COLLABORATORS			
	TITLE :		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 22, 2009	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 PLASMA_Version **1**

1.1 Purpose . . . 1
1.2 Arguments . . . 1
1.3 Return Value: . . . 1
1.4 C Bindings . . . 1
1.5 Fortran Bindings . . . 1
1.6 Online Browsing . . . 1

2 PLASMA_cgelqf **1**

2.1 Purpose . . . 1
2.2 Arguments . . . 1
2.3 Return Value: . . . 2
2.4 C Bindings . . . 2
2.5 Fortran Bindings . . . 2
2.6 Online Browsing . . . 2

3 PLASMA_cgelqf_Tile **2**

3.1 Purpose . . . 2
3.2 Arguments . . . 2
3.3 Return Value: . . . 3
3.4 C Bindings . . . 3
3.5 Fortran Bindings . . . 3
3.6 Online Browsing . . . 3

4 PLASMA_cgelqs **3**

4.1 Purpose . . . 3
4.2 Arguments . . . 3
4.3 Return Value: . . . 4
4.4 C Bindings . . . 4
4.5 Fortran Bindings . . . 4
4.6 Online Browsing . . . 4

5 PLASMA_cgelqs_Tile **4**

5.1 Purpose . . . 4
5.2 Arguments . . . 4
5.3 Return Value: . . . 5
5.4 C Bindings . . . 5
5.5 Fortran Bindings . . . 5
5.6 Online Browsing . . . 5

6 PLASMA_cgels **5**

6.1 Purpose . . . 5
6.2 Arguments . . . 5
6.3 Return Value: . . . 6
6.4 C Bindings . . . 6
6.5 Fortran Bindings . . . 6
6.6 Online Browsing . . . 6

7 PLASMA_cgels_Tile **6**

7.1 Purpose . . . 6
7.2 Arguments . . . 7
7.3 Return Value: . . . 7
7.4 C Bindings . . . 7
7.5 Fortran Bindings . . . 7
7.6 Online Browsing . . . 7

8 PLASMA_cgemm **8**

8.1 Purpose . . . 8
8.2 Arguments . . . 8
8.3 Return Value: . . . 9
8.4 C Bindings . . . 9
8.5 Fortran Bindings . . . 9
8.6 Online Browsing . . . 9

9 PLASMA_cgeqrf **9**

9.1 Purpose . . . 9
9.2 Arguments . . . 9
9.3 Return Value: . . . 10
9.4 C Bindings . . . 10
9.5 Fortran Bindings . . . 10
9.6 Online Browsing . . . 10

10 PLASMA_cgeqrf_Tile **10**

10.1 Purpose . . . 10
10.2 Arguments . . . 10
10.3 Return Value: . . . 11
10.4 C Bindings . . . 11
10.5 Fortran Bindings . . . 11
10.6 Online Browsing . . . 11

11 PLASMA_cgeqrs **11**
11.1 Purpose . . . 11
11.2 Arguments . . . 11
11.3 Return Value: . . . 12
11.4 C Bindings . . . 12
11.5 Fortran Bindings . . . 12
11.6 Online Browsing . . . 12

12 PLASMA_cgeqrs_Tile **12**
12.1 Purpose . . . 12
12.2 Arguments . . . 12
12.3 Return Value: . . . 13
12.4 C Bindings . . . 13
12.5 Fortran Bindings . . . 13
12.6 Online Browsing . . . 13

13 PLASMA_cgesv **13**
13.1 Purpose . . . 13
13.2 Arguments . . . 13
13.3 Return Value: . . . 14
13.4 C Bindings . . . 14
13.5 Fortran Bindings . . . 14
13.6 Online Browsing . . . 14

14 PLASMA_cgesv_Tile **14**
14.1 Purpose . . . 14
14.2 Arguments . . . 14
14.3 Return Value: . . . 15
14.4 C Bindings . . . 15
14.5 Fortran Bindings . . . 15
14.6 Online Browsing . . . 15

15 PLASMA_cgetrf **15**
15.1 Purpose . . . 15
15.2 Arguments . . . 15
15.3 Return Value: . . . 16
15.4 C Bindings . . . 16
15.5 Fortran Bindings . . . 16
15.6 Online Browsing . . . 16

16 PLASMA_cgetrf_Tile **16**
16.1 Purpose . . . 16
16.2 Arguments . . . 16
16.3 Return Value: . . . 16
16.4 C Bindings . . . 17
16.5 Fortran Bindings . . . 17
16.6 Online Browsing . . . 17

17 PLASMA_cgetrs **17**
17.1 Purpose . . . 17
17.2 Arguments . . . 17
17.3 Return Value: . . . 18
17.4 C Bindings . . . 18
17.5 Fortran Bindings . . . 18
17.6 Online Browsing . . . 18

18 PLASMA_cgetrs_Tile **18**
18.1 Purpose . . . 18
18.2 Arguments . . . 18
18.3 Return Value: . . . 18
18.4 C Bindings . . . 19
18.5 Fortran Bindings . . . 19
18.6 Online Browsing . . . 19

19 PLASMA_Enable **19**
19.1 Purpose . . . 19
19.2 Arguments . . . 19
19.3 Return Value: . . . 19
19.4 C Bindings . . . 19
19.5 Fortran Bindings . . . 19
19.6 Online Browsing . . . 19

20 PLASMA_Disable **20**
20.1 Purpose . . . 20
20.2 Arguments . . . 20
20.3 Return Value: . . . 20
20.4 C Bindings . . . 20
20.5 Fortran Bindings . . . 20
20.6 Online Browsing . . . 20

21 PLASMA_Set **20**
21.1 Purpose . . . 20
21.2 Arguments . . . 20
21.3 Return Value: . . . 21
21.4 C Bindings . . . 21
21.5 Fortran Bindings . . . 21
21.6 Online Browsing . . . 21

22 PLASMA_Get **21**
22.1 Purpose . . . 21
22.2 Arguments . . . 21
22.3 Return Value: . . . 21
22.4 C Bindings . . . 21
22.5 Fortran Bindings . . . 21
22.6 Online Browsing . . . 22

23 PLASMA_Init **22**
23.1 Purpose . . . 22
23.2 Arguments . . . 22
23.3 Return Value: . . . 22
23.4 C Bindings . . . 22
23.5 Fortran Bindings . . . 22
23.6 Online Browsing . . . 22

24 PLASMA_Finalize **22**
24.1 Purpose . . . 22
24.2 C Bindings . . . 22
24.3 Fortran Bindings . . . 22
24.4 Online Browsing . . . 23

25 PLASMA_cposv **23**
25.1 Purpose . . . 23
25.2 Arguments . . . 23
25.3 Return Value: . . . 24
25.4 C Bindings . . . 24
25.5 Fortran Bindings . . . 24
25.6 Online Browsing . . . 24

26 PLASMA_cposv_Tile **24**
26.1 Purpose . . . 24
26.2 Arguments . . . 24
26.3 Return Value: . . . 25
26.4 C Bindings . . . 25
26.5 Fortran Bindings . . . 25
26.6 Online Browsing . . . 25

27 PLASMA_cpotrf **25**
27.1 Purpose . . . 25
27.2 Arguments . . . 25
27.3 Return Value: . . . 26
27.4 C Bindings . . . 26
27.5 Fortran Bindings . . . 26
27.6 Online Browsing . . . 26

28 PLASMA_cpotrf_Tile **26**
28.1 Purpose . . . 26
28.2 Arguments . . . 27
28.3 Return Value: . . . 27
28.4 C Bindings . . . 27
28.5 Fortran Bindings . . . 27
28.6 Online Browsing . . . 27

29 PLASMA_cpotrs **27**
29.1 Purpose . . . 27
29.2 Arguments . . . 28
29.3 Return Value: . . . 28
29.4 C Bindings . . . 28
29.5 Fortran Bindings . . . 28
29.6 Online Browsing . . . 28

30 PLASMA_cpotrs_Tile **28**
30.1 Purpose . . . 28
30.2 Arguments . . . 29
30.3 Return Value: . . . 29
30.4 C Bindings . . . 29
30.5 Fortran Bindings . . . 29
30.6 Online Browsing . . . 29

31 PLASMA_ctrsm **29**
31.1 Purpose . . . 29
31.2 Arguments . . . 29
31.3 Return Value: . . . 30
31.4 C Bindings . . . 30
31.5 Fortran Bindings . . . 30
31.6 Online Browsing . . . 31

32 PLASMA_ctrsmpl **31**
32.1 Purpose . . . 31
32.2 Arguments . . . 31
32.3 Return Value: . . . 31
32.4 C Bindings . . . 31
32.5 Fortran Bindings . . . 32
32.6 Online Browsing . . . 32

33 PLASMA_ctrsmpl_Tile **32**
33.1 Purpose . . . 32
33.2 Arguments . . . 32
33.3 Return Value: . . . 32
33.4 C Bindings . . . 32
33.5 Fortran Bindings . . . 32
33.6 Online Browsing . . . 32

34 PLASMA_ctrsm_Tile **33**
34.1 Purpose . . . 33
34.2 Arguments . . . 33
34.3 Return Value: . . . 33
34.4 C Bindings . . . 33
34.5 Fortran Bindings . . . 34
34.6 Online Browsing . . . 34

35 PLASMA_cunglq **34**
35.1 Purpose . . . 34
35.2 Arguments . . . 34
35.3 Return Value: . . . 34
35.4 C Bindings . . . 35
35.5 Fortran Bindings . . . 35
35.6 Online Browsing . . . 35

36 PLASMA_cunglq_Tile **35**
36.1 Purpose . . . 35
36.2 Arguments . . . 35
36.3 Return Value: . . . 35
36.4 C Bindings . . . 35
36.5 Fortran Bindings . . . 35
36.6 Online Browsing . . . 36

37 PLASMA_cungqr **36**
37.1 Purpose . . . 36
37.2 Arguments . . . 36
37.3 Return Value: . . . 36
37.4 C Bindings . . . 36
37.5 Fortran Bindings . . . 37
37.6 Online Browsing . . . 37

38 PLASMA_cungqr_Tile **37**
38.1 Purpose . . . 37
38.2 Arguments . . . 37
38.3 Return Value: . . . 37
38.4 C Bindings . . . 37
38.5 Fortran Bindings . . . 37
38.6 Online Browsing . . . 37

39 PLASMA_cunmlq **38**
39.1 Purpose . . . 38
39.2 Arguments . . . 38
39.3 Return Value: . . . 38
39.4 C Bindings . . . 39
39.5 Fortran Bindings . . . 39
39.6 Online Browsing . . . 39

40 PLASMA_cunmlq_Tile **39**
40.1 Purpose . . . 39
40.2 Arguments . . . 39
40.3 Return Value: . . . 39
40.4 C Bindings . . . 40
40.5 Fortran Bindings . . . 40
40.6 Online Browsing . . . 40

41 PLASMA_cunmqr **40**
41.1 Purpose . . . 40
41.2 Arguments . . . 40
41.3 Return Value: . . . 41
41.4 C Bindings . . . 41
41.5 Fortran Bindings . . . 41
41.6 Online Browsing . . . 41

42 PLASMA_cunmqr_Tile **41**
42.1 Purpose . . . 41
42.2 Arguments . . . 41
42.3 Return Value: . . . 42
42.4 C Bindings . . . 42
42.5 Fortran Bindings . . . 42
42.6 Online Browsing . . . 42

43 PLASMA_Desc_Create **42**
43.1 Purpose . . . 42
43.2 Arguments . . . 42
43.3 Return Value: . . . 43
43.4 C Bindings . . . 43
43.5 Fortran Bindings . . . 43
43.6 Online Browsing . . . 43

44 PLASMA_Desc_Destroy **43**
44.1 Purpose . . . 43
44.2 Arguments . . . 44
44.3 Return Value: . . . 44
44.4 C Bindings . . . 44
44.5 Fortran Bindings . . . 44
44.6 Online Browsing . . . 44

45 PLASMA_dgelqf **44**
45.1 Purpose . . . 44
45.2 Arguments . . . 44
45.3 Return Value: . . . 45
45.4 C Bindings . . . 45
45.5 Fortran Bindings . . . 45
45.6 Online Browsing . . . 45

46 PLASMA_dgelqf_Tile **45**
46.1 Purpose . . . 45
46.2 Arguments . . . 45
46.3 Return Value: . . . 45
46.4 C Bindings . . . 46
46.5 Fortran Bindings . . . 46
46.6 Online Browsing . . . 46

47 PLASMA_dgelqs **46**
47.1 Purpose . . . 46
47.2 Arguments . . . 46
47.3 Return Value: . . . 46
47.4 C Bindings . . . 47
47.5 Fortran Bindings . . . 47
47.6 Online Browsing . . . 47

48 PLASMA_dgelqs_Tile **47**
48.1 Purpose . . . 47
48.2 Arguments . . . 47
48.3 Return Value: . . . 47
48.4 C Bindings . . . 47
48.5 Fortran Bindings . . . 47
48.6 Online Browsing . . . 48

49 PLASMA_dgels **48**
49.1 Purpose . . . 48
49.2 Arguments . . . 48
49.3 Return Value: . . . 49
49.4 C Bindings . . . 49
49.5 Fortran Bindings . . . 49
49.6 Online Browsing . . . 49

50 PLASMA_dgels_Tile **49**
50.1 Purpose . . . 49
50.2 Arguments . . . 50
50.3 Return Value: . . . 50
50.4 C Bindings . . . 50
50.5 Fortran Bindings . . . 50
50.6 Online Browsing . . . 50

51 PLASMA_dgemm **50**
51.1 Purpose . . . 50
51.2 Arguments . . . 51
51.3 Return Value: . . . 52
51.4 C Bindings . . . 52
51.5 Fortran Bindings . . . 52
51.6 Online Browsing . . . 52

52 PLASMA_dgeqrf **52**
52.1 Purpose . . . 52
52.2 Arguments . . . 52
52.3 Return Value: . . . 53
52.4 C Bindings . . . 53
52.5 Fortran Bindings . . . 53
52.6 Online Browsing . . . 53

53 PLASMA_dgeqrf_Tile **53**
53.1 Purpose . . . 53
53.2 Arguments . . . 53
53.3 Return Value: . . . 54
53.4 C Bindings . . . 54
53.5 Fortran Bindings . . . 54
53.6 Online Browsing . . . 54

54 PLASMA_dgeqrs **54**
54.1 Purpose . . . 54
54.2 Arguments . . . 54
54.3 Return Value: . . . 55
54.4 C Bindings . . . 55
54.5 Fortran Bindings . . . 55
54.6 Online Browsing . . . 55

55 PLASMA_dgeqrs_Tile **55**
55.1 Purpose . . . 55
55.2 Arguments . . . 55
55.3 Return Value: . . . 55
55.4 C Bindings . . . 55
55.5 Fortran Bindings . . . 56
55.6 Online Browsing . . . 56

56 PLASMA_dgesv **56**
56.1 Purpose . . . 56
56.2 Arguments . . . 56
56.3 Return Value: . . . 56
56.4 C Bindings . . . 57
56.5 Fortran Bindings . . . 57
56.6 Online Browsing . . . 57

57 PLASMA_dgesv_Tile **57**
57.1 Purpose . . . 57
57.2 Arguments . . . 57
57.3 Return Value: . . . 57
57.4 C Bindings . . . 58
57.5 Fortran Bindings . . . 58
57.6 Online Browsing . . . 58

58 PLASMA_dgetrf **58**
58.1 Purpose . . . 58
58.2 Arguments . . . 58
58.3 Return Value: . . . 58
58.4 C Bindings . . . 59
58.5 Fortran Bindings . . . 59
58.6 Online Browsing . . . 59

59 PLASMA_dgetrf_Tile **59**
59.1 Purpose . . . 59
59.2 Arguments . . . 59
59.3 Return Value: . . . 59
59.4 C Bindings . . . 59
59.5 Fortran Bindings . . . 59
59.6 Online Browsing . . . 60

60 PLASMA_dgetrs **60**
60.1 Purpose . . . 60
60.2 Arguments . . . 60
60.3 Return Value: . . . 60
60.4 C Bindings . . . 61
60.5 Fortran Bindings . . . 61
60.6 Online Browsing . . . 61

61 PLASMA_dgetrs_Tile **61**
61.1 Purpose . . . 61
61.2 Arguments . . . 61
61.3 Return Value: . . . 61
61.4 C Bindings . . . 61
61.5 Fortran Bindings . . . 62
61.6 Online Browsing . . . 62

62 PLASMA_dposv **62**
62.1 Purpose . . . 62
62.2 Arguments . . . 62
62.3 Return Value: . . . 63
62.4 C Bindings . . . 63
62.5 Fortran Bindings . . . 63
62.6 Online Browsing . . . 63

63 PLASMA_dposv_Tile **63**
63.1 Purpose . . . 63
63.2 Arguments . . . 63
63.3 Return Value: . . . 64
63.4 C Bindings . . . 64
63.5 Fortran Bindings . . . 64
63.6 Online Browsing . . . 64

64 PLASMA_dpotrf **64**
64.1 Purpose . . . 64
64.2 Arguments . . . 65
64.3 Return Value: . . . 65
64.4 C Bindings . . . 65
64.5 Fortran Bindings . . . 65
64.6 Online Browsing . . . 65

65 PLASMA_dpotrf_Tile **65**
65.1 Purpose . . . 65
65.2 Arguments . . . 66
65.3 Return Value: . . . 66
65.4 C Bindings . . . 66
65.5 Fortran Bindings . . . 66
65.6 Online Browsing . . . 66

66 PLASMA_dpotrs **66**
66.1 Purpose . . . 66
66.2 Arguments . . . 67
66.3 Return Value: . . . 67
66.4 C Bindings . . . 67
66.5 Fortran Bindings . . . 67
66.6 Online Browsing . . . 67

67 PLASMA_dpotrs_Tile **67**
67.1 Purpose . . . 67
67.2 Arguments . . . 68
67.3 Return Value: . . . 68
67.4 C Bindings . . . 68
67.5 Fortran Bindings . . . 68
67.6 Online Browsing . . . 68

68 PLASMA_dsgesv **68**
68.1 Purpose . . . 68
68.2 Arguments . . . 69
68.3 Return Value: . . . 69
68.4 C Bindings . . . 70
68.5 Fortran Bindings . . . 70
68.6 Online Browsing . . . 70

69 PLASMA_dsgesv **70**
69.1 Purpose . . . 70
69.2 Arguments . . . 71
69.3 Return Value: . . . 71
69.4 C Bindings . . . 71
69.5 Fortran Bindings . . . 71
69.6 Online Browsing . . . 72

70 PLASMA_dtrsm **72**
70.1 Purpose . . . 72
70.2 Arguments . . . 72
70.3 Return Value: . . . 73
70.4 C Bindings . . . 73
70.5 Fortran Bindings . . . 73
70.6 Online Browsing . . . 73

71 PLASMA_dtrsmpl **73**
71.1 Purpose . . . 73
71.2 Arguments . . . 73
71.3 Return Value: . . . 74
71.4 C Bindings . . . 74
71.5 Fortran Bindings . . . 74
71.6 Online Browsing . . . 74

72 PLASMA_dtrsmpl_Tile **74**
72.1 Purpose . . . 74
72.2 Arguments . . . 74
72.3 Return Value: . . . 75
72.4 C Bindings . . . 75
72.5 Fortran Bindings . . . 75
72.6 Online Browsing . . . 75

73 PLASMA_dtrsm_Tile **75**
73.1 Purpose . . . 75
73.2 Arguments . . . 75
73.3 Return Value: . . . 76
73.4 C Bindings . . . 76
73.5 Fortran Bindings . . . 76
73.6 Online Browsing . . . 76

74 PLASMA_sgelqf **76**
74.1 Purpose . . . 76
74.2 Arguments . . . 76
74.3 Return Value: . . . 77
74.4 C Bindings . . . 77
74.5 Fortran Bindings . . . 77
74.6 Online Browsing . . . 77

75 PLASMA_sgelqf_Tile **77**
75.1 Purpose . . . 77
75.2 Arguments . . . 77
75.3 Return Value: . . . 78
75.4 C Bindings . . . 78
75.5 Fortran Bindings . . . 78
75.6 Online Browsing . . . 78

76 PLASMA_sgelqs **78**
76.1 Purpose . . . 78
76.2 Arguments . . . 78
76.3 Return Value: . . . 79
76.4 C Bindings . . . 79
76.5 Fortran Bindings . . . 79
76.6 Online Browsing . . . 79

77 PLASMA_sgelqs_Tile **79**
77.1 Purpose . . . 79
77.2 Arguments . . . 79
77.3 Return Value: . . . 80
77.4 C Bindings . . . 80
77.5 Fortran Bindings . . . 80
77.6 Online Browsing . . . 80

78 PLASMA_sgels **80**
78.1 Purpose . . . 80
78.2 Arguments . . . 80
78.3 Return Value: . . . 81
78.4 C Bindings . . . 81
78.5 Fortran Bindings . . . 81
78.6 Online Browsing . . . 81

79 PLASMA_sgels_Tile **81**
79.1 Purpose . . . 81
79.2 Arguments . . . 82
79.3 Return Value: . . . 82
79.4 C Bindings . . . 82
79.5 Fortran Bindings . . . 82
79.6 Online Browsing . . . 82

80 PLASMA_sgemm **83**
80.1 Purpose . . . 83
80.2 Arguments . . . 83
80.3 Return Value: . . . 84
80.4 C Bindings . . . 84
80.5 Fortran Bindings . . . 84
80.6 Online Browsing . . . 84

81 PLASMA_sgeqrf **84**
81.1 Purpose . . . 84
81.2 Arguments . . . 84
81.3 Return Value: . . . 85
81.4 C Bindings . . . 85
81.5 Fortran Bindings . . . 85
81.6 Online Browsing . . . 85

82 PLASMA_sgeqrf_Tile **85**
82.1 Purpose . . . 85
82.2 Arguments . . . 85
82.3 Return Value: . . . 86
82.4 C Bindings . . . 86
82.5 Fortran Bindings . . . 86
82.6 Online Browsing . . . 86

83 PLASMA_sgeqrs **86**
83.1 Purpose . . . 86
83.2 Arguments . . . 86
83.3 Return Value: . . . 87
83.4 C Bindings . . . 87
83.5 Fortran Bindings . . . 87
83.6 Online Browsing . . . 87

84 PLASMA_sgeqrs_Tile **87**
84.1 Purpose . . . 87
84.2 Arguments . . . 87
84.3 Return Value: . . . 88
84.4 C Bindings . . . 88
84.5 Fortran Bindings . . . 88
84.6 Online Browsing . . . 88

85 PLASMA_sgesv **88**
85.1 Purpose . . . 88
85.2 Arguments . . . 88
85.3 Return Value: . . . 89
85.4 C Bindings . . . 89
85.5 Fortran Bindings . . . 89
85.6 Online Browsing . . . 89

86 PLASMA_sgesv_Tile **89**
86.1 Purpose . . . 89
86.2 Arguments . . . 89
86.3 Return Value: . . . 90
86.4 C Bindings . . . 90
86.5 Fortran Bindings . . . 90
86.6 Online Browsing . . . 90

87 PLASMA_sgetrf **90**
87.1 Purpose . . . 90
87.2 Arguments . . . 90
87.3 Return Value: . . . 91
87.4 C Bindings . . . 91
87.5 Fortran Bindings . . . 91
87.6 Online Browsing . . . 91

88 PLASMA_sgetrf_Tile **91**
88.1 Purpose . . . 91
88.2 Arguments . . . 91
88.3 Return Value: . . . 91
88.4 C Bindings . . . 92
88.5 Fortran Bindings . . . 92
88.6 Online Browsing . . . 92

89 PLASMA_sgetrs **92**
89.1 Purpose . . . 92
89.2 Arguments . . . 92
89.3 Return Value: . . . 93
89.4 C Bindings . . . 93
89.5 Fortran Bindings . . . 93
89.6 Online Browsing . . . 93

90 PLASMA_sgetrs_Tile **93**
90.1 Purpose . . . 93
90.2 Arguments . . . 93
90.3 Return Value: . . . 93
90.4 C Bindings . . . 94
90.5 Fortran Bindings . . . 94
90.6 Online Browsing . . . 94

91 PLASMA_sposv **94**
91.1 Purpose . . . 94
91.2 Arguments . . . 94
91.3 Return Value: . . . 95
91.4 C Bindings . . . 95
91.5 Fortran Bindings . . . 95
91.6 Online Browsing . . . 95

92 PLASMA_sposv_Tile **95**
92.1 Purpose . . . 95
92.2 Arguments . . . 96
92.3 Return Value: . . . 96
92.4 C Bindings . . . 96
92.5 Fortran Bindings . . . 96
92.6 Online Browsing . . . 96

93 PLASMA_spotrf **96**
93.1 Purpose . . . 96
93.2 Arguments . . . 97
93.3 Return Value: . . . 97
93.4 C Bindings . . . 97
93.5 Fortran Bindings . . . 97
93.6 Online Browsing . . . 97

94 PLASMA_spotrf_Tile **97**
94.1 Purpose . . . 97
94.2 Arguments . . . 98
94.3 Return Value: . . . 98
94.4 C Bindings . . . 98
94.5 Fortran Bindings . . . 98
94.6 Online Browsing . . . 98

95 PLASMA_spotrs **98**
95.1 Purpose . . . 98
95.2 Arguments . . . 99
95.3 Return Value: . . . 99
95.4 C Bindings . . . 99
95.5 Fortran Bindings . . . 99
95.6 Online Browsing . . . 99

96 PLASMA_spotrs_Tile **99**
96.1 Purpose . . . 99
96.2 Arguments . . . 100
96.3 Return Value: . . . 100
96.4 C Bindings . . . 100
96.5 Fortran Bindings . . . 100
96.6 Online Browsing . . . 100

97 PLASMA_strsm **100**
97.1 Purpose . . . 100
97.2 Arguments . . . 100
97.3 Return Value: . . . 101
97.4 C Bindings . . . 101
97.5 Fortran Bindings . . . 101
97.6 Online Browsing . . . 101

98 PLASMA_strsmpl **102**
98.1 Purpose . . . 102
98.2 Arguments . . . 102
98.3 Return Value: . . . 102
98.4 C Bindings . . . 102
98.5 Fortran Bindings . . . 102
98.6 Online Browsing . . . 103

99 PLASMA_strsmpl_Tile **103**
99.1 Purpose . . . 103
99.2 Arguments . . . 103
99.3 Return Value: . . . 103
99.4 C Bindings . . . 103
99.5 Fortran Bindings . . . 103
99.6 Online Browsing . . . 103

100PLASMA_strsm_Tile **103**
100.1Purpose . . . 103
100.2Arguments . . . 104
100.3Return Value: . . . 104
100.4C Bindings . . . 104
100.5Fortran Bindings . . . 104
100.6Online Browsing . . . 104

101PLASMA_Lapack_to_Tile **105**
101.1Purpose . . . 105
101.2Arguments . . . 105
101.3Return Value: . . . 105
101.4C Bindings . . . 105
101.5Fortran Bindings . . . 105
101.6Online Browsing . . . 105

102PLASMA_Tile_to_Lapack **105**
102.1Purpose . . . 105
102.2Arguments . . . 105
102.3Return Value: . . . 106
102.4C Bindings . . . 106
102.5Fortran Bindings . . . 106
102.6Online Browsing . . . 106

103PLASMA_Dealloc_Handle **106**
103.1Purpose . . . 106
103.2Arguments . . . 106
103.3Return Value: . . . 106
103.4C Bindings . . . 106
103.5Fortran Bindings . . . 106
103.6Online Browsing . . . 106

104PLASMA_Alloc_Workspace_cgels **107**
104.1Purpose . . . 107
104.2Arguments . . . 107
104.3Return Value: . . . 107
104.4C Bindings . . . 107
104.5Fortran Bindings . . . 107
104.6Online Browsing . . . 107

105PLASMA_Alloc_Workspace_cgeqrf **107**
105.1Purpose . . . 107
105.2Arguments . . . 108
105.3Return Value: . . . 108
105.4C Bindings . . . 108
105.5Fortran Bindings . . . 108
105.6Online Browsing . . . 108

106PLASMA_Alloc_Workspace_cgelqf **108**
106.1Purpose . . . 108
106.2Arguments . . . 108
106.3Return Value: . . . 109
106.4C Bindings . . . 109
106.5Fortran Bindings . . . 109
106.6Online Browsing . . . 109

107PLASMA_Alloc_Workspace_cgesv **109**
107.1Purpose . . . 109
107.2Arguments . . . 109
107.3Return Value: . . . 109
107.4C Bindings . . . 109
107.5Fortran Bindings . . . 110
107.6Online Browsing . . . 110

