[0]	Number of elements used in vectors harmonic_type_pitch and harmonic_coef_pitch	-	>=0
		[1]	Number of elements used in vectors harmonic_type_roll and harmonic_coef_roll	-	>=0
		[2]	Number of elements used in vectors harmonic_type_yaw and harmonic_coef_yaw	-	>=0
harmonic_type_pitch	long[XP_MAX_NUM_HARMONIC]	all	Type of coeficients: =0 for the bias parameter <0 for the sinus coeficients (-n means that correponds to the sinus coeficient of order n) >0 for the cosinus coeficients (+n means that correponds to the cosinus coeficient of order n)	-	
harmonic_type_roll	long[XP_MAX_NUM_HARMONIC]	all	Type of coeficients: =0 for the bias parameter <0 for the sinus coeficients (-n means that correponds to the sinus coeficient of order n) >0 for the cosinus coeficients (+n means that correponds to the cosinus coeficient of order n)	-	-
harmonic_type_yaw	long[XP_MAX_NUM_HARMONIC]	all	Type of coeficients: =0 for the bias parameter <0 for the sinus coeficients (-n means that correponds to the sinus coeficient of order n) >0 for the cosinus coeficients (+n means that correponds to the cosinus coeficient of order n)	-	-

Table 33: Input parameters of xp_sat_att_init_harmonic function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
harmonic_coef_pitch	double[XP_MAX_NUM_HARMONIC]	all	Bias, sinus and cosinus coefficients for the pitch angle	deg	
harmonic_coef_roll	double[XP_MAX_NUM_HARMONIC]	all	Bias, sinus and cosinus coefficients for the roll angle	deg	
harmonic_coef_yaw	double[XP_MAX_NUM_HARMONIC]	all	Bias, sinus and cosinus coefficients for the yaw angle	deg	

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

- Angle Type: See current document, table 3.

7.8.4 Output Parameters

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

Table 34: Output parameters of xp_sat_att_init_harmonic

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_trans_id	xp_sat_trans_id*	-	Structure that contains the Satellite Transformation	-	-
ierr	long	-	Error vector	-	-

7.8.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_sat_att_init_harmonic** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_sat_att_init_harmonic** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 35: Error messages of xp_sat_att_init_harmonic function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Memory allocation error	No calculation performed	XP_CFI_SAT_ATT_INIT_HARMONIC_MEMORY_ERR	0

7.8.6 Runtime Performances

The following runtime performances have been measured.

Table 36: Runtime performances of xp_sat_att_init_harmonic

Ultra Sparc II-400 [ms]
TBD

7.9 xp_sat_att_init_file

7.9.1 Overview

The **xp_sat_att_init_file** CFI function initialises the satellite attitude angles for a given satellite reading values from the attitude file(s). The validity time or orbital range for the attitude angles can be specified by the user. The initalised values will be stored in the *sat_trans_id* output structure.

7.9.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      xl_time_id time_id;
      long n_files, time_init_mode, time_ref;
      char **attitude_file *auxiliary_file;
      double time0, time1;
      double val_time0, val_time1;
      xp_sat_trans_id sat_trans_id;
      long ierr[XP_NUM_ERR_SAT_ATT_INIT_FILE], status;

      status = xp_sat_att_init_file(&time_id, &n_files,
               attitude_file, &auxiliary_file,
               time_init_mode, &time_ref, &time0, &time1,
               &val_time0, &val_time1, &sat_trans_id, ierr);
}
```

The `XP_NUM_ERR_SAT_ATT_INIT_FILE` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, N_FILES, TIME_INIT_MODE, TIME_REF
      INTEGER*4 ORBIT0, ORBIT1
      CHARACTER*LENGTH_NAME ATTITUDE_FILE(NUM_FILES), AUXILIARY_FILE
      REAL*8 TIME0, TIME1
      INTEGER*4 IERR(XP_NUM_ERR_SAT_ATT_INIT_FILE), STATUS

      STATUS = XP_SAT_ATT_INIT_FILE(SAT_ID, N_FILES,
&             ATTITUDE_FILE, AUXILIARY_FILE, TIME_INIT_MODE,
```

```
&                TIME_REF, TIME0, TIME1,
&                ORBIT0, ORBIT1, IERR)
```

7.9.3 Input Parameters

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

Table 37: Input parameters of xp_sat_att_init_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
n_files	long *	-	Number of reference data files	-	> 0
attitude_file	char **	-	Filenames of the reference data files. In case multiple files are used, the files should be time ordered. Details TBD.	-	-
auxiliary_file	char **	-	Filename of an auxiliary file containing the Star-Tracker misalignement matrices (the format must be the same as the attitude file given in function xp_instr_att_init_file)	-	-
time_init_mode	long *	-	Flag for selecting the time range of the initialisation.	-	Select either: · XP_SEL_TIME · XP_SEL_FILE
time_ref	long *	-	Time reference ID	-	Complete
time0	double*	-	If: *time_init_mode=XP_SEL_TIME* Start of the time range defined by [time0,time1]	Decimal days (Processing format)	[-18262.0,36524.0]
time1	double*	-	If: *time_init_mode=XP_SEL_TIME* End of the time range defined by [time0,time1]	Decimal days (Processing format)	[-18262.0,36524.0] > time0

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].
- Time Init Mode ID: time_init_mode. See current document, table 3.
- NOTE (TBC): The supported Attitude File formats will contain:
  - Time vs attitude angles
  - Angle (any of angle_type) vs attitude angles

  where attitude angles will be:
  - pitch, roll and yaw angles
  - quaternions

7.9.4 Output Parameters

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

Table 38: Output parameters of xp_sat_att_init_file

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
val_time0	double*	-	Validity start time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
val_time1	double*	-	Validity end time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
sat_trans_id	xp_sat_trans_id*	-	Structure that contains the Satellite Transformation	-	-
ierr	long	-	Error vector	-	-

7.9.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_sat_att_init_file** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_sat_att_init_file** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 39: Error messages of xp_sat_att_init_file function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Memory allocation error	No calculation performed	XP_CFI_SAT_ATT_INIT_FILE_MEMORY_ERR	0
ERR	Error reading input files	No calculation performed	XP_CFI_SAT_ATT_INIT_FILE_READ_FILES_ERR	1
ERR	No data has been read from the files	No calculation performed	XP_CFI_SAT_ATT_INIT_FILE_NO_READ_DATA_ERR	2
ERR	Error reading auxiliary file	No calculation performed	XP_CFI_SAT_ATT_INIT_FILE_READ_AUX_FILE_ERR	3
ERR	Wrong input time reference	No calculation performed	XP_CFI_SAT_ATT_INIT_FILE_WRONG_TIME_REF_ERR	4
ERR	Could not perform a time transformation	No calculation performed	XP_CFI_SAT_ATT_INIT_FILE_TIME_REF_ERR	5

7.9.6 Runtime Performances

The following runtime performances have been measured.

Table 40: Runtime performances of xp_sat_att_init_file

Ultra Sparc II-400 [ms]
TBD

7.10 xp_sat_att_close

7.10.1 Overview

The **xp_sat_att_close** CFI function cleans up any memory allocation performed by the satellite attitude initialization functions.

7.10.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      xp_sat_trans_id sat_trans_id;
      long ierr[XP_NUM_ERR_SAT_ATT_CLOSE], status;

      status = xp_sat_att_close(&sat_trans_id, ierr);
}
```

The `XP_NUM_ERR_SAT_ATT_CLOSE` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID
      INTEGER*4 IERR(XP_NUM_ERR_SAT_ATT_CLOSE), STATUS

      STATUS = XP_SAT_ATT_CLOSE(SAT_ID, IERR)
```

7.10.3 Input Parameters

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

Table 41: Input parameters of xp_sat_att_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_trans_id	xp_sat_trans_id*	-	Structure that contains the Sat. Trans.	-	-

7.10.4 Output Parameters

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

Table 42: Output parameters of xp_sat_att_close

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

7.10.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_sat_att_close** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_sat_att_close** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 43: Error messages of xp_sat_att_close function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Could not close the Id. as it is not initialized or it is being used	No calculation performed	XP_CFI_SAT_ATT_CLOSE_WRONG_ID_ERR	0

7.10.6 Runtime Performances

The following runtime performances have been measured.

Table 44: Runtime performances of xp_sat_att_close

Ultra Sparc II-400 [ms]
TBD

7.11 xp_instr_att_angle_init

7.11.1 Overview

The **xp_instr_att_angle_init** CFI function initialises the instrument attitude mispointing angles for a given satellite and instrument with a user-provided set of values. The initalised values will be stored in the *instr_trans_id* output structure.

7.11.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      double ang[3], offset[3];
      xp_instr_trans_id instr_trans_id;
      long ierr[XP_NUM_ERR_INSTR_ATT_ANGLE_INIT], status;

      status = xp_instr_att_angle_init(ang, offset,
                                       &instr_trans_id, ierr);
}
```

The `XP_NUM_ERR_INSTR_ATT_ANGLE_INIT` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, INSTRUMENT_ID
      REAL*8 ANG(3), OFFSET(3)
      INTEGER*4 IERR(XP_NUM_ERR_INSTR_ATT_ANGLE_INIT), STATUS

      STATUS = XP_INSTR_ATT_ANGLE_INIT(SAT_ID, INSTRUMENT_ID, ANG,
&                                      OFFSET, IERR)
```

7.11.3 Input Parameters

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

Table 45: Input parameters of xp_instr_att_angle_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
ang	double[3]	[0]	Pitch mispointing angle (Satellite Attitude Frame)	deg	If no better value, assume 0.0
		[1]	Roll mispointing angle (Satellite Attitude Frame)	deg	If no better value, assume 0.0
		[2]	Yaw mispointing angle (Satellite Attitude Frame)	deg	If no better value, assume 0.0
offset	double[3]	all	Instrument Frame Origin position vector (Satellite Attitude Frame)	m	-

7.11.4 Output Parameters

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

Table 46: Output parameters of xp_instr_att_angle_init

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
instr_trans_id	xp_instr_trans_id*	-	Structure that contains the Instrument Transformation	-	-
ierr	long	-	Error vector	-	-

7.11.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_instr_att_angle_init** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_instr_att_angle_init** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 47: Error messages of xp_instr_att_angle_init function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Memory allocation error	No calculation performed	XP_CFI_INSTR_ATT_ANGLE_INIT_MEMORY_ERR	0

7.11.6 Runtime Performances

The following runtime performances have been measured.

Table 48: Runtime performances of xp_instr_att_angle_init

Ultra Sparc II-400 [ms]
TBD

7.12 xp_instr_att_matrix_init

7.12.1 Overview

The **xp_instr_att_matrix_init** CFI function initialises the instrument attitude mispointing angles for a given satellite and instrument with a user-provided matrix. The initalised values will be stored in the *instr_trans_id* output structure.

7.12.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      double att_matrix[3][3], offset[3];
      xp_instr_trans_id instr_trans_id;
      long ierr[XP_NUM_ERR_INSTR_ATT_MATRIX_INIT], status;

      status = xp_instr_att_matrix_init(att_matrix, offset,
                                        &instr_trans_id, ierr);
}
```

The `XP_NUM_ERR_INSTR_ATT_MATRIX_INIT` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, INSTRUMENT_ID
      REAL*8 ATT_MATRIX(3,3), OFFSET(3)
      INTEGER*4 IERR(XP_NUM_ERR_INSTR_ATT_MATRIX_INIT), STATUS

      STATUS = XP_INSTR_ATT_MATRIX_INIT(SAT_ID, INSTRUMENT_ID,
&                                ATT_MATRIX, OFFSET, IERR)
```

`Note:` The matrices are handled differently in C and in ForTran programs. Details TBW.

7.12.3 Input Parameters

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

Table 49: Input parameters of xp_instr_att_matrix_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
att_matrix	double[3][3]	all	Mispointing Matrix	-	-
offset	double[3]	all	Instrument Frame Origin position vector (Satellite Attitude Frame)	m	-

7.12.4 Output Parameters

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

Table 50: Output parameters of xp_instr_att_matrix_init

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
instr_trans_id	xp_instr_trans_id*	-	Structure that contains the Instrument Transformation	-	-
ierr	long	-	Error vector	-	-

7.12.5 Example

TBD

7.12.6 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_instr_att_matrix_init** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_instr_att_matrix_init** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 51: Error messages of xp_instr_att_matrix_init function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Memory allocation error	No calculation performed	XP_CFI_INSTR_ATT_MATRIX_INIT_MEMORY_ERR	0

7.12.7 Runtime Performances

The following runtime performances have been measured.

Table 52: Runtime performances of xp_instr_att_matrix_init

Ultra Sparc II-400 [ms]
TBD

7.13 xp_instr_att_init_harmonic

7.13.1 Overview

The **xp_instr_att_init_harmonic** CFI function initialises the instrument attitude mispointing angles for a given satellite and instrument with a user-provided set of values. The initalised values will be stored in the *instr_trans_id* output structure.

The xp_attitude and xp_change_frame functions will then compute the values as follows:

$$\begin{aligned} attitudeangle = {} & bias + 1stsincoef \cdot \sin\left(\frac{angle \cdot 2\pi}{360}\right) + 1stcoscoef \cdot \cos\left(\frac{angle \cdot 2\pi}{360}\right) \\ & + 2ndsincoef \cdot \sin\left(\frac{2 \cdot angle \cdot 2\pi}{360}\right) + 2ndcoscoef \cdot \cos\left(\frac{2 \cdot angle \cdot 2\pi}{360}\right) \\ & + 3rdsincoef \cdot \sin\left(\frac{3 \cdot angle \cdot 2\pi}{360}\right) + 3rdcoscoef \cdot \cos\left(\frac{3 \cdot angle \cdot 2\pi}{360}\right) \\ & + \ldots \end{aligned}$$

7.13.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long angle_type, num_terms[3];
      long harmonic_type_pitch[XP_MAX_NUM_HARMONIC],
           harmonic_type_roll[XP_MAX_NUM_HARMONIC],
           harmonic_type_yaw[XP_MAX_NUM_HARMONIC];
      double harmonic_coef_pitch[XP_MAX_NUM_HARMONIC],
          harmonic_coef_roll[XP_MAX_NUM_HARMONIC],
          harmonic_coef_yaw[XP_MAX_NUM_HARMONIC];
      double offset[3];
      xp_instr_trans_id instr_trans_id;
      long ierr[XP_NUM_ERR_INSTR_ATT_INIT_HARMONIC], status;

      status = xp_instr_att_init_harmonic(&angle_type, num_terms,
                                          harmonic_type_pitch,
                                          harmonic_type_roll,
                                          harmonic_type_yaw,
                                          harmonic_coef_pitch,
```

```
                                        harmonic_coef_roll,
                                        harmonic_coef_yaw,
                                        offset,
                                        &instr_trans_id, ierr);
}
```

The XP_NUM_ERR_INSTR_ATT_INIT_HARMONIC and XP_MAX_NUM_HARMONIC constants are defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, ANGLE_TYPE, NUM_TERMS(3)
      INTEGER*4 HARMONIC_TYPE_PITCH(XP_MAX_NUM_HARMONIC),
&               HARMONIC_TYPE_ROLL(XP_MAX_NUM_HARMONIC),
&               HARMONIC_TYPE_YAW(XP_MAX_NUM_HARMONIC)
      REAL*8 HARMONIC_COEF_PITCH(XP_MAX_NUM_HARMONIC),
&               HARMONIC_COEF_ROLL(XP_MAX_NUM_HARMONIC),
&               HARMONIC_COEF_YAW(XP_MAX_NUM_HARMONIC)
      INTEGER*4 IERR(XP_NUM_ERR_INSTR_ATT_INIT_HARMONIC),
&          STATUS

      STATUS = XP_INSTR_ATT_INIT_HARMONIC(SAT_ID, INSTRUMENT_ID,
&                                 ANGLE_TYPE,
&                                 NUM_TERMS, HARMONIC_TYPE_PITCH,
&                                 HARMONIC_TYPE_ROLL, HARMONIC_TYPE_YAW,
&                                 HARMONIC_COEF_PITCH,HARMONIC_COEF_ROLL,
&                                 HARMONIC_COEF_YAW, IERR)
```

7.13.3 Input Parameters

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

Table 53: Input parameters of xp_instr_att_init_harmonic function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
angle_type	long *	-	Type of angle	-	XP_ANGLE_TYPE_TRUE_LAT_TOD XP_ANGLE_TYPE_MEAN_LAT_TOD
num_terms[3]	long	[0]	Number of elements used in vectors harmonic_type_pitch and harmonic_coef_pitch	-	>=0
		[1]	Number of elements used in vectors harmonic_type_roll and harmonic_coef_roll	-	>=0
		[2]	Number of elements used in vectors harmonic_type_yaw and harmonic_coef_yaw	-	>=0
harmonic_type_pitch	long[XP_MAX_NUM_HARMONIC]	all	Type of coeficients: =0 for the bias parameter <0 for the sinus coeficients (-n means that correponds to the sinus coeficient of order n) >0 for the cosinus coeficients (+n means that correponds to the cosinus coeficient of order n)	-	
harmonic_type_roll	long[XP_MAX_NUM_HARMONIC]	all	Type of coeficients: =0 for the bias parameter <0 for the sinus coeficients (-n means that correponds to the sinus coeficient of order n) >0 for the cosinus coeficients (+n means that correponds to the cosinus coeficient of order n)	-	-
harmonic_type_yaw	long[XP_MAX_NUM_HARMONIC]	all	Type of coeficients: =0 for the bias parameter <0 for the sinus coeficients (-n means that correponds to the sinus coeficient of order n) >0 for the cosinus coeficients (+n means that correponds to the cosinus coeficient of order n)	-	-

Table 53: Input parameters of xp_instr_att_init_harmonic function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
harmonic_coef_pitch	double[XP_MAX_NUM_HARMONIC]	all	Bias, sinus and cosinus coefficients for the pitch angle	deg	
harmonic_coef_roll	double[XP_MAX_NUM_HARMONIC]	all	Bias, sinus and cosinus coefficients for the roll angle	deg	
harmonic_coef_yaw	double[XP_MAX_NUM_HARMONIC]	all	Bias, sinus and cosinus coefficients for the yaw angle	deg	
offset	double[3]	all	Instrument Frame Origin position vector (Satellite Attitude Frame)	m	-

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

- Angle Type: See current document, table 3.

7.13.4 Output Parameters

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

Table 54: Output parameters of xp_instr_att_init_harmonic

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
instr_trans_id	xp_instr_trans_id*	-	Structure that contains the Instrument Transformation	-	-
ierr	long	-	Error vector	-	-

7.13.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_instr_att_init_harmonic** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_instr_att_init_harmonic** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 55: Error messages of xp_instr_att_init_harmonic function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Memory allocation error	No calculation performed	XP_CFI_INSTR_ATT_INIT_HARMONIC_MEMORY_ERR	0

7.13.6 Runtime Performances

The following runtime performances have been measured.

Table 56: Runtime performances of xp_instr_att_init_harmonic

Ultra Sparc II-400 [ms]
TBD

7.14 xp_instr_att_init_file

7.14.1 Overview

The **xp_instr_att_init_file** CFI function initialises the instrument attitude mispointing angles for a given satellite reading values from the attitude file(s). The validity time or orbital range for the attitude angles can be specified by the user. The initalised values will be kept in memory and used by other CFI functions.

7.14.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      xl_time_id time_id;
      long n_files, time_init_mode, time_ref;
      char **instrument_file;
      double time0, time1;
      double val_time0, val_time1;
      xp_instr_trans_id instr_trans_id;
      long ierr[XP_NUM_ERR_INSTR_ATT_INIT_FILE], status;

      status = xp_instr_att_init_file(&time_id,
                  &n_files, instrument_file,
                  &time_init_mode, &time_ref, &time0, &time1,
                  &val_time0, &val_time1, &instr_trans_id, ierr);
}
```

The `XP_NUM_ERR_INSTR_ATT_INIT_FILE` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, INSTRUMENT_ID, N_FILES, TIME_INIT_MODE, &
&             TIME_REF, ORBIT0, ORBIT1
      CHARACTER*LENGTH_NAME INSTRUMENT_FILE(NUM_FILES)
      REAL*8 TIME0, TIME1
      INTEGER*4 IERR(XP_NUM_ERR_INSTR_ATT_INIT_FILE), STATUS
      STATUS = XP_INSTR_ATT_INIT_FILE(SAT_ID, INSTRUMENT_ID, N_FILES,
&             INSTRUMENT_FILE, TIME_INIT_MODE, TIME_REF, TIME0, TIME1,
&             ORBIT0, ORBIT1, IERR)
```

7.14.3 Input Parameters

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

Table 57: Input parameters of xp_instr_att_init_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
n_files	long *	-	Number of reference data files	-	> 0
instrument_file	char **	-	Filenames of the reference data files. In case multiple files are used, the files should be time ordered. Details TBD.	-	-
time_init_mode	long *	-	Flag for selecting the time range of the initialisation.	-	Select either: · XP_SEL_TIME · XP_SEL_FILE
time_ref	long *	-	Time reference ID	-	Complete
time0	double*	-	If:*time_init_mode=XP_SEL_TIME* Start of the time range defined by [time0,time1]	Decimal days (Processing format)	[-18262.0,36524.0]
time1	double*	-	If:*time_init_mode=XP_SEL_TIME* End of the time range defined by [time0,time1]	Decimal days (Processing format)	[-18262.0,36524.0] > time0

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].
- Time Init Mode ID: time_init_mode. See current document, table 3.

7.14.4 Output Parameters

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

Table 58: Output parameters of xp_instr_att_init_file

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
val_time0	double*	-	Validity start time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
val_time1	double*	-	Validity end time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
instr_trans_id	xp_instr_trans_id*	-	Structure that contains the Instrument Transformation	-	-
ierr	long	-	Error vector	-	-

7.14.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_instr_att_init_file** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_instr_att_init_file** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 59: Error messages of xp_instr_att_init_file function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Memory allocation error	No calculation performed	XP_CFI_INSTR_ATT_INIT_FILE_MEMORY_ERR	0
ERR	Wrong input time reference	No calculation performed	XP_CFI_INSTR_ATT_INIT_FILE_WRONG_TIME_REF_ERR	1

7.14.6 Runtime Performances

The following runtime performances have been measured.

Table 60: Runtime performances of xp_instr_att_init_file

Ultra Sparc II-400 [ms]
TBD

7.15 xp_instr_att_close

7.15.1 Overview

The **xp_instr_att_close** CFI function cleans up any memory allocation performed by the instrument attitude initialization functions.

7.15.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      xp_instr_trans_id instr_trans_id;
      long ierr[XP_NUM_ERR_INSTR_ATT_CLOSE], status;

      status = xp_instr_att_close(&instr_trans_id, ierr);
}
```

The `XP_NUM_ERR_INSTR_ATT_CLOSE` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, INSTRUMENT_ID
      INTEGER*4 IERR(XP_NUM_ERR_INSTR_ATT_CLOSE), STATUS

      STATUS = XP_INSTR_ATT_CLOSE(SAT_ID, INSTRUMENT_ID,IERR)
```

7.15.3 Input Parameters

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

Table 61: Input parameters of xp_instr_att_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
instr_trans_id	xp_instr_trans_id*	-	Structure that contains the Instr. Trans.	-	-

7.15.4 Output Parameters

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

Table 62: Output parameters of xp_instr_att_close

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

7.15.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_instr_att_close** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_instr_att_close** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 63: Error messages of xp_instr_att_close function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Could not close the Id. as it is not initialized or it is being used	No calculation performed	XP_CFI_INSTR_ATT_CLOSE_WRONG_ID_ERR	0

7.15.6 Runtime Performances

The following runtime performances have been measured.