108PLASMA_Alloc_Workspace_cgetrf **110**
108.1Purpose . . . 110
108.2Arguments . . . 110
108.3Return Value: . . . 110
108.4C Bindings . . . 110
108.5Fortran Bindings . . . 110
108.6Online Browsing . . . 110

109PLASMA_Alloc_Workspace_dgels **111**
109.1Purpose . . . 111
109.2Arguments . . . 111
109.3Return Value: . . . 111
109.4C Bindings . . . 111
109.5Fortran Bindings . . . 111
109.6Online Browsing . . . 111

110PLASMA_Alloc_Workspace_dgeqrf **111**
110.1Purpose . . . 111
110.2Arguments . . . 112
110.3Return Value: . . . 112
110.4C Bindings . . . 112
110.5Fortran Bindings . . . 112
110.6Online Browsing . . . 112

111PLASMA_Alloc_Workspace_dgelqf **112**
111.1Purpose . . . 112
111.2Arguments . . . 112
111.3Return Value: . . . 113
111.4C Bindings . . . 113
111.5Fortran Bindings . . . 113
111.6Online Browsing . . . 113

112PLASMA_Alloc_Workspace_dgesv **113**
112.1Purpose . . . 113
112.2Arguments . . . 113
112.3Return Value: . . . 113
112.4C Bindings . . . 113
112.5Fortran Bindings . . . 114
112.6Online Browsing . . . 114

113PLASMA_Alloc_Workspace_dgetrf **114**
113.1Purpose . . . 114
113.2Arguments . . . 114
113.3Return Value: . . . 114
113.4C Bindings . . . 114
113.5Fortran Bindings . . . 114
113.6Online Browsing . . . 114

114PLASMA_Alloc_Workspace_sgels **115**
114.1Purpose . . . 115
114.2Arguments . . . 115
114.3Return Value: . . . 115
114.4C Bindings . . . 115
114.5Fortran Bindings . . . 115
114.6Online Browsing . . . 115

115PLASMA_Alloc_Workspace_sgeqrf **115**
115.1Purpose . . . 115
115.2Arguments . . . 116
115.3Return Value: . . . 116
115.4C Bindings . . . 116
115.5Fortran Bindings . . . 116
115.6Online Browsing . . . 116

116PLASMA_Alloc_Workspace_sgelqf **116**
116.1Purpose . . . 116
116.2Arguments . . . 116
116.3Return Value: . . . 117
116.4C Bindings . . . 117
116.5Fortran Bindings . . . 117
116.6Online Browsing . . . 117

117PLASMA_Alloc_Workspace_sgesv **117**
117.1Purpose . . . 117
117.2Arguments . . . 117
117.3Return Value: . . . 117
117.4C Bindings . . . 117
117.5Fortran Bindings . . . 118
117.6Online Browsing . . . 118

118PLASMA_Alloc_Workspace_sgetrf **118**
118.1Purpose . . . 118
118.2Arguments . . . 118
118.3Return Value: . . . 118
118.4C Bindings . . . 118
118.5Fortran Bindings . . . 118
118.6Online Browsing . . . 118

119PLASMA_Alloc_Workspace_zgels **119**
119.1Purpose . . . 119
119.2Arguments . . . 119
119.3Return Value: . . . 119
119.4C Bindings . . . 119
119.5Fortran Bindings . . . 119
119.6Online Browsing . . . 119

120PLASMA_Alloc_Workspace_zgeqrf **119**
120.1Purpose . . . 119
120.2Arguments . . . 120
120.3Return Value: . . . 120
120.4C Bindings . . . 120
120.5Fortran Bindings . . . 120
120.6Online Browsing . . . 120

121PLASMA_Alloc_Workspace_zgelqf **120**
121.1Purpose . . . 120
121.2Arguments . . . 120
121.3Return Value: . . . 121
121.4C Bindings . . . 121
121.5Fortran Bindings . . . 121
121.6Online Browsing . . . 121

122PLASMA_Alloc_Workspace_zgesv **121**
122.1Purpose . . . 121
122.2Arguments . . . 121
122.3Return Value: . . . 121
122.4C Bindings . . . 121
122.5Fortran Bindings . . . 122
122.6Online Browsing . . . 122

123PLASMA_Alloc_Workspace_zgetrf **122**
123.1Purpose . . . 122
123.2Arguments . . . 122
123.3Return Value: . . . 122
123.4C Bindings . . . 122
123.5Fortran Bindings . . . 122
123.6Online Browsing . . . 122

124PLASMA_zcgesv **123**
124.1Purpose . . . 123
124.2Arguments . . . 123
124.3Return Value: . . . 124
124.4C Bindings . . . 124
124.5Fortran Bindings . . . 124
124.6Online Browsing . . . 124

125PLASMA_zcgesv **124**
125.1Purpose . . . 124
125.2Arguments . . . 125
125.3Return Value: . . . 126
125.4C Bindings . . . 126
125.5Fortran Bindings . . . 126
125.6Online Browsing . . . 126

126PLASMA_zgelqf **126**
126.1Purpose . . . 126
126.2Arguments . . . 126
126.3Return Value: . . . 127
126.4C Bindings . . . 127
126.5Fortran Bindings . . . 127
126.6Online Browsing . . . 127

127PLASMA_zgelqf_Tile **127**
127.1Purpose . . . 127
127.2Arguments . . . 127
127.3Return Value: . . . 127
127.4C Bindings . . . 127
127.5Fortran Bindings . . . 128
127.6Online Browsing . . . 128

128PLASMA_zgelqs **128**
128.1Purpose . . . 128
128.2Arguments . . . 128
128.3Return Value: . . . 128
128.4C Bindings . . . 128
128.5Fortran Bindings . . . 129
128.6Online Browsing . . . 129

129PLASMA_zgelqs_Tile **129**
129.1Purpose . . . 129
129.2Arguments . . . 129
129.3Return Value: . . . 129
129.4C Bindings . . . 129
129.5Fortran Bindings . . . 129
129.6Online Browsing . . . 129

130PLASMA_zgels **130**
130.1Purpose . . . 130
130.2Arguments . . . 130
130.3Return Value: . . . 131
130.4C Bindings . . . 131
130.5Fortran Bindings . . . 131
130.6Online Browsing . . . 131

131PLASMA_zgels_Tile **131**
131.1Purpose . . . 131
131.2Arguments . . . 131
131.3Return Value: . . . 132
131.4C Bindings . . . 132
131.5Fortran Bindings . . . 132
131.6Online Browsing . . . 132

132PLASMA_zgemm **132**
132.1Purpose . . . 132
132.2Arguments . . . 133
132.3Return Value: . . . 133
132.4C Bindings . . . 134
132.5Fortran Bindings . . . 134
132.6Online Browsing . . . 134

133PLASMA_zgeqrf **134**
133.1Purpose . . . 134
133.2Arguments . . . 134
133.3Return Value: . . . 134
133.4C Bindings . . . 135
133.5Fortran Bindings . . . 135
133.6Online Browsing . . . 135

134PLASMA_zgeqrf_Tile **135**
134.1Purpose . . . 135
134.2Arguments . . . 135
134.3Return Value: . . . 135
134.4C Bindings . . . 135
134.5Fortran Bindings . . . 135
134.6Online Browsing . . . 136

135PLASMA_zgeqrs **136**
135.1Purpose . . . 136
135.2Arguments . . . 136
135.3Return Value: . . . 136
135.4C Bindings . . . 136
135.5Fortran Bindings . . . 136
135.6Online Browsing . . . 137

136PLASMA_zgeqrs_Tile **137**
136.1Purpose . . . 137
136.2Arguments . . . 137
136.3Return Value: . . . 137
136.4C Bindings . . . 137
136.5Fortran Bindings . . . 137
136.6Online Browsing . . . 137

137PLASMA_zgesv **137**
137.1Purpose . . . 137
137.2Arguments . . . 138
137.3Return Value: . . . 138
137.4C Bindings . . . 138
137.5Fortran Bindings . . . 138
137.6Online Browsing . . . 138

138PLASMA_zgesv_Tile **139**
138.1Purpose . . . 139
138.2Arguments . . . 139
138.3Return Value: . . . 139
138.4C Bindings . . . 139
138.5Fortran Bindings . . . 139
138.6Online Browsing . . . 139

139PLASMA_zgetrf **140**
139.1Purpose . . . 140
139.2Arguments . . . 140
139.3Return Value: . . . 140
139.4C Bindings . . . 140
139.5Fortran Bindings . . . 140
139.6Online Browsing . . . 140

140PLASMA_zgetrf_Tile **141**
140.1Purpose . . . 141
140.2Arguments . . . 141
140.3Return Value: . . . 141
140.4C Bindings . . . 141
140.5Fortran Bindings . . . 141
140.6Online Browsing . . . 141

141PLASMA_zgetrs **141**
141.1Purpose . . . 141
141.2Arguments . . . 142
141.3Return Value: . . . 142
141.4C Bindings . . . 142
141.5Fortran Bindings . . . 142
141.6Online Browsing . . . 142

142PLASMA_zgetrs_Tile **143**
142.1Purpose . . . 143
142.2Arguments . . . 143
142.3Return Value: . . . 143
142.4C Bindings . . . 143
142.5Fortran Bindings . . . 143
142.6Online Browsing . . . 143

143PLASMA_zposv **143**
143.1Purpose . . . 143
143.2Arguments . . . 144
143.3Return Value: . . . 144
143.4C Bindings . . . 144
143.5Fortran Bindings . . . 145
143.6Online Browsing . . . 145

144PLASMA_zposv_Tile **145**
144.1Purpose . . . 145
144.2Arguments . . . 145
144.3Return Value: . . . 145
144.4C Bindings . . . 146
144.5Fortran Bindings . . . 146
144.6Online Browsing . . . 146

145PLASMA_zpotrf **146**
145.1Purpose . . . 146
145.2Arguments . . . 146
145.3Return Value: . . . 147
145.4C Bindings . . . 147
145.5Fortran Bindings . . . 147
145.6Online Browsing . . . 147

146PLASMA_zpotrf_Tile **147**
146.1Purpose . . . 147
146.2Arguments . . . 147
146.3Return Value: . . . 148
146.4C Bindings . . . 148
146.5Fortran Bindings . . . 148
146.6Online Browsing . . . 148

147PLASMA_zpotrs **148**
147.1Purpose . . . 148
147.2Arguments . . . 148
147.3Return Value: . . . 149
147.4C Bindings . . . 149
147.5Fortran Bindings . . . 149
147.6Online Browsing . . . 149

148PLASMA_zpotrs_Tile **149**
148.1Purpose . . . 149
148.2Arguments . . . 149
148.3Return Value: . . . 149
148.4C Bindings . . . 150
148.5Fortran Bindings . . . 150
148.6Online Browsing . . . 150

149PLASMA_ztrsm **150**
149.1Purpose . . . 150
149.2Arguments . . . 150
149.3Return Value: . . . 151
149.4C Bindings . . . 151
149.5Fortran Bindings . . . 151
149.6Online Browsing . . . 151

150PLASMA_ztrsmpl **151**
150.1Purpose . . . 151
150.2Arguments . . . 151
150.3Return Value: . . . 152
150.4C Bindings . . . 152
150.5Fortran Bindings . . . 152
150.6Online Browsing . . . 152

151PLASMA_ztrsmpl_Tile **152**
151.1Purpose . . . 152
151.2Arguments . . . 153
151.3Return Value: . . . 153
151.4C Bindings . . . 153
151.5Fortran Bindings . . . 153
151.6Online Browsing . . . 153

152PLASMA_ztrsm_Tile **153**
152.1Purpose . . . 153
152.2Arguments . . . 153
152.3Return Value: . . . 154
152.4C Bindings . . . 154
152.5Fortran Bindings . . . 154
152.6Online Browsing . . . 154

153PLASMA_zunglq **154**
153.1Purpose . . . 154
153.2Arguments . . . 155
153.3Return Value: . . . 155
153.4C Bindings . . . 155
153.5Fortran Bindings . . . 155
153.6Online Browsing . . . 155

154PLASMA_zunglq_Tile **156**
154.1Purpose . . . 156
154.2Arguments . . . 156
154.3Return Value: . . . 156
154.4C Bindings . . . 156
154.5Fortran Bindings . . . 156
154.6Online Browsing . . . 156

155PLASMA_zungqr **156**
155.1Purpose . . . 156
155.2Arguments . . . 157
155.3Return Value: . . . 157
155.4C Bindings . . . 157
155.5Fortran Bindings . . . 157
155.6Online Browsing . . . 157

156PLASMA_zungqr_Tile **158**
156.1Purpose . . . 158
156.2Arguments . . . 158
156.3Return Value: . . . 158
156.4C Bindings . . . 158
156.5Fortran Bindings . . . 158
156.6Online Browsing . . . 158

157PLASMA_zunmlq **158**
157.1Purpose . . . 158
157.2Arguments . . . 159
157.3Return Value: . . . 159
157.4C Bindings . . . 159
157.5Fortran Bindings . . . 159
157.6Online Browsing . . . 160

158PLASMA_zunmlq_Tile **160**
158.1Purpose . . . 160
158.2Arguments . . . 160
158.3Return Value: . . . 160
158.4C Bindings . . . 160
158.5Fortran Bindings . . . 160
158.6Online Browsing . . . 161

159PLASMA_zunmqr **161**
159.1Purpose . . . 161
159.2Arguments . . . 161
159.3Return Value: . . . 161
159.4C Bindings . . . 162
159.5Fortran Bindings . . . 162
159.6Online Browsing . . . 162

160PLASMA_zunmqr_Tile **162**
160.1Purpose . . . 162
160.2Arguments . . . 162
160.3Return Value: . . . 163
160.4C Bindings . . . 163
160.5Fortran Bindings . . . 163
160.6Online Browsing . . . 163

1 PLASMA_Version

1.1 Purpose

PLASMA_Version - Reports PLASMA version number.

1.2 Arguments

```
ver_major    int (OUT)
             PLASMA major version number.

ver_minor    int (OUT)
             PLASMA minor version number.

ver_micro    int (OUT)
             PLASMA micro version number.
```

1.3 Return Value:

```
          = PLASMA_SUCCESS: successful exit
```

1.4 C Bindings

```
int PLASMA_Version(int *ver_major, int *ver_minor, int *ver_micro)
```

1.5 Fortran Bindings

```
PLASMA_VERSION(INTEGER VER_MAJOR, INTEGER VER_MINOR, INTEGER VER_MICRO, INTEGER INFO)
```

1.6 Online Browsing

Dive into PLASMA_Version

2 PLASMA_cgelqf

2.1 Purpose

PLASMA_cgelqf - Computes the tile LQ factorization of a complex M-by-N matrix A: A = L * Q.

2.2 Arguments

```
M        int (IN)
         The number of rows of the matrix A. M >= 0.

N        int (IN)
         The number of columns of the matrix A. N >= 0.
```

```
A         PLASMA_Complex32_t* (INOUT)
          On entry, the M-by-N matrix A.
          On exit, the elements on and below the diagonal of the array contain the m-by-min( ↩
              M,N)
          lower trapezoidal matrix L (L is lower triangular if M <= N); the elements above  ↩
              the
          diagonal represent the unitary matrix Q as a product of elementary reflectors,  ↩
              stored
          by tiles.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,M).

T         PLASMA_Complex32_t* (OUT)
          On exit, auxiliary factorization data, required by PLASMA_cgelqs to solve the  ↩
              system
          of equations.
```

2.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

2.4 C Bindings

```
int PLASMA_cgelqf(int M, int N, PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t *T)
```

2.5 Fortran Bindings

```
PLASMA_CGELQF(INTEGER M, INTEGER N, COMPLEX A, INTEGER LDA, INTEGER T, INTEGER INFO)
```

2.6 Online Browsing

Dive into PLASMA_cgelqf

3 PLASMA_cgelqf_Tile

3.1 Purpose

PLASMA_cgelqf_Tile - Computes the tile LQ factorization of a complex M-by-N matrix A: A = L * Q. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

3.2 Arguments

```
A         PLASMA_Complex32_t* (INOUT)
          On entry, the M-by-N matrix A.
          On exit, the elements on and below the diagonal of the array contain the m-by-min( ↩
              M,N)
          lower trapezoidal matrix L (L is lower triangular if M <= N); the elements above  ↩
              the
```

```
          diagonal represent the unitary matrix Q as a product of elementary reflectors,  ↩
              stored
          by tiles.

T         PLASMA_Complex32_t* (OUT)
          On exit, auxiliary factorization data, required by PLASMA_cgelqs to solve the  ↩
              system
          of equations.
```

3.3 Return Value:

```
           = 0: successful exit
```

3.4 C Bindings

```
int PLASMA_cgelqf_Tile(PLASMA_desc *A, PLASMA_desc *T)
```

3.5 Fortran Bindings

```
PLASMA_CGELQF_TILE(INTEGER*4 A, INTEGER*4 T,INTEGER INFO)
```

3.6 Online Browsing

Dive into PLASMA_cgelqf_Tile

4 PLASMA_cgelqs

4.1 Purpose

PLASMA_cgelqs - Compute a minimum-norm solution min || A*X - B || using the LQ factorization A = L*Q computed by PLASMA_cgelqf.

4.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A. N >= M >= 0.

NRHS      int (IN)
          The number of columns of B. NRHS >= 0.

A         PLASMA_Complex32_t* (IN)
          Details of the LQ factorization of the original matrix A as returned by  ↩
              PLASMA_cgelqf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= M.
```

```
T         PLASMA_Complex32_t* (IN)
          Auxiliary factorization data, computed by PLASMA_cgelqf.

B         PLASMA_Complex32_t* (INOUT)
          On entry, the M-by-NRHS right hand side matrix B.
          On exit, the N-by-NRHS solution matrix X.

LDB       int (IN)
          The leading dimension of the array B. LDB >= N.
```

4.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

4.4 C Bindings

```
int PLASMA_cgelqs(int M, int N, int NRHS, PLASMA_Complex32_t *A, int LDA,  ↩
    PLASMA_Complex32_t *T, PLASMA_Complex32_t *B, int LDB)
```

4.5 Fortran Bindings

```
PLASMA_CGELQS(INTEGER M, INTEGER N, INTEGER NRHS, COMPLEX A, INTEGER LDA, INTEGER T,  ↩
    COMPLEX B, INTEGER LDB, INTEGER INFO)
```

4.6 Online Browsing

Dive into PLASMA_cgelqs

5 PLASMA_cgelqs_Tile

5.1 Purpose

PLASMA_cgelqs_Tile - Compute a minimum-norm solution min || A*X - B || using the LQ factorization A = L*Q computed by PLASMA_cgelqf_Tile. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

5.2 Arguments

```
A         PLASMA_Complex32_t* (IN)
          Details of the LQ factorization of the original matrix A as returned by  ↩
              PLASMA_cgelqf.

T         PLASMA_Complex32_t* (IN)
          Auxiliary factorization data, computed by PLASMA_cgelqf.

B         PLASMA_Complex32_t* (INOUT)
          On entry, the M-by-NRHS right hand side matrix B.
          On exit, the N-by-NRHS solution matrix X.
```

5.3 Return Value:

```
          = 0: successful exit
```

5.4 C Bindings

```
int PLASMA_cgelqs_Tile(PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

5.5 Fortran Bindings

```
PLASMA_CGELQS_TILE(INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

5.6 Online Browsing

Dive into PLASMA_cgelqs_Tile

6 PLASMA_cgels

6.1 Purpose

PLASMA_cgels - solves overdetermined or underdetermined linear systems involving an M-by-N matrix A using the QR or the LQ factorization of A. It is assumed that A has full rank. The following options are provided:

1. trans = PlasmaNoTrans and M >= N: find the least squares solution of an overdetermined system, i.e., solve the least squares problem: minimize ‖ B - A*X ‖.
2. trans = PlasmaNoTrans and M < N: find the minimum norm solution of an underdetermined system A * X = B.

Several right hand side vectors B and solution vectors X can be handled in a single call; they are stored as the columns of the M-by-NRHS right hand side matrix B and the N-by-NRHS solution matrix X.

6.2 Arguments

```
trans    PLASMA_enum (IN)
         Intended usage:
         = PlasmaNoTrans:   the linear system involves A;
         = PlasmaConjTrans: the linear system involves A**H.
         Currently only PlasmaNoTrans is supported.

M        int (IN)
         The number of rows of the matrix A. M >= 0.

N        int (IN)
         The number of columns of the matrix A. N >= 0.

NRHS     int (IN)
         The number of right hand sides, i.e., the number of columns of the matrices B and  ↩
             X.
         NRHS >= 0.

A        PLASMA_Complex32_t* (INOUT)
```

```
          On entry, the M-by-N matrix A.
          On exit,
          if M >= N, A is overwritten by details of its QR factorization as returned by
                     PLASMA_cgeqrf;
          if M < N, A is overwritten by details of its LQ factorization as returned by
                      PLASMA_cgelqf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,M).

T         PLASMA_Complex32_t* (OUT)
          On exit, auxiliary factorization data.

B         PLASMA_Complex32_t* (INOUT)
          On entry, the M-by-NRHS matrix B of right hand side vectors, stored columnwise;
          On exit, if return value = 0, B is overwritten by the solution vectors, stored
          columnwise:
          if M >= N, rows 1 to N of B contain the least squares solution vectors; the  ↩
              residual
          sum of squares for the solution in each column is given by the sum of squares of  ↩
              the
          modulus of elements N+1 to M in that column;
          if M < N, rows 1 to N of B contain the minimum norm solution vectors;

LDB       int (IN)
          The leading dimension of the array B. LDB >= MAX(1,M,N).
```

6.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

6.4 C Bindings

```
int PLASMA_cgels(PLASMA_enum trans, int M, int N, int NRHS, PLASMA_Complex32_t *A, int LDA, ↩
     PLASMA_Complex32_t *T, PLASMA_Complex32_t *B, int LDB)
```

6.5 Fortran Bindings

```
PLASMA_CGELS(INTEGER trans, INTEGER M, INTEGER N, INTEGER NRHS, COMPLEX A, INTEGER LDA,  ↩
    INTEGER T, COMPLEX B, INTEGER LDB, INTEGER INFO)
```

6.6 Online Browsing

Dive into PLASMA_cgels

7 PLASMA_cgels_Tile

7.1 Purpose

PLASMA_cgels_Tile - solves overdetermined or underdetermined linear systems involving an M-by-N matrix A using the QR or the LQ factorization of A. It is assumed that A has full rank. The following options are provided:

1. trans = PlasmaNoTrans and M >= N: find the least squares solution of an overdetermined system, i.e., solve the least squares problem: minimize || B - A*X ||.
2. trans = PlasmaNoTrans and M < N: find the minimum norm solution of an underdetermined system A * X = B.

Several right hand side vectors B and solution vectors X can be handled in a single call; they are stored as the columns of the M-by-NRHS right hand side matrix B and the N-by-NRHS solution matrix X. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

7.2 Arguments

```
trans    PLASMA_enum (IN)
         Intended usage:
         = PlasmaNoTrans:   the linear system involves A;
         = PlasmaConjTrans: the linear system involves A**H.
         Currently only PlasmaNoTrans is supported.

A        PLASMA_Complex32_t* (INOUT)
         On entry, the M-by-N matrix A.
         On exit,
         if M >= N, A is overwritten by details of its QR factorization as returned by
                    PLASMA_cgeqrf;
         if M < N, A is overwritten by details of its LQ factorization as returned by
                     PLASMA_cgelqf.

T        PLASMA_Complex32_t* (OUT)
         On exit, auxiliary factorization data.

B        PLASMA_Complex32_t* (INOUT)
         On entry, the M-by-NRHS matrix B of right hand side vectors, stored columnwise;
         On exit, if return value = 0, B is overwritten by the solution vectors, stored
         columnwise:
         if M >= N, rows 1 to N of B contain the least squares solution vectors; the  ↩
             residual
         sum of squares for the solution in each column is given by the sum of squares of  ↩
             the
         modulus of elements N+1 to M in that column;
         if M < N, rows 1 to N of B contain the minimum norm solution vectors;
```

7.3 Return Value:

```
          = 0: successful exit
```

7.4 C Bindings

```
int PLASMA_cgels_Tile(PLASMA_enum trans, PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

7.5 Fortran Bindings

```
PLASMA_CGELS_TILE(INTEGER trans, INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

7.6 Online Browsing

Dive into PLASMA_cgels_Tile

8 PLASMA_cgemm

8.1 Purpose

PLASMA_cgemm - Performs one of the matrix-matrix operations

```
C = alpha*op( A )*op( B ) + beta*C,
```

where op(X) is one of

```
op( X ) = X  or op( X ) = X'
```

alpha and beta are scalars, and A, B and C are matrices, with op(A) an m by k matrix, op(B) a k by n matrix and C an m by n matrix.

8.2 Arguments

```
transA   PLASMA_enum (IN)
         Specifies whether the matrix A is transposed, not transposed or conjugate  ↩
             transposed:
         = PlasmaNoTrans:   A is transposed;
         = PlasmaTrans:     A is not transposed;
         = PlasmaConjTrans: A is conjugate transposed.
         Currently only PlasmaNoTrans is supported

transB   PLASMA_enum (IN)
         Specifies whether the matrix B is transposed, not transposed or conjugate  ↩
             transposed:
         = PlasmaNoTrans:   B is transposed;
         = PlasmaTrans:     B is not transposed;
         = PlasmaConjTrans: B is conjugate transposed.
         Currently only PlasmaNoTrans is supported

M        int (IN)
         M specifies the number of rows of the matrix op( A ) and of the matrix C. M >= 0.

N        int (IN)
         N specifies the number of columns of the matrix op( B ) and of the matrix C. N >=  ↩
             0.

K        int (IN)
         K specifies the number of columns of the matrix op( A ) and the number of rows of
         the matrix op( B ). K >= 0.

alpha    PLASMA_Complex32_t (IN)
         alpha specifies the scalar alpha

A        PLASMA_Complex32_t* (IN)
         A is a LDA-by-ka matrix, where ka is K when  transA = PlasmaNoTrans,
         and is  M  otherwise.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,M).

B        PLASMA_Complex32_t* (IN)
         B is a LDB-by-kb matrix, where kb is N when  transB = PlasmaNoTrans,
         and is  K  otherwise.
```

```
LDB       int (IN)
          The leading dimension of the array B. LDB >= max(1,N).

beta      PLASMA_Complex32_t (IN)
          beta specifies the scalar beta

C         PLASMA_Complex32_t* (INOUT)
          C is a LDC-by-N matrix.
          On exit, the array is overwritten by the M by N matrix ( alpha*op( A )*op( B ) +  ↩
              beta*C )

LDC       int (IN)
          The leading dimension of the array C. LDC >= max(1,M).
```

8.3 Return Value:

```
           = 0: successful exit
```

8.4 C Bindings

```
int PLASMA_cgemm(PLASMA_enum transA, PLASMA_enum transB, int M, int N, int K,  ↩
    PLASMA_Complex32_t alpha, PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t *B, int LDB ↩
    , PLASMA_Complex32_t beta, PLASMA_Complex32_t *C, int LDC)
```

8.5 Fortran Bindings

```
PLASMA_CGEMM(INTEGER transA, INTEGER transB, INTEGER M, INTEGER N, INTEGER K, COMPLEX alpha ↩
    , COMPLEX A, INTEGER LDA, COMPLEX B, INTEGER LDB, COMPLEX beta, COMPLEX C, INTEGER LDC,  ↩
    INTEGER INFO)
```

8.6 Online Browsing

Dive into PLASMA_cgemm

9 PLASMA_cgeqrf

9.1 Purpose

PLASMA_cgeqrf - Computes the tile QR factorization of a complex M-by-N matrix A: A = Q * R.

9.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

A         PLASMA_Complex32_t* (INOUT)
```

```
          On entry, the M-by-N matrix A.
          On exit, the elements on and above the diagonal of the array contain the min(M,N)- ↩
              by-N
          upper trapezoidal matrix R (R is upper triangular if M >= N); the elements below  ↩
              the
          diagonal represent the unitary matrix Q as a product of elementary reflectors  ↩
              stored
          by tiles.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,M).

T         PLASMA_Complex32_t* (OUT)
          On exit, auxiliary factorization data, required by PLASMA_cgeqrs to solve the  ↩
              system
          of equations.
```

9.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

9.4 C Bindings

```
int PLASMA_cgeqrf(int M, int N, PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t *T)
```

9.5 Fortran Bindings

```
PLASMA_CGEQRF(INTEGER M, INTEGER N, COMPLEX A, INTEGER LDA, INTEGER T, INTEGER INFO)
```

9.6 Online Browsing

Dive into PLASMA_cgeqrf

10 PLASMA_cgeqrf_Tile

10.1 Purpose

PLASMA_cgeqrf_Tile - Computes the tile QR factorization of a complex M-by-N matrix A: A = Q * R. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

10.2 Arguments

```
A         PLASMA_Complex32_t* (INOUT)
          On entry, the M-by-N matrix A.
          On exit, the elements on and above the diagonal of the array contain the min(M,N)- ↩
              by-N
          upper trapezoidal matrix R (R is upper triangular if M >= N); the elements below  ↩
              the
```

```
          diagonal represent the unitary matrix Q as a product of elementary reflectors  ↩
              stored
          by tiles.

T         PLASMA_Complex32_t* (OUT)
          On exit, auxiliary factorization data, required by PLASMA_cgeqrs to solve the  ↩
              system
          of equations.
```

10.3 Return Value:

```
           = 0: successful exit
```

10.4 C Bindings

```
int PLASMA_cgeqrf_Tile(PLASMA_desc *A, PLASMA_desc *T)
```

10.5 Fortran Bindings

```
PLASMA_CGEQRF_TILE(INTEGER*4 A, INTEGER*4 T, INTEGER INFO)
```

10.6 Online Browsing

Dive into PLASMA_cgeqrf_Tile

11 PLASMA_cgeqrs

11.1 Purpose

PLASMA_cgeqrs - Compute a minimum-norm solution min || A*X - B || using the RQ factorization A = R*Q computed by PLASMA_cgeqrf.