Table 64: Runtime performances of xp_instr_att_close

Ultra Sparc II-400 [ms]
TBD

7.16 xp_run_init

7.16.1 Overview

The **xp_run_init** CFI function adds to the *run id* the s*at_nom_trans_id, sat_trans_id, instr_trans_id, atmos Id* and *dem Id.*

7.16.2 Calling interface

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

```
#include <explorer_pointing.h>
{
      long run_id;
      xp_sat_nom_trans_id sat_nom_trans_id;
      xp_sat_trans_id     sat_trans_id;
      xp_instr_trans_id   instr_trans_id;
      xp_atmos_id         atmos_id;
      xp_dem_id           dem_id;
      long ierr[XP_NUM_ERR_RUN_INIT], status;
      status = xp_run_init (&run_id, &sat_nom_trans_id,
                            &sat_trans_id, &instr_trans_id,
                            &atmos_id, &dem_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
```

7.16.3 Input parameters

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

Table 65: Input parameters of xp_run_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
run_id	long *	-	Run ID	-	>=0
sat_nom_trans_id	xp_sat_nom_trans_id*	-	Structure that contains the Instr. Trans.	-	-
sat_trans_id	xp_sat_trans_id*	-	Structure that contains the Instr. Trans.	-	-
instr_trans_id	xp_instr_trans_id*	-	Structure that contains the Instr. Trans.	-	-
atmos_id	xp_atmos_id*	-	Structure that contains the atmosphere initialisation.	-	-
dem_id	xp_dem_id*	-	Structure that contains the DEM initialisation.	-	-

7.16.4 Output parameters

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

Table 66: Output parameters of xp_run_init function

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

7.16.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xp_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_POINTING software library **xp_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 ex-

tended status flag returned by the **xp_run_init** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 67: Error messages of xl_run_init function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Wrong input run_id. It is not correctly initialized	No calculation performed	XP_CFI_RUN_INIT_STATUS_ERR	0
ERR	Memory allocation error	No calculation performed	XP_CFI_RUN_INIT_MEMORY_ERR	1
ERR	Incompatible input Ids	No calculation performed	XP_CFI_RUN_INIT_INCONSISTENCY_ERR	2

7.16.6 Runtime performances

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

Table 68: Runtime performances of xp_run_init function

Ultra Sparc II-400 [ms]
TBD

7.17 xp_run_get_ids

7.17.1 Overview

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

7.17.2 Calling interface

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

```
#include <explorer_pointing.h>
{
       long run_id;
       xp_sat_nom_trans_id sat_nom_trans_id;
       xp_sat_trans_id     sat_trans_id;
       xp_instr_trans_id   instr_trans_id;
       xp_atmos_id         atmos_id;
       xp_dem_id           dem_id;
       long status;
       status = xp_run_get_ids (&run_id,
                            &sat_nom_trans_id,
                            &sat_trans_id,
                            &instr_trans_id,
                            &atmos_id,
                            &dem_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
```

7.17.3 Input parameters

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

Table 69: Input parameters of xp_run_get_ids function

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

7.17.4 Output parameters

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

Table 70: Output parameters of xp_run_get_ids function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xl_run_get_ids	long	-	Status flag	-	-
sat_nom_trans_id	xp_sat_nom_trans_id*	-	Structure that contains the Instr. Trans.	-	-
sat_trans_id	xp_sat_trans_id*	-	Structure that contains the Instr. Trans.	-	-
instr_trans_id	xp_instr_trans_id*	-	Structure that contains the Instr. Trans.	-	-
atmos_id	xp_atmos_id*	-	Structure that contains the atmosphere initialisation.	-	-
dem_id	xp_dem_id*	-	Structure that contains the DEM initialisation.	-	-

7.17.5 Warnings and errors

TBW

7.17.6 Runtime performances

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

Table 71: Runtime performances of xp_run_get_ids function

Ultra Sparc II-400 [ms]
TBD

7.18 xp_run_close

7.18.1 Overview

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

7.18.2 Calling interface

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

```
#include <explorer_pointing.h>
{

        long run_id;
        status = xp_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_pointing.inc>

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

7.18.3 Input parameters

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

Table 72: Input parameters of xp_run_close function

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

7.18.4 Output parameters

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

Table 73: Output parameters of xp_run_close function

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

7.18.5 Warnings and errors

TBW

7.18.6 Runtime performances

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

Table 74: Runtime performances of xp_run_close function

Ultra Sparc II-400 [ms]
TBD

7.19 xp_attitude_init

7.19.1 Overview

The **xp_attitude_init** CFI function creates an empty *attitude Id*.

7.19.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      xp_attitude_id attitude_id;
      long ierr[XP_NUM_ERR_ATTITUDE_INIT], status;

      status = xp_attitude_init(&attitude_id, ierr);
}
```

The `XP_NUM_ERR_ATTITUDE_INIT` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 IERR(XP_NUM_ERR_ATTITUDE_INIT), STATUS

      STATUS = XP_ATTITUDE_INIT(SAT_ID, INSTRUMENT_ID,IERR)
```

7.19.3 Input Parameters

The **xp_attitude_init** CFI function has no input parameters.

7.19.4 Output Parameters

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

Table 75: Output parameters of xp_attitude_init

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude.	-	-
ierr	long	-	Error vector	-	-

7.19.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_attitude_init** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_attitude_init** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 76: Error messages of xp_attitude_init function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Memory allocation error	No calculation performed	XP_CFI_ATTITUDE_INIT_MEMORY_ERR	0

7.19.6 Runtime Performances

The following runtime performances have been measured.

Table 77: Runtime performances of xp_attitude_init

Ultra Sparc II-400 [ms]
TBD

7.20 xp_attitude_compute

7.20.1 Overview

The **xp_attitude_compute** CFI function calculates the Attitude Frame for a given S/C state vector.

7.20.2 Calling interface

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

```
#include <explorer_pointing.h>
{
      xl_time_id          time_id;
      xp_sat_nom_trans_id sat_nom_trans_id;
      xp_sat_trans_id     sat_trans_id;
      xp_instr_trans_id   instr_trans_id;
      xp_attitude_id      attitude_id;
      long time_ref, target_frame;
      double time, pos[3], vel[3], acc[3];
      long ierr[XP_NUM_ERR_ATTITUDE_COMPUTE];

      long xp_attitude_compute(&time_id,
                               &sat_nom_trans_id,
                               &sat_trans_id,
                               &instr_trans_id,
                               &attitude_id,        /* input / output */
                               &time_ref, &time, pos, vel, acc,
                               &target_frame,
                               ierr);

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

      long xp_attitude_compute_run(&run_id,
                               &attitude_id,        /* input / output */
                               &time_ref, &time, pos, vel, acc,
                               &target_frame,
                               ierr);
}
```

The `XP_NUM_ERR_ATTITUDE_COMPUTE` constant is defined in the file *explorer_pointing.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):

TBD

7.20.3 Input parameters

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

Table 78: Input parameters of xp_attitude_compute function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
sat_nom_trans_id	xp_sat_nom_trans_id*	-	Structure that contains the Instr. Trans.	-	-
sat_trans_id	xp_sat_trans_id*	-	Structure that contains the Instr. Trans.	-	-
instr_trans_id	xp_instr_trans_id*	-	Structure that contains the Instr. Trans.	-	-
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude (input/output)	-	-
time_ref	long *	-	Time reference ID	-	Complete
time	double	-	Time in Processing Format	Decimal days, MJD2000	[-18262.0,36524.0]
pos[3]	double	all	Satellite position vector (Earth Fixed CS)	m	-
vel[3]	double	all	Satellite velocity vector (Earth Fixed CS)	m/s	-
acc[3]	double	all	Satellite acceleration vector (Earth Fixed CS)	m/s^2	-
target_frame	long *	-	Attitude FrameID	-	Complete

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].
- Attitude Frame ID: attitude_frame_id. See current document, table 3

7.20.4 Output parameters

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

Table 79: Output parameters of xp_attitude_compute function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude. (input/output)	-	-
ierr	long	-	Error vector	-	-

7.20.5 Warnings and errors

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

Table 80: Error messages of xp_attitude_compute function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Time Id. not initialized	No calculation performed	XP_CFI_ATTITUDE_COMPUTE_TIME_STATUS_ERR	0
ERR	Instrument Trans. Id. not initialized	No calculation performed	XP_CFI_ATTITUDE_COMPUTE_INSTR_TRANS_STATUS_ERR	1
ERR	Satellite Att. Trans. not initialized	No calculation performed	XP_CFI_ATTITUDE_COMPUTE_SAT_TRANS_STATUS_ERR	2
ERR	Satellite Nom. Trans not initialized	No calculation performed	XP_CFI_ATTITUDE_COMPUTE_SAT_NOM_TRANS_STATUS_ERR	3
ERR	Attitude Id. not initialized	No calculation performed	XP_CFI_ATTITUDE_COMPUTE_ATTITUDE_STATUS_ERR	4
ERR	Wrong input time reference	No calculation performed	XP_CFI_ATTITUDE_COMPUTE_WRONG_TIME_REF_ERR	5

Table 80: Error messages of xp_attitude_compute function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Attitude Id is being used by another Id.	No calculation performed	XP_CFI_ATTITUDE_COMPUTE_BEING_USED_ERR	6
ERR	Could not compute orbit reference frame	No calculation performed	XP_CFI_ATTITUDE_COMPUTE_ORB_REF_ERR	7
ERR	Could not calculate AOCS parametes	No calculation performed	XP_CFI_ATTITUDE_COMPUTE_AOCS_CALC_ERR	8
ERR	Could not compute Sat. Nom. Trans frame	No calculation performed	XP_CFI_ATTITUDE_COMPUTE_SAT_NOM_TRANS_ERR	9
ERR	"Could not calculate the true latitude"	No calculation performed	XP_CFI_ATTITUDE_COMPUTE_TRUE_LAT_ERR	10
ERR	Could not calculate harmonic angles	No calculation performed	XP_CFI_ATTITUDE_COMPUTE_HARMONIC_CALC_ERR	11
ERR	Could not compute Sat.Trans. frame	No calculation performed	XP_CFI_ATTITUDE_COMPUTE_SAT_TRANS_ERR	12
ERR	Error computing direction cosine matrix from Sat. Att. to BJ2000	No calculation performed	XP_CFI_ATTITUDE_COMPUTE_ATT_TO_J2000_ERR	13
ERR	Could not compute Instrument Trans. frame	No calculation performed	XP_CFI_ATTITUDE_COMPUTE_INSTR_TRANS_ERR	14
ERR	Memory allocation error	No calculation performed	XP_CFI_ATTITUDE_COMPUTE_MEMORY_ERR	15

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 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 81: Runtime performances of xp_attitude_compute function

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

7.21 xp_attitude_user_set

7.21.1 Overview

The **xp_attitude_user_set** CFI function assigns a user defined Attitude Frame to the *attitude Id*.

7.21.2 Calling interface

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

```
#include <explorer_pointing.h>
{
      xl_time_id          time_id;
      xp_attitude_id      attitude_id;
      long time_ref, target_frame;
      double time, pos[3], vel[3], acc[3];
      double matrix[3][3];
      double matrix_rate[3][3];
      double matrix_rate_rate[3][3];
      double offset[3],;
      long ierr[XP_NUM_ERR_ATTITUDE_USER_SET];

      long xp_attitude_user_set(&time_id,
                                &attitude_id,       /* input / output */
                                &time_ref, &time, pos, vel, acc,
                                &target_frame,
                                matrix, matrix_rate, matrix_rate_rate,
                                offset,
                                ierr);

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

      long xp_attitude_user_set_run(&run_id,
                                &attitude_id,       /* input / output */
                                &time_ref, &time, pos, vel, acc,
                                &target_frame,
                                matrix, matrix_rate, matrix_rate_rate,
                                offset,
                                ierr);
}
```

The `XP_NUM_ERR_ATTITUDE_USER_SET` constant is defined in the file *explorer_pointing.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):

TBD

7.21.3 Input parameters

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

Table 82: Input parameters of xp_attitude_user_set function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude (input/output)	-	-
time_ref	long *	-	Time reference ID	-	Complete
time	double	-	Time in Processing Format	Decimal days, MJD2000	[-18262.0,36524.0]
pos[3]	double	all	Satellite position vector (Earth Fixed CS)	m	-
vel[3]	double	all	Satellite velocity vector (Earth Fixed CS)	m/s	-
acc[3]	double	all	Satellite acceleration vector (Earth Fixed CS)	m/s^2	-
target_frame	long *	-	Attitude FrameID	-	Complete
matrix[3][3]	double	all	Matrix representing the transformation from ToD to target_frame	-	-
matrix_rate [3][3]	double	all	Matrix representing the transformation rate from ToD to target_frame	-	-
matrix_rate_rate [3][3]	double	all	Matrix representing the transformation rate rate from ToD to target_frame	-	-
offset[3]	double	all	Offset in the reference frame origin	m	-

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].
- Attitude Frame ID: attitude_frame_id. See current document, table 3

7.21.4 Output parameters

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

Table 83: Output parameters of xp_attitude_user_set function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude. (input/output)	-	-
ierr	long	-	Error vector	-	-

7.21.5 Warnings and errors

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

Table 84: Error messages of xp_attitude_user_set function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Time Id. not initialized	No calculation performed	XP_CFI_ATTITUDE_USER_SET_TIME_STATUS_ERR	0
ERR	Wrong input target frame	No calculation performed	XP_CFI_ATTITUDE_USER_SET_WRONG_TARGET_FRAME_ERR	1
ERR	Attitude Id. not initialized	No calculation performed	XP_CFI_ATTITUDE_USER_SET_ATTITUDE_STATUS_ERR	2
ERR	Attitude Id is being used by another Id	No calculation performed	XP_CFI_ATTITUDE_USER_SET_BEING_USED_ERR	3
ERR	Could not compute orbit reference frame	No calculation performed	XP_CFI_ATTITUDE_USER_SET_ORB_REF_ERR	4
ERR	Memory allocation error	No calculation performed	XP_CFI_ATTITUDE_USER_SET_MEMORY_ERR	5

7.21.6 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 85: Runtime performances of xp_attitude_user_set function

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

7.22 xp_attitude_close

7.22.1 Overview

The **xp_attitude_close** CFI function cleans up any memory allocation performed by the Attitude functions.

.

7.22.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      xp_attitude_id attitude_id;
      long ierr[XP_NUM_ERR_ATTITUDE_CLOSE], status;

      status = xp_attitude_close(&attitude_id, ierr);
}
```

The `XP_NUM_ERR_ATTITUDE_CLOSE` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 IERR(XP_NUM_ERR_ATTITUDE_INIT), STATUS

      STATUS = XP_ATTITUDE_CLOSE(SAT_ID, INSTRUMENT_ID,IERR)
```

7.22.3 Input Parameters

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

Table 86: Input parameters of xp_attitude_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
attitude_id	xp_attitude_id *	-	Structure that contains the Attitude.	-	-

7.22.4 Output Parameters

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

Table 87: Output parameters of xp_attitude_close

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

7.22.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_attitude_close** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_attitude_close** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 88: Error messages of xp_attitude_close function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Could not close Attitude Id. The Attitude Id. is not initialized or it is being used	No calculation performed	XP_CFI_ATTITUDE_CLOSE_WRONG_ID_ERR	0

7.22.6 Runtime Performances

The following runtime performances have been measured.

Table 89: Runtime performances of xp_attitude_close

Ultra Sparc II-400 [ms]
TBD

7.23 xp_change_frame

7.23.1 Overview

The **xp_change_frame** CFI function changes the coordinate or attitude frame of a location or direction by keeping the location or direction in inertial space identical. Both all coordinate frames and all attitude frames are supported. When changing the frame for a location, the difference in origin of the frames is taken into account. While when changing the frame for a direction, the target is assumed to be at infinity.

7.23.2 Calling interface

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

```
#include <explorer_pointing.h>
{
      long sat_id, mode_flag, frame_flag_in, frame_id_in,
           frame_flag_out, frame_id_out, time_ref;
      xl_time_id          time_id;
      xp_sat_nom_trans_id sat_nom_trans_id;
      xp_sat_trans_id     sat_trans_id;
      xp_instr_trans_id   instr_trans_id;
      double time;
      double pos[3], vel[3], acc[3];
      long deriv;
      double vec_in[3], vec_rate_in[3], vec_rate_rate_in[3];
      double vec_out[3], vec_rate_out[3], vec_rate_rate_out[3];
      long ierr[XP_NUM_ERR_CHANGE_FRAME], status;
      status = xp_change_frame (&sat_id,
                                &time_id,
                                &sat_nom_trans_id,
                                &sat_trans_id,
                                &instr_trans_id,
                                &mode_flag,
                                &frame_flag_in, &frame_id_in,
                                &frame_flag_out, &frame_id_out,
                                &instrument_id, &time_ref, &time,
                                pos, vel, acc, &deriv,
                                vec_in, vec_rate_in, vec_rate_rate_in,
                                vec_out, vec_rate_out, vec_rate_rate_out,
                                ierr);
```

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

      status = xp_change_frame_run (&run_id,
                                    &mode_flag,
                                    &frame_flag_in, &frame_id_in,
                                    &frame_flag_out, &frame_id_out,
                                    &instrument_id, &time_ref, &time,
                                    pos, vel, acc, &deriv,
                                    vec_in, vec_rate_in, vec_rate_rate_in,
                                    vec_out, vec_rate_out, vec_rate_rate_out,
                                    ierr);

}
```

The XP_NUM_ERR_CHANGE_FRAME constant is defined in the file *explorer_pointing.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>

      LONG SAT_ID, MODE_FLAG, FRAME_FLAG_IN, FRAME_ID_IN,
&          FRAME_FLAG_OUT, FRAME_ID_OUT, INSTRUMENT_ID, TIME_REF
      REAL*8 TIME
      REAL*8 POS_IN(3),VEL_IN(3), ACC_IN(3)
      REAL*8 VEC_IN(3),VEC_RATE_IN(3), VEC_RATE_RATE_IN(3)
      REAL*8 VEC_OUT(3),VEC_RATE_OUT(3), VEC_RATE_RATE_OUT(3)
      INTEGER*4 IERR(XP_NUM_ERR_ATTITUDE), STATUS

      STATUS = XP_CHANGE_FRAME (SAT_ID, MODE_FLAG,
&                               FRAME_FLAG_IN, FRAME_ID_IN,
&                               FRAME_FLAG_OUT, FRAME_ID_OUT,
&                               INSTRUMENT_ID, TIME_REF, TIME,
&                               POS, VEL, ACC,
&                               VEC_IN, VEC_RATE_IN, VEC_RATE_RATE_IN,
&                               VEC_OUT, VEC_RATE_OUT, VEC_RATE_RATE_OUT,
&                               IERR)
```

7.23.3 Input parameters

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

Table 90: Input parameters of xp_change_frame 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.	-	-
sat_nom_trans_id	xp_sat_nom_trans_id*	-	Structure that contains the Instr. Trans.	-	-
sat_trans_id	xp_sat_trans_id*	-	Structure that contains the Instr. Trans.	-	-
instr_trans_id	xp_instr_trans_id*	-	Structure that contains the Instr. Trans.	-	-
mode_flag	long *	-	Selection of location or direction calculus		Complete
frame_flag_in	long *	-	Selection of Coordinate or Attitude Frame on input		Complete
frame_id_in	long *		Coordinate Frame id or Attitude Frame id on input		Complete
frame_flag_out	long *	-	Selection of Coordinate or Attitude Frame on output		Complete
frame_id_out	long *		Coordinate Frame id or Attitude Frame id on output		Complete
time_ref	long *	-	Time reference ID	-	Complete
time	double	-	Time in Processing Format	Decimal days, MJD2000	[-18262.0,36524.0]
pos[3]	double	all	Satellite position vector (Earth Fixed CS)	m	-
vel[3]	double	all	Satellite velocity vector (Earth Fixed CS)	m/s	-
acc[3]	double	all	Satellite acceleration vector (Earth Fixed CS)	m/s^2	-
deriv	long *	-	Derivative ID	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
vec_in[3]	double	all	Position (direction) vector (Frame in)	m or -	-
vec_rate_in[3]	double	all	Velocity (direction) vector (Frame in)	m/s or 1/s	-
vec_rate_rate_in[3]	double	all	Acceleration (direction) vector (Frame in)	m/s^2 or $1/s^2$	-

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

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

7.23.4 Output parameters

The output parameters of the **xp_change_frame** CFI function are

Table 91: Output parameters of xp_change_frame function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
vec_out[3]	double	all	Position (direction) vector (Frame out)	m or -	-
vec_rate_out[3]	double	all	Velocity (direction) vector (Frame out)	m/s or 1/s	-
vec_rate_rate_out[3]	double	all	Acceleration (direction) vector (Frame out)	m/s^2 or $1/s^2$	-
ierr	long	-	Error vector	-	-

7.23.5 Warnings and errors

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

Table 92: Error messages of xp_change_frame function

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

7.23.6 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 93: Runtime performances of xp_change_frame function

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

7.24 xp_atmos_init

7.24.1 Overview

The **xp_atmos_init** CFI function initialises the atmospheric model for a given satellite. The initalised values will be stored in the *atmos_id* output structure.

7.24.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long atmos_mode, atmos_model;
      char atmos_file[XL_MAX_STR];
      xp_atmos_id *atmos_id;
      long ierr[XP_NUM_ERR_ATMOS_INIT], status;

      status = xp_atmos_init(&atmos_mode, &atmos_model,atmos_file,
                             &atmos_id, ierr);
}
```

The `XP_NUM_ERR_ATMOS_INIT` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID
      INTEGER*4 ATMOS_MODE, ATMOS_MODEL
      CHARACTER*XL_MAX_STR ATMOS_FILE
      INTEGER*4 IERR(XP_NUM_ERR_ATMOS_INIT), STATUS

      STATUS = XP_ATMOS_INIT(SAT_ID, ATMOS_MODE, ATMOS_MODEL,
&                            ATMOS_FILE, IERR)
```

7.24.3 Input Parameters

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

Table 94: Input parameters of xp_atmos_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
atmos_mode	long *	-	Atmosphere initialization mode	-	Complete
atmos_model	long *	-	Atmospheric model (to be used when the XP_COMPLEX_INIT mode is selected)	-	Complete
atmos_file	char[]	-	File used for atmosphere initialization	-	Complete

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

- Atmosphere Initialization Mode: atmos_mode. See current document, table 3.
- Atmosphere Model: atmos_model. See current document, table 3.

7.24.4 Output Parameters

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

Table 95: Output parameters of xp_atmos_init

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
atmos_id	xp_atmos_id*	-	Structure that contains the atmosphere initialisation.	-	-
ierr	long	-	Error vector	-	-

7.24.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_atmos_init** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_atmos_init** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 96: Error messages of xp_atmos_init function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Atmosphere Mode ID is not correct	No calculation performed	XP_CFI_ATMOS_INIT_MODE_ID_ERR	0
ERR	Atmosphere Model ID is not correct	No calculation performed	XP_CFI_ATMOS_INIT_MODEL_ID_ERR	1
ERR	Atmosphere initialization file could not be opened	No calculation performed	XP_CFI_ATMOS_INIT_FILE_NOT_OPEN_ERR	2
ERR	Unable to store atmosphere initialization file (not enough memory	No calculation performed	XP_CFI_ATMOS_INIT_MEMORY_ERR	3
ERR	Error while reading atmos-phere initialization file	No calculation performed	XP_CFI_ATMOS_INIT_FILE_READING_ERR	4

7.24.6 Runtime Performances

The following runtime performances have been measured.

Table 97: Runtime performances of xp_atmos_init

Ultra Sparc II-400 [ms]
TBD

7.25 xp_atmos_close

7.25.1 Overview

The **xp_atmos_close** CFI function cleans up any memory allocation performed by the **xp_atmos_init** functions.

.

7.25.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      xp_atmos_id atmos_id;
      long ierr[XP_NUM_ERR_ATMOS_CLOSE], status;

      status = xp_atmos_close(&atmos_id, ierr);
}
```

The `XP_NUM_ERR_ATMOS_CLOSE` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 IERR(XP_NUM_ERR_ATMOS_CLOSE), STATUS

      STATUS = XP_ATMOS_CLOSE(SAT_ID, INSTRUMENT_ID,IERR)
```

7.25.3 Input Parameters

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

Table 98: Input parameters of xp_atmos_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
atmos_id	xp_atmos_id*	-	Structure that contains the atmosphere initialisation.	-	-

7.25.4 Output Parameters

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

Table 99: Output parameters of xp_atmos_close

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

7.25.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_atmos_close** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_atmos_close** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 100: Error messages of xp_atmos_close function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Could not close the Atmos. Id. as it is not initialized or it is being used	No calculation performed	XP_CFI_ATMOS_CLOSE_WRONG_ID_ERR	0

7.25.6 Runtime Performances

The following runtime performances have been measured.

Table 101: Runtime performances of xp_atmos_close

Ultra Sparc II-400 [ms]
TBD

7.26 xp_dem_init

7.26.1 Overview

The **xp_dem_init** CFI function initialises the digital elevation model for a given satellite. The initalised values will be stored in the *dem_id* output structure.

7.26.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
       long mode, model;
       char dem_file[XL_MAX_STR];
       xp_dem_id *dem_id;
       long ierr[XP_NUM_ERR_DEM_INIT], status;

       status = xp_dem_init(&mode, &model, dem_file, &dem_id, ierr);
}
```

The `XP_NUM_ERR_DEM_INIT` constant is defined in the file *explorer_pointing.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>
       INTEGER*4 SAT_ID
       INTEGER*4 DEM_MODE
       CHARACTER*XL_MAX_STR DEM_FILE
       INTEGER*4 IERR(XP_NUM_ERR_DEM_INIT), STATUS

       STATUS = XP_DEM_INIT(SAT_ID, DEM_MODE, DEM_FILE, IERR)
```

7.26.3 Input Parameters

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

Table 102: Input parameters of xp_dem_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
mode	long *	-	Digital Elevation Model initialization mode	-	Complete
model	long *	-	Digital Elevation Model initialization model	-	Complete
dem_file	char[]	-	File used for DEM initialization	-	Complete

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

- DEM Initilization Mode: dem_mode. See current document, TBW.

7.26.4 Output Parameters

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

Table 103: Output parameters of xp_dem_init

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
dem_id	xp_dem_id*	-	Structure that contains the DEM initialisation.	-	-
ierr	long	-	Error vector	-	-

7.26.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_dem_init** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_dem_init** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 104: Error messages of xp_dem_init function

Error type	Error message	Cause and impact	Error code	Error No
ERR	DEM Mode ID is not correct	No calculation performed	XP_CFI_DEM_INIT_MODE_ID_ERR	0
ERR	DEM Model ID is not correct	No calculation performed	XP_CFI_DEM_INIT_MODEL_ID_ERR	1
ERR	DEM initialization file could not be opened	No calculation performed	XP_CFI_DEM_INIT_FILE_NOT_OPEN_ERR	2
ERR	Unable to store DEM initialization file (not enough memory)	No calculation performed	XP_CFI_DEM_INIT_MEMORY_ERR	3
ERR	Error while reading DEM initialization file	No calculation performed	XP_CFI_DEM_INIT_FILE_READING_ERR	4

7.26.6 Runtime Performances

The following runtime performances have been measured.

Table 105: Runtime performances of xp_dem_init

Ultra Sparc II-400 [ms]
TBD

7.27 xp_dem_close

7.27.1 Overview

The **xp_dem_close** CFI function cleans up any memory allocation performed by the **xp_dem_init** functions.

.

7.27.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      xp_dem_id dem_id;
      long ierr[XP_NUM_ERR_DEM_CLOSE], status;

      status = xp_dem_close(&dem_id, ierr);
}
```

The `XP_NUM_ERR_DEM_CLOSE` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 IERR(XP_NUM_ERR_DEM_CLOSE), STATUS

      STATUS = XP_DEM_CLOSE(SAT_ID, INSTRUMENT_ID,IERR)
```

7.27.3 Input Parameters

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

Table 106: Input parameters of xp_dem_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
dem_id	xp_dem_id*	-	Structure that contains the DEM initialisation.	-	-

7.27.4 Output Parameters

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

Table 107: Output parameters of xp_dem_close

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

7.27.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_dem_close** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_dem_close** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 108: Error messages of xp_dem_close function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Could not close the Dem. Id. as it is not initialized or it is being used	No calculation performed	XP_CFI_DEM_CLOSE_WRONG_ID_ERR	0

7.27.6 Runtime Performances

The following runtime performances have been measured.

Table 109: Runtime performances of xp_dem_close

Ultra Sparc II-400 [ms]
TBD

7.28 xp_target_inter

7.28.1 Overview

The **xp_target_inter** CFI function computes the first or the second intersection point of the line or sight from the satellite (defined by an elevation and an azimuth angle expressed in the selected Attitude Frame) with a surface located at a certain geodetic altitude over the Earth.

7.28.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long            sat_id;
      xp_attitude_id attitude_id;
      xp_atmos_id     atmos_id;
      xp_dem_id       dem_id;
      xp_target_id    target_id;
      long deriv, inter_flag, iray;
      double los_az, los_el, geod_alt, los_az_rate, los_el_rate, freq;
      long ierr[XP_NUM_ERR_TARGET_INTER], status, num_user_target,
          num_los_target;

      status = xp_target_inter(&sat_id,
                &attitude_id,
                &atmos_id,
                &dem_id,
                &deriv, &inter_flag, &los_az, &los_el, &geod_alt,
                &los_az_rate, &los_el_rate, &iray, &freq,
                &num_user_target, &num_los_target,
                &target_id, ierr);

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

      status = xp_target_inter_run(&run_id,
                &attitude_id,
                &deriv, &inter_flag, &los_az, &los_el, &geod_alt,
                &los_az_rate, &los_el_rate, &iray, &freq,
                &num_user_target, &num_los_target,
```

```
                &target_id, ierr);

}
```

The XP_NUM_ERR_TARGET_INTER constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, ATTITUDE_FRAME_ID, INSTRUMENT_ID, TIME_REF
      INTEGER*4 DERIV, INTER_FLAG, IRAY
      REAL*8 TIME
      REAL*8 POS(3), VEL(3), ACC(3)
      REAL*8 LOS_AZ, LOS_EL, GEOD_ALT, LOS_AZ_RATE, LOS_EL_RATE, FREQ
      INTEGER*4 IERR(XP_NUM_ERR_TARGET_INTER), STATUS, NUM_USER_TARGET,
&             NUM_LOS_TARGET


      STATUS = XP_TARGET_INTER(SAT_ID, ATTITUDE_FRAME_ID,
&                              INSTRUMENT_ID, TIME_REF,
&                              TIME, POS, VEL, ACC, DERIV, INTER_FLAG,
&                              LOS_AZ, LOS_EL, GEOD_ALT, LOS_AZ_RATE,
&                              LOS_EL_RATE, IRAY, FREQ,
&                              NUM_USER_TARGET, NUM_LOS_TARGET, IERR)
```

7.28.3 Input Parameters

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

Table 110: Input parameters of xp_target_inter function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude. (input/output)	-	-
atmos_id	xp_atmos_id*	-	Structure that contains the atmosphere initialisation.	-	-
dem_id	xp_dem_id*	-	Structure that contains the DEM initialisation.	-	-
deriv	long *	-	Derivative ID	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
inter_flag	long *	-	Flag for first or second inter-section point selection	-	Allowed values: (1) XP_INTER_1ST (2) XP_INTER_2ND
los_az	double *	-	Azimuth of the LOS (Attitude Frame)	deg	>= 0 < 360
los_el	double *	-	Elevation of the LOS (Attitude Frame)	deg	>= -90 <= 90
geod_alt	double *	-	Geodetic altitude over the Earth	m	>= $-b_{WGS}$
los_az_rate	double *	-	Azimuth-rate of the LOS (Attitude Frame)	deg/s	-
los_el_rate	double *	-	Elevation-rate of the LOS (Attitude Frame)	deg/s	-
iray	long *	-	Ray tracing model switch	-	Accepted values: (0) XP_NO_REF (1) XP_STD_REF (2) XP_USER_REF
freq	double *	-	Frequency of the signal	Hz	>= 0

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

- Derivative switch: deriv. See current document, table 3.
- Intersection flag: inter_flag. See current document, table 3.
- Ray tracing model switch: iray. See current document, table 3.

7.28.4 Output Parameters

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

Table 111: Output parameters of xp_target_inter

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
num_user_target	long*	-	Number of user defined targets calculated	-	>= 0 (Set to 1 for non multi-target routines)
num_los_target	long*	-	Number of LOS targets calculated	-	>= 0
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
ierr	long	-	Error vector	-	-

7.28.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_inter** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_inter** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 112: Error messages of xp_target_inter function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Attitude Id. is not initialized	No calculation performed	XP_CFI_TARGET_INTER_ATTITUDE_STATUS_ERR	0
ERR	Intersection flag is not correct	No calculation performed	XP_CFI_TARGET_INTER_INTER_FLAG_ERR	1
ERR	Invalid Frequency	No calculation performed	XP_CFI_TARGET_INTER_FREQ_ERR	2
ERR	Atmospheric model has not been initialised	No calculation performed	XP_CFI_TARGET_INTER_ATM_NOT_INIT_ERR	3
ERR	Atmosphere initialisation is not compatible with Ray Tracing Model	No calculation performed	XP_CFI_TARGET_INTER_ATM_INIT_IRAY_COMPATIB_ERR	4
ERR	Time reference ID is not correct	No calculation performed	XP_CFI_TARGET_INTER_TIME_REF_ERR	5

Table 112: Error messages of xp_target_inter function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Deriv flag is not correct	No calculation performed	XP_CFI_TARGET_INTER_DERIV_FLAG_ERR	6
ERR	Ray Tracing Model ID is not correct	No calculation performed	XP_CFI_TARGET_INTER_IRAY_ID_ERR	7
ERR	Input state vector does not satisfy loose tolerance requirement	No calculation performed	XP_CFI_TARGET_INTER_INVALID_SV_ERR	8
ERR	Invalid LOS Azimuth	No calculation performed	XP_CFI_TARGET_INTER_LOS_AZIMUTH_ERR	9
ERR	Invalid LOS Elevation	No calculation performed	XP_CFI_TARGET_INTER_LOS_ELEVATION_ERR	10
ERR	Invalid Geodetic Altitude	No calculation performed	XP_CFI_TARGET_INTER_GEODETIC_ALT_ERR	11
ERR	Memory allocation error	No calculation performed	XP_CFI_TARGET_INTER_MEMORY_ERR	12
ERR	Internal computation error # 3	No calculation performed	XP_CFI_TARGET_INTER_INITIAL_LOOK_DIR_OR_PLANE_ERR	13
ERR	Time Reference not initialised	No calculation performed	XP_CFI_TARGET_INTER_TIME_REF_INIT_ERR	14
ERR	No target was found	No calculation performed	XP_CFI_TARGET_INTER_TARGET_NOT_FOUND_ERR	15
ERR	Internal computation error # 4	No calculation performed	XP_CFI_TARGET_INTER_RANGE_OR_POINTING_CALC_ERR	16
WARN	Input state vector does not satisfy tight tolerance requirement	Calculation performed. A message informs the user.	XP_CFI_TARGET_INTER_INVALID_SV_WARN	23
WARN	Path from satellite to target occulted by the Earth	Calculation performed. A message informs the user.	XP_CFI_TARGET_INTER_NEGATIVE_ALTITUDE_WARN	24

7.28.6 Runtime Performances

The following runtime performances have been measured.

Table 113: Runtime performances of xp_target_inter

Ultra Sparc II-400 [ms]
0.696

7.29 xp_target_ground_range

7.29.1 Overview

The **xp_target_ground_range** CFI function computes the location of a point that is placed on a surface at a certain geodetic altitude over the Earth, that lays on the plane defined by the satellite position, the nadir and a reference point, and that is at a certain distance or ground range measured along that surface from that reference point.

This reference point is calculated being the intersection of the previous surface with the line of sight defined by an elevation and azimuth angle in the selected Attitude Frame.

7.29.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long            sat_id;
      xp_attitude_id attitude_id;
      xp_dem_id       dem_id;
      xp_target_id    target_id;
      long deriv;
      double los_az, los_el, geod_alt, distance;
      double los_az_rate, los_el_rate;
      long ierr[XP_NUM_ERR_TARGET_GROUND_RANGE], status,
              num_user_target, num_los_target;

      status = xp_target_ground_range(&sat_id,
               &attitude_id,
               &dem_id,
               &deriv, &los_az,
               &los_el, &geod_alt, &distance, &los_az_rate,
               &los_el_rate, &num_user_target, &num_los_target,
               &target_id, ierr);

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

      status = xp_target_ground_range_run(&run_id,
               &attitude_id,
               &deriv, &los_az,
```

```
                &los_el, &geod_alt, &distance, &los_az_rate,
                &los_el_rate, &num_user_target, &num_los_target,
                &target_id, ierr);
}
```

The XP_NUM_ERR_TARGET_GROUND_RANGE constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, ATTITUDE_FRAME_ID, INSTRUMENT_ID,
&               TIME_REF, DERIV
      REAL*8 TIME
      REAL*8 POS(3), VEL(3), ACC(3)
      REAL*8 LOS_AZ, LOS_EL, GEOD_ALT, DISTANCE
      REAL*8 LOS_AZ_RATE, LOS_EL_RATE
      INTEGER*4 IERR(XP_NUM_ERR_TARGET_GROUND_RANGE), STATUS,
&          NUM_USER_TARGET, NUM_LOS_TARGET

      STATUS = XP_TARGET_GROUND_RANGE(SAT_ID, ATTITUDE_FRAME_ID,
&                                 INSTRUMENT_ID, TIME_REF,
&                                 TIME, POS, VEL, ACC, DERIV, LOS_AZ,
&                                 LOS_EL, GEOD_ALT, DISTANCE, LOS_AZ_RATE,
&                                 LOS_EL_RATE, NUM_USER_TARGET,
&                                 NUM_LOS_TARGET, IERR)
```

7.29.3 Input Parameters

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

Table 114: Input parameters of xp_target_ground_range function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude. (input/output)	-	-
dem_id	xp_dem_id*	-	Structure that contains the DEM initialisation.	-	-
deriv	long *	-	Derivative ID	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
los_az	double *	-	Azimuth of the LOS (Attitude Frame)	deg	>= 0 < 360
los_el	double *	-	Elevation of the LOS (Attitude Frame)	deg	>= -90 <= 90
geod_alt	double *	-	Geodetic altitude over the Earth (Earth fixed CS)	m	>= $-b_{WGS}$
distance	double *	-	Distance or ground range to the reference point, positive from nadir in the azimuth direction (Earth Fixed CS)	m	-
los_az_rate	double *	-	Azimuth-rate of the LOS (Attitude Frame)	deg/s	-
los_el_rate	double *	-	Elevation-rate of the LOS (Attitude Frame)	deg/s	-

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

- Derivative switch: deriv. See current document, table 3.

7.29.4 Output Parameters

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

Table 115: Output parameters of xp_target_ground_range

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
num_user_target	long*	-	Number of user defined targets calculated	-	>= 0 (Set to 1 for non multi-target routines)
num_los_target	long*	-	Number of LOS targets calculated	-	>= 0
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
ierr	long	-	Error vector	-	-

7.29.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_ground_range** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_ground_range** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 116: Error messages of xp_target_ground_range function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Attitude Id. is not initialized	No calculation performed	XP_CFI_TARGET_GR_RANGE_ATTITUDE_STATUS_ERR	0
ERR	Time reference ID is not correct	No calculation performed	XP_CFI_TARGET_GR_RANGE_TIME_REF_ERR	1
ERR	Deriv flag is not correct	No calculation performed	XP_CFI_TARGET_GR_RANGE_DERIV_FLAG_ERR	2
ERR	Invalid LOS Azimuth	No calculation performed	XP_CFI_TARGET_GR_RANGE_LOS_AZIMUTH_ERR	3
ERR	Invalid LOS Elevation	No calculation performed	XP_CFI_TARGET_GR_RANGE_LOS_ELEVATION_ERR	4

Table 116: Error messages of xp_target_ground_range function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Invalid Geodetic Altitude	No calculation performed	XP_CFI_TARGET_GR_RANGE_GEODETIC_ALT_ERR	5
ERR	Input state vector does not satisfy loose tolerance requirement	No calculation performed	XP_CFI_TARGET_GR_RANGE_INVALID_SV_ERR	6
ERR	Memory allocation error	No calculation performed	XP_CFI_TARGET_GR_RANGE_MEMORY_ERR	7
ERR	Internal computation error # 3	No calculation performed	XP_CFI_TARGET_GR_RANGE_INITIAL_LOOK_DIR_OR_PLANE_ERR	8
ERR	Time reference is not initialized	No calculation performed	XP_CFI_TARGET_GR_RANGE_TIME_REF_INIT_ERR	9
ERR	No target was found	No calculation performed	XP_CFI_TARGET_GR_RANGE_TARGET_NOT_FOUND_ERR	10
ERR	Internal computation error #4	No calculation performed	XP_CFI_TARGET_GR_RANGE_RANGE_OR_POINTING_CALC_ERR	11
WARN	Input state vector does not satisfy tight tolerance requirement	Calculation performed. A message informs the user.	XP_CFI_TARGET_GR_RANGE_INVALID_SV_WARN	18

7.29.6 Runtime Performances

The following runtime performances have been measured.

Table 117: Runtime performances of xp_target_ground_range

Ultra Sparc II-400 [ms]
0.624

7.30 xp_target_incidence_angle

7.30.1 Overview

The **xp_target_incidence_angle** CFI function computes the location of a point that is placed on a surface at a certain geodetic altitude over the Earth and that is seen from the satellite on a line of sight that forms a certain azimuth angle in the selected Attitude Frame and that intersects that surface with a certain incidence angle.

7.30.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long           sat_id;
      xp_attitude_id attitude_id;
      xp_dem_id      dem_id;
      xp_target_id   target_id;
      long deriv;
      double los_az, inc_angle, geod_alt, los_az_rate;
      long ierr[XP_NUM_ERR_TARGET_INCIDENCE_ANGLE], status,
              num_user_target, num_los_target;

      status = xp_target_incidence_angle(&sat_id,
               &attitude_id,
               &dem_id,
               &deriv, &los_az,
               &inc_angle, &geod_alt, &los_az_rate,
               &num_user_target, &num_los_target,
               &target_id, ierr);

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

      status = xp_target_incidence_angle_run(&run_id,
               &attitude_id,
               &deriv, &los_az,
               &inc_angle, &geod_alt, &los_az_rate,
               &num_user_target, &num_los_target,
               &target_id, ierr);
```

```
}
```

The `XP_NUM_ERR_TARGET_INCIDENCE_ANGLE` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, ATTITUDE_FRAME_ID, INSTRUMENT_ID, TIME_REF,
&          DERIV
      REAL*8 TIME
      REAL*8 POS(3), VEL(3), ACC(3)
      REAL*8 LOS_AZ, INC_ANGLE, GEOD_ALT, LOS_AZ_RATE
      INTEGER*4 IERR(XP_NUM_ERR_TARGET_INCIDENCE_ANGLE), STATUS,
&             NUM_USER_TARGET, NUM_LOS_TARGET

      STATUS = XP_TARGET_INCIDENCE_ANGLE(SAT_ID, ATTITUDE_FRAME_ID,
&                           INSTRUMENT_ID,
&                           TIME_REF, TIME, POS, VEL, ACC, DERIV,
&                           LOS_AZ, INC_ANGLE, GEOD_ALT, LOS_AZ_RATE,
&                           NUM_USER_TARGET, NUM_LOS_TARGET, IERR)
```

7.30.3 Input Parameters

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

Table 118: Input parameters of xp_target_incidence_angle function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude. (input/output)	-	-
dem_id	xp_dem_id *	-	Structure that contains the DEM initialisation.	-	-
deriv	long *	-	Derivative ID	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
los_az	double *	-	Azimuth of the LOS (Attitude Frame)	deg	>= 0 < 360
inc_angle	double *	-	Incidence angle of the LOS (Earth fixed CS)	deg	>= 0 <= 90
geod_alt	double *	-	Geodetic altitude over the Earth (Earth fixed CS)	m	>= $-b_{WGS}$
los_az_rate	double *	-	Azimuth-rate of the LOS (Attitude Frame)	deg/s	-

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

- Derivative switch: deriv. See current document, table 3.

7.30.4 Output Parameters

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

Table 119: Output parameters of xp_target_incidence_angle

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
num_user_target	long*	-	Number of user defined targets calculated	-	>= 0 (Set to 1 for non multi-target routines)
num_los_target	long*	-	Number of LOS targets calculated	-	>= 0
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
ierr	long	-	Error vector	-	-

7.30.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_incidence_angle** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_incidence_angle** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 120: Error messages of xp_target_incidence_angle function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Attitude Id. is not initialized	No calculation performed	XP_CFI_TARGET_INC_ANGLE_ATTITUDE_STATUS_ERR	0
ERR	Deriv flag is not correct	No calculation performed	XP_CFI_TARGET_INC_ANGLE_DERIV_FLAG_ERR	1
ERR	Invalid LOS Azimuth	No calculation performed	XP_CFI_TARGET_INC_ANGLE_LOS_AZIMUTH_ERR	2
ERR	Invalid Incidence Angle	No calculation performed	XP_CFI_TARGET_INC_ANGLE_INC_ANGLE_ERR	3
ERR	Invalid Geodetic Altitude	No calculation performed	XP_CFI_TARGET_INC_ANGLE_GEODETIC_ALT_ERR	4

Table 120: Error messages of xp_target_incidence_angle function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Input state vector does not satisfy loose tolerance requirement	No calculation performed	XP_CFI_TARGET_INC_ANGLE_INVALID_SV_ERR	5
ERR	Memory allocation error	No calculation performed	XP_CFI_TARGET_INC_ANGLE_MEMORY_ERR	6
ERR	Internal computation error #3	No calculation performed	XP_CFI_TARGET_INC_ANGLE_INITIAL_LOOK_DIR_OR_PLANE_ERR	7
ERR	Time Reference not initialised	No calculation performed	XP_CFI_TARGET_INC_ANGLE_TIME_REF_INIT_ERR	8
ERR	No target was found	No calculation performed	XP_CFI_TARGET_INC_ANGLE_TARGET_NOT_FOUND_ERR	9
WARN	Input state vector does not satisfy tight tolerance requirement	Calculation performed. A message informs the user.	XP_CFI_TARGET_INC_ANGLE_INVALID_SV_WARN	11

7.30.6 Runtime Performances

The following runtime performances have been measured.

Table 121: Runtime performances of xp_target_incidence_angle

Ultra Sparc II-400 [ms]
1.064

7.31 xp_target_range

7.31.1 Overview

The **xp_target_range** CFI function computes the location of a point that is placed on a surface at a certain geodetic altitude over the Earth, that is seen from the satellite on a line of sight that forms a certain azimuth angle in the selected Attitude Frame, and that is at a certain range or slant-range from the satellite.

7.31.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long            sat_id;
      xp_attitude_id attitude_id;
      xp_dem_id       dem_id;
      xp_target_id    target_id;
      long deriv;
      double los_az, range, geod_alt, los_az_rate, range_rate;
      long ierr[XP_NUM_ERR_TARGET_RANGE], status, num_user_target,
              num_los_target;

      status = xp_target_range(&sat_id,
               &attitude_id,
               &dem_id,
               &deriv, &los_az, &range,
               &geod_alt, &los_az_rate, &lrange_rate,
               &num_user_target,&num_los_target,
               &target_id, ierr);

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

      status = xp_target_range_run(&run_id,
               &attitude_id,
               &deriv, &los_az, &range,
               &geod_alt, &los_az_rate, &lrange_rate,
               &num_user_target,&num_los_target,
               &target_id, ierr);
}
```