11.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A. N >= M >= 0.

NRHS      int (IN)
          The number of columns of B. NRHS >= 0.

A         PLASMA_Complex32_t* (INOUT)
          Details of the QR factorization of the original matrix A as returned by  ↩
              PLASMA_cgeqrf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= M.
```

```
T         PLASMA_Complex32_t* (IN)
          Auxiliary factorization data, computed by PLASMA_cgeqrf.

B         PLASMA_Complex32_t* (INOUT)
          On entry, the m-by-nrhs right hand side matrix B.
          On exit, the n-by-nrhs solution matrix X.

LDB       int (IN)
          The leading dimension of the array B. LDB >= max(1,N).
```

11.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

11.4 C Bindings

```
int PLASMA_cgeqrs(int M, int N, int NRHS, PLASMA_Complex32_t *A, int LDA,  ↩
    PLASMA_Complex32_t *T, PLASMA_Complex32_t *B, int LDB)
```

11.5 Fortran Bindings

```
PLASMA_CGEQRS(INTEGER M, INTEGER N, INTEGER NRHS, COMPLEX A, INTEGER LDA, INTEGER T,  ↩
    COMPLEX B, INTEGER LDB, INTEGER INFO)
```

11.6 Online Browsing

Dive into PLASMA_cgeqrs

12 PLASMA_cgeqrs_Tile

12.1 Purpose

PLASMA_cgeqrs_Tile - Compute a minimum-norm solution min || A*X - B || using the RQ factorization A = R*Q computed by PLASMA_cgeqrf_Tile. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

12.2 Arguments

```
A         PLASMA_Complex32_t* (INOUT)
          Details of the QR factorization of the original matrix A as returned by  ↩
              PLASMA_cgeqrf.

T         PLASMA_Complex32_t* (IN)
          Auxiliary factorization data, computed by PLASMA_cgeqrf.

B         PLASMA_Complex32_t* (INOUT)
          On entry, the m-by-nrhs right hand side matrix B.
          On exit, the n-by-nrhs solution matrix X.
```

12.3 Return Value:

```
          = 0: successful exit
```

12.4 C Bindings

```
int PLASMA_cgeqrs_Tile(PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

12.5 Fortran Bindings

```
PLASMA_CGEQRS_TILE(INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

12.6 Online Browsing

Dive into PLASMA_cgeqrs_Tile

13 PLASMA_cgesv

13.1 Purpose

PLASMA_cgesv - Computes the solution to a system of linear equations A **X = B, where A is an N-by-N matrix and X and B are N-by-NRHS matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A. The factored form of A is then used to solve the system of equations A** X = B.

13.2 Arguments

```
N         int (IN)
          The number of linear equations, i.e., the order of the matrix A. N >= 0.

NRHS      int (IN)
          The number of right hand sides, i.e., the number of columns of the matrix B.
          NRHS >= 0.

A         PLASMA_Complex32_t* (INOUT)
          On entry, the N-by-N coefficient matrix A.
          On exit, the tile L and U factors from the factorization (not equivalent to LAPACK ↩
              ).

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,N).

L         PLASMA_Complex32_t* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          necessary to solve the system of equations.

IPIV      int* (OUT)
          On exit, the pivot indices that define the permutations (not equivalent to LAPACK) ↩
              .

B         PLASMA_Complex32_t* (INOUT)
          On entry, the N-by-NRHS matrix of right hand side matrix B.
```

```
          On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB       int (IN)
          The leading dimension of the array B. LDB >= max(1,N).
```

13.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
           > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
                but the factor U is exactly singular, so the solution could not be computed.
```

13.4 C Bindings

```
int PLASMA_cgesv(int N, int NRHS, PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t *L,  ↩
    int *IPIV, PLASMA_Complex32_t *B, int LDB)
```

13.5 Fortran Bindings

```
PLASMA_CGESV(INTEGER N, INTEGER NRHS, COMPLEX A, INTEGER LDA, INTEGER LH, INTEGER IPIVH,  ↩
    COMPLEX B, INTEGER LDB, INTEGER INFO)
```

13.6 Online Browsing

Dive into PLASMA_cgesv

14 PLASMA_cgesv_Tile

14.1 Purpose

PLASMA_cgesv_Tile - Computes the solution to a system of linear equations A **X = B, where A is an N-by-N matrix and X and B are N-by-NRHS matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A. The factored form of A is then used to solve the system of equations A** X = B. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

14.2 Arguments

```
A         PLASMA_Complex32_t* (INOUT)
          On entry, the N-by-N coefficient matrix A.
          On exit, the tile L and U factors from the factorization (not equivalent to LAPACK ↩
              ).

L         PLASMA_Complex32_t* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          necessary to solve the system of equations.

IPIV      int* (OUT)
          On exit, the pivot indices that define the permutations (not equivalent to LAPACK) ↩
              .
```

```
B         PLASMA_Complex32_t* (INOUT)
          On entry, the N-by-NRHS matrix of right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

14.3 Return Value:

```
           = 0: successful exit
           > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
                but the factor U is exactly singular, so the solution could not be computed.
```

14.4 C Bindings

```
int PLASMA_cgesv_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

14.5 Fortran Bindings

```
PLASMA_CGESV_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

14.6 Online Browsing

Dive into PLASMA_cgesv_Tile

15 PLASMA_cgetrf

15.1 Purpose

PLASMA_cgetrf - Computes an LU factorization of a general M-by-N matrix A using the tile LU algorithm with partial tile pivoting with row interchanges.

15.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A. N >= 0.

A         PLASMA_Complex32_t* (INOUT)
          On entry, the M-by-N matrix to be factored.
          On exit, the tile factors L and U from the factorization.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,M).

L         PLASMA_Complex32_t* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          required by PLASMA_cgetrs to solve the system of equations.

IPIV      int* (OUT)
          The pivot indices that define the permutations (not equivalent to LAPACK).
```

15.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
          > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
               but the factor U is exactly singular, and division by zero will occur
               if it is used to solve a system of equations.
```

15.4 C Bindings

```
int PLASMA_cgetrf(int M, int N, PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t *L, int  ↩
    *IPIV)
```

15.5 Fortran Bindings

```
PLASMA_CGETRF(INTEGER M, INTEGER N, COMPLEX A, INTEGER LDA, INTEGER LH, INTEGER IPIVH,  ↩
    INTEGER INFO)
```

15.6 Online Browsing

Dive into PLASMA_cgetrf

16 PLASMA_cgetrf_Tile

16.1 Purpose

PLASMA_cgetrf_Tile - Computes an LU factorization of a general M-by-N matrix A using the tile LU algorithm with partial tile pivoting with row interchanges. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

16.2 Arguments

```
A         PLASMA_Complex32_t* (INOUT)
          On entry, the M-by-N matrix to be factored.
          On exit, the tile factors L and U from the factorization.

L         PLASMA_Complex32_t* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          required by PLASMA_cgetrs to solve the system of equations.

IPIV      int* (OUT)
          The pivot indices that define the permutations (not equivalent to LAPACK).
```

16.3 Return Value:

```
          = 0: successful exit
          > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
               but the factor U is exactly singular, and division by zero will occur
               if it is used to solve a system of equations.
```

16.4 C Bindings

```
int PLASMA_cgetrf_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV)
```

16.5 Fortran Bindings

```
PLASMA_CGETRF_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER INFO)
```

16.6 Online Browsing

Dive into PLASMA_cgetrf_Tile

17 PLASMA_cgetrs

17.1 Purpose

PLASMA_cgetrs - Solves a system of linear equations A * X = B, with a general N-by-N matrix A using the tile LU factorization computed by PLASMA_cgetrf.

17.2 Arguments

```
trans    PLASMA_enum (IN)
         Intended to specify the the form of the system of equations:
         = PlasmaNoTrans:   A * X = B     (No transpose)
         = PlasmaTrans:     A**T * X = B  (Transpose)
         = PlasmaConjTrans: A**H * X = B  (Conjugate transpose)
         Currently only PlasmaNoTrans is supported.

N        int (IN)
         The order of the matrix A.  N >= 0.

NRHS     int (IN)
         The number of right hand sides, i.e., the number of columns of the matrix B.
         NRHS >= 0.

A        PLASMA_Complex32_t* (IN)
         The tile factors L and U from the factorization, computed by PLASMA_cgetrf.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,N).

L        PLASMA_Complex32_t* (IN)
         Auxiliary factorization data, related to the tile L factor, computed by  ↩
             PLASMA_cgetrf.

IPIV     int* (IN)
         The pivot indices from PLASMA_cgetrf (not equivalent to LAPACK).

B        PLASMA_Complex32_t* (INOUT)
         On entry, the N-by-NRHS matrix of right hand side matrix B.
         On exit, the solution matrix X.

LDB      int (IN)
         The leading dimension of the array B. LDB >= max(1,N).
```

17.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
```

17.4 C Bindings

```
int PLASMA_cgetrs(PLASMA_enum uplo, int N, int NRHS, PLASMA_Complex32_t *A, int LDA,  ↩
    PLASMA_Complex32_t *L, int *IPIV, PLASMA_Complex32_t *B, int LDB)
```

17.5 Fortran Bindings

```
PLASMA_CGETRS(INTEGER uplo, INTEGER N, INTEGER NRHS, COMPLEX A, INTEGER LDA, INTEGER LH,  ↩
    INTEGER IPIVH, COMPLEX B, INTEGER LDB, INTEGER INFO)
```

17.6 Online Browsing

Dive into PLASMA_cgetrs

18 PLASMA_cgetrs_Tile

18.1 Purpose

PLASMA_cgetrs_Tile - Solves a system of linear equations A * X = B, with a general N-by-N matrix A using the tile LU factorization computed by PLASMA_cgetrf. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

18.2 Arguments

```
A         PLASMA_Complex32_t* (IN)
          The tile factors L and U from the factorization, computed by PLASMA_cgetrf.

L         PLASMA_Complex32_t* (IN)
          Auxiliary factorization data, related to the tile L factor, computed by  ↩
              PLASMA_cgetrf.

IPIV      int* (IN)
          The pivot indices from PLASMA_cgetrf (not equivalent to LAPACK).

B         PLASMA_Complex32_t* (INOUT)
          On entry, the N-by-NRHS matrix of right hand side matrix B.
          On exit, the solution matrix X.
```

18.3 Return Value:

```
          = 0: successful exit
```

18.4 C Bindings

```
int PLASMA_cgetrs_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

18.5 Fortran Bindings

```
PLASMA_CGETRS_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

18.6 Online Browsing

Dive into PLASMA_cgetrs_Tile

19 PLASMA_Enable

19.1 Purpose

PLASMA_Enable - Enable PLASMA feature.

19.2 Arguments

```
lever    PLASMA_enum (IN)
         Feature to be enabled:
         = PLASMA_WARNINGS:   printing of warning messages,
         = PLASMA_ERRORS:     printing of error messages,
         = PLASMA_AUTOTUNING: autotuning for tile size and inner block size.
```

19.3 Return Value:

```
          = PLASMA_SUCCESS: successful exit
```

19.4 C Bindings

```
int PLASMA_Enable(PLASMA_enum lever)
```

19.5 Fortran Bindings

```
PLASMA_ENABLE(INTEGER lever, INTEGER INFO)
```

19.6 Online Browsing

Dive into PLASMA_Enable

20 PLASMA_Disable

20.1 Purpose

PLASMA_Disable - Disable PLASMA feature.

20.2 Arguments

```
lever    PLASMA_enum (IN)
         Feature to be disabled:
         = PLASMA_WARNINGS:   printing of warning messages,
         = PLASMA_ERRORS:     printing of error messages,
         = PLASMA_AUTOTUNING: autotuning for tile size and inner block size.
```

20.3 Return Value:

```
          = PLASMA_SUCCESS: successful exit
```

20.4 C Bindings

```
int PLASMA_Disable(PLASMA_enum lever)
```

20.5 Fortran Bindings

```
PLASMA_DISABLE(INTEGER lever, INTEGER INFO)
```

20.6 Online Browsing

Dive into PLASMA_Disable

21 PLASMA_Set

21.1 Purpose

PLASMA_Set - Set PLASMA parameter

21.2 Arguments

```
param    PLASMA_enum (IN)
         PLASMA parameter:
         = PLASMA_TILE_SIZE:        size matrix tile,
         = PLASMA_INNER_BLOCK_SIZE: size of tile inner block.

value    int (IN)
         Value of the parameter.
```

21.3 Return Value:

```
          = PLASMA_SUCCESS: successful exit
```

21.4 C Bindings

```
int PLASMA_Set(PLASMA_enum param, int value)
```

21.5 Fortran Bindings

```
PLASMA_SET(INTEGER param, INTEGER value, INTEGER INFO)
```

21.6 Online Browsing

Dive into PLASMA_Set

22 PLASMA_Get

22.1 Purpose

PLASMA_Get - Get value of PLASMA parameter

22.2 Arguments

```
param    PLASMA_enum (IN)
         PLASMA parameter:
         = PLASMA_TILE_SIZE:        size matrix tile,
         = PLASMA_INNER_BLOCK_SIZE: size of tile inner block.

value    int* (OUT)
         Value of the parameter.
```

22.3 Return Value:

```
          = PLASMA_SUCCESS: successful exit
```

22.4 C Bindings

```
int PLASMA_Get(PLASMA_enum param, int *value)
```

22.5 Fortran Bindings

```
PLASMA_GET(INTEGER param, INTEGER value, INTEGER INFO)
```

22.6 Online Browsing

Dive into PLASMA_Get

23 PLASMA_Init

23.1 Purpose

PLASMA_Init - Initialize PLASMA.

23.2 Arguments

```
cores    int (IN)
         Number of cores to use (threads to launch)
```

23.3 Return Value:

```
          = PLASMA_SUCCESS: successful exit
```

23.4 C Bindings

```
int PLASMA_Init(int cores)
```

23.5 Fortran Bindings

```
PLASMA_INIT(INTEGER CORES, INTEGER INFO)
```

23.6 Online Browsing

Dive into PLASMA_Init

24 PLASMA_Finalize

24.1 Purpose

PLASMA_Finalize - Finalize PLASMA.

24.2 C Bindings

```
int PLASMA_Finalize()
```

24.3 Fortran Bindings

```
PLASMA_FINALIZE(INTEGER INFO)
```

24.4 Online Browsing

Dive into PLASMA_Finalize

25 PLASMA_cposv

25.1 Purpose

PLASMA_cposv - Computes the solution to a system of linear equations A * X = B, where A is an N-by-N symmetric positive definite (or Hermitian positive definite in the complex case) matrix and X and B are N-by-NRHS matrices. The Cholesky decomposition is used to factor A as

```
A = U**H * U, if uplo = PlasmaUpper, or
A = L * L**H, if uplo =  PlasmaLower,
```

where U is an upper triangular matrix and L is a lower triangular matrix. The factored form of A is then used to solve the system of equations A * X = B.

25.2 Arguments

```
uplo     PLASMA_enum (IN)
         Specifies whether the matrix A is upper triangular or lower triangular:
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

N        int (IN)
         The number of linear equations, i.e., the order of the matrix A. N >= 0.

NRHS     int (IN)
         The number of right hand sides, i.e., the number of columns of the matrix B. NRHS  ↩
             >= 0.

A        PLASMA_Complex32_t* (INOUT)
         On entry, the symmetric positive definite (or Hermitian) matrix A.
         If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A
         contains the upper triangular part of the matrix A, and the strictly lower  ↩
             triangular
         part of A is not referenced.
         If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower
         triangular part of the matrix A, and the strictly upper triangular part of A is  ↩
             not
         referenced.
         On exit, if return value = 0, the factor U or L from the Cholesky factorization
         A = U**H*U or A = L*L**H.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,N).

B        PLASMA_Complex32_t* (INOUT)
         On entry, the N-by-NRHS right hand side matrix B.
         On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB      int (IN)
         The leading dimension of the array B. LDB >= max(1,N).
```

25.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
          > 0: if i, the leading minor of order i of A is not positive definite, so the
               factorization could not be completed, and the solution has not been computed ↩
                   .
```

25.4 C Bindings

```
int PLASMA_cposv(PLASMA_enum uplo, int N, int NRHS, PLASMA_Complex32_t *A, int LDA,  ↩
    PLASMA_Complex32_t *B, int LDB)
```

25.5 Fortran Bindings

```
PLASMA_CPOSV(INTEGER uplo, INTEGER N, INTEGER NRHS, COMPLEX A, INTEGER LDA, COMPLEX B,  ↩
    INTEGER LDB, INTEGER INFO)
```

25.6 Online Browsing

Dive into PLASMA_cposv

26 PLASMA_cposv_Tile

26.1 Purpose

PLASMA_cposv_Tile - Computes the solution to a system of linear equations A * X = B, where A is an N-by-N symmetric positive definite (or Hermitian positive definite in the complex case) matrix and X and B are N-by-NRHS matrices. The Cholesky decomposition is used to factor A as

```
A = U**H * U, if uplo = PlasmaUpper, or
A = L * L**H, if uplo =  PlasmaLower,
```

where U is an upper triangular matrix and L is a lower triangular matrix. The factored form of A is then used to solve the system of equations A * X = B. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

26.2 Arguments

```
uplo     PLASMA_enum (IN)
         Specifies whether the matrix A is upper triangular or lower triangular:
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

A        PLASMA_Complex32_t* (INOUT)
         On entry, the symmetric positive definite (or Hermitian) matrix A.
         If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A
         contains the upper triangular part of the matrix A, and the strictly lower  ↩
             triangular
         part of A is not referenced.
         If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower
```

```
          triangular part of the matrix A, and the strictly upper triangular part of A is  ↩
              not
          referenced.
          On exit, if return value = 0, the factor U or L from the Cholesky factorization
          A = U**H*U or A = L*L**H.

B         PLASMA_Complex32_t* (INOUT)
          On entry, the N-by-NRHS right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

26.3 Return Value:

```
           = 0: successful exit
           > 0: if i, the leading minor of order i of A is not positive definite, so the
                factorization could not be completed, and the solution has not been computed ↩
                    .
```

26.4 C Bindings

```
int PLASMA_cposv_Tile(PLASMA_enum uplo, PLASMA_desc *A, PLASMA_desc *B)
```

26.5 Fortran Bindings

```
PLASMA_CPOSV_TILE(INTEGER uplo, INTEGER*4 A, INTEGER*4 B, INTEGER INFO)
```

26.6 Online Browsing

Dive into PLASMA_cposv_Tile

27 PLASMA_cpotrf

27.1 Purpose

PLASMA_cpotrf - Computes the Cholesky factorization of a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A. The factorization has the form

```
A = U**H * U, if uplo = PlasmaUpper, or
A = L * L**H, if uplo =  PlasmaLower,
```

where U is an upper triangular matrix and L is a lower triangular matrix.

27.2 Arguments

```
uplo      PLASMA_enum (IN)
          = PlasmaUpper: Upper triangle of A is stored;
          = PlasmaLower: Lower triangle of A is stored.

N         int (IN)
          The order of the matrix A. N >= 0.
```

```
A         PLASMA_Complex32_t* (INOUT)
          On entry, the symmetric positive definite (or Hermitian) matrix A.
          If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A
          contains the upper triangular part of the matrix A, and the strictly lower  ↩
              triangular
          part of A is not referenced.
          If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower
          triangular part of the matrix A, and the strictly upper triangular part of A is  ↩
              not
          referenced.
          On exit, if return value = 0, the factor U or L from the Cholesky factorization
          A = U**H*U or A = L*L**H.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,N).
```

27.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
          > 0: if i, the leading minor of order i of A is not positive definite, so the
               factorization could not be completed, and the solution has not been computed ↩
                   .
```

27.4 C Bindings

```
int PLASMA_cpotrf(PLASMA_enum uplo, int N, PLASMA_Complex32_t *A, int LDA)
```

27.5 Fortran Bindings

```
PLASMA_CPOTRF(INTEGER uplo, INTEGER N, COMPLEX A, INTEGER LDA, INTEGER INFO)
```

27.6 Online Browsing

Dive into PLASMA_cpotrf

28 PLASMA_cpotrf_Tile

28.1 Purpose

PLASMA_cpotrf_Tile - Computes the Cholesky factorization of a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A. The factorization has the form

```
A = U**H * U, if uplo = PlasmaUpper, or
A = L * L**H, if uplo =  PlasmaLower,
```

where U is an upper triangular matrix and L is a lower triangular matrix. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

28.2 Arguments

```
uplo     PLASMA_enum (IN)
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

A        PLASMA_Complex32_t* (INOUT)
         On entry, the symmetric positive definite (or Hermitian) matrix A.
         If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A
         contains the upper triangular part of the matrix A, and the strictly lower  ↩
             triangular
         part of A is not referenced.
         If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower
         triangular part of the matrix A, and the strictly upper triangular part of A is  ↩
             not
         referenced.
         On exit, if return value = 0, the factor U or L from the Cholesky factorization
         A = U**H*U or A = L*L**H.
```

28.3 Return Value:

```
          = 0: successful exit
          > 0: if i, the leading minor of order i of A is not positive definite, so the
               factorization could not be completed, and the solution has not been computed ↩
                   .
```

28.4 C Bindings

```
int PLASMA_cpotrf_Tile(PLASMA_enum uplo, PLASMA_desc *A)
```

28.5 Fortran Bindings

```
PLASMA_CPOTRF_TILE(INTEGER uplo, INTEGER*4 A, INTEGER INFO)
```

28.6 Online Browsing

Dive into PLASMA_cpotrf_Tile

29 PLASMA_cpotrs

29.1 Purpose

PLASMA_cpotrs - Solves a system of linear equations A * X = B with a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A using the Cholesky factorization A = U**H*U or A = L*L**H computed by PLASMA_cpotrf.

29.2 Arguments

```
uplo     PLASMA_enum (IN)
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

N        int (IN)
         The order of the matrix A. N >= 0.

NRHS     int (IN)
         The number of right hand sides, i.e., the number of columns of the matrix B. NRHS  ↩
             >= 0.

A        PLASMA_Complex32_t* (IN)
         The triangular factor U or L from the Cholesky factorization A = U**H*U or A = L*L ↩
             **H,
         computed by PLASMA_cpotrf.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,N).

B        PLASMA_Complex32_t* (INOUT)
         On entry, the N-by-NRHS right hand side matrix B.
         On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB      int (IN)
         The leading dimension of the array B. LDB >= max(1,N).
```

29.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
```

29.4 C Bindings

```
int PLASMA_cpotrs(PLASMA_enum uplo, int N, int NRHS, PLASMA_Complex32_t *A, int LDA,  ↩
    PLASMA_Complex32_t *B, int LDB)
```

29.5 Fortran Bindings

```
PLASMA_CPOTRS(INTEGER uplo, INTEGER N, INTEGER NRHS, COMPLEX A, INTEGER LDA, COMPLEX B,  ↩
    INTEGER  LDB, INTEGER  INFO)
```

29.6 Online Browsing

Dive into PLASMA_cpotrs

30 PLASMA_cpotrs_Tile

30.1 Purpose

PLASMA_cpotrs_Tile - Solves a system of linear equations A * X = B with a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A using the Cholesky factorization A = U**H*U or A = L*L**H computed by PLASMA_cpotrf.

All matrices are passed through descriptors. All dimensions are taken from the descriptors.

30.2 Arguments

```
uplo     PLASMA_enum (IN)
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

A        PLASMA_Complex32_t* (IN)
         The triangular factor U or L from the Cholesky factorization A = U**H*U or A = L*L ↩
             **H,
         computed by PLASMA_cpotrf.

B        PLASMA_Complex32_t* (INOUT)
         On entry, the N-by-NRHS right hand side matrix B.
         On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

30.3 Return Value:

```
          = 0: successful exit
```

30.4 C Bindings

```
int PLASMA_cpotrs_Tile(PLASMA_enum uplo, PLASMA_desc *A, PLASMA_desc *B)
```

30.5 Fortran Bindings

```
PLASMA_CPOTRS_TILE(INTEGER uplo, INTEGER*4 A, INTEGER*4 B, INTEGER INFO)
```

30.6 Online Browsing

Dive into PLASMA_cpotrs_Tile

31 PLASMA_ctrsm

31.1 Purpose

PLASMA_ctrsm - Computes triangular solve A*X = B or X*A = B

31.2 Arguments

```
side     PLASMA_enum (IN)
         Specifies whether A appears on the left or on the right of X:
         = PlasmaLeft:  A*X = B
         = PlasmaRight: X*A = B

uplo     PLASMA_enum (IN)
         Specifies whether the matrix A is upper triangular or lower triangular:
         = PlasmaUpper: Upper triangle of A is stored;
```

```
          = PlasmaLower: Lower triangle of A is stored.

transA    PLASMA_enum (IN)
          Specifies whether the matrix A is transposed, not transposed or conjugate  ↩
              transposed:
          = PlasmaNoTrans:   A is transposed;
          = PlasmaTrans:     A is not transposed;
          = PlasmaConjTrans: A is conjugate transposed.

diag      PLASMA_enum (IN)
          Specifies whether or not A is unit triangular:
          = PlasmaNonUnit: A is non unit;
          = PlasmaUnit:    A us unit.

N         int (IN)
          The order of the matrix A. N >= 0.

NRHS      int (IN)
          The number of right hand sides, i.e., the number of columns of the matrix B. NRHS  ↩
              >= 0.

A         PLASMA_Complex32_t* (IN)
          The triangular matrix A. If uplo = PlasmaUpper, the leading N-by-N upper  ↩
              triangular
          part of the array A contains the upper triangular matrix, and the strictly lower
          triangular part of A is not referenced. If uplo = PlasmaLower, the leading N-by-N
          lower triangular part of the array A contains the lower triangular matrix, and the
          strictly upper triangular part of A is not referenced. If diag = PlasmaUnit, the
          diagonal elements of A are also not referenced and are assumed to be 1.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,N).

B         PLASMA_Complex32_t* (INOUT)
          On entry, the N-by-NRHS right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB       PLASMA_Complex32_t* (IN)
          The leading dimension of the array B. LDB >= max(1,N).
```

31.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

31.4 C Bindings

```
int PLASMA_ctrsm(PLASMA_enum side, PLASMA_enum uplo, PLASMA_enum transA, PLASMA_enum diag,  ↩
    int N, int NRHS, PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t *B, int LDB)
```

31.5 Fortran Bindings

```
PLASMA_CTRSM(INTEGER side, INTEGER uplo, INTEGER transA, INTEGER diag, INTEGER N, INTEGER  ↩
    NRHS, COMPLEX A, INTEGER LDA, COMPLEX B, INTEGER LDB, INTEGER INFO)
```

31.6 Online Browsing

Dive into PLASMA_ctrsm

32 PLASMA_ctrsmpl

32.1 Purpose

PLASMA_ctrsmpl - Performs the forward substitution step of solving a system of linear equations after the tile LU factorization of the matrix.

32.2 Arguments

```
N         int (IN)
          The order of the matrix A. N >= 0.

NRHS      int (IN)
          The number of right hand sides, i.e., the number of columns of the matrix B. NRHS  ↩
              >= 0.