The XP_NUM_ERR_TARGET_RANGE constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, ATTITUDE_FRAME_ID, INSTRUMENT_ID,
&               TIME_REF, DERIV
      REAL*8 TIME
      REAL*8 POS(3), VEL(3), ACC(3)
      REAL*8 LOS_AZ, RANGE, GEOD_ALT, LOS_AZ_RATE, RANGE_RATE
      INTEGER*4 IERR(XP_NUM_ERR_TARGET_RANGE), STATUS, NUM_USER_TARGET,
&             NUM_LOS_TARGET

      STATUS = XP_TARGET_RANGE(SAT_ID, ATTITUDE_FRAME_ID,
&                              INSTRUMENT_ID, TIME_REF,
&                              TIME, POS, VEL, ACC, DERIV, LOS_AZ,
&                              RANGE, GEOD_ALT, LOS_AZ_RATE,
&                              RANGE_RATE, NUM_USER_TARGET,
&                              NUM_LOS_TARGET, IERR)
```

7.31.3 Input Parameters

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

Table 122: Input parameters of xp_target_range function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude. (input/output)	-	-
dem_id	xp_dem_id *	-	Structure that contains the DEM initialisation.	-	-
deriv	long *	-	Derivative ID	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
los_az	double *	-	Azimuth of the LOS (Attitude Frame)	deg	>= 0 < 360
range	double *	-	Range to the satellite (Earth fixed CS)	m	> 0
geod_alt	double *	-	Geodetic altitude over the Earth	m	>= $-b_{WGS}$
los_az_rate	double *	-	Azimuth-rate of the LOS (Attitude Frame)	deg/s	-
range_rate	double *	-	Range-rate to the satellite (Earth fixed CS)	m/s	-

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

- Derivative switch: deriv. See current document, table 3.

7.31.4 Output Parameters

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

Table 123: Output parameters of xp_target_range

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
num_user_target	long*	-	Number of user defined targets calculated	-	>= 0 (Set to 1 for non multi-target routines)
num_los_target	long*	-	Number of LOS targets calculated	-	>= 0
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
ierr	long	-	Error vector	-	-

7.31.5 Warnings and Errors

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

Table 124: Error messages of xp_target_range function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Attitude Id. is not initialized	No calculation performed	XP_CFI_TARGET_RANGE_ATTITUDE_STATUS_ERR	0
ERR	Deriv flag is not correct	No calculation performed	XP_CFI_TARGET_RANGE_DERIV_FLAG_ERR	1
ERR	Invalid LOS Azimuth	No calculation performed	XP_CFI_TARGET_RANGE_LOS_AZIMUTH_ERR	2
ERR	Invalid Range	No calculation performed	XP_CFI_TARGET_RANGE_RANGE_ERR	3
ERR	Invalid Geodetic Altitude	No calculation performed	XP_CFI_TARGET_RANGE_GEODETIC_ALT_ERR	4
ERR	Input state vector does not satisfy loose tolerance requirement	No calculation performed	XP_CFI_TARGET_RANGE_INVALID_SV_ERR	5

Table 124: Error messages of xp_target_range function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Memory allocation error	No calculation performed	XP_CFI_TARGET_RANGE_MEMORY_ERR	6
ERR	Internal computation error #3	No calculation performed	XP_CFI_TARGET_RANGE_INITIAL_LOOK_DIR_OR_PLANE_ERR	7
ERR	Could not perform a time transformation	No calculation performed	XP_CFI_TARGET_RANGE_TIME_TRANSFORMATION_ERR	8
ERR	No target was found	No calculation performed	XP_CFI_TARGET_RANGE_TARGET_NOT_FOUND_ERR	9
WARN	Input state vector does not satisfy tight tolerance requirement	Calculation performed. A message informs the user.	XP_CFI_TARGET_RANGE_INVALID_SV_WARN	11

7.31.6 Runtime Performances

The following runtime performances have been measured.

Table 125: Runtime performances of xp_target_range

Ultra Sparc II-400 [ms]
0.198

7.32 xp_target_range_rate

7.32.1 Overview

The **xp_target_range_rate** CFI function computes the location of a point that is placed on a surface at a certain geodetic altitude over the Earth, that is at a certain range from the satellite, and whose associated Earth-fixed target has a certain range-rate value.

7.32.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long            sat_id;
      xp_attitude_id attitude_id;
      xp_dem_id       dem_id;
      xp_target_id    target_id;
      long deriv;
      double ef_range_rate, range, geod_alt;
      double ef_range_rate_rate, range_rate;
      long ierr[XP_NUM_ERR_TARGET_RANGE_RATE], status, num_user_target,
              num_los_target;

      status = xp_target_range_rate(&sat_id,
               &attitude_id,
               &dem_id,
               &deriv, &ef_range_rate, &range,
               &geod_alt, &ef_range_rate_rate, &range_rate,
               &num_user_target, &num_los_target, &target_id, ierr);

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

      status = xp_target_range_rate_run(&run_id,
               &attitude_id,
               &deriv, &ef_range_rate, &range,
               &geod_alt, &ef_range_rate_rate, &range_rate,
               &num_user_target, &num_los_target, &target_id, ierr);
}
```

The XP_NUM_ERR_TARGET_RANGE_RATE constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, ATTITUDE_FRAME_ID, INSTRUMENT_ID,
&               TIME_REF, DERIV
      REAL*8 TIME
      REAL*8 POS(3), VEL(3), ACC(3)
      REAL*8 EF_RANGE_RATE, RANGE, GEOD_ALT
      REAL*8 EF_RANGE_RATE_RATE, RANGE_RATE
      INTEGER*4 IERR(XP_NUM_ERR_TARGET_RANGE_RATE), STATUS,
&             NUM_USER_TARGET, NUM_LOS_TARGET


      STATUS = XP_TARGET_RANGE_RATE(SAT_ID, ATTITUDE_FRAME_ID,
&                                   INSTRUMENT_ID, TIME_REF,
&                                   TIME, POS, VEL, ACC, DERIV,
&                                   EF_RANGE_RATE, RANGE, GEOD_ALT,
&                                   EF_RANGE_RATE_RATE, RANGE_RATE,
&                                   NUM_USER_TARGET, NUM_LOS_TARGET, IERR)
```

7.32.3 Input Parameters

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

Table 126: Input parameters of xp_target_range_rate function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude. (input/output)	-	-
dem_id	xp_dem_id *	-	Structure that contains the DEM initialisation.	-	-
attitude_frame_id	long *	-	Attitude Frame ID	-	Complete
deriv	long *	-	Derivative ID	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
ef_range_rate	double *	-	Range-rate of the related Earth-fixed target (Earth fixed CS)	m/s	-
range	double *	-	Range or slant-range from target to satellite (Earth fixed CS)	m	> 0
geod_alt	double *	-	Geodetic altitude over the Earth	m	>= $-b_{WGS}$
ef_range_rate_rate	double *	-	Range-rate-rate of the related Earth-fixed target (Earth fixed CS)	m/s^2	-
range_rate	double *	-	Range-rate from target to satellite (Earth fixed CS)	m/s	-

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

- Derivative switch: deriv. See current document, table 3.

7.32.4 Output Parameters

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

Table 127: Output parameters of xp_target_range_rate

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
num_user_target	long*	-	Number of user defined targets calculated	-	>= 0 (Set to 1 for non multi-target routines)
num_los_target	long*	-	Number of LOS targets calculated	-	>= 0
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
ierr	long	-	Error vector	-	-

7.32.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_range_rate** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_range_rate** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 128: Error messages of xp_target_range_rate function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Attitude Id. is not initialized	No calculation performed	XP_CFI_TARGET_RANGE_RATE_ATTITUDE_STATUS_ERR	0
ERR	Deriv flag is not correct	No calculation performed	XP_CFI_TARGET_RANGE_RATE_DERIV_FLAG_ERR	1
ERR	Invalid Range	No calculation performed	XP_CFI_TARGET_RANGE_RATE_RANGE_ERR	2
ERR	Invalid Geodetic Altitude	No calculation performed	XP_CFI_TARGET_RANGE_RATE_GEODETIC_ALT_ERR	3
ERR	Input state vector does not satisfy loose tolerance requirement	No calculation performed	XP_CFI_TARGET_RANGE_RATE_INVALID_SV_ERR	4

Table 128: Error messages of xp_target_range_rate function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Memory allocation error	No calculation performed	XP_CFI_TARGET_RANGE_RATE_MEMORY_ERR	5
ERR	Time Reference not initialised	No calculation performed	XP_CFI_TARGET_RANGE_RATE_TIME_REF_INIT_ERR	6
ERR	Internal computation error #3	No calculation performed	XP_CFI_TARGET_RANGE_RATE_INITIAL_LOOK_DIR_OR_PLANE_ERR	7
ERR	No target was found	No calculation performed	XP_CFI_TARGET_RANGE_RATE_TARGET_NOT_FOUND_ERR	8
WARN	Input state vector does not satisfy tight tolerance requirement	Calculation performed. A message informs the user.	XP_CFI_TARGET_RANGE_RATE_INVALID_SV_WARN	9

7.32.6 Runtime Performances

The following runtime performances have been measured.

Table 129: Runtime performances of xp_target_range_rate

Ultra Sparc II-400 [ms]
0.220

7.33 xp_target_tangent

7.33.1 Overview

The **xp_target_tangent** CFI function computes the location of the tangent point over the Earth that is located on the line of sight defined by an elevation and azimuth angles expressed in the selected Attitude Frame.

7.33.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long            sat_id;
      xp_attitude_id attitude_id;
      xp_atmos_id     atmos_id;
      xp_dem_id       dem_id;
      xp_target_id    target_id;
      long deriv, iray;
      double los_az, los_el, los_az_rate, los_el_rate, freq;
      long ierr[XP_NUM_ERR_TARGET_TANGENT], status, num_user_target,
          num_los_target;

      status = xp_target_tangent(&sat_id,
                &attitude_id,
                &atmos_id,
                &dem_id,
                &deriv, &los_az, &los_el,
                &los_az_rate, &los_el_rate, &iray, &freq,
                &num_user_target, &num_los_target,
                &target_id, ierr);

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

      status = xp_target_tangent_run(&run_id,
                &attitude_id,
                &deriv, &los_az, &los_el,
                &los_az_rate, &los_el_rate, &iray, &freq,
                &num_user_target, &num_los_target,
```

```
                &target_id, ierr);
}
```

The `XP_NUM_ERR_TARGET_TANGENT` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, ATTITUDE_FRAME_ID, INSTRUMENT_ID,
&               TIME_REF, DERIV, IRAY
      REAL*8 TIME
      REAL*8 POS(3), VEL(3), ACC(3)
      REAL*8 LOS_AZ, LOS_EL, LOS_AZ_RATE, LOS_EL_RATE, FREQ
      INTEGER*4 IERR(XP_NUM_ERR_TARGET_TANGENT), STATUS,
&             NUM_USER_TARGET, NUM_LOS_TARGET

      STATUS = XP_TARGET_TANGENT(SAT_ID, ATTITUDE_FRAME_ID,
&                                INSTRUMENT_ID, TIME_REF,
&                                TIME, POS, VEL, ACC, DERIV, LOS_AZ,
&                                LOS_EL, LOS_AZ_RATE, LOS_EL_RATE,
&                                IRAY, FREQ, NUM_USER_TARGET,
&                                NUM_LOS_TARGET, IERR)
```

7.33.3 Input Parameters

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

Table 130: Input parameters of xp_target_tangent function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude. (input/output)	-	-
atmos_id	xp_atmos_id*	-	Structure that contains the atmosphere initialisation.	-	-
dem_id	xp_dem_id*	-	Structure that contains the DEM initialisation.	-	-
deriv	long *	-	Derivative ID	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
los_az	double *	-	Azimuth of the LOS (Attitude Frame)	deg	>= 0 < 360
los_el	double *	-	Elevation of the LOS (Attitude Frame)	deg	>= -90 <= 90
los_az_rate	double *	-	Azimuth-rate of the LOS (Attitude Frame)	deg/s	-
los_el_rate	double *	-	Elevation-rate of the LOS (Attitude Frame)	deg/s	-
iray	long *	-	Ray tracing model switch	-	Accepted values: All (see table 3)
freq	double *	-	Frequency of the signal	Hz	>= 0

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

- Derivative switch: deriv. See current document, table 3.
- Ray tracing model switch: iray. See current document, table 3.

7.33.4 Output Parameters

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

Table 131: Output parameters of xp_target_tangent

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
num_user_target	long*	-	Number of user defined targets calculated	-	>= 0 (Set to 1 for non multi-target routines)
num_los_target	long*	-	Number of LOS targets calculated	-	>= 0
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
ierr	long	-	Error vector	-	-

7.33.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_tangent** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_tangent** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 132: Error messages of xp_target_tangent function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Attitude Id. is not initialized	No calculation performed	XP_CFI_TARGET_TANGENT_ATTITUDE_STATUS_ERR	0
ERR	Deriv flag is not correct	No calculation performed	XP_CFI_TARGET_TANGENT_DERIV_FLAG_ERR	1
ERR	Invalid LOS Azimuth	No calculation performed	XP_CFI_TARGET_TANGENT_LOS_AZIMUTH_ERR	2
ERR	Invalid LOS Elevation	No calculation performed	XP_CFI_TARGET_TANGENT_LOS_ELEVATION_ERR	3
ERR	Ray Tracing Model ID is not correct	No calculation performed	XP_CFI_TARGET_TANGENT_IRAY_ID_ERR	4
ERR	Invalid Frequency	No calculation performed	XP_CFI_TARGET_TANGENT_FREQ_ERR	5

Table 132: Error messages of xp_target_tangent function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Atmospheric model has not been initialised	No calculation performed	XP_CFI_TARGET_TANGENT_ATM_NOT_INIT_ERR	6
ERR	Atmosphere initialisation is not compatible with Ray Tracing Model	No calculation performed	XP_CFI_TARGET_TANGENT_ATM_INIT_IRAY_COMPATIB_ERR	7
ERR	Input state vector does not satisfy loose tolerance requirement	No calculation performed	XP_CFI_TARGET_TANGENT_INVALID_SV_ERR	8
ERR	Memory allocation error	No calculation performed	XP_CFI_TARGET_TANGENT_MEMORY_ERR	9
ERR	Internal computation error # 3	No calculation performed	XP_CFI_TARGET_TANGENT_INITIAL_LOOK_DIR_OR_PLANE_ERR	10
ERR	Time Reference not initialised	No calculation performed	XP_CFI_TARGET_TANGENT_TIME_REF_INIT_ERR	11
ERR	No target was found	No calculation performed	XP_CFI_TARGET_TANGENT_TARGET_NOT_FOUND_ERR	12
ERR	Tangent point is behind looking direction	No calculation performed	XP_CFI_TARGET_TANGENT_TG_PT_BEHIND_LOOK_DIR_ERR	13
ERR	Internal computation error # 4	No calculation performed	XP_CFI_TARGET_TANGENT_RANGE_OR_POINTING_CALC_ERR	14
WARN	Input state vector does not satisfy tight tolerance requirement	Calculation performed. A message informs the user.	XP_CFI_TARGET_TANGENT_INVALID_SV_WARN	15
WARN	Path from satellite to target occulted by the Earth	Calculation performed. A message informs the user.	XP_CFI_TARGET_TANGENT_NEGATIVE_ALTITUDE_WARN	16
WARN	Tangent point latitude is outside the selected corrective function latitude band	Calculation performed. A message informs the user.	XP_CFI_TARGET_TANGENT_PRED_WRONG_LAT_WARN	17

7.33.6 Runtime Performances

The following runtime performances have been measured.

Table 133: Runtime performances of xp_target_tangent

Ultra Sparc II-400 [ms]
0.746

7.34 xp_target_altitude

7.34.1 Overview

The **xp_target_altitude** CFI function computes the location of the tangent point over the Earth that is located on a surface at a certain geodetic altitude over the Earth and that is on a line of sight that forms a certain azimuth angle in the selected Attitude Frame.

7.34.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long            sat_id;
      xp_attitude_id attitude_id;
      xp_atmos_id     atmos_id;
      xp_dem_id       dem_id;
      xp_target_id    target_id;
      long deriv, iray;
      double los_az, geod_alt, los_az_rate, freq;
      long ierr[XP_NUM_ERR_TARGET_ALTITUDE], status, num_user_target,
              num_los_target;

      status = xp_target_altitude(sat_id,
               &attitude_id,
               &atmos_id,
               &dem_id,
               &deriv, &los_az, &geod_alt,
               &los_az_rate, &iray, &freq,
               &num_user_target, &num_los_target,
               &target_id, ierr);

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

      status = xp_target_altitude_run(run_id,
               &attitude_id,
               &deriv, &los_az, &geod_alt,
               &los_az_rate, &iray, &freq,
               &num_user_target, &num_los_target,
```

```
                &target_id, ierr);
}
```

The XP_NUM_ERR_TARGET_ALTITUDE constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID,ATTITUDE_FRAME_ID, INSTRUMENT_ID,
&               TIME_REF, DERIV, IRAY
      REAL*8 TIME, POS(3), VEL(3), ACC(3)
      REAL*8 LOS_AZ, GEOD_ALT, LOS_AZ_RATE, FREQ
      INTEGER*4 IERR(XP_NUM_ERR_TARGET_ALTITUDE), STATUS,
&             NUM_USER_TARGET, NUM_LOS_TARGET

      STATUS = XP_TARGET_ALTITUDE(SAT_ID, ATTITUDE_FRAME_ID,
&                                 INSTRUMENT_ID, TIME_REF,
&                                 TIME, POS, VEL, ACC, DERIV, LOS_AZ,
&                                 GEOD_ALT, LOS_AZ_RATE, IRAY, FREQ,
&                                 NUM_USER_TARGET, NUM_LOS_TARGET, IERR)
```

7.34.3 Input Parameters

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

Table 134: Input parameters of xp_target_altitude function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude. (input/output)	-	-
atmos_id	xp_atmos_id*	-	Structure that contains the atmosphere initialisation.	-	-
dem_id	xp_dem_id*	-	Structure that contains the DEM initialisation.	-	-
deriv	long *	-	Derivative ID	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
los_az	double *	-	Azimuth of the LOS (Attitude Frame)	deg	>= 0 < 360
geod_alt	double *	-	Geodetic altitude over the Earth	m	>= $-b_{WGS}$
los_az_rate	double *	-	Azimuth-rate of the LOS (Attitude Frame)	deg/s	-
iray	long *	-	Ray tracing model switch	-	Accepted values: All (see table 3)
freq	double *	-	Frequency of the signal	Hz	>= 0

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

- Derivative switch: deriv. See current document, table 3.
- Ray tracing model switch: iray. See current document, table 3.

7.34.4 Output Parameters

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

Table 135: Output parameters of xp_target_altitude

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
num_user_target	long*	-	Number of user defined targets calculated	-	>= 0 (Set to 1 for non multi-target routines)
num_los_target	long*	-	Number of LOS targets calculated	-	>= 0
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
ierr	long	-	Error vector	-	-

7.34.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_altitude** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_altitude** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 136: Error messages of xp_target_altitude function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Attitude Id. is not initialized	No calculation performed	XP_CFI_TARGET_ALT_ATTITUDE_STATUS_ERR	0
ERR	Deriv flag is not correct	No calculation performed	XP_CFI_TARGET_ALT_DERIV_FLAG_ERR	1
ERR	Invalid LOS Azimuth	No calculation performed	XP_CFI_TARGET_ALT_LOS_AZIMUTH_ERR	2
ERR	Invalid Geodetic Altitude	No calculation performed	XP_CFI_TARGET_ALT_GEODETIC_ALT_ERR	3
ERR	Ray Tracing Model ID is not correct	No calculation performed	XP_CFI_TARGET_ALT_IRAY_ID_ERR	4
ERR	Invalid Frequency	No calculation performed	XP_CFI_TARGET_ALT_FREQ_ERR	5
ERR	Atmospheric model has not been initialised	No calculation performed	XP_CFI_TARGET_ALT_ATM_NOT_INIT_ERR	6

Table 136: Error messages of xp_target_altitude function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Atmosphere initialisation is not compatible with Ray Tracing Model	No calculation performed	XP_CFI_TARGET_ALT_ATM_INIT_IRAY_COMPATIB_ERR	7
ERR	Input state vector does not satisfy loose tolerance requirement	No calculation performed	XP_CFI_TARGET_ALT_INVALID_SV_ERR	8
ERR	Memory allocation error	No calculation performed	XP_CFI_TARGET_ALT_MEMORY_ERR	9
ERR	Time Reference not initialised	No calculation performed	XP_CFI_TARGET_ALT_TIME_REF_INIT_ERR	10
ERR	Internal computation error #3	No calculation performed	XP_CFI_TARGET_ALT_INITIAL_LOOK_DIR_OR_PLANE_ERR	11
ERR	No target was found	No calculation performed	XP_CFI_TARGET_ALT_TARGET_NOT_FOUND_ERR	12
ERR	Internal computation error #4	No calculation performed	XP_CFI_TARGET_ALT_RANGE_OR_POINTING_CALC_ERR	13
WARN	Input state vector does not satisfy tight tolerance requirement	Calculation performed. A message informs the user.	XP_TARGET_ALT_INVALID_SV_WARN	14
WARN	Path from satellite to target occulted by the Earth	Calculation performed. A message informs the user.	XP_CFI_TARGET_ALT_NEGATIVE_ALTITUDE_WARN	15
WARN	Tangent point latitude is outside the selected corrective function latitude band	Calculation performed. A message informs the user.	XP_CFI_TARGET_ALT_PRED_WRONG_LAT_WARN	16

7.34.6 Runtime Performances

The following runtime performances have been measured.

Table 137: Runtime performances of xp_target_altitude

Ultra Sparc II-400 [ms]
0.716

7.35 xp_target_star

7.35.1 Overview

The **xp_target_star** CFI function computes the location of the tangent point over the Earth that is located on the line of sight that points to a star defined by its right ascension and declination coordinates.

7.35.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long            sat_id;
      xp_attitude_id attitude_id;
      xp_atmos_id     atmos_id;
      xp_dem_id       dem_id;
      xp_target_id    target_id;
      long deriv, iray;
      double star_ra, star_dec, star_ra_rate, star_dec_rate, freq;
      long ierr[XP_NUM_ERR_TARGET_STAR], status, num_user_target,
              num_los_target;

      status = xp_target_star(&sat_id,
               &attitude_id,
               &atmos_id,
               &dem_id,
               &deriv, &star_ra, star_dec,
               &&star_ra_rate, &star_dec_rate, &iray, &freq,
               &num_user_target, &num_los_target,
               &target_id, ierr);

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

      status = xp_target_star_run(&run_id,
               &attitude_id,
               &deriv, &star_ra, star_dec,
               &&star_ra_rate, &star_dec_rate, &iray, &freq,
               &num_user_target, &num_los_target,
               &target_id, ierr);
```

```
}
```

The `XP_NUM_ERR_TARGET_STAR` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, ATTITUDE_FRAME_ID, INSTRUMENT_ID,
&               TIME_REF, DERIV, IRAY
      REAL*8 TIME, POS(3), VEL(3), ACC(3)
      REAL*8 LOS_AZ, LOS_EL, GEOD_ALT, LOS_AZ_RATE, LOS_EL_RATE, FREQ
      INTEGER*4 IERR(XP_NUM_ERR_TARGET_STAR), STATUS, NUM_USER_TARGET,
&             NUM_LOS_TARGET

      STATUS = XP_TARGET_STAR(SAT_ID, ATTITUDE_FRAME_ID,
&                             INSTRUMENT_ID, TIME_REF,
&                             TIME, POS, VEL, ACC, DERIV, STAR_RA,
&                             STAR_DEC, STAR_RA_RATE, STAR_DEC_RATE,
&                             IRAY, FREQ, NUM_USER_TARGET,
&                             NUM_LOS_TARGET, IERR)
```

7.35.3 Input Parameters

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

Table 138: Input parameters of xp_target_star function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude. (input/output)	-	-
atmos_id	xp_atmos_id*	-	Structure that contains the atmosphere initialisation.	-	-
dem_id	xp_dem_id*	-	Structure that contains the DEM initialisation.	-	-
deriv	long *	-	Derivative ID	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
star_ra	double *	-	Right ascension of the star (True of Date CS)	deg	>= 0 < 360
star_dec	double *	-	Declination of the star (True of Date CS)	deg	>= -90 <= +90
star_ra_rate	double *	-	Right ascension rate of the star (True of Date CS)	deg/s	-
star_dec_rate	double *	-	Declination rate of the star (True of Date CS)	deg/s	-
iray	long *	-	Ray tracing model switch	-	Accepted values: All (see table 3)
freq	double *	-	Frequency of the signal	Hz	>= 0

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

- Derivative switch: deriv. See current document, table 3.
- Ray tracing model switch: iray. See current document, table 3.

7.35.4 Output Parameters

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

Table 139: Output parameters of xp_target_star

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
num_user_target	long*	-	Number of user defined targets calculated	-	>= 0 (Set to 1 for non multi-target routines)
num_los_target	long*	-	Number of LOS targets calculated	-	>= 0
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
ierr	long	-	Error vector	-	-

7.35.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_star** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_star** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 140: Error messages of xp_target_star function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Attitude Id. is not initialized	No calculation performed	XP_CFI_TARGET_STAR_ATTITUDE_STATUS_ERR	0
ERR	Deriv flag is not correct	No calculation performed	XP_CFI_TARGET_STAR_DERIV_FLAG_ERR	1
ERR	Invalid Right Ascension of the star	No calculation performed	XP_CFI_TARGET_STAR_RA_ERR	2
ERR	Invalid Declination of the star	No calculation performed	XP_CFI_TARGET_STAR_DEC_ERR	3
ERR	Ray Tracing Model ID is not correct	No calculation performed	XP_CFI_TARGET_STAR_IRAY_ID_ERR	4
ERR	Invalid Frequency	No calculation performed	XP_CFI_TARGET_STAR_FREQ_ERR	5
ERR	Atmospheric model has not been initialised	No calculation performed	XP_CFI_TARGET_STAR_ATM_NOT_INIT_ERR	6

Table 140: Error messages of xp_target_star function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Atmosphere initialisation is not compatible with Ray Tracing Model	No calculation performed	XP_CFI_TARGET_STAR_ATM_INIT_IRAY_COMPATIB_ERR	7
ERR	Input state vector does not satisfy loose tolerance requirement	No calculation performed	XP_CFI_TARGET_STAR_INVALID_SV_ERR	8
ERR	Memory allocation error	No calculation performed	XP_CFI_TARGET_STAR_MEMORY_ERR	9
ERR	Internal computation error # 3	No calculation performed	XP_CFI_TARGET_STAR_INITIAL_LOOK_DIR_OR_PLANE_ERR	10
ERR	Time reference ID is not correct	No calculation performed	XP_CFI_TARGET_STAR_TIME_REF_INIT_ERR	11
ERR	No target was found	No calculation performed	XP_CFI_TARGET_STAR_TARGET_NOT_FOUND_ERR	12
ERR	Tangent point is behind looking direction	No calculation performed	XP_CFI_TARGET_STAR_TG_PT_BEHIND_LOOK_DIR_ERR	13
ERR	Internal computation error # 4	No calculation performed	XP_CFI_TARGET_STAR_RANGE_OR_POINTING_CALC_ERR	14
WARN	Input state vector does not satisfy tight tolerance requirement	Calculation performed. A message informs the user.	XP_CFI_TARGET_STAR_INVALID_SV_WARN	15
WARN	Path from satellite to target occulted by the Earth	Calculation performed. A message informs the user.	XP_CFI_TARGET_STAR_NEGATIVE_ALTITUDE_WARN	16
WARN	Tangent point latitude is outside the selected corrective function latitude band	Calculation performed. A message informs the user.	XP_CFI_TARGET_STAR_PRED_WRONG_LAT_WARN	17

7.35.6 Runtime Performances

The following runtime performances have been measured.

Table 141: Runtime performances of xp_target_star

Ultra Sparc II-400 [ms]
0.750

7.36 xp_target_station

7.36.1 Overview

The **xp_target_station** CFI function computes the most relevant observation parameters of the link between the satellite and a ground station.

7.36.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long            sat_id;
      xp_attitude_id attitude_id;
      xp_dem_id       dem_id;
      xp_target_id    target_id;
      long deriv;
      double geoc_long, geod_lat, geod_alt, min_link_el;
      long ierr[XP_NUM_ERR_TARGET_STATION], status, num_user_target,
              num_los_target;

      status = xp_target_station(&sat_id,
               &attitude_id,
               &dem_id,
               &deriv, &geoc_long, &geod_lat,
               &geod_alt, &min_link_el,
               &num_user_target, &num_los_target,
               &target_id, ierr);

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

      status = xp_target_station_run(&run_id,
               &attitude_id,
               &deriv, &geoc_long, &geod_lat,
               &geod_alt, &min_link_el,
               &num_user_target, &num_los_target,
               &target_id, ierr);
}
```

The XP_NUM_ERR_TARGET_STATION constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, ATTITUDE_FRAME_ID, INSTRUMENT_ID,
&               TIME_REF, DERIV
      REAL*8 TIME
      REAL*8 POS(3), VEL(3), ACC(3)
      REAL*8 GEOC_LONG, GEOD_LAT, GEOD_ALT, MIN_LINK_EL
      INTEGER*4 IERR(XP_NUM_ERR_TARGET_STATION), STATUS,
&          NUM_USER_TARGET, NUM_LOS_TARGET

      STATUS = XP_TARGET_STATION(SAT_ID, ATTITUDE_FRAME_ID,
&                                INSTRUMENT_ID, TIME_REF,
&                                TIME, POS, VEL, ACC, DERIV, GEOC_LONG,
&                                GEOD_LAT, GEOD_ALT, MIN_LINK_EL,
&                                NUM_USER_TARGET, NUM_LOS_TARGET, IERR)
```

7.36.3 Input Parameters

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

Table 142: Input parameters of xp_target_station function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude. (input/output)	-	-
dem_id	xp_dem_id *	-	Structure that contains the DEM initialisation.	-	-
deriv	long *	-	Derivative ID	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
geoc_long	double *	-	GS geocentric longitude (Earth fixed CS)	deg	>= 0 < 360
geod_lat	double *	-	GS geodetic latitude (Earth fixed CS)	deg	>= -90 < = 90
geod_alt	double *	-	GS geodetic altitude (Earth fixed CS)	m	>= $-b_{WGS}$
min_link_el	double *	-	GS minimum link elevation (Topocentric CS)	deg	>= -90 <= 90

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

- Derivative switch: deriv. See current document, table 3.

7.36.4 Output Parameters

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

Table 143: Output parameters of xp_target_station

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
num_user_target	long*	-	Number of user defined targets calculated	-	>= 0 (Set to 1 for non multi-target routines)
num_los_target	long*	-	Number of LOS targets calculated	-	>= 0
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
ierr	long	-	Error vector	-	-

7.36.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_station** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_station** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 144: Error messages of xp_target_station function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Attitude Id. is not initialized	No calculation performed	XP_CFI_TARGET_STATION_ATTITUDE_STATUS_ERR	0
ERR	Deriv flag is not correct	No calculation performed	XP_CFI_TARGET_STATION_DERIV_FLAG_ERR	1
ERR	Invalid GS Geocentric Longitude	No calculation performed	XP_CFI_TARGET_STATION_GEOC_LONG_ERR	2
ERR	Invalid GS Geodetic Latitude	No calculation performed	XP_CFI_TARGET_STATION_GEOD_LAT_ERR	3
ERR	Invalid GS Geodetic Altitude	No calculation performed	XP_CFI_TARGET_STATION_GEODETIC_ALT_ERR	4
ERR	Invalid GS Minimum Link Elevation	No calculation performed	XP_CFI_TARGET_STATION_MIN_LINK_EL_ERR	5
ERR	Input state vector does not satisfy loose tolerance requirement	No calculation performed	XP_CFI_TARGET_STATION_INVALID_SV_ERR	6

Table 144: Error messages of xp_target_station function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Memory allocation error	No calculation performed	XP_CFI_TARGET_STATION_MEMORY_ERR	7
ERR	Internal computation error #3	No calculation performed	XP_CFI_TARGET_STATION_STAVIS_COMP_FAILED_ERR	8
WARN	Input state vector does not satisfy tight tolerance requirement	Calculation performed. A message informs the user.	XP_CFI_TARGET_STATION_INVALID_SV_WARN	17

7.36.6 Runtime Performances

The following runtime performances have been measured.

Table 145: Runtime performances of xp_target_station

Ultra Sparc II-400 [ms]
0.350

7.37 xp_target_drs

7.37.1 Overview

The **xp_target_drs** CFI function computes the most relevant observation parameters of the link between the satellite and a Data Relay Satellite (DRS).

7.37.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long            sat_id;
      xp_attitude_id attitude_id;
      xp_dem_id       dem_id;
      xp_target_id    target_id;
      long deriv;
      double drs_pos[3], drs_vel[3];
      long ierr[XP_NUM_ERR_TARGET_DRS], status, num_user_target,
          num_los_target;

      status = xp_target_drs(&sat_id,
               &attitude_id,
               &dem_id,
               &deriv, &drs_pos, &drs_vel,
               &num_user_target, &num_los_target,
               &target_id, ierr);

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

      status = xp_target_drs_run(&run_id,
               &attitude_id,
               &deriv, &drs_pos, &drs_vel,
               &num_user_target, &num_los_target,
               &target_id, ierr);
}
```

The `XP_NUM_ERR_TARGET_DRS` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, ATTITUDE_FRAME_ID, INSTRUMENT_ID,
&               TIME_REF, DERIV
      REAL*8 TIME
      REAL*8 POS(3), VEL(3), ACC(3), DRS_POS(3), DRS_VEL(3)
      INTEGER*4 IERR(XP_NUM_ERR_TARGET_DRS), STATUS, NUM_USER_TARGET,
&          NUM_LOS_TARGET

      STATUS = XP_TARGET_DRS(SAT_ID, ATTITUDE_FRAME_ID,
&                            INSTRUMENT_ID, TIME_REF,
&                            TIME, POS, VEL, ACC, DERIV, DRS_POS,
&                            DRS_VEL, NUM_USER_TARGET,
&                            NUM_LOS_TARGET, IERR)
```

7.37.3 Input Parameters

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

Table 146: Input parameters of xp_target_drs function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude. (input/output)	-	-
dem_id	xp_dem_id *	-	Structure that contains the DEM initialisation.	-	-
deriv	long *	-	Derivative ID	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
drs_pos	double[3]	[0-2]	DRS position vector (Earth Fixed CS)	m	-
drs_vel	double[3]	[0-2]	DRS velocity vector (Earth Fixed CS)	m/s	-

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

- Derivative switch: deriv. See current document, table 3.

7.37.4 Output Parameters

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

Table 147: Output parameters of xp_target_drs

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
num_user_target	long*	-	Number of user defined targets calculated	-	>= 0 (Set to 1 for non multi-target routines)
num_los_target	long*	-	Number of LOS targets calculated	-	>= 0
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
ierr	long	-	Error vector	-	-

7.37.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_drs** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_drs** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 148: Error messages of xp_target_drs function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Attitude Id. is not initialized	No calculation performed	XP_CFI_TARGET_DRS_ATTITUDE_STATUS_ERR	0
ERR	Deriv flag is not correct	No calculation performed	XP_CFI_TARGET_DRS_DERIV_FLAG_ERR	1
ERR	Input state vector does not satisfy loose tolerance requirement	No calculation performed	XP_CFI_TARGET_DRS_INVALID_SV_ERR	2
ERR	Memory allocation error	No calculation performed	XP_CFI_TARGET_DRS_MEMORY_ERR	3
ERR	Input DRS state vector does not satisfy loose tolerance requirement	No calculation performed	XP_CFI_TARGET_DRS_INVALID_DRS_SV_ERR	4
ERR	Internal computation error #3	No calculation performed	XP_CFI_TARGET_DRS_DRSVIS_COMP_FAILED_ERR	5

Table 148: Error messages of xp_target_drs function

Error type	Error message	Cause and impact	Error code	Error No
WARN	Input state vector does not satisfy tight tolerance requirement	Calculation performed. A message informs the user.	XP_CFI_TARGET_DRS_INVALID_SV_WARN	6
WARN	Input DRS state vector does not satisfy tight tolerance requirement	Calculation performed. A message informs the user.	XP_CFI_TARGET_DRS_INVALID_DRS_SV_WARN	7
WARN	Path from Satellite to DRS occulted by the Earth	Calculation performed. A message informs the user.	XP_CFI_TARGET_DRS_NEGATIVE_TANG_DRS_ALT_WARN	8
WARN	Internal computation warning # 1	Calculation performed. A message informs the user.	XP_CFI_TARGET_DRS_NEGATIVE_TANG_SUN_ALT_WARN	9
WARN	Internal computation warning # 2	Calculation performed. A message informs the user.	XP_CFI_TARGET_DRS_TANGENT_DRS_NOT_VALID_WARN	10
WARN	Internal computation warning # 3	Calculation performed. A message informs the user.	XP_CFI_TARGET_DRS_TANGENT_SUN_NOT_VALID_WARN	11

7.37.6 Runtime Performances

The following runtime performances have been measured.

Table 149: Runtime performances of xp_target_drs

Ultra Sparc II-400 [ms]
0.800

7.38 xp_target_generic

7.38.1 Overview

The **xp_target_generic** CFI function allows the user to provide the target location (position and velocity) and later calculate extra results from it.

7.38.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long            sat_id;
      xp_attitude_id attitude_id;
      xp_dem_id       dem_id;
      xp_target_id    target_id;
      long deriv;
      double targ_pos[3], targ_vel[3], targ_acc[3];
      long ierr[XP_NUM_ERR_TARGET_GENERIC], status, num_user_target,
              num_los_target;

      status = xp_target_generic(&sat_id,
               &attitude_id,
               &atmos_id,
               &dem_id,
               &deriv, &targ_pos, &targ_vel, &targ_acc,
               &num_user_target, &num_los_target,
               &target_id, ierr);

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

      status = xp_target_generic_run(&run_id,
               &attitude_id,
               &deriv, &targ_pos, &targ_vel, &targ_acc,
               &num_user_target, &num_los_target,
               &target_id, ierr);
}
```

The XP_NUM_ERR_TARGET_GENERIC constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, ATTITUDE_FRAME_ID, INSTRUMENT_ID
      INTEGER*4 TIME_REF, DERIV
      REAL*8 TIME
      REAL*8 POS(3), VEL(3), ACC(3), TARG_POS(3), TARG_VEL(3)
      INTEGER*4 IERR(XP_NUM_ERR_TARGET_GENERIC), STATUS,
&             NUM_USER_TARGET, NUM_LOS_TARGET

      STATUS = XP_TARGET_GENERIC(SAT_ID, ATTITUDE_FRAME_ID,
&                                INSTRUMENT_ID, TIME_REF,
&                                TIME, POS, VEL, ACC, DERIV, TARG_POS,
&                                TARG_VEL, NUM_USER_TARGET,
&                                NUM_LOS_TARGET, IERR)
```

7.38.3 Input Parameters

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

Table 150: Input parameters of xp_target_generic function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude. (input/output)	-	-
dem_id	xp_dem_id *	-	Structure that contains the DEM initialisation.	-	-
deriv	long *	-	Derivative ID	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
targ_pos	double[3]	[0-2]	Target position vector (Earth Fixed CS)	m	-
targ_vel	double[3]	[0-2]	Target velocity vector (Earth Fixed CS)	m/s	-
targ_acc	double[3]	[0-2]	Target acceleration vector (Earth Fixed CS)	m/s^2	-

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

- Derivative switch: deriv. See current document, table 3.

7.38.4 Output Parameters

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

Table 151: Output parameters of xp_target_generic

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
num_user_target	long*	-	Number of user defined targets calculated	-	>= 0 (Set to 1 for non multi-target routines)
num_los_target	long*	-	Number of LOS targets calculated	-	>= 0
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
ierr	long	-	Error vector	-	-

7.38.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_generic** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_generic** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 152: Error messages of xp_target_generic function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Attitude Id. is not initialized	No calculation performed	XP_CFI_TARGET_GENERIC_ATTITUDE_STATUS_ERR	0
ERR	Deriv flag is not correct	No calculation performed	XP_CFI_TARGET_GENERIC_DERIV_FLAG_ERR	1
ERR	Input state vector does not satisfy loose tolerance requirement	No calculation performed	XP_CFI_TARGET_GENERIC_INVALID_SV_ERR	2
ERR	Memory allocation error	No calculation performed	XP_CFI_TARGET_GENERIC_MEMORY_ERR	3
ERR	Internal computation error # 3	No calculation performed	XP_CFI_TARGET_GENERIC_INITIAL_LOOK_DIR_OR_PLANE_ERR	4

Table 152: Error messages of xp_target_generic function

Error type	Error message	Cause and impact	Error code	Error No
WARN	Input state vector does not satisfy tight tolerance requirement	Calculation performed. A message informs the user.	XP_CFI_TARGET_GENERIC_INVALID_SV_WARN	14

7.38.6 Runtime Performances

The following runtime performances have been measured.

Table 153: Runtime performances of xp_target_generic

Ultra Sparc II-400 [ms]
TBD

7.39 xp_target_travel_time

7.39.1 Overview

The **xp_target_travel_time** CFI function computes the point of the line or sight from the satellite (defined by an elevation and an azimuth angle expressed in the selected Attitude Frame) at a given travel time along the (curved) line of sight.

7.39.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long            sat_id;
      xp_attitude_id attitude_id;
      xp_atmos_id     atmos_id;
      xp_dem_id       dem_id;
      xp_target_id    target_id;
      long deriv, iray;
      double los_az, los_el, travel_time
      double los_az_rate, los_el_rate, travel_time_rate, freq;
      long ierr[XP_NUM_ERR_TARGET_TRAVEL_TIME], status,
              num_user_target, num_los_target;

      status = xp_target_travel_time(&sat_id,
               &attitude_id,
               &atmos_id,
               &dem_id,
               &deriv, &los_az,
               &los_el, &travel_time, &los_az_rate, &los_el_rate,
               &travel_time_rate, &iray, &freq,
               &num_user_target, &num_los_target,
               &target_id, ierr);

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

      status = xp_target_travel_time_run(&run_id,
               &attitude_id,
               &deriv, &los_az,
```

```
                &los_el, &travel_time, &los_az_rate, &los_el_rate,
                &travel_time_rate, &iray, &freq,
                &num_user_target, &num_los_target,
                &target_id, ierr);
}
```

The `XP_NUM_ERR_TARGET_TRAVEL_TIME` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, ATTITUDE_FRAME_ID, INSTRUMENT_ID, TIME_REF
      INTEGER*4 DERIV, IRAY
      REAL*8 TIME
      REAL*8 POS(3), VEL(3), ACC(3)
      REAL*8 LOS_AZ, LOS_EL, TRAVEL_TIME,
      REAL*8 LOS_AZ_RATE, LOS_EL_RATE, TRAVEL_TIME_RATE, FREQ
      INTEGER*4 IERR(XP_NUM_ERR_TARGET_TRAVEL_TIME), STATUS,
&             NUM_USER_TARGET, NUM_LOS_TARGET


      STATUS = XP_TARGET_TRAVEL_TIME(SAT_ID, ATTITUDE_FRAME_ID,
&                                    INSTRUMENT_ID, TIME_REF,
&                                    TIME, POS, VEL, ACC, DERIV, INTER_FLAG,
&                                    LOS_AZ, LOS_EL, TRAVEL_TIME,
&                                    LOS_AZ_RATE, LOS_EL_RATE,
&                                    TRAVEL_TIME_RATE, IRAY, FREQ,
&                                    NUM_USER_TARGET, NUM_LOS_TARGET, IERR)
```

7.39.3 Input Parameters

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

Table 154: Input parameters of xp_target_travel_time function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude. (input/output)	-	-
atmos_id	xp_atmos_id*	-	Structure that contains the atmosphere initialisation.	-	-
dem_id	xp_dem_id*	-	Structure that contains the DEM initialisation.	-	-
deriv	long *	-	Derivative ID	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
los_az	double *	-	Azimuth of the LOS (Attitude Frame)	deg	>= 0 < 360
los_el	double *	-	Elevation of the LOS (Attitude Frame)	deg	>= -90 <= 90
travel_time	double *	-	Travel time along the (curved) line of sight	s	> 0
los_az_rate	double *	-	Azimuth-rate of the LOS (Attitude Frame)	deg/s	-
los_el_rate	double *	-	Elevation-rate of the LOS (Attitude Frame)	deg/s	-
travel_time_rate	double *	-	Travel time-rate along the (curved) line of sight	s/s	-
iray	long *	-	Ray tracing model switch	-	Accepted values: (0) XP_NO_REF (1) XP_STD_REF (2) XP_USER_REF
freq	double *	-	Frequency of the signal	Hz	>= 0

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

- Derivative switch: deriv. See current document, table 3.
- Ray tracing model switch: iray. See current document, table 3.

7.39.4 Output Parameters

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

Table 155: Output parameters of xp_target_travel_time

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
num_user_target	long*	-	Number of user defined targets calculated	-	>= 0 (Set to 1 for non multi-target routines)
num_los_target	long*	-	Number of LOS targets calculated	-	>= 0
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
ierr	long	-	Error vector	-	-

7.39.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_travel_time** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_travel_time** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 156: Error messages of xp_target_travel_time function

Error type	Error message	Cause and impact	Error code	Error No
ERR		No calculation performed		0
TBW				

7.39.6 Runtime Performances

The following runtime performances have been measured.

Table 157: Runtime performances of xp_target_travel_time

Ultra Sparc II-400 [ms]
TBD

7.40 xp_target_tangent_sun

7.40.1 Overview

The **xp_target_tangent_sun** CFI function computes the location of the tangent point over the Earth that is located on the line of sight that points to the Sun.

7.40.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long           sat_id;
      xp_attitude_id attitude_id;
      xp_atmos_id    atmos_id;
      xp_dem_id      dem_id;
      xp_target_id   target_id;
      long deriv, iray;
      double freq;
      long ierr[XP_NUM_ERR_TARGET_TANGENT_SUN], status,
          num_user_target, num_los_target;

      status = xp_target_tangent_sun(&sat_id,
               &attitude_id,
               &atmos_id,
               &dem_id,
               &deriv, &iray, &freq,
               &num_user_target, &num_los_target,
               &target_id, ierr);

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

      status = xp_target_tangent_sun_run(&run_id,
               &attitude_id,
               &deriv, &iray, &freq,
               &num_user_target, &num_los_target,
               &target_id, ierr);
}
```