A         PLASMA_Complex32_t* (IN)
          The tile factor L from the factorization, computed by PLASMA_cgetrf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,N).

L         PLASMA_Complex32_t* (IN)
          Auxiliary factorization data, related to the tile L factor, computed by  ↩
              PLASMA_cgetrf.

IPIV      int* (IN)
          The pivot indices from PLASMA_cgetrf (not equivalent to LAPACK).

B         PLASMA_Complex32_t* (INOUT)
          On entry, the N-by-NRHS right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB       PLASMA_Complex32_t* (IN)
          The leading dimension of the array B. LDB >= max(1,N).
```

32.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

32.4 C Bindings

```
int PLASMA_ctrsmpl(int N, int NRHS, PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t *L,  ↩
    int *IPIV, PLASMA_Complex32_t *B, int LDB)
```

32.5 Fortran Bindings

```
PLASMA_CTRSMPL(INTEGER N, INTEGER NRHS, COMPLEX A, INTEGER LDA, INTEGER LH, INTEGER IPIVH,  ↩
    COMPLEX B, INTEGER LDB, INTEGER INFO)
```

32.6 Online Browsing

Dive into PLASMA_ctrsmpl

33 PLASMA_ctrsmpl_Tile

33.1 Purpose

PLASMA_ctrsmpl_Tile - Performs the forward substitution step of solving a system of linear equations after the tile LU factorization of the matrix. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

33.2 Arguments

```
A         PLASMA_Complex32_t* (IN)
          The tile factor L from the factorization, computed by PLASMA_cgetrf.

L         PLASMA_Complex32_t* (IN)
          Auxiliary factorization data, related to the tile L factor, computed by  ↩
              PLASMA_cgetrf.

IPIV      int* (IN)
          The pivot indices from PLASMA_cgetrf (not equivalent to LAPACK).

B         PLASMA_Complex32_t* (INOUT)
          On entry, the N-by-NRHS right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

33.3 Return Value:

```
          = 0: successful exit
```

33.4 C Bindings

```
int PLASMA_ctrsmpl_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

33.5 Fortran Bindings

```
PLASMA_CTRSMPL_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

33.6 Online Browsing

Dive into PLASMA_ctrsmpl_Tile

34 PLASMA_ctrsm_Tile

34.1 Purpose

PLASMA_ctrsm_Tile - Computes triangular solve A*X = B or X*A = B All matrices are passed through descriptors. All dimensions are taken from the descriptors.

34.2 Arguments

```
side     PLASMA_enum (IN)
         Specifies whether A appears on the left or on the right of X:
         = PlasmaLeft:  A*X = B
         = PlasmaRight: X*A = B

uplo     PLASMA_enum (IN)
         Specifies whether the matrix A is upper triangular or lower triangular:
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

transA   PLASMA_enum (IN)
         Specifies whether the matrix A is transposed, not transposed or conjugate  ↩
             transposed:
         = PlasmaNoTrans:   A is transposed;
         = PlasmaTrans:     A is not transposed;
         = PlasmaConjTrans: A is conjugate transposed.

diag     PLASMA_enum (IN)
         Specifies whether or not A is unit triangular:
         = PlasmaNonUnit: A is non unit;
         = PlasmaUnit:    A us unit.

A        PLASMA_Complex32_t* (IN)
         The triangular matrix A. If uplo = PlasmaUpper, the leading N-by-N upper  ↩
             triangular
         part of the array A contains the upper triangular matrix, and the strictly lower
         triangular part of A is not referenced. If uplo = PlasmaLower, the leading N-by-N
         lower triangular part of the array A contains the lower triangular matrix, and the
         strictly upper triangular part of A is not referenced. If diag = PlasmaUnit, the
         diagonal elements of A are also not referenced and are assumed to be 1.

B        PLASMA_Complex32_t* (INOUT)
         On entry, the N-by-NRHS right hand side matrix B.
         On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

34.3 Return Value:

```
          = 0: successful exit
```

34.4 C Bindings

```
int PLASMA_ctrsm_Tile(PLASMA_enum side, PLASMA_enum uplo, PLASMA_enum transA, PLASMA_enum  ↩
    diag, PLASMA_desc *A, PLASMA_desc *B)
```

34.5 Fortran Bindings

```
PLASMA_CTRSM_TILE(INTEGER side, INTEGER uplo, INTEGER transA, INTEGER diag, INTEGER*4 A,  ↩
    INTEGER*4 B, INTEGER INFO)
```

34.6 Online Browsing

Dive into PLASMA_ctrsm_Tile

35 PLASMA_cunglq

35.1 Purpose

PLASMA_cunglq - Generates an M-by-N matrix Q with orthonormal rows, which is defined as the first M rows of a product of the elementary reflectors returned by PLASMA_cgelqf.

35.2 Arguments

```
M         int (IN)
          The number of rows of the matrix Q. M >= 0.

N         int (IN)
          The number of columns of the matrix Q. N >= M.

K         int (IN)
          The number of rows of elementary tile reflectors whose product defines the matrix  ↩
              Q.
          M >= K >= 0.

A         PLASMA_Complex32_t* (IN)
          Details of the LQ factorization of the original matrix A as returned by  ↩
              PLASMA_cgelqf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,M).

T         PLASMA_Complex32_t* (IN)
          Auxiliary factorization data, computed by PLASMA_cgelqf.

B         PLASMA_Complex32_t* (OUT)
          On exit, the M-by-N matrix Q.

LDA       int (IN)
          The leading dimension of the array B. LDB >= max(1,M).
```

35.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

35.4 C Bindings

```
int PLASMA_cunglq(int M, int N, int K, PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t * ↩
    T, PLASMA_Complex32_t *B, int LDB)
```

35.5 Fortran Bindings

```
PLASMA_CUNGLQ(INTEGER M, INTEGER N, INTEGER K, COMPLEX A, INTEGER LDA, INTEGER T, COMPLEX B ↩
    , INTEGER LDB, INTEGER INFO)
```

35.6 Online Browsing

Dive into PLASMA_cunglq

36 PLASMA_cunglq_Tile

36.1 Purpose

PLASMA_cunglq_Tile - Generates an M-by-N matrix Q with orthonormal rows, which is defined as the first M rows of a product of the elementary reflectors returned by PLASMA_cgelqf_Tile. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

36.2 Arguments

```
A         PLASMA_Complex32_t* (IN)
          Details of the LQ factorization of the original matrix A as returned by  ↩
              PLASMA_cgelqf.

T         PLASMA_Complex32_t* (IN)
          Auxiliary factorization data, computed by PLASMA_cgelqf.

B         PLASMA_Complex32_t* (OUT)
          On exit, the M-by-N matrix Q.
```

36.3 Return Value:

```
           = 0: successful exit
```

36.4 C Bindings

```
int PLASMA_cunglq_Tile(PLASMA_desc *A, PLASMA_desc *T, PLASMA_desc *B)
```

36.5 Fortran Bindings

36.6 Online Browsing

Dive into PLASMA_cunglq_Tile

37 PLASMA_cungqr

37.1 Purpose

PLASMA_cungqr - Generates an M-by-N matrix Q with orthonormal columns, which is defined as the first N columns of a product of the elementary reflectors returned by PLASMA_cgeqrf.

37.2 Arguments

```
M         int (IN)
          The number of rows of the matrix Q. M >= 0.

N         int (IN)
          The number of columns of the matrix Q. N >= M.

K         int (IN)
          The number of columns of elementary tile reflectors whose product defines the  ↩
              matrix Q.
          M >= K >= 0.

A         PLASMA_Complex32_t* (IN)
          Details of the QR factorization of the original matrix A as returned by  ↩
              PLASMA_cgeqrf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,M).

T         PLASMA_Complex32_t* (IN)
          Auxiliary factorization data, computed by PLASMA_cgeqrf.

B         PLASMA_Complex32_t* (OUT)
          On exit, the M-by-N matrix Q.

LDB       int (IN)
          The leading dimension of the array B. LDB >= max(1,M).
```

37.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

37.4 C Bindings

```
int PLASMA_cungqr(int M, int N, int K, PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t * ↩
    T, PLASMA_Complex32_t *B, int LDB)
```

37.5 Fortran Bindings

```
PLASMA_CUNGQR(INTEGER M, INTEGER N, INTEGER K, COMPLEX A, INTEGER LDA, INTEGER T, COMPLEX B ↩
    , INTEGER LDB, INTEGER INFO)
```

37.6 Online Browsing

Dive into PLASMA_cungqr

38 PLASMA_cungqr_Tile

38.1 Purpose

PLASMA_cungqr_Tile - Generates an M-by-N matrix Q with orthonormal columns, which is defined as the first N columns of a product of the elementary reflectors returned by PLASMA_cgeqrf_Tile. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

38.2 Arguments

```
A         PLASMA_Complex32_t* (IN)
          Details of the QR factorization of the original matrix A as returned by  ↩
              PLASMA_cgeqrf.

T         PLASMA_Complex32_t* (IN)
          Auxiliary factorization data, computed by PLASMA_cgeqrf.

B         PLASMA_Complex32_t* (OUT)
          On exit, the M-by-N matrix Q.
```

38.3 Return Value:

```
           = 0: successful exit
```

38.4 C Bindings

```
int PLASMA_cungqr_Tile(PLASMA_desc *A, PLASMA_desc *T, PLASMA_desc *B)
```

38.5 Fortran Bindings

```
PLASMA_CUNGQR_TILE(INTEGER*4 A, INTEGER*4 T, INTEGER*4 B, INTEGER INFO)
```

38.6 Online Browsing

Dive into PLASMA_cungqr_Tile

39 PLASMA_cunmlq

39.1 Purpose

PLASMA_cunmlq - overwrites the general M-by-N matrix C with Q*C, where Q is an orthogonal matrix (unitary in the complex case) defined as the product of elementary reflectors returned by PLASMA_cgelqf. Q is of order M.

39.2 Arguments

```
side     PLASMA_enum (IN)
         Intended usage:
         = PlasmaLeft:  apply Q or Q**H from the left;
         = PlasmaRight: apply Q or Q**H from the right.
         Currently only PlasmaLeft is supported.

trans    PLASMA_enum (IN)
         Intended usage:
         = PlasmaNoTrans:   no transpose, apply Q;
         = PlasmaConjTrans: conjugate transpose, apply Q**H.
         Currently only PlasmaConjTrans is supported.

M        int (IN)
         The number of rows of the matrix C. M >= 0.

N        int (IN)
         The number of columns of the matrix C. N >= 0.

K        int (IN)
         The number of rows of elementary tile reflectors whose product defines the matrix ↩
             Q.
         M >= K >= 0.

A        PLASMA_Complex32_t* (IN)
         Details of the LQ factorization of the original matrix A as returned by ↩
             PLASMA_cgelqf.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,K).

T        PLASMA_Complex32_t* (IN)
         Auxiliary factorization data, computed by PLASMA_cgelqf.

B        PLASMA_Complex32_t* (INOUT)
         On entry, the M-by-N matrix B.
         On exit, B is overwritten by Q*B or Q**H*B.

LDB      int (IN)
         The leading dimension of the array C. LDC >= max(1,M).
```

39.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
```

39.4 C Bindings

```
int PLASMA_cunmlq(PLASMA_enum side, PLASMA_enum trans, int M, int N, int K,  ↩
    PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t *T, PLASMA_Complex32_t *B, int LDB)
```

39.5 Fortran Bindings

```
PLASMA_CUNMLQ(INTEGER side, INTEGER trans, INTEGER M, INTEGER N, INTEGER K, COMPLEX A,  ↩
    INTEGER LDA, INTEGER T, COMPLEX B, INTEGER LDB, INTEGER INFO)
```

39.6 Online Browsing

Dive into PLASMA_cunmlq

40 PLASMA_cunmlq_Tile

40.1 Purpose

PLASMA_cunmlq_Tile - overwrites the general M-by-N matrix C with Q*C, where Q is an orthogonal matrix (unitary in the complex case) defined as the product of elementary reflectors returned by PLASMA_cgelqf_Tile Q is of order M. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

40.2 Arguments

```
side     PLASMA_enum (IN)
         Intended usage:
         = PlasmaLeft:  apply Q or Q**H from the left;
         = PlasmaRight: apply Q or Q**H from the right.
         Currently only PlasmaLeft is supported.

trans    PLASMA_enum (IN)
         Intended usage:
         = PlasmaNoTrans:   no transpose, apply Q;
         = PlasmaConjTrans: conjugate transpose, apply Q**H.
         Currently only PlasmaConjTrans is supported.

A        PLASMA_Complex32_t* (IN)
         Details of the LQ factorization of the original matrix A as returned by  ↩
             PLASMA_cgelqf.

T        PLASMA_Complex32_t* (IN)
         Auxiliary factorization data, computed by PLASMA_cgelqf.

B        PLASMA_Complex32_t* (INOUT)
         On entry, the M-by-N matrix B.
         On exit, B is overwritten by Q*B or Q**H*B.
```

40.3 Return Value:

```
          = 0: successful exit
```

40.4 C Bindings

```
int PLASMA_cunmlq_Tile(PLASMA_enum side, PLASMA_enum trans, PLASMA_desc *A, PLASMA_desc *T, ↩
    PLASMA_desc *B)
```

40.5 Fortran Bindings

```
PLASMA_CUNMLQ_TILE(INTEGER side, INTEGER trans, INTEGER*4 A, INTEGER*4 T, INTEGER*4 B,  ↩
    INTEGER INFO)
```

40.6 Online Browsing

Dive into PLASMA_cunmlq_Tile

41 PLASMA_cunmqr

41.1 Purpose

PLASMA_cunmqr - overwrites the general M-by-N matrix C with Q*C, where Q is an orthogonal matrix (unitary in the complex case) defined as the product of elementary reflectors returned by PLASMA_cgeqrf. Q is of order M.

41.2 Arguments

```
side     PLASMA_enum (IN)
         Intended usage:
         = PlasmaLeft:  apply Q or Q**H from the left;
         = PlasmaRight: apply Q or Q**H from the right.
         Currently only PlasmaLeft is supported.

trans    PLASMA_enum (IN)
         Intended usage:
         = PlasmaNoTrans:   no transpose, apply Q;
         = PlasmaConjTrans: conjugate transpose, apply Q**H.
         Currently only PlasmaConjTrans is supported.

M        int (IN)
         The number of rows of the matrix C. M >= 0.

N        int (IN)
         The number of columns of the matrix C. N >= 0.

K        int (IN)
         The number of columns of elementary tile reflectors whose product defines the  ↩
             matrix Q.
         M >= K >= 0.

A        PLASMA_Complex32_t* (IN)
         Details of the QR factorization of the original matrix A as returned by  ↩
             PLASMA_cgeqrf.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,M);

T        PLASMA_Complex32_t* (IN)
```

```
          Auxiliary factorization data, computed by PLASMA_cgeqrf.

B         PLASMA_Complex32_t* (INOUT)
          On entry, the M-by-N matrix B.
          On exit, B is overwritten by Q*B or Q**H*B.

LDB       int (IN)
          The leading dimension of the array C. LDC >= max(1,M).
```

41.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

41.4 C Bindings

```
int PLASMA_cunmqr(PLASMA_enum side, PLASMA_enum trans, int M, int N, int K,  ↩
    PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t *T, PLASMA_Complex32_t *B, int LDB)
```

41.5 Fortran Bindings

```
PLASMA_CUNMQR(INTEGER side, INTEGER trans, INTEGER M, INTEGER N, INTEGER K, COMPLEX A,  ↩
    INTEGER LDA, INTEGER T, COMPLEX B, INTEGER LDB, INTEGER INFO)
```

41.6 Online Browsing

Dive into PLASMA_cunmqr

42 PLASMA_cunmqr_Tile

42.1 Purpose

PLASMA_cunmqr_Tile - overwrites the general M-by-N matrix C with Q*C, where Q is an orthogonal matrix (unitary in the complex case) defined as the product of elementary reflectors returned by PLASMA_cgeqrf_Tile Q is of order M. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

42.2 Arguments

```
side      PLASMA_enum (IN)
          Intended usage:
          = PlasmaLeft:  apply Q or Q**H from the left;
          = PlasmaRight: apply Q or Q**H from the right.
          Currently only PlasmaLeft is supported.

trans     PLASMA_enum (IN)
          Intended usage:
          = PlasmaNoTrans:   no transpose, apply Q;
          = PlasmaConjTrans: conjugate transpose, apply Q**H.
          Currently only PlasmaConjTrans is supported.
```

```
A         PLASMA_Complex32_t* (IN)
          Details of the QR factorization of the original matrix A as returned by  ↩
              PLASMA_cgeqrf.

T         PLASMA_Complex32_t* (IN)
          Auxiliary factorization data, computed by PLASMA_cgeqrf.

B         PLASMA_Complex32_t* (INOUT)
          On entry, the M-by-N matrix B.
          On exit, B is overwritten by Q*B or Q**H*B.
```

42.3 Return Value:

```
           = 0: successful exit
```

42.4 C Bindings

```
int PLASMA_cunmqr_Tile(PLASMA_enum side, PLASMA_enum trans, PLASMA_desc *A, PLASMA_desc *T, ↩
     PLASMA_desc *B)
```

42.5 Fortran Bindings

```
PLASMA_CUNMQR_TILE(INTEGER side, INTEGER trans, INTEGER*4 A, INTEGER*4 T, INTEGER*4 B,  ↩
    INTEGER INFO)
```

42.6 Online Browsing

Dive into PLASMA_cunmqr_Tile

43 PLASMA_Desc_Create

43.1 Purpose

PLASMA_Desc_Create - Create matrix descriptor.

43.2 Arguments

```
desc      PLASMA_desc** (OUT)
          On exit, descriptor of the matrix.

mat       void* (IN)
          Memory location of the matrix.

dtyp      PLASMA_enum (IN)
          Data type of the matrix:
          = PlasmaRealFloat:     single precision real (S),
          = PlasmaRealDouble:    double precision real (D),
          = PlasmaComplexFloat:  single precision complex (C),
          = PlasmaComplexDouble: double precision complex (Z).
```

```
mb        int (IN)
          Number of rows in a tile.

nb        int (IN)
          Number of columns in a tile.

bsiz      int (IN)
          Nize in elements including padding.

lm        int (IN)
          Number of rows of the entire matrix.

ln        int (IN)
          Number of columns of the entire matrix.

i         int (IN)
          Row index to the beginning of the submatrix.

j         int (IN)
          Column indes to the beginning of the submatrix.

m         int (IN)
          Number of rows of the submatrix.

n         int (IN)
          Number of columns of the submatrix.
```

43.3 Return Value:

```
          = PLASMA_SUCCESS: successful exit
```

43.4 C Bindings

```
int PLASMA_Desc_Create(PLASMA_desc **desc, void *mat, PLASMA_enum dtyp, int mb, int nb, int ↩
     bsiz, int lm, int ln, int i, int j, int m, int n)
```

43.5 Fortran Bindings

```
PLASMA_DESC_CREATE(INTEGER*4 desc, INTEGER mat, INTEGER dtyp, INTEGER mb, INTEGER nb,  ↩
    INTEGER bsiz, INTEGER lm, INTEGER ln, INTEGER i, INTEGER j, INTEGER m, INTEGER n,  ↩
    INTEGER INFO)
```

43.6 Online Browsing

Dive into PLASMA_Desc_Create

44 PLASMA_Desc_Destroy

44.1 Purpose

PLASMA_Desc_Destroy - Destroys matrix descriptor.

44.2 Arguments

```
desc     PLASMA_desc** (IN)
         Matrix descriptor.
```

44.3 Return Value:

```
          = PLASMA_SUCCESS: successful exit
```

44.4 C Bindings

```
int PLASMA_Desc_Destroy(PLASMA_desc **desc)
```

44.5 Fortran Bindings

```
PLASMA_DESC_DESTROY(INTEGER*4 desc, INTEGER INFO)
```

44.6 Online Browsing

Dive into PLASMA_Desc_Destroy

45 PLASMA_dgelqf

45.1 Purpose

PLASMA_dgelqf - Computes the tile LQ factorization of a complex M-by-N matrix A: A = L * Q.

45.2 Arguments

```
M        int (IN)
         The number of rows of the matrix A. M >= 0.

N        int (IN)
         The number of columns of the matrix A. N >= 0.

A        double* (INOUT)
         On entry, the M-by-N matrix A.
         On exit, the elements on and below the diagonal of the array contain the m-by-min( ↩
             M,N)
         lower trapezoidal matrix L (L is lower triangular if M <= N); the elements above  ↩
             the
         diagonal represent the unitary matrix Q as a product of elementary reflectors,  ↩
             stored
         by tiles.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,M).

T        double* (OUT)
         On exit, auxiliary factorization data, required by PLASMA_dgelqs to solve the  ↩
             system
         of equations.
```

45.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
```

45.4 C Bindings

```
int PLASMA_dgelqf(int M, int N, double *A, int LDA, double *T)
```

45.5 Fortran Bindings

```
PLASMA_DGELQF(INTEGER M, INTEGER N, DOUBLE PRECISION A, INTEGER LDA, INTEGER T, INTEGER  ↩
    INFO)
```

45.6 Online Browsing

Dive into PLASMA_dgelqf

46 PLASMA_dgelqf_Tile

46.1 Purpose

PLASMA_dgelqf_Tile - Computes the tile LQ factorization of a complex M-by-N matrix A: A = L * Q. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

46.2 Arguments

```
A         double* (INOUT)
          On entry, the M-by-N matrix A.
          On exit, the elements on and below the diagonal of the array contain the m-by-min( ↩
              M,N)
          lower trapezoidal matrix L (L is lower triangular if M <= N); the elements above  ↩
              the
          diagonal represent the unitary matrix Q as a product of elementary reflectors,  ↩
              stored
          by tiles.

T         double* (OUT)
          On exit, auxiliary factorization data, required by PLASMA_dgelqs to solve the  ↩
              system
          of equations.
```

46.3 Return Value:

```
          = 0: successful exit
```

46.4 C Bindings

```
int PLASMA_dgelqf_Tile(PLASMA_desc *A, PLASMA_desc *T)
```

46.5 Fortran Bindings

```
PLASMA_DGELQF_TILE(INTEGER*4 A, INTEGER*4 T,INTEGER INFO)
```

46.6 Online Browsing

Dive into PLASMA_dgelqf_Tile

47 PLASMA_dgelqs

47.1 Purpose

PLASMA_dgelqs - Compute a minimum-norm solution min || A*X - B || using the LQ factorization A = L*Q computed by PLASMA_dgelqf.

47.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A. N >= M >= 0.

NRHS      int (IN)
          The number of columns of B. NRHS >= 0.

A         double* (IN)
          Details of the LQ factorization of the original matrix A as returned by  ↩
              PLASMA_dgelqf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= M.

T         double* (IN)
          Auxiliary factorization data, computed by PLASMA_dgelqf.

B         double* (INOUT)
          On entry, the M-by-NRHS right hand side matrix B.
          On exit, the N-by-NRHS solution matrix X.

LDB       int (IN)
          The leading dimension of the array B. LDB >= N.
```

47.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

47.4 C Bindings

```
int PLASMA_dgelqs(int M, int N, int NRHS, double *A, int LDA, double *T, double *B, int LDB ↩
    )
```

47.5 Fortran Bindings

```
PLASMA_DGELQS(INTEGER M, INTEGER N, INTEGER NRHS, DOUBLE PRECISION A, INTEGER LDA, INTEGER  ↩
    T, DOUBLE PRECISION B, INTEGER LDB, INTEGER INFO)
```

47.6 Online Browsing

Dive into PLASMA_dgelqs

48 PLASMA_dgelqs_Tile

48.1 Purpose

PLASMA_dgelqs_Tile - Compute a minimum-norm solution min || A*X - B || using the LQ factorization A = L*Q computed by PLASMA_dgelqf_Tile. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

48.2 Arguments

```
A         double* (IN)
          Details of the LQ factorization of the original matrix A as returned by  ↩
              PLASMA_dgelqf.

T         double* (IN)
          Auxiliary factorization data, computed by PLASMA_dgelqf.

B         double* (INOUT)
          On entry, the M-by-NRHS right hand side matrix B.
          On exit, the N-by-NRHS solution matrix X.
```

48.3 Return Value:

```
           = 0: successful exit
```

48.4 C Bindings

```
int PLASMA_dgelqs_Tile(PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

48.5 Fortran Bindings

```
PLASMA_DGELQS_TILE(INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

48.6 Online Browsing

Dive into PLASMA_dgelqs_Tile

49 PLASMA_dgels

49.1 Purpose

PLASMA_dgels - solves overdetermined or underdetermined linear systems involving an M-by-N matrix A using the QR or the LQ factorization of A. It is assumed that A has full rank. The following options are provided:

1. trans = PlasmaNoTrans and M >= N: find the least squares solution of an overdetermined system, i.e., solve the least squares problem: minimize || B - A*X ||.
2. trans = PlasmaNoTrans and M < N: find the minimum norm solution of an underdetermined system A * X = B.

Several right hand side vectors B and solution vectors X can be handled in a single call; they are stored as the columns of the M-by-NRHS right hand side matrix B and the N-by-NRHS solution matrix X.

49.2 Arguments

```
trans    PLASMA_enum (IN)
         Intended usage:
         = PlasmaNoTrans:   the linear system involves A;
         = PlasmaTrans: the linear system involves A**T.
         Currently only PlasmaNoTrans is supported.

M        int (IN)
         The number of rows of the matrix A. M >= 0.

N        int (IN)
         The number of columns of the matrix A. N >= 0.

NRHS     int (IN)
         The number of right hand sides, i.e., the number of columns of the matrices B and  ↩
             X.
         NRHS >= 0.

A        double* (INOUT)
         On entry, the M-by-N matrix A.
         On exit,
         if M >= N, A is overwritten by details of its QR factorization as returned by
                    PLASMA_dgeqrf;
         if M < N, A is overwritten by details of its LQ factorization as returned by
                     PLASMA_dgelqf.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,M).

T        double* (OUT)
         On exit, auxiliary factorization data.

B        double* (INOUT)
         On entry, the M-by-NRHS matrix B of right hand side vectors, stored columnwise;
         On exit, if return value = 0, B is overwritten by the solution vectors, stored
         columnwise:
         if M >= N, rows 1 to N of B contain the least squares solution vectors; the  ↩
             residual
```

```
          sum of squares for the solution in each column is given by the sum of squares of  ↩
              the
          modulus of elements N+1 to M in that column;
          if M < N, rows 1 to N of B contain the minimum norm solution vectors;

LDB       int (IN)
          The leading dimension of the array B. LDB >= MAX(1,M,N).
```

49.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

49.4 C Bindings

```
int PLASMA_dgels(PLASMA_enum trans, int M, int N, int NRHS, double *A, int LDA, double *T,  ↩
    double *B, int LDB)
```

49.5 Fortran Bindings

```
PLASMA_DGELS(INTEGER trans, INTEGER M, INTEGER N, INTEGER NRHS, DOUBLE PRECISION A, INTEGER ↩
     LDA, INTEGER T, DOUBLE PRECISION B, INTEGER LDB, INTEGER INFO)
```

49.6 Online Browsing

Dive into PLASMA_dgels

50 PLASMA_dgels_Tile

50.1 Purpose

PLASMA_dgels_Tile - solves overdetermined or underdetermined linear systems involving an M-by-N matrix A using the QR or the LQ factorization of A. It is assumed that A has full rank. The following options are provided:

1. trans = PlasmaNoTrans and M >= N: find the least squares solution of an overdetermined system, i.e., solve the least squares problem: minimize ‖ B - A*X ‖.
2. trans = PlasmaNoTrans and M < N: find the minimum norm solution of an underdetermined system A * X = B.