The XP_NUM_ERR_TARGET_TANGENT_SUN constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, ATTITUDE_FRAME_ID, INSTRUMENT_ID,
&          TIME_REF, DERIV, IRAY
      REAL*8 TIME, FREQ
      REAL*8 POS(3), VEL(3), ACC(3)
      INTEGER*4 IERR(XP_NUM_ERR_TARGET_TANGENT_SUN), STATUS,
&             NUM_USER_TARGET, NUM_LOS_TARGET

      STATUS = XP_TARGET_TANGENT_SUN(SAT_ID, ATTITUDE_FRAME_ID,
&                                INSTRUMENT_ID,
&                                TIME_REF, TIME,
&                                POS, VEL, ACC, DERIV, IRAY, FREQ,
&                                NUM_USER_TARGET, NUM_LOS_TARGET, IERR)
```

7.40.3 Input Parameters

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

Table 158: Input parameters of xp_target_tangent_sun function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude. (input/output)	-	-
atmos_id	xp_atmos_id*	-	Structure that contains the atmosphere initialisation.	-	-
dem_id	xp_dem_id*	-	Structure that contains the DEM initialisation.	-	-
deriv	long *	-	Derivative ID	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
iray	long *	-	Ray tracing model switch	-	Accepted values: All (see table 3)
freq	double *	-	Frequency of the signal	Hz	>= 0

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

- Derivative switch: deriv. See current document, table 3.
- Ray tracing model switch: iray. See current document, table 3

7.40.4 Output Parameters

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

Table 159: Output parameters of xp_target_tangent_sun

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
num_user_target	long*	-	Number of user defined targets calculated	-	>= 0 (Set to 1 for non multi-target routines)
num_los_target	long*	-	Number of LOS targets calculated	-	>= 0
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
ierr	long	-	Error vector	-	-

.

7.40.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_tangent_sun** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_tangent_sun** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 160: Error messages of xp_target_tangent_sun function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Attitude ID is not initialized	No calculation performed	XP_CFI_SUN_ATTITUDE_STATUS_ERR	0
ERR	Deriv flag is not correct	No calculation performed	XP_CFI_SUN_DERIV_FLAG_ERR	1
ERR	Ray Tracing Model ID is not correct	No calculation performed	XP_CFI_SUN_IRAY_ID_ERR	2
ERR	Invalid Frequency	No calculation performed	XP_CFI_SUN_FREQ_ERR	3
ERR	Atmospheric model has not been initialised	No calculation performed	XP_CFI_SUN_ATM_NOT_INIT_ERR	4
ERR	Atmosphere initialisation is not compatible with Ray Tracing Model	No calculation performed	XP_CFI_SUN_ATM_INIT_IRAY_COMPATIB_ERR	5
ERR	Input state vector does not satisfy loose tolerance requirement	No calculation performed	XP_CFI_SUN_INVALID_SV_ERR	6
ERR	Time Reference not initialised	No calculation performed	XP_CFI_SUN_TIME_REF_INIT_ERR	7
ERR	Internal computation error # 1	No calculation performed	XP_CFI_SUN_SUN_POSITION_CALC_ERR	8
ERR	Internal computation error # 2	No calculation performed	XP_CFI_SUN_SUN_CS_CALC_ERR	9
ERR	Internal computation error # 3	No calculation performed	XP_CFI_SUN_SUN_POINTING_CALC_ERR	10
ERR	Internal computation error # 4	No calculation performed	XP_CFI_SUN_TARGET_STAR_ERR	11
ERR	Internal computation error # 5	No calculation performed	XP_CFI_SUN_TG_PT_BEHIND_LOOK_DIR_ERR	12
WARN	Input state vector does not satisfy tight tolerance requirement	Calculation performed. A message informs the user.	XP_CFI_SUN_INVALID_SV_WARN	13

Table 160: Error messages of xp_target_tangent_sun function

Error type	Error message	Cause and impact	Error code	Error No
WARN	Tangent point is behind looking direction	Calculation performed. A message informs the user.	XP_CFI_SUN_TG_PT_BEHIND_LOOK_DIR_WARN	14

7.40.6 Runtime Performances

The following runtime performances have been measured.

Table 161: Runtime performances of xp_target_tangent_sun

Ultra Sparc II-400 [ms]
0.925

7.41 xp_target_tangent_moon

7.41.1 Overview

The **xp_target_tangent_moon** CFI function computes the location of the tangent point over the Earth that is located on the line of sight that points to the Moon.

7.41.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
{
      long           sat_id;
      xp_attitude_id attitude_id;
      xp_atmos_id    atmos_id;
      xp_dem_id      dem_id;
      xp_target_id   target_id;
      long deriv, iray;
      double freq;
      long ierr[XP_NUM_ERR_TARGET_TANGENT_MOON], status,
          num_user_target, num_los_target;

      status = xp_target_tangent_moon(&sat_id,
               &attitude_id,
               &atmos_id,
               &dem_id,
               &deriv, &iray, &freq,
               &num_user_target, &num_los_target,
               &target_id, ierr);

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

      status = xp_target_tangent_moon_run(&run_id,
               &attitude_id,
               &deriv, &iray, &freq,
               &num_user_target, &num_los_target,
               &target_id, ierr);
}
```

The XP_NUM_ERR_TARGET_TANGENT_MOON constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, ATTITUDE_FRAME_ID, INSTRUMENT_ID, TIME_REF,
&          DERIV, IRAY
      REAL*8 TIME, FREQ
      REAL*8 POS(3), VEL(3), ACC(3)
      INTEGER*4 IERR(XP_NUM_ERR_TARGET_TANGENT_MOON), STATUS,
&             NUM_USER_TARGET, NUM_LOS_TARGET

      STATUS = XP_TARGET_TANGENT_MOON(SAT_ID, INSTRUMENT_ID, TIME_REF,
&                                     TIME, POS, VEL, ACC, DERIV, IRAY, FREQ,
&                                     NUM_USER_TARGET, NUM_LOS_TARGET, IERR)
```

7.41.3 Input Parameters

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

Table 162: Input parameters of xp_tangent_target_moon function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude. (input/output)	-	-
atmos_id	xp_atmos_id*	-	Structure that contains the atmosphere initialisation.	-	-
dem_id	xp_dem_id*	-	Structure that contains the DEM initialisation.	-	-
deriv	long *	-	Derivative ID	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
iray	long *	-	Ray tracing model switch	-	Accepted values: All (see table 3)
freq	double *	-	Frequency of the signal	Hz	>= 0

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

- Derivative switch: deriv. See current document, table 3.
- Ray tracing model switch: iray. See current document, table 3

7.41.4 Output Parameters

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

Table 163: Output parameters of xp_tangent_target_moon

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
num_user_target	long*	-	Number of user defined targets calculated	-	>= 0 (Set to 1 for non multi-target routines)
num_los_target	long*	-	Number of LOS targets calculated	-	>= 0
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
ierr	long	-	Error vector	-	-

7.41.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_tangent_moon** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_tangent_moon** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 164: Error messages of xp_target_tangent_moon function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Attitude ID is not initialized	No calculation performed	XP_CFI_MOON_ATTITUDE_STATUS_ERR	0
ERR	Deriv flag is not correct	No calculation performed	XP_CFI_MOON_DERIV_FLAG_ERR	1
ERR	Ray Tracing Model ID is not correct	No calculation performed	XP_CFI_MOON_IRAY_ID_ERR	2
ERR	Invalid Frequency	No calculation performed	XP_CFI_MOON_FREQ_ERR	3
ERR	Atmospheric model has not been initialised	No calculation performed	XP_CFI_MOON_ATM_NOT_INIT_ERR	4
ERR	Atmosphere initialisation is not compatible with Ray Tracing Model	No calculation performed	XP_CFI_MOON_ATM_INIT_IRAY_COMPATIB_ERR	5

Table 164: Error messages of xp_target_tangent_moon function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Input state vector does not satisfy loose tolerance requirement	No calculation performed	XP_CFI_MOON_INVALID_SV_ERR	6
ERR	Time Reference not initialised	No calculation performed	XP_CFI_MOON_TIME_REF_INIT_ERR	7
ERR	Internal computation error #1	No calculation performed	XP_CFI_MOON_MOON_POSITION_CALC_ERR	8
ERR	Internal computation error #2	No calculation performed	XP_CFI_MOON_MOON_CS_CALC_ERR	9
ERR	Internal computation error #3	No calculation performed	XP_CFI_MOON_MOON_POINTING_CALC_ERR	10
ERR	Internal computation error #4	No calculation performed	XP_CFI_MOON_TARGET_STAR_ERR	11
ERR	Internal computation error #5	No calculation performed	XP_CFI_MOON_TG_PT_BEHIND_LOOK_DIR_ERR	12
WARN	Input state vector does not satisfy tight tolerance requirement	Calculation performed. A message informs the user.	XP_CFI_MOON_INVALID_SV_WARN	19
WARN	Tangent point is behind looking direction	Calculation performed. A message informs the user.	XP_CFI_MOON_TG_PT_BEHIND_LOOK_DIR_WARN	20

7.41.6 Runtime Performances

The following runtime performances have been measured.

Table 165: Runtime performances of xp_target_tangent_moon

Ultra Sparc II-400 [ms]
0.925

7.42 xp_multi_target_inter

7.42.1 Overview

The **xp_multi_target_inter** CFI function computes the first or the second intersection points of the line of sight from the satellite (defined by an elevation and an azimuth angle expressed in the selected Attitude Frame) with surfaces located at certain geodetic altitudes over the Earth.

7.42.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long            sat_id;
      xp_attitude_id attitude_id;
      xp_atmos_id     atmos_id;
      xp_dem_id       dem_id;
      xp_target_id    target_id;
      long deriv, inter_flag, iray;
      double los_az, los_el, geod_alt[XP_MAX_NUM_MULTI_TARGET],
             los_az_rate, los_el_rate, freq;
      long ierr[XP_NUM_ERR_MULTI_TARGET_INTER], num_target, status
             num_user_target, num_los_target;

      status = xp_multi_target_inter(&sat_id,
               &attitude_id,
               &atmos_id,
               &dem_id,
               &deriv, &inter_flag, &los_az,
               &los_el, &num_target, geod_alt, &los_az_rate,
               &los_el_rate, &iray, &freq,
               &num_user_target, &num_los_target,
               &target_id, ierr);

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

      status = xp_multi_target_inter_run(&run_id,
               &attitude_id,
               &deriv, &inter_flag, &los_az,
```

```
                &los_el, &num_target, geod_alt, &los_az_rate,
                &los_el_rate, &iray, &freq,
                &num_user_target, &num_los_target,
                &target_id, ierr);
}
```

The XP_NUM_ERR_MULTI_TARGET_INTER constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, ATTITUDE_FRAME_ID, INSTRUMENT_ID, TIME_REF
      INTEGER*4 DERIV, INTER_FLAG, IRAY
      REAL*8 TIME
      REAL*8 POS(3), VEL(3), ACC(3)
      REAL*8 LOS_AZ, LOS_EL, GEOD_ALT, LOS_AZ_RATE, LOS_EL_RATE, FREQ
      INTEGER*4 IERR(XP_NUM_ERR_MULTI_TARGET_INTER), STATUS
      INTEGER*4 NUM_TARGET, NUM_USER_TARGET, NUM_LOS_TARGET


      STATUS = XP_MULTI_TARGET_INTER(SAT_ID, ATTITUDE_FRAME_ID,
&                                    INSTRUMENT_ID, TIME_REF,
&                                    TIME, POS, VEL, ACC, DERIV, INTER_FLAG,
&                                    LOS_AZ, LOS_EL, NUM_TARGET, GEOD_ALT,
&                                    LOS_AZ_RATE, LOS_EL_RATE, IRAY, FREQ,
&                                    NUM_USER_TARGET, NUM_LOS_TARGET, IERR)
```

7.42.3 Input Parameters

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

Table 166: Input parameters of xp_multi_target_inter function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude. (input/output)	-	-
atmos_id	xp_atmos_id*	-	Structure that contains the atmosphere initialisation.	-	-
dem_id	xp_dem_id*	-	Structure that contains the DEM initialisation.	-	-
deriv	long *	-	Derivative ID	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
inter_flag	long *	-	Flag for first or second intersection point selection	-	Allowed values: (1) XP_INTER_1ST (2) XP_INTER_2ND
los_az	double *	-	Azimuth of the LOS (Attitude Frame)	deg	>= 0 < 360
los_el	double *	-	Elevation of the LOS (Attitude Frame)	deg	>= -90 <= 90
num_target	long *	-	Number of user defined altitudes	-	> 0
geod_alt	double [XP_MAX_NUM_MULTI_TARGET]	-	Geodetic altitude over the Earth, sorted vector, strict monotonic decreasing	m	>= $-b_{WGS}$
los_az_rate	double *	-	Azimuth-rate of the LOS (Attitude Frame)	deg/s	-
los_el_rate	double *	-	Elevation-rate of the LOS (Attitude Frame)	deg/s	-
iray	long *	-	Ray tracing model switch	-	Accepted values: (0) XP_NO_REF (1) XP_STD_REF (2) XP_USER_REF
freq	double *	-	Frequency of the signal	Hz	>= 0

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

- Derivative switch: deriv. See current document, table 3.
- Intersection flag: inter_flag. See current document, table 3.
- Ray tracing model switch: iray. See current document, table 3.

7.42.4 Output Parameters

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

Table 167: Output parameters of xp_multi_target_inter

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
num_user_target	long*		Number of user defined targets calculated		>= 0 <= num_target
num_los_target	long*		Number of LOS targets calculated		>= 0
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
ierr	long	-	Error vector	-	-

7.42.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_multi_target_inter** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_multi_target_inter** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 168: Error messages of xp_multi_target_inter function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Attitude Id. is not initialized	No calculation performed	XP_CFI_MULTI_TARGET_INTER_ATTITUDE_STATUS_ERR	0
ERR	Intersection flag is not correct	No calculation performed	XP_CFI_MULTI_TARGET_INTER_INTER_FLAG_ERR	1
ERR	Invalid Frequency	No calculation performed	XP_CFI_MULTI_TARGET_INTER_FREQ_ERR	2
ERR	Atmospheric model has not been initialised	No calculation performed	XP_CFI_MULTI_TARGET_INTER_ATM_NOT_INIT_ERR	3
ERR	Atmosphere initialisation is not compatible with Ray Tracing Model	No calculation performed	XP_CFI_MULTI_TARGET_INTER_ATM_INIT_IRAY_COMPATIB_ERR	4
ERR	Time reference ID is not correct	No calculation performed	XP_CFI_MULTI_TARGET_INTER_TIME_REF_ERR	5

Table 168: Error messages of xp_multi_target_inter function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Deriv flag is not correct	No calculation performed	XP_CFI_MULTI_TARGET_INTER_DERIV_FLAG_ERR	6
ERR	Ray Tracing Model ID is not correct	No calculation performed	XP_CFI_MULTI_TARGET_INTER_IRAY_ID_ERR	7
ERR	Input state vector does not satisfy loose tolerance requirement	No calculation performed	XP_CFI_MULTI_TARGET_INTER_INVALID_SV_ERR	8
ERR	Invalid LOS Azimuth	No calculation performed	XP_CFI_MULTI_TARGET_INTER_LOS_AZIMUTH_ERR	9
ERR	Invalid LOS Elevation	No calculation performed	XP_CFI_MULTI_TARGET_INTER_LOS_ELEVATION_ERR	10
ERR	Invalid Geodetic Altitude	No calculation performed	XP_CFI_MULTI_TARGET_INTER_GEODETIC_ALT_ERR	11
ERR	Memory allocation error	No calculation performed	XP_CFI_MULTI_TARGET_INTER_MEMORY_ERR	12
ERR	Internal computation error #3	No calculation performed	XP_CFI_MULTI_TARGET_INTER_INITIAL_LOOK_DIR_OR_PLANE_ERR	13
ERR	Time Reference not initialised	No calculation performed	XP_CFI_MULTI_TARGET_INTER_TIME_REF_INIT_ERR	14
ERR	No target was found	No calculation performed	XP_CFI_MULTI_TARGET_INTER_TARGET_NOT_FOUND_ERR	15
ERR	Internal computation error #4	No calculation performed	XP_CFI_MULTI_TARGET_INTER_RANGE_OR_POINTING_CALC_ERR	16
WARN	Input state vector does not satisfy tight tolerance requirement	Calculation performed. A message informs the user.	XP_CFI_MULTI_TARGET_INTER_INVALID_SV_WARN	23
WARN	Path from satellite to target occulted by the Earth	Calculation performed. A message informs the user.	XP_CFI_MULTI_TARGET_INTER_NEGATIVE_ALTITUDE_WARN	24

7.42.6 Runtime Performances

The following runtime performances have been measured.

Table 169: Runtime performances of xp_multi_target_inter

Ultra Sparc II-400 [ms]
TBD

7.43 xp_multi_target_travel_time

7.43.1 Overview

The **xp_multi_target_travel_time** CFI function computes the points of the line or sight from the satellite (defined by an elevation and an azimuth angle expressed in the selected Attitude Frame) at given travel times along the (curved) line of sight.

7.43.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long            sat_id;
      xp_attitude_id attitude_id;
      xp_atmos_id     atmos_id;
      xp_dem_id       dem_id;
      xp_target_id    target_id;
      long deriv, iray;
      double los_az, los_el, travel_time[XP_MAX_NUM_MULTI_TARGET];
      double los_az_rate, los_el_rate, travel_time_rate, freq;
      long num_target, num_user_target, num_los_target;
      long ierr[XP_NUM_ERR_MULTI_TARGET_TRAVEL_TIME], status;

      status = xp_multi_target_travel_time(&sat_id,
               &attitude_id,
               &atmos_id,
               &dem_id,
               &deriv, &los_az, &los_el,
               &num_target, travel_time, &los_az_rate,
               &los_el_rate,&travel_time_rate, &iray, &freq,
               &num_user_target, &num_los_target,
               &target_id, ierr);

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

      status = xp_multi_target_travel_time_run(&run_id,
               &attitude_id,
               &deriv, &los_az, &los_el,
```

```
                &num_target, travel_time, &los_az_rate,
                &los_el_rate,&travel_time_rate, &iray, &freq,
                &num_user_target, &num_los_target,
                &target_id, ierr);
}
```

The XP_NUM_ERR_MULTI_TARGET_TRAVEL_TIME constant is defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, ATTITUDE_FRAME_ID, INSTRUMENT_ID, TIME_REF
      INTEGER*4 DERIV, IRAY
      REAL*8 TIME
      REAL*8 POS(3), VEL(3), ACC(3)
      REAL*8 LOS_AZ, LOS_EL, TRAVEL_TIME(XP_MAX_NUM_MULTI_TARGET)
      REAL*8 LOS_AZ_RATE, LOS_EL_RATE, TRAVEL_TIME_RATE, FREQ
      INTEGER*4 NUM_TARGET, NUM_USER_TARGET, NUM_LOS_TARGET
      INTEGER*4 IERR(XP_NUM_ERR_MULTI_TARGET_TRAVEL_TIME), STATUS


      STATUS = XP_MULTI_TARGET_TRAVEL_TIME(SAT_ID, ATTITUDE_FRAME_ID,
&                                INSTRUMENT_ID, TIME_REF,
&                                TIME, POS, VEL, ACC, DERIV, INTER_FLAG,
&                                LOS_AZ, LOS_EL, NUM_TARGET, TRAVEL_TIME,
&                                LOS_AZ_RATE, LOS_EL_RATE,
&                                TRAVEL_TIME_RATE, IRAY, FREQ,
&                                NUM_USER_TARGET, NUM_LOS_TARGET, IERR)
```

7.43.3 Input Parameters

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

Table 170: Input parameters of xp_multi_target_travel_time function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
attitude_id	xp_attitude_id*	-	Structure that contains the Attitude. (input/output)	-	-
atmos_id	xp_atmos_id*	-	Structure that contains the atmosphere initialisation.	-	-
dem_id	xp_dem_id*	-	Structure that contains the DEM initialisation.	-	-
deriv	long *	-	Derivative ID	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
los_az	double *	-	Azimuth of the LOS (Attitude Frame)	deg	>= 0 < 360
los_el	double *	-	Elevation of the LOS (Attitude Frame)	deg	>= -90 <= 90
num_target	long *	-	Number of user defined times	-	> 0
travel_time	double [XP_MAX_NUM_MULTI_TARGET]	-	Travel time along the (curved) line of sight,sorted vector, strict monotonic increasing	s	> 0
los_az_rate	double *	-	Azimuth-rate of the LOS (Attitude Frame)	deg/s	-
los_el_rate	double *	-	Elevation-rate of the LOS (Attitude Frame)	deg/s	-
travel_time_rate	double *	-	Travel time-rate along the (curved) line of sight. Constant number.	s/s	-
iray	long *	-	Ray tracing model switch	-	Accepted values: (0) XP_NO_REF (1) XP_STD_REF (2) XP_USER_REF
freq	double *	-	Frequency of the signal	Hz	>= 0

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

- Derivative switch: deriv. See current document, table 3.
- Ray tracing model switch: iray. See current document, table 3.

7.43.4 Output Parameters

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

Table 171: Output parameters of xp_multi_target_travel_time

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
num_user_target	long*	-	Number of user defined targets calculated	-	>= 0 <= num_target
num_los_target	long*	-	Number of LOS targets calculated	-	>= 0
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
ierr	long	-	Error vector	-	-

7.43.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_multi_target_travel_time** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_multi_target_travel_time** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 172: Error messages of xp_multi_target_travel_time function

Error type	Error message	Cause and impact	Error code	Error No
ERR		No calculation performed		0
TBW				

7.43.6 Runtime Performances

The following runtime performances have been measured.

Table 173: Runtime performances of xp_multi_target_travel_time

Ultra Sparc II-400 [ms]
TBD

7.44 xp_target_extra_vector

7.44.1 Overview

The **xp_target_extra_vector** CFI function provides the following output parameters for the target(s) in input data structure.: target position, velocity and acceleration vectors, line of sight direction, range, travel time and their corresponding derivatives.

7.44.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long choice, target_type, target_number;
      xp_target_id   target_id;
      double vector_results[XP_SIZE_TARGET_RESULT_VECTOR],
          vector_results_rate[XP_SIZE_TARGET_RESULT_VECTOR],
          vector_results_rate_rate[XP_SIZE_TARGET_RESULT_VECTOR];
      long ierr[XP_NUM_ERR_TARGET_EXTRA_VECTOR], status;

      status = xp_target_extra_vector (&target_id, &choice,
                                       &target_type, &target_number,
                                       vector_results,
                                       vector_results_rate,
                                       vector_results_rate_rate, ierr);
}
```

The `XP_SIZE_TARGET_RESULT_VECTOR` and `XP_NUM_ERR_TARGET_EXTRA_VECTOR` constants aredefined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, CHOICE, TARGET_ID
      REAL*8 VECTOR_RESULTS(XP_SIZE_TARGET_RESULT_VECTOR),
&          VECTOR_RESULTS_RATE(XP_SIZE_TARGET_RESULT_VECTOR),
&          VECTOR_RESULTS_RATE_RATE(XP_SIZE_TARGET_RESULT_VECTOR)
      INTEGER*4 IERR(XP_NUM_ERR_TARGET_EXTRA_VECTOR), STATUS

      STATUS = XP_TARGET_EXTRA_VECTOR (SAT_ID, CHOICE, TARGET_TYPE,
```

```
&                                   TARGET_ID,
&                                   VECTOR_RESULTS, VECTOR_RESULTS_RATE,
&                                   VECTOR_RESULTS_RATE_RATE, IERR)
```

7.44.3 Input Parameters

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

Table 174: Input parameters of xp_target_extra_vector function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
choice	long *	-	Flag to select the extra results to be computed. Even though derivatives could be requested by user, they will not be calculated if the target was computed without derivatives (a warning is raised in this case).	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
target_type	long *		Flag to select the type of target		XP_USER_TARGET_TYPE XP_LOS_TARGET_TYPE XP_DEM_TARGET_TYPE
target_number	long *	-	Target number		>= 0

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

- Choice. (See table 3).

7.44.4 Output Parameters

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

Table 175: Output parameters of xp_target_extra_vector

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
vector_results	double[XP_SIZE_TARGET_RESULT_VECTOR]	[0]	Target Position X (Earth-Fixed)	m	
		[1]	Target Position Y (Earth-Fixed)	m	
		[2]	Target Position Z (Earth-Fixed)	m	
		[3]	Direction LOS X (Earth-Fixed)	-	
		[4]	Direction LOS Y (Earth-Fixed)	-	
		[5]	Direction LOS Z (Earth-Fixed)	-	
		[6]	Range to Attitude Frame Origin	m	
		[7]	Travel Time to Attitude Frame Origin	s	
		[8:XP_SIZE_TARGET_RESULT_VECTOR]	*(dummy)*	-	-
vector_results_rate	double[XP_SIZE_TARGET_RESULT_VECTOR]	[0]	Target Velocity X (Earth-Fixed)	m/ s	
		[1]	Target Velocity Y (Earth-Fixed)	m/ s	
		[2]	Target Velocity Z (Earth-Fixed)	m/ s	
		[3]	Direction Rate LOS X (Earth-Fixed)	1/s	
		[4]	Direction Rate LOS Y (Earth-Fixed)	1/s	
		[5]	Direction Rate LOS Z (Earth-Fixed)	1/s	
		[6]	Range Rate to Attitude Frame Origin	m/s	
		[7]	Travel Time Rate to Attitude Frame Origin	s/s	
		[8:XP_SIZE_TARGET_RESULT_VECTOR]	*(dummy)*	-	-

Table 175: Output parameters of xp_target_extra_vector

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
vector_results_rate_rate double[XP_SIZE_TARGET_RESULT_VECTOR]		[0]	Target Acceleration X (Earth-Fixed)	m/s^2	
		[1]	Target Acceleration Y (Earth-Fixed)	m/s^2	
		[2]	Target Acceleration Z (Earth-Fixed)	m/s^2	
		[3]	Direction Rate Rate LOS X (Earth-Fixed)	$1/s^2$	
		[4]	Direction Rate Rate LOS Y (Earth-Fixed)	$1/s^2$	
		[5]	Direction Rate Rate LOS Z (Earth-Fixed)	$1/s^2$	
		[6]	Range Rate Rate to Attitude Frame Origin	m/s^2	
		[7]	Travel Time Rate Rate to Attitude Frame Origin	s/s^2	
		[8:XP_SIZE_TARGET_RESULT_VECTOR]	*(dummy)*	-	-
ierr	long	-	Error vector	-	-

Note that first derivative parameters (vector_results_rate) are returned as zeros if derivative flag (deriv) was set to NO_DER when the target was computed and that second derivative parameters (vector_results_rate_rate) are returned as zeros if derivative flag (deriv) was set to NO_DER or 1ST_DER.

Note also that when a refraction mode is selected, the second derivative parameters (vector_results_rate_rate) are returned as zeros.

7.44.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_extra_vector** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_extra_vector** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 176: Error messages of xp_target_extra_vector function

Error type	Error message	Cause and impact	Error code	Error No
ERR	The Target ID does not contain any data	No calculation performed	XP_CFI_TARGET_EXTRA_VECTOR_NO_DATA_ERR	0

Table 176: Error messages of xp_target_extra_vector function

Error type	Error message	Cause and impact	Error code	Error No
ERR	The target does not exist	No calculation performed	XP_CFI_TARGET_EXTRA_VECTOR_NO_SUCH_USER_TARGET_ERR	1
ERR	The target does not exist	No calculation performed	XP_CFI_TARGET_EXTRA_VECTOR_NO_SUCH_LOS_TARGET_ERR	2
ERR	The target does not exist	No calculation performed	XP_CFI_TARGET_EXTRA_VECTOR_NO_SUCH_EARTH_TARGET_ERR	3
ERR	Wrong target type	No calculation performed	XP_CFI_TARGET_EXTRA_VECTOR_WRONG_TARGET_TYPE_ERR	4
ERR	Wrong deriv input flag	No calculation performed	XP_CFI_TARGET_EXTRA_VECTOR_DERIV_FLAG_ERR	5
WARN	1st. Derivatives are not available	Calculation performed. A message informs the user.	XP_CFI_TARGET_EXTRA_VECTOR_DER_1ST_NOT_AVAIL_WARN	6
WARN	2nd. Derivatives are not available	Calculation performed. A message informs the user.	XP_CFI_TARGET_EXTRA_VECTOR_DER_2ND_NOT_AVAIL_WARN	7

7.44.6 Runtime Performances

The following runtime performances have been measured.

Table 177: Runtime performances of xp_target_extra_vector

Ultra Sparc II-400 [ms]
TBD

7.45 xp_target_extra_main

7.45.1 Overview

The **xp_target_extra_main** CFI function computes the extra parameter for the target(s) in input data structure.

7.45.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long choice, target_type, target_number;
      double main_results[XP_SIZE_TARGET_RESULT_MAIN],
          main_results_rate[XP_SIZE_TARGET_RESULT_MAIN],
          main_results_rate_rate[XP_SIZE_TARGET_RESULT_MAIN];
      xp_target_id   target_id;
      long ierr[XP_NUM_ERR_TARGET_EXTRA_MAIN], status;

      status = xp_target_extra_main (&target_id, &choice, &target_type,
                                     &target_number,
                                     main_results, main_results_rate,
                                     main_results_rate_rate, ierr);
}
```

The `XP_SIZE_TARGET_EXTRA_MAIN` and `XP_NUM_ERR_TARGET_RESULT_MAIN` constants are defined in the file *explorer_pointing.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>
      INTEGER*4 SAT_ID, CHOICE, TARGET_ID, TARGET_TYPE
      REAL*8 MAIN_RESULTS(XP_SIZE_TARGET_RESULT_MAIN),
&          MAIN_RESULTS_RATE(XP_SIZE_TARGET_RESULT_MAIN),
&          MAIN_RESULTS_RATE_RATE(XP_SIZE_TARGET_RESULT_MAIN)
      INTEGER*4 IERR(XP_NUM_ERR_TARGET_EXTRA_MAIN), STATUS

      STATUS = XP_TARGET_EXTRA_MAIN (SAT_ID, CHOICE, TARGET_TYPE,
&                                    TARGET_ID,
&                                    MAIN_RESULTS, MAIN_RESULTS_RATE,
&                                    MAIN_RESULTS_RATE_RATE, IERR)
```

7.45.3 Input Parameters

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

Table 178: Input parameters of xp_target_extra_main function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
choice	long *	-	Flag to select the extra results to be computed. Even though derivatives could be requested by user, they will not be calculated if the target was computed without derivatives (a warning is raised in this case).	-	Complete
target_type	long *		Flag to select the type of target		XP_USER_TARGET_TYPE XP_LOS_TARGET_TYPE XP_DEM_TARGET_TYPE
target_number	long *	-	Target number		>= 0

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

- Choice. (See table 3).

7.45.4 Output Parameters

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

Table 179: Output parameters of xp_target_extra_main

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
main_results	double[XP_SIZE_TARGET_RESULT_MAIN]	[0]	Target geocentric longitude (Earth Fixed CS)	deg	>= 0 < 360
		[1]	Target geocentric latitude (Earth Fixed CS)	deg	>= -90 <= +90
		[2]	Target geodetic latitude (Earth Fixed CS)	deg	>= -90 <= +90
		[3]	Target geodetic altitude (Earth Fixed CS)	m	-
		[4]	Satellite to target azimuth (Topocentric CS)	deg	>= 0 < 360
		[5]	Satellite to target elevation (Topocentric CS)	deg	>= -90 <= +90
		[6]	Satellite to target pointing: Azimuth (attitude frame)	deg	>= 0 < 360
		[7]	Satellite to target pointing: Elevation (attitude frame)	deg	>= -90 <= +90
		[8]	Target to satellite pointing: Azimuth (Topocentric)	deg	>= 0 < 360
		[9]	Target to satellite pointing: Elevation (Topocentric)	deg	>= -90 <= +90
		[10:XP_SIZE_TARGET_RESULT_MAIN]	*(dummy)*	-	-

Table 179: Output parameters of xp_target_extra_main

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
main_results_rate	double[XP_SIZE_TARGET_RESULT_MAIN]	[0]	Target geocentric longitude rate (Earth Fixed CS)	deg/s	-
		[1]	Target geocentric latitude rate (Earth Fixed CS)	deg/s	-
		[2]	Target geodetic latitude rate (Earth Fixed CS)	deg/s	-
		[3]	Target geodetic altitude rate (Earth Fixed CS)	m/s	-
		[4]	Satellite to target azimuth rate (Topocentric CS)	deg/s	-
		[5]	Satellite to target elevation rate (Topocentric CS)	deg/s	-
		[6]	Satellite to target pointing: Azimuth rate (attitude frame)	deg/s	-
		[7]	Satellite to target pointing: Elevation rate (attitude frame)	deg/s	-
		[8]	Target to satellite pointing: Azimuth rate (Topocentric)	deg/s	-
		[9]	Target to satellite pointing: Elevation rate (Topocentric)	deg/s	-
		[10:XP_SIZE_TARGET_RESULT_MAIN]	*(dummy)*	-	-

Table 179: Output parameters of xp_target_extra_main

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
main_results_rate_rate double[XP_SIZE_TARGET_RESULT_MAIN]		[0]	Target geocentric longitude rate-rate (Earth Fixed CS)	deg/s^2	-
		[1]	Target geocentric latitude rate-rate (Earth Fixed CS)	deg/s^2	-
		[2]	Target geodetic latitude rate-rate (Earth Fixed CS)	deg/s^2	-
		[3]	Target geodetic altitude rate-rate (Earth Fixed CS)	m/s^2	-
		[4]	Satellite to target azimuth rate-rate (Topocentric CS)	deg/s^2	-
		[5]	Satellite to target elevation rate-rate (Topocentric CS)	deg/s^2	-
		[6]	Satellite to target pointing: Azimuth rate-rate (attitude frame)	deg/s^2	-
		[7]	Satellite to target pointing: Elevation rate-rate (attitude frame)	deg/s^2	-
		[8]	Target to satellite pointing: Azimuth rate-rate (Topocentric)	deg/s^2	-
		[9]	Target to satellite pointing: Elevation rate-rate (Topocentric)	deg/s^2	-
		[10:XP_SIZE_TARGET_RESULT_MAIN]	*(dummy)*	-	-
ierr	long	-	Error vector	-	-

Note that first derivative parameters (vector_results_rate) are returned as zeros if derivative flag (deriv) was set to NO_DER when the target was computed and that second derivative parameters (vector_results_rate_rate) are returned as zeros if derivative flag (deriv) was set to NO_DER or 1ST_DER.

Note also that when a refraction mode is selected, the second derivative parameters (vector_results_rate_rate) are returned as zeros.

7.45.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_extra_main** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_extra_main** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 180: Error messages of xp_target_extra_main function

Error type	Error message	Cause and impact	Error code	Error No
ERR	No target data available	No calculation performed	XP_CFI_TARGET_EXTRA_MAIN_NO_DATA_ERR	0
ERR	The target does not exist	No calculation performed	XP_CFI_TARGET_EXTRA_MAIN_NO_SUCH_USER_TARGET_ERR	1
ERR	The target does not exist	No calculation performed	XP_CFI_TARGET_EXTRA_MAIN_NO_SUCH_LOS_TARGET_ERR	2
ERR	The target does not exist	No calculation performed	XP_CFI_TARGET_EXTRA_MAIN_NO_SUCH_EARTH_TARGET_ERR	3
ERR	Wrong target type	No calculation performed	XP_CFI_TARGET_EXTRA_MAIN_WRONG_TARGET_TYPE_ERR	4
ERR	Invalid time reference in target data	No calculation performed	XP_CFI_TARGET_EXTRA_MAIN_INVALID_TIME_REF_ERR	5
ERR	Error calling to XL_Car_Geo CFI function	No calculation performed	XP_CFI_TARGET_EXTRA_MAIN_CAR_TO_GEO_ERR	6
ERR	Error getting tranformation matrix to Topocentric CS	No calculation performed	XP_CFI_TARGET_EXTRA_MAIN_TOPO_ERR	7
ERR	Error getting direction angles	No calculation performed	XP_CFI_TARGET_EXTRA_MAIN_DIR_POINTING_ERR	8
ERR	Error while changing coordinate system	No calculation performed	XP_CFI_TARGET_EXTRA_MAIN_CS_CHANGE_ERR	9
WARN	Warning: Derivatives cannot be calculated	Calculation performed	XP_CFI_TARGET_EXTRA_MAIN_DERIV_WARN	10
WARN	Warning calling to XL_Car_Geo CFI function	Calculation performed, but derivatives will not be computed	XP_CFI_TARGET_EXTRA_MAIN_AMBIGUOUS_SINGULAR_WARN	11

7.45.6 Runtime Performances

The following runtime performances have been measured.

Table 181: Runtime performances of xp_target_extra_main

Ultra Sparc II-400 [ms]
TBD

7.46 xp_target_extra_aux

7.46.1 Overview

The **xp_target_extra_aux** CFI function computes auxiliary parameters for the target in input data structure.

7.46.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long choice, target_type, target_number;
      double aux_results[XP_SIZE_TARGET_RESULT_AUX],
          aux_results_rate[XP_SIZE_TARGET_RESULT_AUX],
          aux_results_rate_rate[XP_SIZE_TARGET_RESULT_AUX];
      xp_target_id   target_id;
      long ierr[XP_NUM_ERR_TARGET_EXTRA_AUX], status;

      status = xp_target_extra_aux(&target_id, &choice, &target_type,
                                   &target_number,
                                   aux_results, aux_results_rate,
                                   aux_results_rate_rate, ierr);
}
```

The `XP_SIZE_TARGET_RESULT_AUX` and `XP_NUM_ERR_TARGET_EXTRA_AUX` constants are defined in the file *explorer_pointing.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>
       INTEGER*4 SAT_ID, CHOICE, TARGET_TYPE, TARGET_ID
       REAL*8 AUX_RESULTS(XP_SIZE_TARGET_RESULT_AUX),
&         AUX_RESULTS_RATE(XP_SIZE_TARGET_RESULT_AUX),
&         AUX_RESULTS_RATE_RATE(XP_SIZE_TARGET_RESULT_AUX)
       INTEGER*4 IERR(XP_NUM_ERR_TARGET_RESULT_AUX), STATUS

       STATUS = XP_TARGET_RESULT_AUX(SAT_ID, CHOICE, TARGET_TYPE,
&                                    TARGET_ID,
&                                    AUX_RESULTS, AUX_RESULTS_RATE,
&                                    AUX_RESULTS_RATE_RATE, IERR)
```

7.46.3 Input Parameters

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

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
choice	long *	-	Flag to select the extra results to be computed. Even though derivatives could be requested by user, they will not be calculated if the target was computed without derivatives (a warning is raised in this case).	-	Complete
target_type	long *		Flag to select the type of target		XP_USER_TARGET_TYPE XP_LOS_TARGET_TYPE XP_DEM_TARGET_TYPE
target_number	long *	-	Target number		>= 0

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

- Choice. (See table 3).

7.46.4 Output Parameters

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

Table 182: Output parameters of xp_target_extra_aux

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
aux_results	dou-ble[XP_SIZE_TARGET_RESULT_AUX]	[0]	Radius of curvature in the look direction at the nadir of the target (Earth fixed CS)	m	>= 0
		[1]	Distance from the nadir of the target to the satellite nadir. (Earth fixed CS)	m	>= 0
		[2]	Minimum distance from the nadir of the target to the ground track (Earth Fixed CS). It is regarded as positive distance when the nadir of the target is located on the left hand side of the ground track.	m	-
		[3]	Distance from the SSP to the point located on the ground track that is at a minimum distance from the nadir of the target (Earth fixed CS) It is regarded as positive distance when that point is located on the ground track ahead the SSP (in the direction of the motion of the SSP)	m	-
		[4]	Mean Local Solar Time at target.	decimal hour	>= 0 < 24
		[5]	True Local Solar Time at target.	decimal hour	>= 0 < 24
		[6]	Right ascension at which the look direction from the satellite to the target points at target point. (True of Date CS)	deg	>= 0 < 360
		[7]	Declination at which the look direction from the satellite to the target points at target point. (True of Date CS)	deg	>= -90 < 90
		[8:XP_SIZE_TARGET_RESULT_AUX]	*(dummy)*	-	-

Table 182: Output parameters of xp_target_extra_aux

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
aux_results_rate double[XP_SIZE_TARGET_RESULT_AUX]		[0]	Radius of curvature-rate in the look direction at the nadir of the target (Earth fixed CS)	m/s	-
		[1]	Distance-rate from the nadir of the target to the satellite nadir. (Earth fixed CS)	m	>= 0
		[2]	Distance-rate from the nadir of the target to the ground track (Earth fixed CS)	m/s	-
		[3]	Distance-rate from the SSP to the point located on the ground track that is at a minimum distance from the nadir of the target (Earth fixed CS)		
		[4:7]	*(dummy)*	-	-
		[8]	Northward component of the velocity relative to the Earth of the nadir of the target (Topocentric CS)	m/s	-
		[9]	Eastward component of the velocity relative to the Earth of the nadir of the target (Topocentric CS)	m/s	-
		[10]	Azimuth of the velocity relative to the Earth of the nadir of the target. (Topocentric CS)	deg	>= 0 < 360
		[11]	Magnitude of the velocity relative to the Earth of the nadir of the target. (Topocentric CS)	m/s	>= 0
		[12:XP_SIZE_TARGET_RESULT_AUX]	*(dummy)*	-	-
aux_results_rate_rate	double[XP_SIZE_TARGET_RESULT_AUX]	[0]	Radius of curvature-rate-rate in the look direction at the nadir of the target (Earth fixed CS)	m/s	-
		[1]	Distance-rate-rate from the nadir of the target to the satellite nadir. (Earth fixed CS)	m	>= 0
		[2]	Distance-rate-rate from the nadir of the target to the ground track (Earth fixed CS)	m/s	-
		[3]	Distance-rate-rate from the SSP to the point located on the ground track that is at a minimum distance from the nadir of the target (Earth fixed CS)		
		[4:XP_SIZE_TARGET_RESULT_AUX]	*(dummy)*	-	-

Table 182: Output parameters of xp_target_extra_aux

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

7.46.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_extra_aux** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_extra_aux** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 183: Error messages of xp_target_extra_aux function

Error type	Error message	Cause and impact	Error code	Error No
ERR	No target data available	No calculation performed	XP_CFI_TARGET_EXTRA_AUX_NO_DATA_ERR	0
ERR	The target does not exist	No calculation performed	XP_CFI_TARGET_EXTRA_AUX_NO_SUCH_USER_TARGET_ERR	1
ERR	The target does not exist	No calculation performed.	XP_CFI_TARGET_EXTRA_AUX_NO_SUCH_LOS_TARGET_ERR	2
ERR	The target does not exist	No calculation performed	XP_CFI_TARGET_EXTRA_AUX_NO_SUCH_EARTH_TARGET_ERR	3
ERR	Wrong target type	No calculation performed	XP_CFI_TARGET_EXTRA_AUX_WRONG_TARGET_TYPE_ERR	4
ERR	Invalid time reference in target data	No calculation performed	XP_CFI_TARGET_EXTRA_AUX_INVALID_TIME_REF_ERR	5
ERR	Error calling to XL_Car_Geo CFI function	No calculation performed	XP_CFI_TARGET_EXTRA_AUX_CAR_TO_GEO_ERR	6
ERR	Error getting tranformation matrix to Topocentric CS.	No calculation performed	XP_TARGET_EXTRA_AUX_TOPO_ERR	7
ERR	Error getting direction angles	No calculation performed	XP_CFI_TARGET_EXTRA_AUX_DIR_POINTING_ERR	8

Table 183: Error messages of xp_target_extra_aux function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Error computing radius of curvature	No calculation performed	XP_CFI_TARGET_EXTRA_AUX_RADII_CURVATURE_CALC_ERR	9
ERR	Error computing pointing after crossing the Earth atmosphere	No calculation performed	XP_CFI_TARGET_EXTRA_AUX_POINTING_AFTER_ATM_CALC_ERR	10
ERR	Error computing distance	No calculation performed	XP_CFI_TARGET_EXTRA_AUX_DISTANCE_CALC_ERR	11
ERR	Error computing velocity of the target's nadir	No calculation performed	XP_CFI_TARGET_EXTRA_AUX_TOP_VEL_CALC_ERR	12
WARN	Error computing MLST of TLST	Calculation performed	XP_CFI_TARGET_EXTRA_AUX_MLST_OR_TLST_CALC_ERR	13
WARN	Warning: Path from satellite to target occulted by the Earth	Calculation performed	XP_CFI_TARGET_EXTRA_AUX_NEGATIVE_ALTITUDE_WARN	14
WARN	Warning calling to XL_Car_Geo CFI function	Calculation performed,	XP_CFI_TARGET_EXTRA_AUX_AMBIGUOUS_SINGULAR_WARN	15
WARN	Warning: Derivatives cannot be calculated	Calculation performed,	XP_CFI_TARGET_EXTRA_AUX_DERIV_WARN	16

7.46.6 Runtime Performances

The following runtime performances have been measured.

Table 184: Runtime performances of xp_target_extra_aux

Ultra Sparc II-400 [ms]
TBD

7.47 xp_target_extra_ef_target

7.47.1 Overview

The **xp_target_extra_ef_target** CFI function computes the parameter for an Earth fixed target related to the target in input data structure.

7.47.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long target_type, target_number, choice;
      double freq;
      double ef_target_results_rate[XP_SIZE_TARGET_RESULT_EF_TARGET],
      ef_target_results_rate_rate[XP_SIZE_TARGET_RESULT_EF_TARGET];
      xp_target_id   target_id;
      long ierr[XP_NUM_ERR_TARGET_EXTRA_EF_TARGET], status;

      status = xp_target_extra_ef_target(&target_id, &choice,
                              &target_type, &target_number, &freq,
                              ef_target_results_rate,
                              ef_target_results_rate_rate, ierr);
}
```