Several right hand side vectors B and solution vectors X can be handled in a single call; they are stored as the columns of the M-by-NRHS right hand side matrix B and the N-by-NRHS solution matrix X. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

50.2 Arguments

```
trans    PLASMA_enum (IN)
         Intended usage:
         = PlasmaNoTrans:   the linear system involves A;
         = PlasmaTrans: the linear system involves A**T.
         Currently only PlasmaNoTrans is supported.

A        double* (INOUT)
         On entry, the M-by-N matrix A.
         On exit,
         if M >= N, A is overwritten by details of its QR factorization as returned by
                    PLASMA_dgeqrf;
         if M < N, A is overwritten by details of its LQ factorization as returned by
                     PLASMA_dgelqf.

T        double* (OUT)
         On exit, auxiliary factorization data.

B        double* (INOUT)
         On entry, the M-by-NRHS matrix B of right hand side vectors, stored columnwise;
         On exit, if return value = 0, B is overwritten by the solution vectors, stored
         columnwise:
         if M >= N, rows 1 to N of B contain the least squares solution vectors; the  ↩
             residual
         sum of squares for the solution in each column is given by the sum of squares of  ↩
             the
         modulus of elements N+1 to M in that column;
         if M < N, rows 1 to N of B contain the minimum norm solution vectors;
```

50.3 Return Value:

```
          = 0: successful exit
```

50.4 C Bindings

```
int PLASMA_dgels_Tile(PLASMA_enum trans, PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

50.5 Fortran Bindings

```
PLASMA_DGELS_TILE(INTEGER trans, INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

50.6 Online Browsing

Dive into PLASMA_dgels_Tile

51 PLASMA_dgemm

51.1 Purpose

PLASMA_dgemm - Performs one of the matrix-matrix operations

```
C = alpha*op( A )*op( B ) + beta*C,
```

where op(X) is one of

```
op( X ) = X   or op( X ) = X'
```

alpha and beta are scalars, and A, B and C are matrices, with op(A) an m by k matrix, op(B) a k by n matrix and C an m by n matrix.

51.2 Arguments

```
transA   PLASMA_enum (IN)
         Specifies whether the matrix A is transposed, not transposed or conjugate  ↩
             transposed:
         = PlasmaNoTrans:   A is transposed;
         = PlasmaTrans:     A is not transposed;
         = PlasmaTrans: A is conjugate transposed.
         Currently only PlasmaNoTrans is supported

transB   PLASMA_enum (IN)
         Specifies whether the matrix B is transposed, not transposed or conjugate  ↩
             transposed:
         = PlasmaNoTrans:   B is transposed;
         = PlasmaTrans:     B is not transposed;
         = PlasmaTrans: B is conjugate transposed.
         Currently only PlasmaNoTrans is supported

M        int (IN)
         M specifies the number of rows of the matrix op( A ) and of the matrix C. M >= 0.

N        int (IN)
         N specifies the number of columns of the matrix op( B ) and of the matrix C. N >=  ↩
             0.

K        int (IN)
         K specifies the number of columns of the matrix op( A ) and the number of rows of
         the matrix op( B ). K >= 0.

alpha    double (IN)
         alpha specifies the scalar alpha

A        double* (IN)
         A is a LDA-by-ka matrix, where ka is K when  transA = PlasmaNoTrans,
         and is  M  otherwise.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,M).

B        double* (IN)
         B is a LDB-by-kb matrix, where kb is N when  transB = PlasmaNoTrans,
         and is  K  otherwise.

LDB      int (IN)
         The leading dimension of the array B. LDB >= max(1,N).

beta     double (IN)
         beta specifies the scalar beta

C        double* (INOUT)
```

```
          C is a LDC-by-N matrix.
          On exit, the array is overwritten by the M by N matrix ( alpha*op( A )*op( B ) +  ↩
              beta*C )

LDC       int (IN)
          The leading dimension of the array C. LDC >= max(1,M).
```

51.3 Return Value:

```
           = 0: successful exit
```

51.4 C Bindings

```
int PLASMA_dgemm(PLASMA_enum transA, PLASMA_enum transB, int M, int N, int K, double alpha, ↩
     double *A, int LDA, double *B, int LDB, double beta, double *C, int LDC)
```

51.5 Fortran Bindings

```
PLASMA_DGEMM(INTEGER transA, INTEGER transB, INTEGER M, INTEGER N, INTEGER K, DOUBLE  ↩
    PRECISION alpha, DOUBLE PRECISION A, INTEGER LDA, DOUBLE PRECISION B, INTEGER LDB,  ↩
    DOUBLE PRECISION beta, DOUBLE PRECISION C, INTEGER LDC, INTEGER INFO)
```

51.6 Online Browsing

Dive into PLASMA_dgemm

52 PLASMA_dgeqrf

52.1 Purpose

PLASMA_dgeqrf - Computes the tile QR factorization of a complex M-by-N matrix A: A = Q * R.

52.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

A         double* (INOUT)
          On entry, the M-by-N matrix A.
          On exit, the elements on and above the diagonal of the array contain the min(M,N)- ↩
              by-N
          upper trapezoidal matrix R (R is upper triangular if M >= N); the elements below  ↩
              the
          diagonal represent the unitary matrix Q as a product of elementary reflectors  ↩
              stored
          by tiles.
```

```
LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,M).

T         double* (OUT)
          On exit, auxiliary factorization data, required by PLASMA_dgeqrs to solve the  ↩
              system
          of equations.
```

52.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

52.4 C Bindings

```
int PLASMA_dgeqrf(int M, int N, double *A, int LDA, double *T)
```

52.5 Fortran Bindings

```
PLASMA_DGEQRF(INTEGER M, INTEGER N, DOUBLE PRECISION A, INTEGER LDA, INTEGER T, INTEGER  ↩
    INFO)
```

52.6 Online Browsing

Dive into PLASMA_dgeqrf

53 PLASMA_dgeqrf_Tile

53.1 Purpose

PLASMA_dgeqrf_Tile - Computes the tile QR factorization of a complex M-by-N matrix A: A = Q * R. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

53.2 Arguments

```
A         double* (INOUT)
          On entry, the M-by-N matrix A.
          On exit, the elements on and above the diagonal of the array contain the min(M,N)- ↩
              by-N
          upper trapezoidal matrix R (R is upper triangular if M >= N); the elements below  ↩
              the
          diagonal represent the unitary matrix Q as a product of elementary reflectors  ↩
              stored
          by tiles.

T         double* (OUT)
          On exit, auxiliary factorization data, required by PLASMA_dgeqrs to solve the  ↩
              system
          of equations.
```

53.3 Return Value:

```
          = 0: successful exit
```

53.4 C Bindings

```
int PLASMA_dgeqrf_Tile(PLASMA_desc *A, PLASMA_desc *T)
```

53.5 Fortran Bindings

```
PLASMA_DGEQRF_TILE(INTEGER*4 A, INTEGER*4 T, INTEGER INFO)
```

53.6 Online Browsing

Dive into PLASMA_dgeqrf_Tile

54 PLASMA_dgeqrs

54.1 Purpose

PLASMA_dgeqrs - Compute a minimum-norm solution min || A*X - B || using the RQ factorization A = R*Q computed by PLASMA_dgeqrf.

54.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A. N >= M >= 0.

NRHS      int (IN)
          The number of columns of B. NRHS >= 0.

A         double* (INOUT)
          Details of the QR factorization of the original matrix A as returned by  ↩
              PLASMA_dgeqrf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= M.

T         double* (IN)
          Auxiliary factorization data, computed by PLASMA_dgeqrf.

B         double* (INOUT)
          On entry, the m-by-nrhs right hand side matrix B.
          On exit, the n-by-nrhs solution matrix X.

LDB       int (IN)
          The leading dimension of the array B. LDB >= max(1,N).
```

56.4 C Bindings

```
int PLASMA_dgesv(int N, int NRHS, double *A, int LDA, double *L, int *IPIV, double *B, int  ↩
    LDB)
```

56.5 Fortran Bindings

```
PLASMA_DGESV(INTEGER N, INTEGER NRHS, DOUBLE PRECISION A, INTEGER LDA, INTEGER LH, INTEGER  ↩
    IPIVH, DOUBLE PRECISION B, INTEGER LDB, INTEGER INFO)
```

56.6 Online Browsing

Dive into PLASMA_dgesv

57 PLASMA_dgesv_Tile

57.1 Purpose

PLASMA_dgesv_Tile - Computes the solution to a system of linear equations A **X = B, where A is an N-by-N matrix and X and B are N-by-NRHS matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A. The factored form of A is then used to solve the system of equations** A X = B. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

57.2 Arguments

```
A         double* (INOUT)
          On entry, the N-by-N coefficient matrix A.
          On exit, the tile L and U factors from the factorization (not equivalent to LAPACK ↩
              ).

L         double* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          necessary to solve the system of equations.

IPIV      int* (OUT)
          On exit, the pivot indices that define the permutations (not equivalent to LAPACK) ↩
              .

B         double* (INOUT)
          On entry, the N-by-NRHS matrix of right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

57.3 Return Value:

```
           = 0: successful exit
           > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
                but the factor U is exactly singular, so the solution could not be computed.
```

57.4 C Bindings

```
int PLASMA_dgesv_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

57.5 Fortran Bindings

```
PLASMA_DGESV_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

57.6 Online Browsing

Dive into PLASMA_dgesv_Tile

58 PLASMA_dgetrf

58.1 Purpose

PLASMA_dgetrf - Computes an LU factorization of a general M-by-N matrix A using the tile LU algorithm with partial tile pivoting with row interchanges.

58.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A. N >= 0.

A         double* (INOUT)
          On entry, the M-by-N matrix to be factored.
          On exit, the tile factors L and U from the factorization.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,M).

L         double* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          required by PLASMA_dgetrs to solve the system of equations.

IPIV      int* (OUT)
          The pivot indices that define the permutations (not equivalent to LAPACK).
```

58.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
          > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
               but the factor U is exactly singular, and division by zero will occur
               if it is used to solve a system of equations.
```

58.4 C Bindings

```
int PLASMA_dgetrf(int M, int N, double *A, int LDA, double *L, int *IPIV)
```

58.5 Fortran Bindings

```
PLASMA_DGETRF(INTEGER M, INTEGER N, DOUBLE PRECISION A, INTEGER LDA, INTEGER LH, INTEGER ↩
    IPIVH, INTEGER INFO)
```

58.6 Online Browsing

Dive into PLASMA_dgetrf

59 PLASMA_dgetrf_Tile

59.1 Purpose

PLASMA_dgetrf_Tile - Computes an LU factorization of a general M-by-N matrix A using the tile LU algorithm with partial tile pivoting with row interchanges. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

59.2 Arguments

```
A         double* (INOUT)
          On entry, the M-by-N matrix to be factored.
          On exit, the tile factors L and U from the factorization.

L         double* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          required by PLASMA_dgetrs to solve the system of equations.

IPIV      int* (OUT)
          The pivot indices that define the permutations (not equivalent to LAPACK).
```

59.3 Return Value:

```
           = 0: successful exit
           > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
                but the factor U is exactly singular, and division by zero will occur
                if it is used to solve a system of equations.
```

59.4 C Bindings

```
int PLASMA_dgetrf_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV)
```

59.5 Fortran Bindings

```
PLASMA_DGETRF_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER INFO)
```

59.6 Online Browsing

Dive into PLASMA_dgetrf_Tile

60 PLASMA_dgetrs

60.1 Purpose

PLASMA_dgetrs - Solves a system of linear equations A * X = B, with a general N-by-N matrix A using the tile LU factorization computed by PLASMA_dgetrf.

60.2 Arguments

```
trans    PLASMA_enum (IN)
         Intended to specify the the form of the system of equations:
         = PlasmaNoTrans:   A * X = B     (No transpose)
         = PlasmaTrans:     A**T * X = B  (Transpose)
         = PlasmaTrans: A**T * X = B  (Conjugate transpose)
         Currently only PlasmaNoTrans is supported.

N        int (IN)
         The order of the matrix A.  N >= 0.

NRHS     int (IN)
         The number of right hand sides, i.e., the number of columns of the matrix B.
         NRHS >= 0.

A        double* (IN)
         The tile factors L and U from the factorization, computed by PLASMA_dgetrf.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,N).

L        double* (IN)
         Auxiliary factorization data, related to the tile L factor, computed by  ↩
             PLASMA_dgetrf.

IPIV     int* (IN)
         The pivot indices from PLASMA_dgetrf (not equivalent to LAPACK).

B        double* (INOUT)
         On entry, the N-by-NRHS matrix of right hand side matrix B.
         On exit, the solution matrix X.

LDB      int (IN)
         The leading dimension of the array B. LDB >= max(1,N).
```

60.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
```

60.4 C Bindings

```
int PLASMA_dgetrs(PLASMA_enum uplo, int N, int NRHS, double *A, int LDA, double *L, int * ↩
    IPIV, double *B, int LDB)
```

60.5 Fortran Bindings

```
PLASMA_DGETRS(INTEGER uplo, INTEGER N, INTEGER NRHS, DOUBLE PRECISION A, INTEGER LDA,  ↩
    INTEGER LH, INTEGER IPIVH, DOUBLE PRECISION B, INTEGER LDB, INTEGER INFO)
```

60.6 Online Browsing

Dive into PLASMA_dgetrs

61 PLASMA_dgetrs_Tile

61.1 Purpose

PLASMA_dgetrs_Tile - Solves a system of linear equations A * X = B, with a general N-by-N matrix A using the tile LU factorization computed by PLASMA_dgetrf. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

61.2 Arguments

```
A         double* (IN)
          The tile factors L and U from the factorization, computed by PLASMA_dgetrf.

L         double* (IN)
          Auxiliary factorization data, related to the tile L factor, computed by  ↩
              PLASMA_dgetrf.

IPIV      int* (IN)
          The pivot indices from PLASMA_dgetrf (not equivalent to LAPACK).

B         double* (INOUT)
          On entry, the N-by-NRHS matrix of right hand side matrix B.
          On exit, the solution matrix X.
```

61.3 Return Value:

```
           = 0: successful exit
```

61.4 C Bindings

```
int PLASMA_dgetrs_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

61.5 Fortran Bindings

```
PLASMA_DGETRS_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

61.6 Online Browsing

Dive into PLASMA_dgetrs_Tile

62 PLASMA_dposv

62.1 Purpose

PLASMA_dposv - Computes the solution to a system of linear equations A * X = B, where A is an N-by-N symmetric positive definite (or Hermitian positive definite in the complex case) matrix and X and B are N-by-NRHS matrices. The Cholesky decomposition is used to factor A as

```
A = U**T * U, if uplo = PlasmaUpper, or
A = L * L**T, if uplo =  PlasmaLower,
```

where U is an upper triangular matrix and L is a lower triangular matrix. The factored form of A is then used to solve the system of equations A * X = B.

62.2 Arguments

```
uplo     PLASMA_enum (IN)
         Specifies whether the matrix A is upper triangular or lower triangular:
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

N        int (IN)
         The number of linear equations, i.e., the order of the matrix A. N >= 0.

NRHS     int (IN)
         The number of right hand sides, i.e., the number of columns of the matrix B. NRHS  ↩
             >= 0.

A        double* (INOUT)
         On entry, the symmetric positive definite (or Hermitian) matrix A.
         If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A
         contains the upper triangular part of the matrix A, and the strictly lower  ↩
             triangular
         part of A is not referenced.
         If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower
         triangular part of the matrix A, and the strictly upper triangular part of A is  ↩
             not
         referenced.
         On exit, if return value = 0, the factor U or L from the Cholesky factorization
         A = U**T*U or A = L*L**T.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,N).

B        double* (INOUT)
         On entry, the N-by-NRHS right hand side matrix B.
```

```
          If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A
          contains the upper triangular part of the matrix A, and the strictly lower  ↩
              triangular
          part of A is not referenced.
          If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower
          triangular part of the matrix A, and the strictly upper triangular part of A is  ↩
              not
          referenced.
          On exit, if return value = 0, the factor U or L from the Cholesky factorization
          A = U**T*U or A = L*L**T.

B         double* (INOUT)
          On entry, the N-by-NRHS right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

63.3 Return Value:

```
           = 0: successful exit
           > 0: if i, the leading minor of order i of A is not positive definite, so the
                factorization could not be completed, and the solution has not been computed ↩
                    .
```

63.4 C Bindings

```
int PLASMA_dposv_Tile(PLASMA_enum uplo, PLASMA_desc *A, PLASMA_desc *B)
```

63.5 Fortran Bindings

```
PLASMA_DPOSV_TILE(INTEGER uplo, INTEGER*4 A, INTEGER*4 B, INTEGER INFO)
```

63.6 Online Browsing

Dive into PLASMA_dposv_Tile

64 PLASMA_dpotrf

64.1 Purpose

PLASMA_dpotrf - Computes the Cholesky factorization of a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A. The factorization has the form

```
A = U**T * U, if uplo = PlasmaUpper, or
A = L * L**T, if uplo =  PlasmaLower,
```

where U is an upper triangular matrix and L is a lower triangular matrix.

64.2 Arguments

```
uplo     PLASMA_enum (IN)
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

N        int (IN)
         The order of the matrix A. N >= 0.

A        double* (INOUT)
         On entry, the symmetric positive definite (or Hermitian) matrix A.
         If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A
         contains the upper triangular part of the matrix A, and the strictly lower  ↩
             triangular
         part of A is not referenced.
         If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower
         triangular part of the matrix A, and the strictly upper triangular part of A is  ↩
             not
         referenced.
         On exit, if return value = 0, the factor U or L from the Cholesky factorization
         A = U**T*U or A = L*L**T.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,N).
```

64.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
          > 0: if i, the leading minor of order i of A is not positive definite, so the
               factorization could not be completed, and the solution has not been computed ↩
                   .
```

64.4 C Bindings

```
int PLASMA_dpotrf(PLASMA_enum uplo, int N, double *A, int LDA)
```

64.5 Fortran Bindings

```
PLASMA_DPOTRF(INTEGER uplo, INTEGER N, DOUBLE PRECISION A, INTEGER LDA, INTEGER INFO)
```

64.6 Online Browsing

Dive into PLASMA_dpotrf

65 PLASMA_dpotrf_Tile

65.1 Purpose

PLASMA_dpotrf_Tile - Computes the Cholesky factorization of a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A. The factorization has the form

```
A = U**T * U, if uplo = PlasmaUpper, or
A = L * L**T, if uplo =  PlasmaLower,
```

where U is an upper triangular matrix and L is a lower triangular matrix. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

65.2 Arguments

```
uplo     PLASMA_enum (IN)
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

A        double* (INOUT)
         On entry, the symmetric positive definite (or Hermitian) matrix A.
         If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A
         contains the upper triangular part of the matrix A, and the strictly lower ↩
             triangular
         part of A is not referenced.
         If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower
         triangular part of the matrix A, and the strictly upper triangular part of A is ↩
             not
         referenced.
         On exit, if return value = 0, the factor U or L from the Cholesky factorization
         A = U**T*U or A = L*L**T.
```

65.3 Return Value:

```
          = 0: successful exit
          > 0: if i, the leading minor of order i of A is not positive definite, so the
               factorization could not be completed, and the solution has not been computed ↩
                   .
```

65.4 C Bindings

```
int PLASMA_dpotrf_Tile(PLASMA_enum uplo, PLASMA_desc *A)
```

65.5 Fortran Bindings

```
PLASMA_DPOTRF_TILE(INTEGER uplo, INTEGER*4 A, INTEGER INFO)
```

65.6 Online Browsing

Dive into PLASMA_dpotrf_Tile

66 PLASMA_dpotrs

66.1 Purpose

PLASMA_dpotrs - Solves a system of linear equations A * X = B with a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A using the Cholesky factorization A = U**T*U or A = L*LT computed by PLASMA_dpotrf.

66.2 Arguments

```
uplo     PLASMA_enum (IN)
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

N        int (IN)
         The order of the matrix A. N >= 0.

NRHS     int (IN)
         The number of right hand sides, i.e., the number of columns of the matrix B. NRHS  ↩
             >= 0.

A        double* (IN)
         The triangular factor U or L from the Cholesky factorization A = U**T*U or A = L*L ↩
             **T,
         computed by PLASMA_dpotrf.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,N).

B        double* (INOUT)
         On entry, the N-by-NRHS right hand side matrix B.
         On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB      int (IN)
         The leading dimension of the array B. LDB >= max(1,N).
```

66.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
```

66.4 C Bindings

```
int PLASMA_dpotrs(PLASMA_enum uplo, int N, int NRHS, double *A, int LDA, double *B, int LDB ↩
    )
```

66.5 Fortran Bindings

```
PLASMA_DPOTRS(INTEGER uplo, INTEGER N, INTEGER NRHS, DOUBLE PRECISION A, INTEGER LDA,  ↩
    DOUBLE PRECISION B, INTEGER  LDB, INTEGER  INFO)
```

66.6 Online Browsing

Dive into PLASMA_dpotrs

67 PLASMA_dpotrs_Tile

67.1 Purpose

PLASMA_dpotrs_Tile - Solves a system of linear equations A * X = B with a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A using the Cholesky factorization A = U**T*U or A = L*LT** computed by PLASMA_dpotrf.

All matrices are passed through descriptors. All dimensions are taken from the descriptors.

67.2 Arguments

```
uplo     PLASMA_enum (IN)
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

A        double* (IN)
         The triangular factor U or L from the Cholesky factorization A = U**T*U or A = L*L ↩
             **T,
         computed by PLASMA_dpotrf.

B        double* (INOUT)
         On entry, the N-by-NRHS right hand side matrix B.
         On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

67.3 Return Value:

```
          = 0: successful exit
```

67.4 C Bindings

```
int PLASMA_dpotrs_Tile(PLASMA_enum uplo, PLASMA_desc *A, PLASMA_desc *B)
```

67.5 Fortran Bindings

```
PLASMA_DPOTRS_TILE(INTEGER uplo, INTEGER*4 A, INTEGER*4 B, INTEGER INFO)
```

67.6 Online Browsing

Dive into PLASMA_dpotrs_Tile

68 PLASMA_dsgesv

68.1 Purpose

PLASMA_dsgesv - Computes the solution to a system of linear equations A **X = B, where A is an N-by-N matrix and X and B are N-by-NRHS matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A. The factored form of A is then used to solve the system of equations A** X = B.

IMPORTANT NOTICE: in its current state, this routine only intends to be a proof-of-concept. There are still some costly serial parts and one may NOT expect to achieve high performance.

PLASMA_dsgesv first attempts to factorize the matrix in COMPLEX and use this factorization within an iterative refinement procedure to produce a solution with COMPLEX*16 normwise backward error quality (see below). If the approach fails the method switches to a COMPLEX*16 factorization and solve.

The iterative refinement is not going to be a winning strategy if the ratio COMPLEX performance over COMPLEX*16 performance is too small. A reasonable strategy should take the number of right-hand sides and the size of the matrix into account. This might be done with a call to ILAENV in the future. Up to now, we always try iterative refinement.

The iterative refinement process is stopped if ITER > ITERMAX or for all the RHS we have: RNRM < SQRT(N)*XNRM*ANRM*EPS* where

- ITER is the number of the current iteration in the iterative refinement process
- RNRM is the infinity-norm of the residual
- XNRM is the infinity-norm of the solution
- ANRM is the infinity-operator-norm of the matrix A
- EPS is the machine epsilon returned by DLAMCH(*Epsilon*).

The values ITERMAX and BWDMAX are fixed to 30 and 1.0D+00 respectively.

68.2 Arguments

```
N         int (IN)
          The number of linear equations, i.e., the order of the matrix A. N >= 0.

NRHS      int (IN)
          The number of right hand sides, i.e., the number of columns of the matrix B.
          NRHS >= 0.

A         double* (INOUT)
          On entry, the N-by-N coefficient matrix A.
          On exit, the tile L and U factors from the factorization (not equivalent to LAPACK ↩
              ).

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,N).

L         double* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          necessary to solve the system of equations.

IPIV      int* (OUT)
          On exit, the pivot indices that define the permutations (not equivalent to LAPACK) ↩
              .

B         double* (IN)
          The N-by-NRHS matrix of right hand side matrix B.

LDB       int (IN)
          The leading dimension of the array B. LDB >= max(1,N).

X         double* (OUT)
          If return value = 0, the N-by-NRHS solution matrix X.

LDX       int (IN)
          The leading dimension of the array B. LDX >= max(1,N).

ITER      int* (OUT)is the number of the current iteration in the iterative refinement  ↩
    process
```

68.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
          > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
               but the factor U is exactly singular, so the solution could not be computed.
```

68.4 C Bindings

```
int PLASMA_dsgesv(int N, int NRHS, double *A, int LDA, double *L, int *IPIV, double *B, int ↩
    LDB, double *X, int LDX, int *ITER)
```

68.5 Fortran Bindings

```
PLASMA_DSGESV(INTEGER N, INTEGER NRHS, DOUBLE PRECISION A, INTEGER LDA, INTEGER LH, INTEGER ↩
    IPIVH, DOUBLE PRECISION B, INTEGER LDB, DOUBLE PRECISION X, INTEGER LDX, INTEGER ITER,  ↩
   INTEGER INFO)
```

68.6 Online Browsing

Dive into PLASMA_dsgesv

69 PLASMA_dsgesv

69.1 Purpose

PLASMA_dsgesv - Computes the solution to a system of linear equations A **X = B, where A is an N-by-N matrix and X and B are N-by-NRHS matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A. The factored form of A is then used to solve the system of equations A** X = B. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

IMPORTANT NOTICE: in its current state, this routine only intends to be a proof-of-concept. There are still some costly serial parts and one may NOT expect to achieve high performance.

PLASMA_dsgesv first attempts to factorize the matrix in COMPLEX and use this factorization within an iterative refinement procedure to produce a solution with COMPLEX*16 normwise backward error quality (see below). If the approach fails the method switches to a COMPLEX*16 factorization and solve.

The iterative refinement is not going to be a winning strategy if the ratio COMPLEX performance over COMPLEX*16 performance is too small. A reasonable strategy should take the number of right-hand sides and the size of the matrix into account. This might be done with a call to ILAENV in the future. Up to now, we always try iterative refinement.

The iterative refinement process is stopped if ITER > ITERMAX or for all the RHS we have: RNRM < SQRT(N)*XNRM*ANRM*EPS* where

- ITER is the number of the current iteration in the iterative refinement process
- RNRM is the infinity-norm of the residual
- XNRM is the infinity-norm of the solution
- ANRM is the infinity-operator-norm of the matrix A
- EPS is the machine epsilon returned by DLAMCH(*Epsilon*).

The values ITERMAX and BWDMAX are fixed to 30 and 1.0D+00 respectively.

69.2 Arguments

```
N         int (IN)
          The number of linear equations, i.e., the order of the matrix A. N >= 0.

NRHS      int (IN)
          The number of right hand sides, i.e., the number of columns of the matrix B.
          NRHS >= 0.

A         double* (INOUT)
          On entry, the N-by-N coefficient matrix A.
          On exit, the tile L and U factors from the factorization (not equivalent to LAPACK ↩
              ).

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,N).

L         double* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          necessary to solve the system of equations.

IPIV      int* (OUT)
          On exit, the pivot indices that define the permutations (not equivalent to LAPACK) ↩
              .

B         double* (IN)
          The N-by-NRHS matrix of right hand side matrix B.

LDB       int (IN)
          The leading dimension of the array B. LDB >= max(1,N).

X         double* (OUT)
          If return value = 0, the N-by-NRHS solution matrix X.

LDX       int (IN)
          The leading dimension of the array B. LDX >= max(1,N).

ITER      int* (OUT)is the number of the current iteration in the iterative refinement  ↩
    process
```

69.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
           > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
                but the factor U is exactly singular, so the solution could not be computed.
```

69.4 C Bindings

```
int PLASMA_dsgesv(int N, int NRHS, double *A, int LDA, double *L, int *IPIV, double *B, int ↩
     LDB, double *X, int LDX, int *ITER)
```

69.5 Fortran Bindings

```
PLASMA_DSGESV(INTEGER N, INTEGER NRHS, DOUBLE PRECISION A, INTEGER LDA, INTEGER LH, INTEGER ↩
     IPIVH, DOUBLE PRECISION B, INTEGER LDB, DOUBLE PRECISION X, INTEGER LDX, INTEGER ITER,  ↩
    INTEGER INFO)
```

69.6 Online Browsing

Dive into PLASMA_dsgesv

70 PLASMA_dtrsm

70.1 Purpose

PLASMA_dtrsm - Computes triangular solve A*X = B or X*A = B

70.2 Arguments

```
side     PLASMA_enum (IN)
         Specifies whether A appears on the left or on the right of X:
         = PlasmaLeft:  A*X = B
         = PlasmaRight: X*A = B

uplo     PLASMA_enum (IN)
         Specifies whether the matrix A is upper triangular or lower triangular:
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

transA   PLASMA_enum (IN)
         Specifies whether the matrix A is transposed, not transposed or conjugate  ↩
             transposed:
         = PlasmaNoTrans:   A is transposed;
         = PlasmaTrans:     A is not transposed;
         = PlasmaTrans: A is conjugate transposed.

diag     PLASMA_enum (IN)
         Specifies whether or not A is unit triangular:
         = PlasmaNonUnit: A is non unit;
         = PlasmaUnit:    A us unit.

N        int (IN)
         The order of the matrix A. N >= 0.

NRHS     int (IN)
         The number of right hand sides, i.e., the number of columns of the matrix B. NRHS  ↩
             >= 0.

A        double* (IN)
         The triangular matrix A. If uplo = PlasmaUpper, the leading N-by-N upper  ↩
             triangular
         part of the array A contains the upper triangular matrix, and the strictly lower
         triangular part of A is not referenced. If uplo = PlasmaLower, the leading N-by-N
         lower triangular part of the array A contains the lower triangular matrix, and the
         strictly upper triangular part of A is not referenced. If diag = PlasmaUnit, the
         diagonal elements of A are also not referenced and are assumed to be 1.

LDA      int (IN)
```

```
          The leading dimension of the array A. LDA >= max(1,N).

B         double* (INOUT)
          On entry, the N-by-NRHS right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB       double* (IN)
          The leading dimension of the array B. LDB >= max(1,N).
```

70.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

70.4 C Bindings

```
int PLASMA_dtrsm(PLASMA_enum side, PLASMA_enum uplo, PLASMA_enum transA, PLASMA_enum diag,  ↩
    int N, int NRHS, double *A, int LDA, double *B, int LDB)
```

70.5 Fortran Bindings

```
PLASMA_DTRSM(INTEGER side, INTEGER uplo, INTEGER transA, INTEGER diag, INTEGER N, INTEGER  ↩
    NRHS, DOUBLE PRECISION A, INTEGER LDA, DOUBLE PRECISION B, INTEGER LDB, INTEGER INFO)
```

70.6 Online Browsing

Dive into PLASMA_dtrsm

71 PLASMA_dtrsmpl

71.1 Purpose

PLASMA_dtrsmpl - Performs the forward substitution step of solving a system of linear equations after the tile LU factorization of the matrix.

71.2 Arguments

```
N         int (IN)
          The order of the matrix A. N >= 0.

NRHS      int (IN)
          The number of right hand sides, i.e., the number of columns of the matrix B. NRHS  ↩
              >= 0.

A         double* (IN)
          The tile factor L from the factorization, computed by PLASMA_dgetrf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,N).
```

```
L         double* (IN)
          Auxiliary factorization data, related to the tile L factor, computed by  ↩
              PLASMA_dgetrf.

IPIV      int* (IN)
          The pivot indices from PLASMA_dgetrf (not equivalent to LAPACK).

B         double* (INOUT)
          On entry, the N-by-NRHS right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB       double* (IN)
          The leading dimension of the array B. LDB >= max(1,N).
```

71.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

71.4 C Bindings

```
int PLASMA_dtrsmpl(int N, int NRHS, double *A, int LDA, double *L, int *IPIV, double *B,  ↩
    int LDB)
```

71.5 Fortran Bindings

```
PLASMA_DTRSMPL(INTEGER N, INTEGER NRHS, DOUBLE PRECISION A, INTEGER LDA, INTEGER LH,  ↩
    INTEGER IPIVH, DOUBLE PRECISION B, INTEGER LDB, INTEGER INFO)
```

71.6 Online Browsing

Dive into PLASMA_dtrsmpl

72 PLASMA_dtrsmpl_Tile

72.1 Purpose

PLASMA_dtrsmpl_Tile - Performs the forward substitution step of solving a system of linear equations after the tile LU factorization of the matrix. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

72.2 Arguments

```
A         double* (IN)
          The tile factor L from the factorization, computed by PLASMA_dgetrf.

L         double* (IN)
          Auxiliary factorization data, related to the tile L factor, computed by  ↩
              PLASMA_dgetrf.

IPIV      int* (IN)
```

```
          The pivot indices from PLASMA_dgetrf (not equivalent to LAPACK).

B         double* (INOUT)
          On entry, the N-by-NRHS right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

72.3 Return Value:

```
           = 0: successful exit
```

72.4 C Bindings

```
int PLASMA_dtrsmpl_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

72.5 Fortran Bindings

```
PLASMA_DTRSMPL_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

72.6 Online Browsing

Dive into PLASMA_dtrsmpl_Tile

73 PLASMA_dtrsm_Tile

73.1 Purpose

PLASMA_dtrsm_Tile - Computes triangular solve A*X = B or X*A = B All matrices are passed through descriptors. All dimensions are taken from the descriptors.

73.2 Arguments

```
side      PLASMA_enum (IN)
          Specifies whether A appears on the left or on the right of X:
          = PlasmaLeft:  A*X = B
          = PlasmaRight: X*A = B

uplo      PLASMA_enum (IN)
          Specifies whether the matrix A is upper triangular or lower triangular:
          = PlasmaUpper: Upper triangle of A is stored;
          = PlasmaLower: Lower triangle of A is stored.

transA    PLASMA_enum (IN)
          Specifies whether the matrix A is transposed, not transposed or conjugate  ↩
              transposed:
          = PlasmaNoTrans:   A is transposed;
          = PlasmaTrans:     A is not transposed;
          = PlasmaTrans: A is conjugate transposed.

diag      PLASMA_enum (IN)
          Specifies whether or not A is unit triangular:
```

```
          = PlasmaNonUnit: A is non unit;
          = PlasmaUnit:    A us unit.

A         double* (IN)
          The triangular matrix A. If uplo = PlasmaUpper, the leading N-by-N upper  ↩
              triangular
          part of the array A contains the upper triangular matrix, and the strictly lower
          triangular part of A is not referenced. If uplo = PlasmaLower, the leading N-by-N
          lower triangular part of the array A contains the lower triangular matrix, and the
          strictly upper triangular part of A is not referenced. If diag = PlasmaUnit, the
          diagonal elements of A are also not referenced and are assumed to be 1.

B         double* (INOUT)
          On entry, the N-by-NRHS right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

73.3 Return Value:

```
           = 0: successful exit
```

73.4 C Bindings

```
int PLASMA_dtrsm_Tile(PLASMA_enum side, PLASMA_enum uplo, PLASMA_enum transA, PLASMA_enum  ↩
    diag, PLASMA_desc *A, PLASMA_desc *B)
```

73.5 Fortran Bindings

```
PLASMA_DTRSM_TILE(INTEGER side, INTEGER uplo, INTEGER transA, INTEGER diag, INTEGER*4 A,  ↩
    INTEGER*4 B, INTEGER INFO)
```

73.6 Online Browsing

Dive into PLASMA_dtrsm_Tile

74 PLASMA_sgelqf

74.1 Purpose

PLASMA_sgelqf - Computes the tile LQ factorization of a complex M-by-N matrix A: A = L * Q.

74.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A. N >= 0.

A         float* (INOUT)
          On entry, the M-by-N matrix A.
```

```
          On exit, the elements on and below the diagonal of the array contain the m-by-min( ↩
              M,N)
          lower trapezoidal matrix L (L is lower triangular if M <= N); the elements above  ↩
              the
          diagonal represent the unitary matrix Q as a product of elementary reflectors,  ↩
              stored
          by tiles.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,M).

T         float* (OUT)
          On exit, auxiliary factorization data, required by PLASMA_sgelqs to solve the  ↩
              system
          of equations.
```

74.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

74.4 C Bindings

```
int PLASMA_sgelqf(int M, int N, float *A, int LDA, float *T)
```

74.5 Fortran Bindings

```
PLASMA_SGELQF(INTEGER M, INTEGER N, REAL A, INTEGER LDA, INTEGER T, INTEGER INFO)
```

74.6 Online Browsing

Dive into PLASMA_sgelqf

75 PLASMA_sgelqf_Tile

75.1 Purpose

PLASMA_sgelqf_Tile - Computes the tile LQ factorization of a complex M-by-N matrix A: A = L * Q. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

75.2 Arguments

```
A         float* (INOUT)
          On entry, the M-by-N matrix A.
          On exit, the elements on and below the diagonal of the array contain the m-by-min( ↩
              M,N)
          lower trapezoidal matrix L (L is lower triangular if M <= N); the elements above  ↩
              the
          diagonal represent the unitary matrix Q as a product of elementary reflectors,  ↩
              stored
```

```
          by tiles.

T         float* (OUT)
          On exit, auxiliary factorization data, required by PLASMA_sgelqs to solve the  ↩
              system
          of equations.
```

75.3 Return Value:

```
           = 0: successful exit
```

75.4 C Bindings

```
int PLASMA_sgelqf_Tile(PLASMA_desc *A, PLASMA_desc *T)
```

75.5 Fortran Bindings

```
PLASMA_SGELQF_TILE(INTEGER*4 A, INTEGER*4 T,INTEGER INFO)
```

75.6 Online Browsing

Dive into PLASMA_sgelqf_Tile

76 PLASMA_sgelqs

76.1 Purpose

PLASMA_sgelqs - Compute a minimum-norm solution min || A*X - B || using the LQ factorization A = L*Q computed by PLASMA_sgelqf.

76.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A. N >= M >= 0.

NRHS      int (IN)
          The number of columns of B. NRHS >= 0.

A         float* (IN)
          Details of the LQ factorization of the original matrix A as returned by  ↩
              PLASMA_sgelqf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= M.

T         float* (IN)
          Auxiliary factorization data, computed by PLASMA_sgelqf.
```

```
B         float* (INOUT)
          On entry, the M-by-NRHS right hand side matrix B.
          On exit, the N-by-NRHS solution matrix X.

LDB       int (IN)
          The leading dimension of the array B. LDB >= N.
```

76.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

76.4 C Bindings

```
int PLASMA_sgelqs(int M, int N, int NRHS, float *A, int LDA, float *T, float *B, int LDB)
```

76.5 Fortran Bindings

```
PLASMA_SGELQS(INTEGER M, INTEGER N, INTEGER NRHS, REAL A, INTEGER LDA, INTEGER T, REAL B,  ↩
    INTEGER LDB, INTEGER INFO)
```

76.6 Online Browsing

Dive into PLASMA_sgelqs

77 PLASMA_sgelqs_Tile

77.1 Purpose

PLASMA_sgelqs_Tile - Compute a minimum-norm solution min || A*X - B || using the LQ factorization A = L*Q computed by PLASMA_sgelqf_Tile. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

77.2 Arguments

```
A         float* (IN)
          Details of the LQ factorization of the original matrix A as returned by  ↩
              PLASMA_sgelqf.

T         float* (IN)
          Auxiliary factorization data, computed by PLASMA_sgelqf.

B         float* (INOUT)
          On entry, the M-by-NRHS right hand side matrix B.
          On exit, the N-by-NRHS solution matrix X.
```

77.3 Return Value:

```
          = 0: successful exit
```

77.4 C Bindings

```
int PLASMA_sgelqs_Tile(PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

77.5 Fortran Bindings

```
PLASMA_SGELQS_TILE(INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

77.6 Online Browsing

Dive into PLASMA_sgelqs_Tile

78 PLASMA_sgels

78.1 Purpose

PLASMA_sgels - solves overdetermined or underdetermined linear systems involving an M-by-N matrix A using the QR or the LQ factorization of A. It is assumed that A has full rank. The following options are provided:

1. trans = PlasmaNoTrans and M >= N: find the least squares solution of an overdetermined system, i.e., solve the least squares problem: minimize || B - A*X ||.
2. trans = PlasmaNoTrans and M < N: find the minimum norm solution of an underdetermined system A * X = B.

Several right hand side vectors B and solution vectors X can be handled in a single call; they are stored as the columns of the M-by-NRHS right hand side matrix B and the N-by-NRHS solution matrix X.

78.2 Arguments

```
trans    PLASMA_enum (IN)
         Intended usage:
         = PlasmaNoTrans:   the linear system involves A;
         = PlasmaTrans: the linear system involves A**T.
         Currently only PlasmaNoTrans is supported.

M        int (IN)
         The number of rows of the matrix A. M >= 0.

N        int (IN)
         The number of columns of the matrix A. N >= 0.

NRHS     int (IN)
         The number of right hand sides, i.e., the number of columns of the matrices B and  ↩
             X.
         NRHS >= 0.

A        float* (INOUT)
```

```
          On entry, the M-by-N matrix A.
          On exit,
          if M >= N, A is overwritten by details of its QR factorization as returned by
                     PLASMA_sgeqrf;
          if M < N, A is overwritten by details of its LQ factorization as returned by
                      PLASMA_sgelqf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,M).

T         float* (OUT)
          On exit, auxiliary factorization data.

B         float* (INOUT)
          On entry, the M-by-NRHS matrix B of right hand side vectors, stored columnwise;
          On exit, if return value = 0, B is overwritten by the solution vectors, stored
          columnwise:
          if M >= N, rows 1 to N of B contain the least squares solution vectors; the  ↩
              residual
          sum of squares for the solution in each column is given by the sum of squares of  ↩
              the
          modulus of elements N+1 to M in that column;
          if M < N, rows 1 to N of B contain the minimum norm solution vectors;

LDB       int (IN)
          The leading dimension of the array B. LDB >= MAX(1,M,N).
```

78.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

78.4 C Bindings

```
int PLASMA_sgels(PLASMA_enum trans, int M, int N, int NRHS, float *A, int LDA, float *T,  ↩
    float *B, int LDB)
```

78.5 Fortran Bindings

```
PLASMA_SGELS(INTEGER trans, INTEGER M, INTEGER N, INTEGER NRHS, REAL A, INTEGER LDA,  ↩
    INTEGER T, REAL B, INTEGER LDB, INTEGER INFO)
```

78.6 Online Browsing

Dive into PLASMA_sgels

79 PLASMA_sgels_Tile

79.1 Purpose

PLASMA_sgels_Tile - solves overdetermined or underdetermined linear systems involving an M-by-N matrix A using the QR or the LQ factorization of A. It is assumed that A has full rank. The following options are provided:

1. trans = PlasmaNoTrans and M >= N: find the least squares solution of an overdetermined system, i.e., solve the least squares problem: minimize || B - A*X ||.
2. trans = PlasmaNoTrans and M < N: find the minimum norm solution of an underdetermined system A * X = B.

Several right hand side vectors B and solution vectors X can be handled in a single call; they are stored as the columns of the M-by-NRHS right hand side matrix B and the N-by-NRHS solution matrix X. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

79.2 Arguments

```
trans    PLASMA_enum (IN)
         Intended usage:
         = PlasmaNoTrans:   the linear system involves A;
         = PlasmaTrans: the linear system involves A**T.
         Currently only PlasmaNoTrans is supported.

A        float* (INOUT)
         On entry, the M-by-N matrix A.
         On exit,
         if M >= N, A is overwritten by details of its QR factorization as returned by
                    PLASMA_sgeqrf;
         if M < N, A is overwritten by details of its LQ factorization as returned by
                     PLASMA_sgelqf.

T        float* (OUT)
         On exit, auxiliary factorization data.

B        float* (INOUT)
         On entry, the M-by-NRHS matrix B of right hand side vectors, stored columnwise;
         On exit, if return value = 0, B is overwritten by the solution vectors, stored
         columnwise:
         if M >= N, rows 1 to N of B contain the least squares solution vectors; the  ↩
             residual
         sum of squares for the solution in each column is given by the sum of squares of  ↩
             the
         modulus of elements N+1 to M in that column;
         if M < N, rows 1 to N of B contain the minimum norm solution vectors;
```

79.3 Return Value:

```
          = 0: successful exit
```

79.4 C Bindings

```
int PLASMA_sgels_Tile(PLASMA_enum trans, PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

79.5 Fortran Bindings

```
PLASMA_SGELS_TILE(INTEGER trans, INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

79.6 Online Browsing

Dive into PLASMA_sgels_Tile

80 PLASMA_sgemm

80.1 Purpose

PLASMA_sgemm - Performs one of the matrix-matrix operations

```
C = alpha*op( A )*op( B ) + beta*C,
```

where op(X) is one of

```
op( X ) = X  or op( X ) = X'
```

alpha and beta are scalars, and A, B and C are matrices, with op(A) an m by k matrix, op(B) a k by n matrix and C an m by n matrix.

80.2 Arguments

```
transA   PLASMA_enum (IN)
         Specifies whether the matrix A is transposed, not transposed or conjugate  ↩
             transposed:
         = PlasmaNoTrans:   A is transposed;
         = PlasmaTrans:     A is not transposed;
         = PlasmaTrans: A is conjugate transposed.
         Currently only PlasmaNoTrans is supported

transB   PLASMA_enum (IN)
         Specifies whether the matrix B is transposed, not transposed or conjugate  ↩
             transposed:
         = PlasmaNoTrans:   B is transposed;
         = PlasmaTrans:     B is not transposed;
         = PlasmaTrans: B is conjugate transposed.
         Currently only PlasmaNoTrans is supported

M        int (IN)
         M specifies the number of rows of the matrix op( A ) and of the matrix C. M >= 0.

N        int (IN)
         N specifies the number of columns of the matrix op( B ) and of the matrix C. N >=  ↩
             0.

K        int (IN)
         K specifies the number of columns of the matrix op( A ) and the number of rows of
         the matrix op( B ). K >= 0.

alpha    float (IN)
         alpha specifies the scalar alpha

A        float* (IN)
         A is a LDA-by-ka matrix, where ka is K when  transA = PlasmaNoTrans,
         and is  M  otherwise.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,M).

B        float* (IN)
         B is a LDB-by-kb matrix, where kb is N when  transB = PlasmaNoTrans,
         and is  K  otherwise.
```

```
LDB       int (IN)
          The leading dimension of the array B. LDB >= max(1,N).

beta      float (IN)
          beta specifies the scalar beta

C         float* (INOUT)
          C is a LDC-by-N matrix.
          On exit, the array is overwritten by the M by N matrix ( alpha*op( A )*op( B ) +  ↩
              beta*C )

LDC       int (IN)
          The leading dimension of the array C. LDC >= max(1,M).
```

80.3 Return Value:

```
           = 0: successful exit
```

80.4 C Bindings

```
int PLASMA_sgemm(PLASMA_enum transA, PLASMA_enum transB, int M, int N, int K, float alpha,  ↩
    float *A, int LDA, float *B, int LDB, float beta, float *C, int LDC)
```

80.5 Fortran Bindings

```
PLASMA_SGEMM(INTEGER transA, INTEGER transB, INTEGER M, INTEGER N, INTEGER K, REAL alpha,  ↩
    REAL A, INTEGER LDA, REAL B, INTEGER LDB, REAL beta, REAL C, INTEGER LDC, INTEGER INFO)
```

80.6 Online Browsing

Dive into PLASMA_sgemm

81 PLASMA_sgeqrf

81.1 Purpose

PLASMA_sgeqrf - Computes the tile QR factorization of a complex M-by-N matrix A: A = Q * R.

81.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

A         float* (INOUT)
          On entry, the M-by-N matrix A.
```

```
          On exit, the elements on and above the diagonal of the array contain the min(M,N)- ↩
              by-N
          upper trapezoidal matrix R (R is upper triangular if M >= N); the elements below  ↩
              the
          diagonal represent the unitary matrix Q as a product of elementary reflectors  ↩
              stored
          by tiles.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,M).

T         float* (OUT)
          On exit, auxiliary factorization data, required by PLASMA_sgeqrs to solve the  ↩
              system
          of equations.
```

81.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

81.4 C Bindings

```
int PLASMA_sgeqrf(int M, int N, float *A, int LDA, float *T)
```

81.5 Fortran Bindings

```
PLASMA_SGEQRF(INTEGER M, INTEGER N, REAL A, INTEGER LDA, INTEGER T, INTEGER INFO)
```

81.6 Online Browsing

Dive into PLASMA_sgeqrf

82 PLASMA_sgeqrf_Tile

82.1 Purpose

PLASMA_sgeqrf_Tile - Computes the tile QR factorization of a complex M-by-N matrix A: A = Q * R. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

82.2 Arguments

```
A         float* (INOUT)
          On entry, the M-by-N matrix A.
          On exit, the elements on and above the diagonal of the array contain the min(M,N)- ↩
              by-N
          upper trapezoidal matrix R (R is upper triangular if M >= N); the elements below  ↩
              the
          diagonal represent the unitary matrix Q as a product of elementary reflectors  ↩
              stored
```

```
          by tiles.

T         float* (OUT)
          On exit, auxiliary factorization data, required by PLASMA_sgeqrs to solve the  ↩
              system
          of equations.
```

82.3 Return Value:

```
           = 0: successful exit
```

82.4 C Bindings

```
int PLASMA_sgeqrf_Tile(PLASMA_desc *A, PLASMA_desc *T)
```

82.5 Fortran Bindings

```
PLASMA_SGEQRF_TILE(INTEGER*4 A, INTEGER*4 T, INTEGER INFO)
```

82.6 Online Browsing

Dive into PLASMA_sgeqrf_Tile

83 PLASMA_sgeqrs

83.1 Purpose

PLASMA_sgeqrs - Compute a minimum-norm solution min || A*X - B || using the RQ factorization A = R*Q computed by PLASMA_sgeqrf.

83.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A. N >= M >= 0.

NRHS      int (IN)
          The number of columns of B. NRHS >= 0.

A         float* (INOUT)
          Details of the QR factorization of the original matrix A as returned by  ↩
              PLASMA_sgeqrf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= M.

T         float* (IN)
          Auxiliary factorization data, computed by PLASMA_sgeqrf.
```

```
B         float* (INOUT)
          On entry, the m-by-nrhs right hand side matrix B.
          On exit, the n-by-nrhs solution matrix X.

LDB       int (IN)
          The leading dimension of the array B. LDB >= max(1,N).
```

83.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

83.4 C Bindings

```
int PLASMA_sgeqrs(int M, int N, int NRHS, float *A, int LDA, float *T, float *B, int LDB)
```

83.5 Fortran Bindings

```
PLASMA_SGEQRS(INTEGER M, INTEGER N, INTEGER NRHS, REAL A, INTEGER LDA, INTEGER T, REAL B,  ↩
    INTEGER LDB, INTEGER INFO)
```

83.6 Online Browsing

Dive into PLASMA_sgeqrs

84 PLASMA_sgeqrs_Tile

84.1 Purpose

PLASMA_sgeqrs_Tile - Compute a minimum-norm solution min || A*X - B || using the RQ factorization A = R*Q computed by PLASMA_sgeqrf_Tile. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

84.2 Arguments

```
A         float* (INOUT)
          Details of the QR factorization of the original matrix A as returned by  ↩
              PLASMA_sgeqrf.

T         float* (IN)
          Auxiliary factorization data, computed by PLASMA_sgeqrf.

B         float* (INOUT)
          On entry, the m-by-nrhs right hand side matrix B.
          On exit, the n-by-nrhs solution matrix X.
```

84.3 Return Value:

```
          = 0: successful exit
```

84.4 C Bindings

```
int PLASMA_sgeqrs_Tile(PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

84.5 Fortran Bindings

```
PLASMA_SGEQRS_TILE(INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

84.6 Online Browsing

Dive into PLASMA_sgeqrs_Tile

85 PLASMA_sgesv

85.1 Purpose

PLASMA_sgesv - Computes the solution to a system of linear equations A **X = B, where A is an N-by-N matrix and X and B are N-by-NRHS matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A. The factored form of A is then used to solve the system of equations A** X = B.

85.2 Arguments

```
N         int (IN)
          The number of linear equations, i.e., the order of the matrix A. N >= 0.

NRHS      int (IN)
          The number of right hand sides, i.e., the number of columns of the matrix B.
          NRHS >= 0.

A         float* (INOUT)
          On entry, the N-by-N coefficient matrix A.
          On exit, the tile L and U factors from the factorization (not equivalent to LAPACK ↩
              ).

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,N).

L         float* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          necessary to solve the system of equations.

IPIV      int* (OUT)
          On exit, the pivot indices that define the permutations (not equivalent to LAPACK) ↩
              .

B         float* (INOUT)
          On entry, the N-by-NRHS matrix of right hand side matrix B.
```

```
          On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB       int (IN)
          The leading dimension of the array B. LDB >= max(1,N).
```

85.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
           > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
                but the factor U is exactly singular, so the solution could not be computed.
```

85.4 C Bindings

```
int PLASMA_sgesv(int N, int NRHS, float *A, int LDA, float *L, int *IPIV, float *B, int LDB ↩
    )
```

85.5 Fortran Bindings

```
PLASMA_SGESV(INTEGER N, INTEGER NRHS, REAL A, INTEGER LDA, INTEGER LH, INTEGER IPIVH, REAL  ↩
    B, INTEGER LDB, INTEGER INFO)
```

85.6 Online Browsing

Dive into PLASMA_sgesv

86 PLASMA_sgesv_Tile

86.1 Purpose

PLASMA_sgesv_Tile - Computes the solution to a system of linear equations A **X = B, where A is an N-by-N matrix and X and B are N-by-NRHS matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A. The factored form of A is then used to solve the system of equations A** X = B. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

86.2 Arguments

```
A         float* (INOUT)
          On entry, the N-by-N coefficient matrix A.
          On exit, the tile L and U factors from the factorization (not equivalent to LAPACK ↩
              ).

L         float* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          necessary to solve the system of equations.

IPIV      int* (OUT)
          On exit, the pivot indices that define the permutations (not equivalent to LAPACK) ↩
              .
```

```
B         float* (INOUT)
          On entry, the N-by-NRHS matrix of right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

86.3 Return Value:

```
           = 0: successful exit
           > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
                but the factor U is exactly singular, so the solution could not be computed.
```

86.4 C Bindings

```
int PLASMA_sgesv_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

86.5 Fortran Bindings

```
PLASMA_SGESV_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

86.6 Online Browsing

Dive into PLASMA_sgesv_Tile

87 PLASMA_sgetrf

87.1 Purpose

PLASMA_sgetrf - Computes an LU factorization of a general M-by-N matrix A using the tile LU algorithm with partial tile pivoting with row interchanges.

87.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A. N >= 0.

A         float* (INOUT)
          On entry, the M-by-N matrix to be factored.
          On exit, the tile factors L and U from the factorization.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,M).

L         float* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          required by PLASMA_sgetrs to solve the system of equations.

IPIV      int* (OUT)
          The pivot indices that define the permutations (not equivalent to LAPACK).
```

87.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
          > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
               but the factor U is exactly singular, and division by zero will occur
               if it is used to solve a system of equations.
```

87.4 C Bindings

```
int PLASMA_sgetrf(int M, int N, float *A, int LDA, float *L, int *IPIV)
```

87.5 Fortran Bindings

```
PLASMA_SGETRF(INTEGER M, INTEGER N, REAL A, INTEGER LDA, INTEGER LH, INTEGER IPIVH, INTEGER ↩
     INFO)
```

87.6 Online Browsing

Dive into PLASMA_sgetrf

88 PLASMA_sgetrf_Tile

88.1 Purpose

PLASMA_sgetrf_Tile - Computes an LU factorization of a general M-by-N matrix A using the tile LU algorithm with partial tile pivoting with row interchanges. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

88.2 Arguments

```
A         float* (INOUT)
          On entry, the M-by-N matrix to be factored.
          On exit, the tile factors L and U from the factorization.

L         float* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          required by PLASMA_sgetrs to solve the system of equations.

IPIV      int* (OUT)
          The pivot indices that define the permutations (not equivalent to LAPACK).
```

88.3 Return Value:

```
          = 0: successful exit
          > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
               but the factor U is exactly singular, and division by zero will occur
               if it is used to solve a system of equations.
```

88.4 C Bindings

```
int PLASMA_sgetrf_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV)
```

88.5 Fortran Bindings

```
PLASMA_SGETRF_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER INFO)
```

88.6 Online Browsing

Dive into PLASMA_sgetrf_Tile

89 PLASMA_sgetrs

89.1 Purpose

PLASMA_sgetrs - Solves a system of linear equations A * X = B, with a general N-by-N matrix A using the tile LU factorization computed by PLASMA_sgetrf.

89.2 Arguments

```
trans    PLASMA_enum (IN)
         Intended to specify the the form of the system of equations:
         = PlasmaNoTrans:   A * X = B     (No transpose)
         = PlasmaTrans:     A**T * X = B  (Transpose)
         = PlasmaTrans: A**T * X = B  (Conjugate transpose)
         Currently only PlasmaNoTrans is supported.

N        int (IN)
         The order of the matrix A.  N >= 0.

NRHS     int (IN)
         The number of right hand sides, i.e., the number of columns of the matrix B.
         NRHS >= 0.

A        float* (IN)
         The tile factors L and U from the factorization, computed by PLASMA_sgetrf.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,N).

L        float* (IN)
         Auxiliary factorization data, related to the tile L factor, computed by  ↩
             PLASMA_sgetrf.

IPIV     int* (IN)
         The pivot indices from PLASMA_sgetrf (not equivalent to LAPACK).

B        float* (INOUT)
         On entry, the N-by-NRHS matrix of right hand side matrix B.
         On exit, the solution matrix X.

LDB      int (IN)
         The leading dimension of the array B. LDB >= max(1,N).
```

89.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
```

89.4 C Bindings

```
int PLASMA_sgetrs(PLASMA_enum uplo, int N, int NRHS, float *A, int LDA, float *L, int *IPIV ↩
    , float *B, int LDB)
```

89.5 Fortran Bindings

```
PLASMA_SGETRS(INTEGER uplo, INTEGER N, INTEGER NRHS, REAL A, INTEGER LDA, INTEGER LH,  ↩
    INTEGER IPIVH, REAL B, INTEGER LDB, INTEGER INFO)
```

89.6 Online Browsing

Dive into PLASMA_sgetrs

90 PLASMA_sgetrs_Tile

90.1 Purpose

PLASMA_sgetrs_Tile - Solves a system of linear equations A * X = B, with a general N-by-N matrix A using the tile LU factorization computed by PLASMA_sgetrf. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

90.2 Arguments

```
A         float* (IN)
          The tile factors L and U from the factorization, computed by PLASMA_sgetrf.