The `XP_SIZE_TARGET_RESULT_EF_TARGET` and `XP_NUM_ERR_TARGET_EXTRA_EF_TARGET` constants are defined in the file *explorer_pointing.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>
       INTEGER*4 SAT_ID, TARGET_TYPE, TARGET_ID
       REAL*8 EF_TARGET_RESULTS_RATE(XP_SIZE_TARGET_RESULT_EF_TARGET),
      & EF_TARGET_RESULTS_RATE_RATE(XP_SIZE_TARGET_RESULT_EF_TARGET),
      & FREQ
       INTEGER*4 IERR(XP_NUM_ERR_TARGET_RESULT_EF_TARGET), STATUS

       STATUS = XP_TARGET_EXTRA_EF_TARGET(SAT_ID, TARGET_TYPE,
      &                                   TARGET_ID, FREQ,
      &                                   EF_TARGET_RESULTS_RATE,
```

```
     &                                EF_TARGET_RESULTS_RATE_RATE,
     &                                IERR)
```

7.47.3 Input Parameters

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

Table 185: Input parameters of xp_target_extra_ef_target function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
choice	long *	-	Flag to select the extra results to be computed. Even though derivatives could be requested by user, they will not be calculated if the target was computed without derivatives (a warning is raised in this case).	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
target_type	long *		Flag to select the type of target		XP_USER_TARGET_TYPE XP_LOS_TARGET_TYPE XP_DEM_TARGET_TYPE
target_number	long *	-	Target number		>= 0
freq	double *	-	Frequency of the signal	Hz	>=0

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

- Choice. (See table 3).

7.47.4 Output Parameters

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

Table 186: Output parameters of xp_target_extra_ef_target

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
ef_target_results_rate	double[XP_SIZE_TARGET_RESULT_EF_TARGET]	[0]	2-way Doppler shift of the signal (Earth Fixed CS)	Hz	-
		[1]	Earthfixed target to satellite range-rate. (Earth Fixed CS)	m/s	-
		[2]	Earthfixed target to satellite azimuth-rate. (Topocentric CS)	deg/s	-
		[3]	Earthfixed target to satellite elevation-rate. (Topocentric CS)	deg/s	-
		[4]	Satellite to earthfixed target azimuth-rate. (Topocentric CS)	deg/s	-
		[5]	Satellite to earthfixed target elevation-rate. (Topocentric CS)	deg/s	-
		[6]	Satellite to earthfixed target azimuth-rate. (Attitude Frame)	deg/s	-
		[7]	Satellite to earthfixed target elevation-rate. (Attitude Frame)	deg/s	-
ef_target_results_rate_rate	double[XP_SIZE_TARGET_RESULT_EF_TARGET]	[0]	*(dummy)*	-	-
		[1]	Earthfixed target to satellite range-rate-rate. (Earth Fixed CS)	m/s^2	-
		[2]	Earthfixed target to satellite azimuth-rate-rate. (Topocentric CS)	deg/s^2	-
		[3]	Earthfixed target to satellite elevation-rate-rate. (Topocentric CS)	deg/s^2	-
		[4]	Satellite to earthfixed target azimuth-rate-rate. (Topocentric CS)	deg/s^2	-

Table 186: Output parameters of xp_target_extra_ef_target

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
		[5]	Satellite to earthfixed target elevation-rate-rate. (Topocentric CS)	deg/s^2	-
		[6]	Satellite to earthfixed target azimuth-rate-rate-rate. (Attitude Frame)	deg/s^2	-
		[7]	Satellite to earthfixed target elevation-rate-rate. (Attitude Frame)	deg/s^2	-
ierr	long	-	Error vector	-	-

7.47.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_extra_ef_target** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_extra_ef_target** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 187: Error messages of xp_target_extra_ef_target function

Error type	Error message	Cause and impact	Error code	Error No
ERR	No target data available	No calculation performed	XP_CFI_TARGET_EXTRA_EF_TARGET_NO_DATA_ERR	0
ERR	The target does not exist	No calculation performed	XP_CFI_TARGET_EXTRA_EF_TARGET_NO_SUCH_USER_TARGET_ERR	1
ERR	The target does not exist	No calculation performed	XP_CFI_TARGET_EXTRA_EF_TARGET_NO_SUCH_LOS_TARGET_ERR	2
ERR	The target does not exist	No calculation performed	XP_CFI_TARGET_EXTRA_EF_TARGET_NO_SUCH_EARTH_TARGET_ERR	3
ERR	Wrong target type	No calculation performed	XP_CFI_TARGET_EXTRA_EF_TARGET_WRONG_TARGET_TYPE_ERR	4
ERR	Wrong input deriv flag	No calculation performed	XP_CFI_TARGET_EXTRA_EF_TARGET_DERIV_FLAG_ERR	5

Table 187: Error messages of xp_target_extra_ef_target function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Error getting target geodetic coordinates	No calculation performed	XP_CFI_TARGET_EXTRA_EF_TARGET_GEO_COORD_ERR	6
ERR	Invalid time reference in target data	No calculation performed	XP_CFI_TARGET_EXTRA_EF_TARGET_INVALID_TIME_REF_ERR	7
ERR	Internal computation error	No calculation performed	XP_CFI_TARGET_EXTRA_EF_TARGET_RANGE_OR_POINTING_CALC_ERR	8
ERR	Wrong Atmospheric model in target data	No calculation performed	XP_CFI_TARGET_EXTRA_EF_TARGET_MODE_COMBINATION_SWITCHES_ERR	9
ERR	Error calling to XL_Car_Geo CFI function	No calculation performed	XP_CFI_TARGET_EXTRA_EF_TARGET_CAR_GEO_ERR	10
WARN	2nd. Derivatives are not available	Calculation performed	XP_CFI_TARGET_EXTRA_EF_TARGET_DER_2ND_NOT_AVAIL_WARN	11
WARN	Warning calling to XL_Car_Geo CFI function	Calculation performed	XP_CFI_TARGET_EXTRA_EF_TARGET_AMBIGUOUS_SINGULAR_WARN	12
WARN	1ST Derivative not computed for target. Satellite to target azimuth and elevation rates (SRAR CS) cannot be calculated	Calculation performed, except for azimuth and elevation rates in SRAR coordinate system.	XP_CFI_TARGET_EXTRA_EF_TARGET_DERIV_FLAG_WARN	13

7.47.6 Runtime Performances

The following runtime performances have been measured.

Table 188: Runtime performances of xp_target_extra_ef_target

Ultra Sparc II-400 [ms]
TBD

7.48 xp_target_extra_target_to_sun

7.48.1 Overview

The **xp_target_extra_target_to_sun** CFI function computes extra parameters related to the pointing from the target in input data structure to the sun.

7.48.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      long target_type, target_number, choice, iray;
      double freq;
      double sun_results[XP_SIZE_TARGET_RESULT_TARGET_TO_SUN],
          sun_results_rate[XP_SIZE_TARGET_RESULT_TARGET_TO_SUN],
          sun_results_rate_rate[XP_SIZE_TARGET_RESULT_TARGET_TO_SUN];
      xp_target_id   target_id;
      long ierr[XP_NUM_ERR_TARGET_EXTRA_TARGET_TO_SUN], status;

      status = xp_target_extra_target_to_sun
                              (&target_id, &choice, &target_type,
                               &target_number, &iray, &freq,
                               sun_results, sun_results_rate,
                               sun_results_rate_rate, ierr);
}
```

The `XP_SIZE_TARGET_RESULT_TARGET_TO_SUN` and `XP_NUM_ERR_TARGET_EXTRA_TARGET_TO_SUN` constants are defined in the file *explorer_pointing.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>
       INTEGER*4 SAT_ID, TARGET_TYPE, TARGET_ID, DERIV_FLAG, IRAY
       REAL*8  SUN_RESULTS(XP_SIZE_TARGET_RESULT_TARGET_TO_SUN),
      &    SUN_RESULTS_RATE(XP_SIZE_TARGET_RESULT_TARGET_TO_SUN),
      &    SUN_RESULTS_RATE_RATE(XP_SIZE_TARGET_RESULT_TARGET_TO_SUN),
      &    FREQ
       INTEGER*4 IERR(XP_NUM_ERR_TARGET_EXTRA_TARGET_TO_SUN), STATUS
```

```
      STATUS = XP_TARGET_EXTRA_TARGET_TO_SUN
                          (SAT_ID, TARGET_TYPE, TARGET_ID, DERIV_FLAG,
     &                     IRAY, FREQ,
     &                     SUN_RESULTS, SUN_RESULTS_RATE,
     &                     SUN_RESULTS_RATE_RATE, IERR)
```

7.48.3 Input Parameters

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

Table 189: Input parameters of xp_target_extra_target_to_sun function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
target_id	xp_target_id*	-	Structure that contains the Target results	-	-
choice	long *	-	Flag to select the extra results to be computed. Even though derivatives could be requested by user, they will not be calculated if the target was computed without derivatives (a warning is raised in this case).	-	Allowed values: (0) XP_NO_DER (1) XP_DER_1ST (2) XP_DER_2ND
target_type	long *		Flag to select the type of target		XP_USER_TARGET_TYPE XP_LOS_TARGET_TYPE XP_DEM_TARGET_TYPE
target_number	long *	-	Target number		>= 0
iray	long *	-	Ray tracing model switch	-	Complete
freq	double *	-	Frequency of the signal	Hz	>=0

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

- Choice. (See table 3).
- Ray tracing model: iray. (See table 3).

7.48.4 Output Parameters

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

Table 190: Output parameters of xp_target_extra_target_to_sun

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sun_results	double[XP_SIZE_TARGET_RESULT_TARGET_TO_SUN]	[0]	Target to Sun (centre) azimuth. (Topocentric CS)	deg	>= 0 < 360
		[1]	Target to Sun (centre) elevation. (Topocentric CS)	deg	>= -90 <= +90
		[2]	Tangent altitude over the Earth in the target to Sun (centre) look direction. (Earth fixed CS)	m	-
		[3]	Target to Sun visibility flag: • - 1: Sun eclipsed by the Earth. • +1: Sun in sight.	-	+1, -1
		[4:XP_SIZE_TARGET_RESULT_TARGET_TO_SUN]	*(dummy)*	-	-
sun_results_rate	double[XP_SIZE_TARGET_RESULT_TARGET_TO_SUN]	[0]	Target to Sun (centre) azimuth-rate. (Topocentric CS)	deg/s	-
		[1]	Target to Sun (centre) elevation-rate. (Topocentric CS)	deg/s	-
		[2:XP_SIZE_TARGET_RESULT_TARGET_TO_SUN]	*(dummy)*	-	-

Table 190: Output parameters of xp_target_extra_target_to_sun

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sun_results_rate_rate	double[XP_SIZE_TARGET_RESULT_TARGET_TO_SUN]	[0]	Target to Sun (centre) azimuth-rate. (Topocentric CS)	deg/s^2	-
		[1]	Target to Sun (centre) elevation-rate. (Topocentric CS)	deg/s^2	-
		[2:XP_SIZE_TARGET_RESULT_TARGET_TO_SUN]	*(dummy)*	-	-
ierr	long	-	Error vector	-	-

7.48.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_extra_target_to_sun** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_extra_target_to_sun** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 191: Error messages of xp_target_extra_target_to_sun function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Attitude ID is not initialized	No calculation performed	XP_CFI_SUN_ATTITUDE_STATUS_ERR	0
ERR	Deriv flag is not correct	No calculation performed	XP_CFI_SUN_DERIV_FLAG_ERR	1
ERR	Ray Tracing Model ID is not correct	No calculation performed	XP_CFI_SUN_IRAY_ID_ERR	2
ERR	Invalid Frequency	No calculation performed	XP_CFI_SUN_FREQ_ERR	3

Table 191: Error messages of xp_target_extra_target_to_sun function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Atmospheric model has not been initialised	No calculation performed	XP_CFI_SUN_ATM_NOT_INIT_ERR	4
ERR	Atmosphere initialisation is not compatible with Ray Tracing Model	No calculation performed	XP_CFI_SUN_ATM_INIT_IRAY_COMPATIB_ERR	5
ERR	Input state vector does not satisfy loose tolerance requirement	No calculation performed	XP_CFI_SUN_INVALID_SV_ERR	6
ERR	Time Reference not initialised	No calculation performed	XP_CFI_SUN_TIME_REF_INIT_ERR	7
ERR	Internal computation error #1	No calculation performed	XP_CFI_SUN_SUN_POSITION_CALC_ERR	8
ERR	Internal computation error #2	No calculation performed	XP_CFI_SUN_SUN_CS_CALC_ERR	9
ERR	Internal computation error #3	No calculation performed	XP_CFI_SUN_SUN_POINTING_CALC_ERR	10
ERR	Internal computation error #4	No calculation performed	XP_CFI_SUN_TARGET_STAR_ERR	11
ERR	Internal computation error #5	No calculation performed	XP_CFI_SUN_TG_PT_BEHIND_LOOK_DIR_ERR	11
WARN	Input state vector does not satisfy tight tolerance requirement	Calculation performed	XP_CFI_SUN_INVALID_SV_WARN	12
WARN	Tangent point is behind looking direction	Calculation performed	XP_CFI_SUN_TG_PT_BEHIND_LOOK_DIR_WARN	13

7.48.6 Runtime Performances

The following runtime performances have been measured.

Table 192: Runtime performances of xp_target_extra_target_to_sun

Ultra Sparc II-400 [ms]
TBD

7.49 xp_target_close

7.49.1 Overview

The **xp_target_close** CFI function cleans up any memory allocation performed by the Target functions.

.

7.49.2 Calling Interface

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

```
#include <explorer_pointing.h>
{
      xp_target_id target_id;
      long ierr[XP_NUM_ERR_TARGET_CLOSE], status;

      status = xp_target_close(&target_id, ierr);
}
```

The `XP_NUM_ERR_TARGET_CLOSE` constant is defined in the file *explorer_pointing.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>
      INTEGER*4 IERR(XP_NUM_ERR_TARGET_INIT), STATUS

      STATUS = XP_TARGET_CLOSE(SAT_ID, INSTRUMENT_ID,IERR)
```

7.49.3 Input Parameters

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

Table 193: Input parameters of xp_target_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
target_id	xp_target_id*	-	Structure that contains the Target results	-	-

7.49.4 Output Parameters

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

Table 194: Output parameters of xp_target_close

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

7.49.5 Warnings and Errors

Next table lists the possible error messages that can be returned by the **xp_target_close** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_POINTING software library **xp_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 **xp_target_close** function by calling the function of the EXPLORER_POINTING software library **xp_get_code** (see [GEN_SUM])

Table 195: Error messages of xp_target_close function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Target ID is not initialized or it is being used	No calculation performed	XP_CFI_TARGET_CLOSE_WRONG_ID_ERR	11

7.49.6 Runtime Performances

The following runtime performances have been measured.

Table 196: Runtime performances of xp_target_close

Ultra Sparc II-400 [ms]
TBD

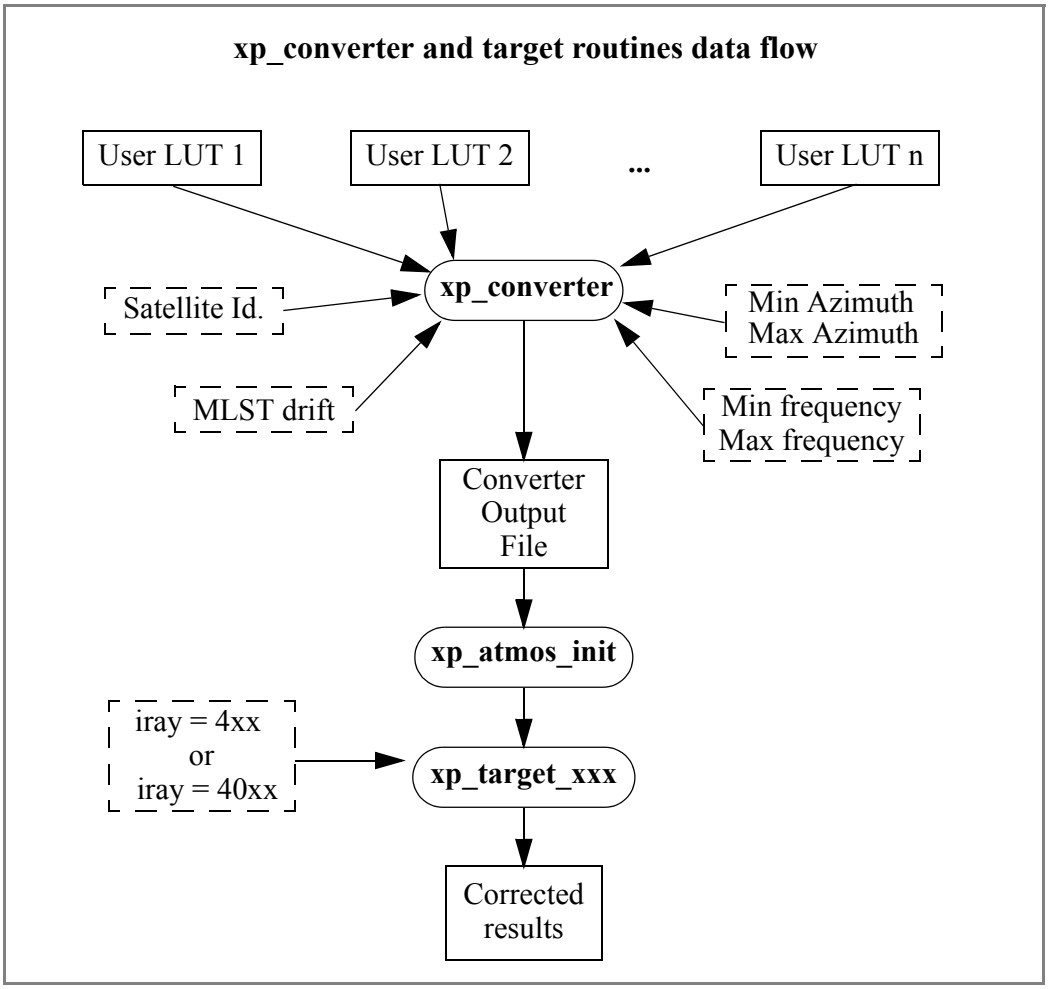
7.50 xp_converter

7.50.1 Overview

Before calling a target CFI function using iray = 400 or iray = 4000 (user's predefined refraction ray tracing models), the atmosphere model should be initialised using a user's file containing the description of the atmosphere. This file can be generated using the executable program **xp_converter.**

xp_converter allows the user to select different atmosphere models to obtain different corrective functions, each of them with a given latitude band and a given validity time. The output file name should be defined in the enviromental variable XP_USER_REF_CONV_FILE_NAME, before running **xp_converter**. If the variable is not defined the function returns an error.

The overall data flow of **xp_converter** and **xp_target_xxx** for a user predefined reraction ray tracing model is as follows:



7.50.2 Calling interface

xp_converter shall be called from a UNIX or WINDOWS shell as follows:

```
xp_converter -sat satellite_id
             -cif User_LUT_1
             [-cif User_LUT_2]
              ...
             [-cif User_LUT_n]
             -min_az min_azimuth
             -max_az max_azimuth
             -min_freq min_frequency
             -max_freq max_frequency
             -mlst_dr mlst_drift
             [-v]
             [-xl_v]
             [-xo_v]
             [-xp_v]
             [-help]
```

taking into account the following considerations:

- Order of parameters does not matter
- Bracketed parameters are not mandatory
- *xl_v* for *explorer_lib* verbose mode.
- *xo_v* for *explorer_orbit* verbose mode.
- *xp_v* for *explorer_pointing* verbose mode.
- *v* for Verbose mode for the 3 libraries (default is silent).
- *help* option will print the above text on stderr (no execution).

7.50.3 Input parameters

The **xp_converter** executable program has the following input parameters:

Table 197: Input parameters for xp_converter

Keyword	Value after keyword	Type	Description	Unit	Allowed Range
cif	User_LUT_i	string	Name of the LUT file (input file, path included). The number of LUTs is arbitrary.	-	-
sat	satellite_id	long	Satellite ID	-	Complete

Table 197: Input parameters for xp_converter

Keyword	Value after keyword	Type	Description	Unit	Allowed Range
min_az	min_azimuth	double	Minimum azimuth value of the looking direction to be considered in computations.	deg	0<=min_az<360.0
max_az	max_azimuth	double	Maximum azimuth value of the looking direction to be considered in computations.	deg	0<=max_az<360.0
min_freq	min_frequency	double	Minimum frequency value to be considered in computations.	MHz	0<=min_freq
max_freq	max_frequency	double	Maximum frequency value to be considered in computations.	MHz	0<=max_freq
mlst_dr	mlst_drift	double	MLST drift of the orbit	secs/day	mlst_drift >= 0

7.50.4 Output

In addition to the **xp_converter** output file, some intermediate output files are produced with data that can be plotted. They are named *interm_outp_file_xx.dat* (where *xx* is the number of the LUT file) and *interm_outp_file_av.dat* (for the average) and they can be plotted with **gnuplot** (in a UNIX shell).

For **xp_target_xxx** CFI function, the selection of the user atmosphere determines the correction to be applied to the parameters of the unrefracted tangent point (tangent altitude, etc.). The following table defines the relation between the iray input parameter and the selected corrective function:

Table 198: iray input vs corrective function.

IRAY	Mnemonic	Description
400	PP_LUT_REF	Average User-defined Corrective function
401	PP_LUT_REF + 1	First User-defined Corrective function
...		
400 + n	PP_LUT_REF + n	Last User-defined Corrective function (*n* being the number of LUT input files for xp_converter)
4000	PP_LUT_REF_N	Average User-defined Corrective function (No refraction in Sun and Moon related parameters).
4001	PP_LUT_REF_N + 1	First User-defined Corrective function (No refraction in Sun and Moon related parameters).
...		

Table 198: iray input vs corrective function.

IRAY	Mnemonic	Description
4000 + n	PP_LUT_REF_N + n	Last User-defined Corrective function (No refraction in Sun and Moon related parameters).

8 LIBRARY PRECAUTIONS

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

- When a message like

  EXPLORER_POINTING >>> ERROR in *xp_function*: Internal computation error # *n*

  or

  EXPLORER_POINTING >>> WARNING in *xp_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 199: Known problems

CFI library	Problem	Work around solution
xp_sat_nominal_att_init	Only Envisat mode is currently available	
xp_sat_nominal_att_init_model	Functionality is not currently available	
xp_sat_nominal_att_init_harmonic	Functionality is not currently available	
xp_sat_nominal_att_init_file	Functionality is not currently available	
xp_sat_att_init_harmonic	Functionality is not currently available	
xp_sat_att_init_file	Only CryoSat Star Tracker file is currently supported	
xp_instr_att_init_harmonic	Functionality is not currently available	
xp_instr_att_init_file	Functionality is not currently available	
xp_change_frame	Functionality is not currently available	
xp_dem_init	Functionality is not currently available.	
xp_target_travel_time	Functionality is not currently available	
xp_multi_target_inter	Only available without refraction	
xp_multi_target_travel_time	Only available without refraction	
xp_converter	Not available yet with the updated interfaces	