L         float* (IN)
          Auxiliary factorization data, related to the tile L factor, computed by  ↩
              PLASMA_sgetrf.

IPIV      int* (IN)
          The pivot indices from PLASMA_sgetrf (not equivalent to LAPACK).

B         float* (INOUT)
          On entry, the N-by-NRHS matrix of right hand side matrix B.
          On exit, the solution matrix X.
```

90.3 Return Value:

```
          = 0: successful exit
```

90.4 C Bindings

```
int PLASMA_sgetrs_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

90.5 Fortran Bindings

```
PLASMA_SGETRS_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

90.6 Online Browsing

Dive into PLASMA_sgetrs_Tile

91 PLASMA_sposv

91.1 Purpose

PLASMA_sposv - Computes the solution to a system of linear equations A * X = B, where A is an N-by-N symmetric positive definite (or Hermitian positive definite in the complex case) matrix and X and B are N-by-NRHS matrices. The Cholesky decomposition is used to factor A as

```
A = U**T * U, if uplo = PlasmaUpper, or
A = L * L**T, if uplo =  PlasmaLower,
```

where U is an upper triangular matrix and L is a lower triangular matrix. The factored form of A is then used to solve the system of equations A * X = B.

91.2 Arguments

```
uplo     PLASMA_enum (IN)
         Specifies whether the matrix A is upper triangular or lower triangular:
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

N        int (IN)
         The number of linear equations, i.e., the order of the matrix A. N >= 0.

NRHS     int (IN)
         The number of right hand sides, i.e., the number of columns of the matrix B. NRHS  ↩
             >= 0.

A        float* (INOUT)
         On entry, the symmetric positive definite (or Hermitian) matrix A.
         If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A
         contains the upper triangular part of the matrix A, and the strictly lower  ↩
             triangular
         part of A is not referenced.
         If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower
         triangular part of the matrix A, and the strictly upper triangular part of A is  ↩
             not
         referenced.
         On exit, if return value = 0, the factor U or L from the Cholesky factorization
         A = U**T*U or A = L*L**T.
```

```
LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,N).

B        float* (INOUT)
         On entry, the N-by-NRHS right hand side matrix B.
         On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB      int (IN)
         The leading dimension of the array B. LDB >= max(1,N).
```

91.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
          > 0: if i, the leading minor of order i of A is not positive definite, so the
               factorization could not be completed, and the solution has not been computed ↩
                   .
```

91.4 C Bindings

```
int PLASMA_sposv(PLASMA_enum uplo, int N, int NRHS, float *A, int LDA, float *B, int LDB)
```

91.5 Fortran Bindings

```
PLASMA_SPOSV(INTEGER uplo, INTEGER N, INTEGER NRHS, REAL A, INTEGER LDA, REAL B, INTEGER  ↩
    LDB, INTEGER INFO)
```

91.6 Online Browsing

Dive into PLASMA_sposv

92 PLASMA_sposv_Tile

92.1 Purpose

PLASMA_sposv_Tile - Computes the solution to a system of linear equations A * X = B, where A is an N-by-N symmetric positive definite (or Hermitian positive definite in the complex case) matrix and X and B are N-by-NRHS matrices. The Cholesky decomposition is used to factor A as

```
A = U**T * U, if uplo = PlasmaUpper, or
A = L * L**T, if uplo =  PlasmaLower,
```

where U is an upper triangular matrix and L is a lower triangular matrix. The factored form of A is then used to solve the system of equations A * X = B. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

93.2 Arguments

```
uplo     PLASMA_enum (IN)
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

N        int (IN)
         The order of the matrix A. N >= 0.

A        float* (INOUT)
         On entry, the symmetric positive definite (or Hermitian) matrix A.
         If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A
         contains the upper triangular part of the matrix A, and the strictly lower  ↩
            triangular
         part of A is not referenced.
         If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower
         triangular part of the matrix A, and the strictly upper triangular part of A is  ↩
            not
         referenced.
         On exit, if return value = 0, the factor U or L from the Cholesky factorization
         A = U**T*U or A = L*L**T.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,N).
```

93.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
          > 0: if i, the leading minor of order i of A is not positive definite, so the
               factorization could not be completed, and the solution has not been computed ↩
                   .
```

93.4 C Bindings

```
int PLASMA_spotrf(PLASMA_enum uplo, int N, float *A, int LDA)
```

93.5 Fortran Bindings

```
PLASMA_SPOTRF(INTEGER uplo, INTEGER N, REAL A, INTEGER LDA, INTEGER INFO)
```

93.6 Online Browsing

Dive into PLASMA_spotrf

94 PLASMA_spotrf_Tile

94.1 Purpose

PLASMA_spotrf_Tile - Computes the Cholesky factorization of a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A. The factorization has the form

```
A = U**T * U, if uplo = PlasmaUpper, or
A = L * L**T, if uplo =  PlasmaLower,
```

where U is an upper triangular matrix and L is a lower triangular matrix. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

94.2 Arguments

```
uplo     PLASMA_enum (IN)
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

A        float* (INOUT)
         On entry, the symmetric positive definite (or Hermitian) matrix A.
         If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A
         contains the upper triangular part of the matrix A, and the strictly lower ↩
             triangular
         part of A is not referenced.
         If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower
         triangular part of the matrix A, and the strictly upper triangular part of A is ↩
             not
         referenced.
         On exit, if return value = 0, the factor U or L from the Cholesky factorization
         A = U**T*U or A = L*L**T.
```

94.3 Return Value:

```
          = 0: successful exit
          > 0: if i, the leading minor of order i of A is not positive definite, so the
               factorization could not be completed, and the solution has not been computed ↩
                   .
```

94.4 C Bindings

```
int PLASMA_spotrf_Tile(PLASMA_enum uplo, PLASMA_desc *A)
```

94.5 Fortran Bindings

```
PLASMA_SPOTRF_TILE(INTEGER uplo, INTEGER*4 A, INTEGER INFO)
```

94.6 Online Browsing

Dive into PLASMA_spotrf_Tile

95 PLASMA_spotrs

95.1 Purpose

PLASMA_spotrs - Solves a system of linear equations A * X = B with a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A using the Cholesky factorization A = U**T*U or A = L*LT computed by PLASMA_spotrf.

95.2 Arguments

```
uplo     PLASMA_enum (IN)
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

N        int (IN)
         The order of the matrix A. N >= 0.

NRHS     int (IN)
         The number of right hand sides, i.e., the number of columns of the matrix B. NRHS
             >= 0.

A        float* (IN)
         The triangular factor U or L from the Cholesky factorization A = U**T*U or A = L*L
             **T,
         computed by PLASMA_spotrf.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,N).

B        float* (INOUT)
         On entry, the N-by-NRHS right hand side matrix B.
         On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB      int (IN)
         The leading dimension of the array B. LDB >= max(1,N).
```

95.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
```

95.4 C Bindings

```
int PLASMA_spotrs(PLASMA_enum uplo, int N, int NRHS, float *A, int LDA, float *B, int LDB)
```

95.5 Fortran Bindings

```
PLASMA_SPOTRS(INTEGER uplo, INTEGER N, INTEGER NRHS, REAL A, INTEGER LDA, REAL B, INTEGER
    LDB, INTEGER  INFO)
```

95.6 Online Browsing

Dive into PLASMA_spotrs

96 PLASMA_spotrs_Tile

96.1 Purpose

PLASMA_spotrs_Tile - Solves a system of linear equations A * X = B with a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A using the Cholesky factorization A = U**T*U or A = L*LT** computed by PLASMA_spotrf. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

96.2 Arguments

```
uplo     PLASMA_enum (IN)
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

A        float* (IN)
         The triangular factor U or L from the Cholesky factorization A = U**T*U or A = L*L ↩
             **T,
         computed by PLASMA_spotrf.

B        float* (INOUT)
         On entry, the N-by-NRHS right hand side matrix B.
         On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

96.3 Return Value:

```
          = 0: successful exit
```

96.4 C Bindings

```
int PLASMA_spotrs_Tile(PLASMA_enum uplo, PLASMA_desc *A, PLASMA_desc *B)
```

96.5 Fortran Bindings

```
PLASMA_SPOTRS_TILE(INTEGER uplo, INTEGER*4 A, INTEGER*4 B, INTEGER INFO)
```

96.6 Online Browsing

Dive into PLASMA_spotrs_Tile

97 PLASMA_strsm

97.1 Purpose

PLASMA_strsm - Computes triangular solve A*X = B or X*A = B

97.2 Arguments

```
side     PLASMA_enum (IN)
         Specifies whether A appears on the left or on the right of X:
         = PlasmaLeft:  A*X = B
         = PlasmaRight: X*A = B

uplo     PLASMA_enum (IN)
         Specifies whether the matrix A is upper triangular or lower triangular:
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

transA   PLASMA_enum (IN)
```

```
          Specifies whether the matrix A is transposed, not transposed or conjugate  ↩
              transposed:
          = PlasmaNoTrans:   A is transposed;
          = PlasmaTrans:     A is not transposed;
          = PlasmaTrans: A is conjugate transposed.

diag      PLASMA_enum (IN)
          Specifies whether or not A is unit triangular:
          = PlasmaNonUnit: A is non unit;
          = PlasmaUnit:    A us unit.

N         int (IN)
          The order of the matrix A. N >= 0.

NRHS      int (IN)
          The number of right hand sides, i.e., the number of columns of the matrix B. NRHS  ↩
              >= 0.

A         float* (IN)
          The triangular matrix A. If uplo = PlasmaUpper, the leading N-by-N upper  ↩
              triangular
          part of the array A contains the upper triangular matrix, and the strictly lower
          triangular part of A is not referenced. If uplo = PlasmaLower, the leading N-by-N
          lower triangular part of the array A contains the lower triangular matrix, and the
          strictly upper triangular part of A is not referenced. If diag = PlasmaUnit, the
          diagonal elements of A are also not referenced and are assumed to be 1.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,N).

B         float* (INOUT)
          On entry, the N-by-NRHS right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB       float* (IN)
          The leading dimension of the array B. LDB >= max(1,N).
```

97.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

97.4 C Bindings

```
int PLASMA_strsm(PLASMA_enum side, PLASMA_enum uplo, PLASMA_enum transA, PLASMA_enum diag,  ↩
    int N, int NRHS, float *A, int LDA, float *B, int LDB)
```

97.5 Fortran Bindings

```
PLASMA_STRSM(INTEGER side, INTEGER uplo, INTEGER transA, INTEGER diag, INTEGER N, INTEGER  ↩
    NRHS, REAL A, INTEGER LDA, REAL B, INTEGER LDB, INTEGER INFO)
```

97.6 Online Browsing

Dive into PLASMA_strsm

98 PLASMA_strsmpl

98.1 Purpose

PLASMA_strsmpl - Performs the forward substitution step of solving a system of linear equations after the tile LU factorization of the matrix.

98.2 Arguments

```
N         int (IN)
          The order of the matrix A. N >= 0.

NRHS      int (IN)
          The number of right hand sides, i.e., the number of columns of the matrix B. NRHS  ↩
              >= 0.

A         float* (IN)
          The tile factor L from the factorization, computed by PLASMA_sgetrf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,N).

L         float* (IN)
          Auxiliary factorization data, related to the tile L factor, computed by  ↩
              PLASMA_sgetrf.

IPIV      int* (IN)
          The pivot indices from PLASMA_sgetrf (not equivalent to LAPACK).

B         float* (INOUT)
          On entry, the N-by-NRHS right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB       float* (IN)
          The leading dimension of the array B. LDB >= max(1,N).
```

98.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

98.4 C Bindings

```
int PLASMA_strsmpl(int N, int NRHS, float *A, int LDA, float *L, int *IPIV, float *B, int  ↩
    LDB)
```

98.5 Fortran Bindings

```
PLASMA_STRSMPL(INTEGER N, INTEGER NRHS, REAL A, INTEGER LDA, INTEGER LH, INTEGER IPIVH,  ↩
    REAL B, INTEGER LDB, INTEGER INFO)
```

98.6 Online Browsing

Dive into PLASMA_strsmpl

99 PLASMA_strsmpl_Tile

99.1 Purpose

PLASMA_strsmpl_Tile - Performs the forward substitution step of solving a system of linear equations after the tile LU factorization of the matrix. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

99.2 Arguments

```
A         float* (IN)
          The tile factor L from the factorization, computed by PLASMA_sgetrf.

L         float* (IN)
          Auxiliary factorization data, related to the tile L factor, computed by  ↩
              PLASMA_sgetrf.

IPIV      int* (IN)
          The pivot indices from PLASMA_sgetrf (not equivalent to LAPACK).

B         float* (INOUT)
          On entry, the N-by-NRHS right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

99.3 Return Value:

```
           = 0: successful exit
```

99.4 C Bindings

```
int PLASMA_strsmpl_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

99.5 Fortran Bindings

```
PLASMA_STRSMPL_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

99.6 Online Browsing

Dive into PLASMA_strsmpl_Tile

100 PLASMA_strsm_Tile

100.1 Purpose

PLASMA_strsm_Tile - Computes triangular solve A*X = B or X*A = B All matrices are passed through descriptors. All dimensions are taken from the descriptors.

100.2 Arguments

```
side     PLASMA_enum (IN)
         Specifies whether A appears on the left or on the right of X:
         = PlasmaLeft:  A*X = B
         = PlasmaRight: X*A = B

uplo     PLASMA_enum (IN)
         Specifies whether the matrix A is upper triangular or lower triangular:
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

transA   PLASMA_enum (IN)
         Specifies whether the matrix A is transposed, not transposed or conjugate  ↩
             transposed:
         = PlasmaNoTrans:   A is transposed;
         = PlasmaTrans:     A is not transposed;
         = PlasmaTrans: A is conjugate transposed.

diag     PLASMA_enum (IN)
         Specifies whether or not A is unit triangular:
         = PlasmaNonUnit: A is non unit;
         = PlasmaUnit:    A us unit.

A        float* (IN)
         The triangular matrix A. If uplo = PlasmaUpper, the leading N-by-N upper  ↩
             triangular
         part of the array A contains the upper triangular matrix, and the strictly lower
         triangular part of A is not referenced. If uplo = PlasmaLower, the leading N-by-N
         lower triangular part of the array A contains the lower triangular matrix, and the
         strictly upper triangular part of A is not referenced. If diag = PlasmaUnit, the
         diagonal elements of A are also not referenced and are assumed to be 1.

B        float* (INOUT)
         On entry, the N-by-NRHS right hand side matrix B.
         On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

100.3 Return Value:

```
          = 0: successful exit
```

100.4 C Bindings

```
int PLASMA_strsm_Tile(PLASMA_enum side, PLASMA_enum uplo, PLASMA_enum transA, PLASMA_enum  ↩
    diag, PLASMA_desc *A, PLASMA_desc *B)
```

100.5 Fortran Bindings

```
PLASMA_STRSM_TILE(INTEGER side, INTEGER uplo, INTEGER transA, INTEGER diag, INTEGER*4 A,  ↩
    INTEGER*4 B, INTEGER INFO)
```

100.6 Online Browsing

Dive into PLASMA_strsm_Tile

101 PLASMA_Lapack_to_Tile

101.1 Purpose

PLASMA_Lapack_to_Tile - Conversion from LAPACK layout to tile layout.

101.2 Arguments

```
Af77     void* (IN)
         LAPACK matrix.

LDA      int (IN)
         The leading dimension of the matrix Af77.

A        PLASMA_desc* (OUT)
         Descriptor of the PLASMA matrix in tile layout.
```

101.3 Return Value:

```
          = PLASMA_SUCCESS: successful exit
```

101.4 C Bindings

```
int PLASMA_Lapack_to_Tile(void *Af77, int LDA, PLASMA_desc *A)
```

101.5 Fortran Bindings

```
PLASMA_LAPACK_TO_TILE(INTEGER*4 Af77, INTEGER LDA, INTEGER*4 A, INTEGER INFO)
```

101.6 Online Browsing

Dive into PLASMA_Lapack_to_Tile

102 PLASMA_Tile_to_Lapack

102.1 Purpose

PLASMA_Tile_to_Lapack - Conversion from tile layout to LAPACK layout

102.2 Arguments

```
A        PLASMA_desc* (OUT)
         Descriptor of the PLASMA matrix in tile layout.

Af77     void* (IN)
         LAPACK matrix.

LDA      int (IN)
         The leading dimension of the matrix Af77.
```

102.3 Return Value:

```
          = PLASMA_SUCCESS: successful exit
```

102.4 C Bindings

```
int PLASMA_Tile_to_Lapack(PLASMA_desc *A, void *Af77, int LDA)
```

102.5 Fortran Bindings

```
PLASMA_TILE_TO_LAPACK(INTEGER*4 A, INTEGER*4 Af77, INTEGER LDA, INTEGER INFO)
```

102.6 Online Browsing

Dive into PLASMA_Tile_to_Lapack

103 PLASMA_Dealloc_Handle

103.1 Purpose

PLASMA_Dealloc_Handle - Deallocate workspace handle allocated by any workspace allocation routine.

103.2 Arguments

```
handle   void** (IN)
         Workspace handle
```

103.3 Return Value:

```
          = PLASMA_SUCCESS: successful exit
```

103.4 C Bindings

```

```

103.5 Fortran Bindings

```
PLASMA_DEALLOC_HANDLE(INTEGER sp, INTEGER INFO)
```

103.6 Online Browsing

Dive into PLASMA_Dealloc_Handle

104 PLASMA_Alloc_Workspace_cgels

104.1 Purpose

PLASMA_Alloc_Workspace_cgels - Allocates workspace for PLASMA_cgels or PLASMA_cgels_Tile routine.

104.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

T         PLASMA_Complex32_t* (OUT)
          On exit, workspace handle for storage of the extra T factors required by the tile  ↩
              QR
          or the tile LQ factorization.
```

104.3 Return Value:

```
           = 0: successful exit
```

104.4 C Bindings

```
int PLASMA_Alloc_Workspace_cgels(int M, int N, PLASMA_Complex32_t **T)
```

104.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_CGELS(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

104.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_cgels

105 PLASMA_Alloc_Workspace_cgeqrf

105.1 Purpose

PLASMA_Alloc_Workspace_cgeqrf - Allocates workspace for PLASMA_cgeqrf or PLASMA_cgeqrf_Tile routine.

105.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

T         PLASMA_Complex32_t* (OUT)
          On exit, workspace handle for storage of the extra T factors required by the tile  ↩
              QR
          factorization.
```

105.3 Return Value:

```
           = 0: successful exit
```

105.4 C Bindings

```
int PLASMA_Alloc_Workspace_cgeqrf(int M, int N, PLASMA_Complex32_t **T)
```

105.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_CGEQRF(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

105.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_cgeqrf

106 PLASMA_Alloc_Workspace_cgelqf

106.1 Purpose

PLASMA_Alloc_Workspace_cgelqf - Allocates workspace for PLASMA_cgelqf or PLASMA_cgelqf_Tile routines.

106.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

T         PLASMA_Complex32_t* (OUT)
          On exit, workspace handle for storage of the extra T factors required by the tile  ↩
              LQ
          factorization.
```

106.3 Return Value:

```
          = 0: successful exit
```

106.4 C Bindings

```
int PLASMA_Alloc_Workspace_cgelqf(int M, int N, PLASMA_Complex32_t **T)
```

106.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_CGELQF(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

106.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_cgelqf

107 PLASMA_Alloc_Workspace_cgesv

107.1 Purpose

PLASMA_Alloc_Workspace_cgesv - Allocates workspace for PLASMA_cgesv or PLASMA_cgesv_Tile routines.

107.2 Arguments

```
N         int (IN)
          The number of linear equations, i.e., the order of the matrix A. N >= 0.

L         PLASMA_Complex32_t* (OUT)
          On exit, workspace handle for storage of the extra L factors required by the tile  ↩
              LU
          factorization.

IPIV      int** (OUT)
          On exit, workspace handle for storage of pivot indexes required by the tile LU
          factorization (not equivalent to LAPACK).
```

107.3 Return Value:

```
          = 0: successful exit
```

107.4 C Bindings

```
int PLASMA_Alloc_Workspace_cgesv(int N, PLASMA_Complex32_t **L, int **IPIV)
```

107.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_CGESV(INTEGER N, INTEGER L, INTEGER IPIV, INTEGER INFO)
```

107.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_cgesv

108 PLASMA_Alloc_Workspace_cgetrf

108.1 Purpose

PLASMA_Alloc_Workspace_cgetrf - Allocates workspace for PLASMA_cgetrf or PLASMA_cgetrf_Tile routines.

108.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

L         PLASMA_Complex32_t* (OUT)
          On exit, workspace handle for storage of the extra L factors required by the tile  ↩
              LU
          factorization.

IPIV      int** (OUT)
          On exit, workspace handle for storage of pivot indexes required by the tile LU
          factorization (not equivalent to LAPACK).
```

108.3 Return Value:

```
           = 0: successful exit
```

108.4 C Bindings

```
int PLASMA_Alloc_Workspace_cgetrf(int M, int N, PLASMA_Complex32_t **L, int **IPIV)
```

108.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_CGETRF(INTEGER M, INTEGER N, INTEGER L, INTEGER IPIV, INTEGER INFO)
```

108.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_cgetrf

109 PLASMA_Alloc_Workspace_dgels

109.1 Purpose

PLASMA_Alloc_Workspace_dgels - Allocates workspace for PLASMA_dgels or PLASMA_dgels_Tile routine.

109.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

T         double* (OUT)
          On exit, workspace handle for storage of the extra T factors required by the tile  ↩
              QR
          or the tile LQ factorization.
```

109.3 Return Value:

```
           = 0: successful exit
```

109.4 C Bindings

```
int PLASMA_Alloc_Workspace_dgels(int M, int N, double **T)
```

109.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_DGELS(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

109.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_dgels

110 PLASMA_Alloc_Workspace_dgeqrf

110.1 Purpose

PLASMA_Alloc_Workspace_dgeqrf - Allocates workspace for PLASMA_dgeqrf or PLASMA_dgeqrf_Tile routine.

110.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

T         double* (OUT)
          On exit, workspace handle for storage of the extra T factors required by the tile  ↩
              QR
          factorization.
```

110.3 Return Value:

```
           = 0: successful exit
```

110.4 C Bindings

```
int PLASMA_Alloc_Workspace_dgeqrf(int M, int N, double **T)
```

110.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_DGEQRF(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

110.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_dgeqrf

111 PLASMA_Alloc_Workspace_dgelqf

111.1 Purpose

PLASMA_Alloc_Workspace_dgelqf - Allocates workspace for PLASMA_dgelqf or PLASMA_dgelqf_Tile routines.

111.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

T         double* (OUT)
          On exit, workspace handle for storage of the extra T factors required by the tile  ↩
              LQ
          factorization.
```

111.3 Return Value:

```
          = 0: successful exit
```

111.4 C Bindings

```
int PLASMA_Alloc_Workspace_dgelqf(int M, int N, double **T)
```

111.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_DGELQF(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

111.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_dgelqf

112 PLASMA_Alloc_Workspace_dgesv

112.1 Purpose

PLASMA_Alloc_Workspace_dgesv - Allocates workspace for PLASMA_dgesv or PLASMA_dgesv_Tile routines.

112.2 Arguments

```
N         int (IN)
          The number of linear equations, i.e., the order of the matrix A. N >= 0.

L         double* (OUT)
          On exit, workspace handle for storage of the extra L factors required by the tile  ↩
              LU
          factorization.

IPIV      int** (OUT)
          On exit, workspace handle for storage of pivot indexes required by the tile LU
          factorization (not equivalent to LAPACK).
```

112.3 Return Value:

```
          = 0: successful exit
```

112.4 C Bindings

```
int PLASMA_Alloc_Workspace_dgesv(int N, double **L, int **IPIV)
```

112.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_DGESV(INTEGER N, INTEGER L, INTEGER IPIV, INTEGER INFO)
```

112.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_dgesv

113 PLASMA_Alloc_Workspace_dgetrf

113.1 Purpose

PLASMA_Alloc_Workspace_dgetrf - Allocates workspace for PLASMA_dgetrf or PLASMA_dgetrf_Tile routines.

113.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

L         double* (OUT)
          On exit, workspace handle for storage of the extra L factors required by the tile  ↩
              LU
          factorization.

IPIV      int** (OUT)
          On exit, workspace handle for storage of pivot indexes required by the tile LU
          factorization (not equivalent to LAPACK).
```

113.3 Return Value:

```
           = 0: successful exit
```

113.4 C Bindings

```
int PLASMA_Alloc_Workspace_dgetrf(int M, int N, double **L, int **IPIV)
```

113.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_DGETRF(INTEGER M, INTEGER N, INTEGER L, INTEGER IPIV, INTEGER INFO)
```

113.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_dgetrf

114 PLASMA_Alloc_Workspace_sgels

114.1 Purpose

PLASMA_Alloc_Workspace_sgels - Allocates workspace for PLASMA_sgels or PLASMA_sgels_Tile routine.

114.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

T         float* (OUT)
          On exit, workspace handle for storage of the extra T factors required by the tile QR
          or the tile LQ factorization.
```

114.3 Return Value:

```
           = 0: successful exit
```

114.4 C Bindings

```
int PLASMA_Alloc_Workspace_sgels(int M, int N, float **T)
```

114.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_SGELS(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

114.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_sgels

115 PLASMA_Alloc_Workspace_sgeqrf

115.1 Purpose

PLASMA_Alloc_Workspace_sgeqrf - Allocates workspace for PLASMA_sgeqrf or PLASMA_sgeqrf_Tile routine.

115.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

T         float* (OUT)
          On exit, workspace handle for storage of the extra T factors required by the tile  ↩
              QR
          factorization.
```

115.3 Return Value:

```
           = 0: successful exit
```

115.4 C Bindings

```
int PLASMA_Alloc_Workspace_sgeqrf(int M, int N, float **T)
```

115.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_SGEQRF(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

115.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_sgeqrf

116 PLASMA_Alloc_Workspace_sgelqf

116.1 Purpose

PLASMA_Alloc_Workspace_sgelqf - Allocates workspace for PLASMA_sgelqf or PLASMA_sgelqf_Tile routines.

116.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

T         float* (OUT)
          On exit, workspace handle for storage of the extra T factors required by the tile  ↩
              LQ
          factorization.
```

116.3 Return Value:

```
          = 0: successful exit
```

116.4 C Bindings

```
int PLASMA_Alloc_Workspace_sgelqf(int M, int N, float **T)
```

116.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_SGELQF(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

116.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_sgelqf

117 PLASMA_Alloc_Workspace_sgesv

117.1 Purpose

PLASMA_Alloc_Workspace_sgesv - Allocates workspace for PLASMA_sgesv or PLASMA_sgesv_Tile routines.

117.2 Arguments

```
N         int (IN)
          The number of linear equations, i.e., the order of the matrix A. N >= 0.

L         float* (OUT)
          On exit, workspace handle for storage of the extra L factors required by the tile  ↩
              LU
          factorization.

IPIV      int** (OUT)
          On exit, workspace handle for storage of pivot indexes required by the tile LU
          factorization (not equivalent to LAPACK).
```

117.3 Return Value:

```
          = 0: successful exit
```

117.4 C Bindings

```
int PLASMA_Alloc_Workspace_sgesv(int N, float **L, int **IPIV)
```

117.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_SGESV(INTEGER N, INTEGER L, INTEGER IPIV, INTEGER INFO)
```

117.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_sgesv

118 PLASMA_Alloc_Workspace_sgetrf

118.1 Purpose

PLASMA_Alloc_Workspace_sgetrf - Allocates workspace for PLASMA_sgetrf or PLASMA_sgetrf_Tile routines.

118.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

L         float* (OUT)
          On exit, workspace handle for storage of the extra L factors required by the tile  ↩
              LU
          factorization.

IPIV      int** (OUT)
          On exit, workspace handle for storage of pivot indexes required by the tile LU
          factorization (not equivalent to LAPACK).
```

118.3 Return Value:

```
           = 0: successful exit
```

118.4 C Bindings

```
int PLASMA_Alloc_Workspace_sgetrf(int M, int N, float **L, int **IPIV)
```

118.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_SGETRF(INTEGER M, INTEGER N, INTEGER L, INTEGER IPIV, INTEGER INFO)
```

118.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_sgetrf

119 PLASMA_Alloc_Workspace_zgels

119.1 Purpose

PLASMA_Alloc_Workspace_zgels - Allocates workspace for PLASMA_zgels or PLASMA_zgels_Tile routine.

119.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

T         PLASMA_Complex64_t* (OUT)
          On exit, workspace handle for storage of the extra T factors required by the tile  ↩
              QR
          or the tile LQ factorization.
```

119.3 Return Value:

```
           = 0: successful exit
```

119.4 C Bindings

```
int PLASMA_Alloc_Workspace_zgels(int M, int N, PLASMA_Complex64_t **T)
```

119.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_ZGELS(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

119.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_zgels

120 PLASMA_Alloc_Workspace_zgeqrf

120.1 Purpose

PLASMA_Alloc_Workspace_zgeqrf - Allocates workspace for PLASMA_zgeqrf or PLASMA_zgeqrf_Tile routine.

120.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

T         PLASMA_Complex64_t* (OUT)
          On exit, workspace handle for storage of the extra T factors required by the tile  ↩
              QR
          factorization.
```

120.3 Return Value:

```
           = 0: successful exit
```

120.4 C Bindings

```
int PLASMA_Alloc_Workspace_zgeqrf(int M, int N, PLASMA_Complex64_t **T)
```

120.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_ZGEQRF(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

120.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_zgeqrf

121 PLASMA_Alloc_Workspace_zgelqf

121.1 Purpose

PLASMA_Alloc_Workspace_zgelqf - Allocates workspace for PLASMA_zgelqf or PLASMA_zgelqf_Tile routines.

121.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

T         PLASMA_Complex64_t* (OUT)
          On exit, workspace handle for storage of the extra T factors required by the tile  ↩
              LQ
          factorization.
```

121.3 Return Value:

```
          = 0: successful exit
```

121.4 C Bindings

```
int PLASMA_Alloc_Workspace_zgelqf(int M, int N, PLASMA_Complex64_t **T)
```

121.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_ZGELQF(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

121.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_zgelqf

122 PLASMA_Alloc_Workspace_zgesv

122.1 Purpose

PLASMA_Alloc_Workspace_zgesv - Allocates workspace for PLASMA_zgesv or PLASMA_zgesv_Tile routines.

122.2 Arguments

```
N         int (IN)
          The number of linear equations, i.e., the order of the matrix A. N >= 0.

L         PLASMA_Complex64_t* (OUT)
          On exit, workspace handle for storage of the extra L factors required by the tile  ↩
              LU
          factorization.

IPIV      int** (OUT)
          On exit, workspace handle for storage of pivot indexes required by the tile LU
          factorization (not equivalent to LAPACK).
```

122.3 Return Value:

```
          = 0: successful exit
```

122.4 C Bindings

```
int PLASMA_Alloc_Workspace_zgesv(int N, PLASMA_Complex64_t **L, int **IPIV)
```

122.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_ZGESV(INTEGER N, INTEGER L, INTEGER IPIV, INTEGER INFO)
```

122.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_zgesv

123 PLASMA_Alloc_Workspace_zgetrf

123.1 Purpose

PLASMA_Alloc_Workspace_zgetrf - Allocates workspace for PLASMA_zgetrf or PLASMA_zgetrf_Tile routines.

123.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

L         PLASMA_Complex64_t* (OUT)
          On exit, workspace handle for storage of the extra L factors required by the tile  ↩
              LU
          factorization.

IPIV      int** (OUT)
          On exit, workspace handle for storage of pivot indexes required by the tile LU
          factorization (not equivalent to LAPACK).
```

123.3 Return Value:

```
           = 0: successful exit
```

123.4 C Bindings

```
int PLASMA_Alloc_Workspace_zgetrf(int M, int N, PLASMA_Complex64_t **L, int **IPIV)
```

123.5 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_ZGETRF(INTEGER M, INTEGER N, INTEGER L, INTEGER IPIV, INTEGER INFO)
```

123.6 Online Browsing

Dive into PLASMA_Alloc_Workspace_zgetrf

124 PLASMA_zcgesv

124.1 Purpose

PLASMA_zcgesv - Computes the solution to a system of linear equations A **X = B, where A is an N-by-N matrix and X and B are N-by-NRHS matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A. The factored form of A is then used to solve the system of equations A** X = B.

IMPORTANT NOTICE: in its current state, this routine only intends to be a proof-of-concept. There are still some costly serial parts and one may NOT expect to achieve high performance.

PLASMA_zcgesv first attempts to factorize the matrix in COMPLEX and use this factorization within an iterative refinement procedure to produce a solution with COMPLEX*16 normwise backward error quality (see below). If the approach fails the method switches to a COMPLEX*16 factorization and solve.

The iterative refinement is not going to be a winning strategy if the ratio COMPLEX performance over COMPLEX*16 performance is too small. A reasonable strategy should take the number of right-hand sides and the size of the matrix into account. This might be done with a call to ILAENV in the future. Up to now, we always try iterative refinement.

The iterative refinement process is stopped if ITER > ITERMAX or for all the RHS we have: RNRM < SQRT(N)*XNRM*ANRM*EPS* where

- ITER is the number of the current iteration in the iterative refinement process
- RNRM is the infinity-norm of the residual
- XNRM is the infinity-norm of the solution
- ANRM is the infinity-operator-norm of the matrix A
- EPS is the machine epsilon returned by DLAMCH(*Epsilon*).

The values ITERMAX and BWDMAX are fixed to 30 and 1.0D+00 respectively.

124.2 Arguments

```
N         int (IN)
          The number of linear equations, i.e., the order of the matrix A. N >= 0.

NRHS      int (IN)
          The number of right hand sides, i.e., the number of columns of the matrix B.
          NRHS >= 0.

A         PLASMA_Complex64_t* (INOUT)
          On entry, the N-by-N coefficient matrix A.
          On exit, the tile L and U factors from the factorization (not equivalent to LAPACK ↩
              ).

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,N).

L         PLASMA_Complex64_t* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          necessary to solve the system of equations.

IPIV      int* (OUT)
          On exit, the pivot indices that define the permutations (not equivalent to LAPACK) ↩
              .

B         PLASMA_Complex64_t* (IN)
```

```
          The N-by-NRHS matrix of right hand side matrix B.

LDB       int (IN)
          The leading dimension of the array B. LDB >= max(1,N).

X         PLASMA_Complex64_t* (OUT)
          If return value = 0, the N-by-NRHS solution matrix X.

LDX       int (IN)
          The leading dimension of the array B. LDX >= max(1,N).

ITER      int* (OUT)is the number of the current iteration in the iterative refinement  ↩
    process
```

124.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
           > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
                but the factor U is exactly singular, so the solution could not be computed.
```

124.4 C Bindings

```
int PLASMA_zcgesv(int N, int NRHS, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *L,  ↩
    int *IPIV, PLASMA_Complex64_t *B, int LDB, PLASMA_Complex64_t *X, int LDX, int *ITER)
```

124.5 Fortran Bindings

```
PLASMA_ZCGESV(INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA, INTEGER LH, INTEGER IPIVH ↩
    , COMPLEX*16 B, INTEGER LDB, COMPLEX*16 X, INTEGER LDX, INTEGER ITER, INTEGER INFO)
```

124.6 Online Browsing

Dive into PLASMA_zcgesv

125 PLASMA_zcgesv

125.1 Purpose

PLASMA_zcgesv - Computes the solution to a system of linear equations A **X = B, where A is an N-by-N matrix and X and B are N-by-NRHS matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A. The factored form of A is then used to solve the system of equations A** X = B. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

IMPORTANT NOTICE: in its current state, this routine only intends to be a proof-of-concept. There are still some costly serial parts and one may NOT expect to achieve high performance.

PLASMA_zcgesv first attempts to factorize the matrix in COMPLEX and use this factorization within an iterative refinement procedure to produce a solution with COMPLEX*16 normwise backward error quality (see below). If the approach fails the method switches to a COMPLEX*16 factorization and solve.

The iterative refinement is not going to be a winning strategy if the ratio COMPLEX performance over COMPLEX*16 performance is too small. A reasonable strategy should take the number of right-hand sides and the size of the matrix into account. This might be done with a call to ILAENV in the future. Up to now, we always try iterative refinement.

The iterative refinement process is stopped if ITER > ITERMAX or for all the RHS we have: RNRM < SQRT(N)*XNRM*ANRM*EPS* where

- ITER is the number of the current iteration in the iterative refinement process
- RNRM is the infinity-norm of the residual
- XNRM is the infinity-norm of the solution
- ANRM is the infinity-operator-norm of the matrix A
- EPS is the machine epsilon returned by DLAMCH(*Epsilon*).

The values ITERMAX and BWDMAX are fixed to 30 and 1.0D+00 respectively.

125.2 Arguments

```
N         int (IN)
          The number of linear equations, i.e., the order of the matrix A. N >= 0.

NRHS      int (IN)
          The number of right hand sides, i.e., the number of columns of the matrix B.
          NRHS >= 0.

A         PLASMA_Complex64_t* (INOUT)
          On entry, the N-by-N coefficient matrix A.
          On exit, the tile L and U factors from the factorization (not equivalent to LAPACK ↩
              ).

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,N).

L         PLASMA_Complex64_t* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          necessary to solve the system of equations.

IPIV      int* (OUT)
          On exit, the pivot indices that define the permutations (not equivalent to LAPACK) ↩
              .

B         PLASMA_Complex64_t* (IN)
          The N-by-NRHS matrix of right hand side matrix B.

LDB       int (IN)
          The leading dimension of the array B. LDB >= max(1,N).

X         PLASMA_Complex64_t* (OUT)
          If return value = 0, the N-by-NRHS solution matrix X.

LDX       int (IN)
          The leading dimension of the array B. LDX >= max(1,N).

ITER      int* (OUT)is the number of the current iteration in the iterative refinement  ↩
    process
```

125.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
          > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
               but the factor U is exactly singular, so the solution could not be computed.
```

125.4 C Bindings

```
int PLASMA_zcgesv(int N, int NRHS, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *L,  ↩
    int *IPIV, PLASMA_Complex64_t *B, int LDB, PLASMA_Complex64_t *X, int LDX, int *ITER)
```

125.5 Fortran Bindings

```
PLASMA_ZCGESV(INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA, INTEGER LH, INTEGER IPIVH ↩
    , COMPLEX*16 B, INTEGER LDB, COMPLEX*16 X, INTEGER LDX, INTEGER ITER, INTEGER INFO)
```

125.6 Online Browsing

Dive into PLASMA_zcgesv

126 PLASMA_zgelqf

126.1 Purpose

PLASMA_zgelqf - Computes the tile LQ factorization of a complex M-by-N matrix A: A = L * Q.

126.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A. N >= 0.

A         PLASMA_Complex64_t* (INOUT)
          On entry, the M-by-N matrix A.
          On exit, the elements on and below the diagonal of the array contain the m-by-min( ↩
              M,N)
          lower trapezoidal matrix L (L is lower triangular if M <= N); the elements above  ↩
              the
          diagonal represent the unitary matrix Q as a product of elementary reflectors,  ↩
              stored
          by tiles.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,M).

T         PLASMA_Complex64_t* (OUT)
          On exit, auxiliary factorization data, required by PLASMA_zgelqs to solve the  ↩
              system
          of equations.
```

126.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
```

126.4 C Bindings

```
int PLASMA_zgelqf(int M, int N, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *T)
```

126.5 Fortran Bindings

```
PLASMA_ZGELQF(INTEGER M, INTEGER N, COMPLEX*16 A, INTEGER LDA, INTEGER T, INTEGER INFO)
```

126.6 Online Browsing

Dive into PLASMA_zgelqf

127 PLASMA_zgelqf_Tile

127.1 Purpose

PLASMA_zgelqf_Tile - Computes the tile LQ factorization of a complex M-by-N matrix A: A = L * Q. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

127.2 Arguments

```
A         PLASMA_Complex64_t* (INOUT)
          On entry, the M-by-N matrix A.
          On exit, the elements on and below the diagonal of the array contain the m-by-min( ↩
              M,N)
          lower trapezoidal matrix L (L is lower triangular if M <= N); the elements above  ↩
              the
          diagonal represent the unitary matrix Q as a product of elementary reflectors,  ↩
              stored
          by tiles.

T         PLASMA_Complex64_t* (OUT)
          On exit, auxiliary factorization data, required by PLASMA_zgelqs to solve the  ↩
              system
          of equations.
```

127.3 Return Value:

```
          = 0: successful exit
```

127.4 C Bindings

```
int PLASMA_zgelqf_Tile(PLASMA_desc *A, PLASMA_desc *T)
```

128.5 Fortran Bindings

```
PLASMA_ZGELQS(INTEGER M, INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA, INTEGER T,  ↩
    COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

128.6 Online Browsing

Dive into PLASMA_zgelqs

129 PLASMA_zgelqs_Tile

129.1 Purpose

PLASMA_zgelqs_Tile - Compute a minimum-norm solution min || A*X - B || using the LQ factorization A = L*Q computed by PLASMA_zgelqf_Tile. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

129.2 Arguments

```
A         PLASMA_Complex64_t* (IN)
          Details of the LQ factorization of the original matrix A as returned by  ↩
              PLASMA_zgelqf.

T         PLASMA_Complex64_t* (IN)
          Auxiliary factorization data, computed by PLASMA_zgelqf.

B         PLASMA_Complex64_t* (INOUT)
          On entry, the M-by-NRHS right hand side matrix B.
          On exit, the N-by-NRHS solution matrix X.
```

129.3 Return Value:

```
           = 0: successful exit
```

129.4 C Bindings

```
int PLASMA_zgelqs_Tile(PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

129.5 Fortran Bindings

```
PLASMA_ZGELQS_TILE(INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

129.6 Online Browsing

Dive into PLASMA_zgelqs_Tile

130 PLASMA_zgels

130.1 Purpose

PLASMA_zgels - solves overdetermined or underdetermined linear systems involving an M-by-N matrix A using the QR or the LQ factorization of A. It is assumed that A has full rank. The following options are provided:

1. trans = PlasmaNoTrans and M >= N: find the least squares solution of an overdetermined system, i.e., solve the least squares problem: minimize || B - A*X ||.
2. trans = PlasmaNoTrans and M < N: find the minimum norm solution of an underdetermined system A * X = B.

Several right hand side vectors B and solution vectors X can be handled in a single call; they are stored as the columns of the M-by-NRHS right hand side matrix B and the N-by-NRHS solution matrix X.

130.2 Arguments

```
trans    PLASMA_enum (IN)
         Intended usage:
         = PlasmaNoTrans:   the linear system involves A;
         = PlasmaConjTrans: the linear system involves A**H.
         Currently only PlasmaNoTrans is supported.

M        int (IN)
         The number of rows of the matrix A. M >= 0.

N        int (IN)
         The number of columns of the matrix A. N >= 0.

NRHS     int (IN)
         The number of right hand sides, i.e., the number of columns of the matrices B and  ↩
             X.
         NRHS >= 0.

A        PLASMA_Complex64_t* (INOUT)
         On entry, the M-by-N matrix A.
         On exit,
         if M >= N, A is overwritten by details of its QR factorization as returned by
                    PLASMA_zgeqrf;
         if M < N, A is overwritten by details of its LQ factorization as returned by
                     PLASMA_zgelqf.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,M).

T        PLASMA_Complex64_t* (OUT)
         On exit, auxiliary factorization data.

B        PLASMA_Complex64_t* (INOUT)
         On entry, the M-by-NRHS matrix B of right hand side vectors, stored columnwise;
         On exit, if return value = 0, B is overwritten by the solution vectors, stored
         columnwise:
         if M >= N, rows 1 to N of B contain the least squares solution vectors; the  ↩
             residual
         sum of squares for the solution in each column is given by the sum of squares of  ↩
             the
         modulus of elements N+1 to M in that column;
         if M < N, rows 1 to N of B contain the minimum norm solution vectors;
```

```
LDB       int (IN)
          The leading dimension of the array B. LDB >= MAX(1,M,N).
```

130.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

130.4 C Bindings

```
int PLASMA_zgels(PLASMA_enum trans, int M, int N, int NRHS, PLASMA_Complex64_t *A, int LDA, ↩
     PLASMA_Complex64_t *T, PLASMA_Complex64_t *B, int LDB)
```

130.5 Fortran Bindings

```
PLASMA_ZGELS(INTEGER trans, INTEGER M, INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA,  ↩
    INTEGER T, COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

130.6 Online Browsing

Dive into PLASMA_zgels

131 PLASMA_zgels_Tile

131.1 Purpose

PLASMA_zgels_Tile - solves overdetermined or underdetermined linear systems involving an M-by-N matrix A using the QR or the LQ factorization of A. It is assumed that A has full rank. The following options are provided:

1. trans = PlasmaNoTrans and M >= N: find the least squares solution of an overdetermined system, i.e., solve the least squares problem: minimize ‖ B - A*X ‖.
2. trans = PlasmaNoTrans and M < N: find the minimum norm solution of an underdetermined system A * X = B.

Several right hand side vectors B and solution vectors X can be handled in a single call; they are stored as the columns of the M-by-NRHS right hand side matrix B and the N-by-NRHS solution matrix X. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

131.2 Arguments

```
trans     PLASMA_enum (IN)
          Intended usage:
          = PlasmaNoTrans:   the linear system involves A;
          = PlasmaConjTrans: the linear system involves A**H.
          Currently only PlasmaNoTrans is supported.

A         PLASMA_Complex64_t* (INOUT)
          On entry, the M-by-N matrix A.
          On exit,
```

```
          if M >= N, A is overwritten by details of its QR factorization as returned by
                     PLASMA_zgeqrf;
          if M < N, A is overwritten by details of its LQ factorization as returned by
                      PLASMA_zgelqf.

T         PLASMA_Complex64_t* (OUT)
          On exit, auxiliary factorization data.

B         PLASMA_Complex64_t* (INOUT)
          On entry, the M-by-NRHS matrix B of right hand side vectors, stored columnwise;
          On exit, if return value = 0, B is overwritten by the solution vectors, stored
          columnwise:
          if M >= N, rows 1 to N of B contain the least squares solution vectors; the  ↩
              residual
          sum of squares for the solution in each column is given by the sum of squares of  ↩
              the
          modulus of elements N+1 to M in that column;
          if M < N, rows 1 to N of B contain the minimum norm solution vectors;
```

131.3 Return Value:

```
           = 0: successful exit
```

131.4 C Bindings

```
int PLASMA_zgels_Tile(PLASMA_enum trans, PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

131.5 Fortran Bindings

```
PLASMA_ZGELS_TILE(INTEGER trans, INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

131.6 Online Browsing

Dive into PLASMA_zgels_Tile

132 PLASMA_zgemm

132.1 Purpose

PLASMA_zgemm - Performs one of the matrix-matrix operations

```
C = alpha*op( A )*op( B ) + beta*C,
```

where op(X) is one of

```
op( X ) = X  or op( X ) = X'
```

alpha and beta are scalars, and A, B and C are matrices, with op(A) an m by k matrix, op(B) a k by n matrix and C an m by n matrix.

132.2 Arguments

```
transA   PLASMA_enum (IN)
         Specifies whether the matrix A is transposed, not transposed or conjugate  ↩
             transposed:
         = PlasmaNoTrans:   A is transposed;
         = PlasmaTrans:     A is not transposed;
         = PlasmaConjTrans: A is conjugate transposed.
         Currently only PlasmaNoTrans is supported

transB   PLASMA_enum (IN)
         Specifies whether the matrix B is transposed, not transposed or conjugate  ↩
             transposed:
         = PlasmaNoTrans:   B is transposed;
         = PlasmaTrans:     B is not transposed;
         = PlasmaConjTrans: B is conjugate transposed.
         Currently only PlasmaNoTrans is supported

M        int (IN)
         M specifies the number of rows of the matrix op( A ) and of the matrix C. M >= 0.

N        int (IN)
         N specifies the number of columns of the matrix op( B ) and of the matrix C. N >=  ↩
             0.

K        int (IN)
         K specifies the number of columns of the matrix op( A ) and the number of rows of
         the matrix op( B ). K >= 0.

alpha    PLASMA_Complex64_t (IN)
         alpha specifies the scalar alpha

A        PLASMA_Complex64_t* (IN)
         A is a LDA-by-ka matrix, where ka is K when  transA = PlasmaNoTrans,
         and is  M  otherwise.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,M).

B        PLASMA_Complex64_t* (IN)
         B is a LDB-by-kb matrix, where kb is N when  transB = PlasmaNoTrans,
         and is  K  otherwise.

LDB      int (IN)
         The leading dimension of the array B. LDB >= max(1,N).

beta     PLASMA_Complex64_t (IN)
         beta specifies the scalar beta

C        PLASMA_Complex64_t* (INOUT)
         C is a LDC-by-N matrix.
         On exit, the array is overwritten by the M by N matrix ( alpha*op( A )*op( B ) +  ↩
             beta*C )

LDC      int (IN)
         The leading dimension of the array C. LDC >= max(1,M).
```

132.3 Return Value:

```
          = 0: successful exit
```

132.4 C Bindings

```
int PLASMA_zgemm(PLASMA_enum transA, PLASMA_enum transB, int M, int N, int K,  ↩
    PLASMA_Complex64_t alpha, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *B, int LDB ↩
    , PLASMA_Complex64_t beta, PLASMA_Complex64_t *C, int LDC)
```

132.5 Fortran Bindings

```
PLASMA_ZGEMM(INTEGER transA, INTEGER transB, INTEGER M, INTEGER N, INTEGER K, COMPLEX*16  ↩
    alpha, COMPLEX*16 A, INTEGER LDA, COMPLEX*16 B, INTEGER LDB, COMPLEX*16 beta, COMPLEX*16 ↩
     C, INTEGER LDC, INTEGER INFO)
```

132.6 Online Browsing

Dive into PLASMA_zgemm

133 PLASMA_zgeqrf

133.1 Purpose

PLASMA_zgeqrf - Computes the tile QR factorization of a complex M-by-N matrix A: A = Q * R.

133.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A.  N >= 0.

A         PLASMA_Complex64_t* (INOUT)
          On entry, the M-by-N matrix A.
          On exit, the elements on and above the diagonal of the array contain the min(M,N)- ↩
              by-N
          upper trapezoidal matrix R (R is upper triangular if M >= N); the elements below  ↩
              the
          diagonal represent the unitary matrix Q as a product of elementary reflectors  ↩
              stored
          by tiles.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,M).

T         PLASMA_Complex64_t* (OUT)
          On exit, auxiliary factorization data, required by PLASMA_zgeqrs to solve the  ↩
              system
          of equations.
```

133.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

133.4 C Bindings

```
int PLASMA_zgeqrf(int M, int N, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *T)
```

133.5 Fortran Bindings

```
PLASMA_ZGEQRF(INTEGER M, INTEGER N, COMPLEX*16 A, INTEGER LDA, INTEGER T, INTEGER INFO)
```

133.6 Online Browsing

Dive into PLASMA_zgeqrf

134 PLASMA_zgeqrf_Tile

134.1 Purpose

PLASMA_zgeqrf_Tile - Computes the tile QR factorization of a complex M-by-N matrix A: A = Q * R. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

134.2 Arguments

```
A         PLASMA_Complex64_t* (INOUT)
          On entry, the M-by-N matrix A.
          On exit, the elements on and above the diagonal of the array contain the min(M,N)- ↩
              by-N
          upper trapezoidal matrix R (R is upper triangular if M >= N); the elements below  ↩
              the
          diagonal represent the unitary matrix Q as a product of elementary reflectors  ↩
              stored
          by tiles.

T         PLASMA_Complex64_t* (OUT)
          On exit, auxiliary factorization data, required by PLASMA_zgeqrs to solve the  ↩
              system
          of equations.
```

134.3 Return Value:

```
          = 0: successful exit
```

134.4 C Bindings

```
int PLASMA_zgeqrf_Tile(PLASMA_desc *A, PLASMA_desc *T)
```

134.5 Fortran Bindings

```
PLASMA_ZGEQRF_TILE(INTEGER*4 A, INTEGER*4 T, INTEGER INFO)
```

134.6 Online Browsing

Dive into PLASMA_zgeqrf_Tile

135 PLASMA_zgeqrs

135.1 Purpose

PLASMA_zgeqrs - Compute a minimum-norm solution min || A*X - B || using the RQ factorization A = R*Q computed by PLASMA_zgeqrf.

135.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A. N >= M >= 0.

NRHS      int (IN)
          The number of columns of B. NRHS >= 0.

A         PLASMA_Complex64_t* (INOUT)
          Details of the QR factorization of the original matrix A as returned by  ↩
              PLASMA_zgeqrf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= M.

T         PLASMA_Complex64_t* (IN)
          Auxiliary factorization data, computed by PLASMA_zgeqrf.

B         PLASMA_Complex64_t* (INOUT)
          On entry, the m-by-nrhs right hand side matrix B.
          On exit, the n-by-nrhs solution matrix X.

LDB       int (IN)
          The leading dimension of the array B. LDB >= max(1,N).
```

135.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

135.4 C Bindings

```
int PLASMA_zgeqrs(int M, int N, int NRHS, PLASMA_Complex64_t *A, int LDA,  ↩
    PLASMA_Complex64_t *T, PLASMA_Complex64_t *B, int LDB)
```

135.5 Fortran Bindings

```
PLASMA_ZGEQRS(INTEGER M, INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA, INTEGER T,  ↩
    COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

135.6 Online Browsing

Dive into PLASMA_zgeqrs

136 PLASMA_zgeqrs_Tile

136.1 Purpose

PLASMA_zgeqrs_Tile - Compute a minimum-norm solution min || A*X - B || using the RQ factorization A = R*Q computed by PLASMA_zgeqrf_Tile. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

136.2 Arguments

```
A         PLASMA_Complex64_t* (INOUT)
          Details of the QR factorization of the original matrix A as returned by  ↩
              PLASMA_zgeqrf.

T         PLASMA_Complex64_t* (IN)
          Auxiliary factorization data, computed by PLASMA_zgeqrf.

B         PLASMA_Complex64_t* (INOUT)
          On entry, the m-by-nrhs right hand side matrix B.
          On exit, the n-by-nrhs solution matrix X.
```

136.3 Return Value:

```
           = 0: successful exit
```

136.4 C Bindings

```
int PLASMA_zgeqrs_Tile(PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

136.5 Fortran Bindings

```
PLASMA_ZGEQRS_TILE(INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

136.6 Online Browsing

Dive into PLASMA_zgeqrs_Tile

137 PLASMA_zgesv

137.1 Purpose

PLASMA_zgesv - Computes the solution to a system of linear equations A **X = B, where A is an N-by-N matrix and X and B are N-by-NRHS matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A. The factored form of A is then used to solve the system of equations A** X = B.

137.2 Arguments

```
N         int (IN)
          The number of linear equations, i.e., the order of the matrix A. N >= 0.

NRHS      int (IN)
          The number of right hand sides, i.e., the number of columns of the matrix B.
          NRHS >= 0.

A         PLASMA_Complex64_t* (INOUT)
          On entry, the N-by-N coefficient matrix A.
          On exit, the tile L and U factors from the factorization (not equivalent to LAPACK ↩
              ).

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,N).

L         PLASMA_Complex64_t* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          necessary to solve the system of equations.

IPIV      int* (OUT)
          On exit, the pivot indices that define the permutations (not equivalent to LAPACK) ↩
              .

B         PLASMA_Complex64_t* (INOUT)
          On entry, the N-by-NRHS matrix of right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB       int (IN)
          The leading dimension of the array B. LDB >= max(1,N).
```

137.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
           > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
                but the factor U is exactly singular, so the solution could not be computed.
```

137.4 C Bindings

```
int PLASMA_zgesv(int N, int NRHS, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *L,  ↩
    int *IPIV, PLASMA_Complex64_t *B, int LDB)
```

137.5 Fortran Bindings

```
PLASMA_ZGESV(INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA, INTEGER LH, INTEGER IPIVH, ↩
     COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

137.6 Online Browsing

Dive into PLASMA_zgesv

138 PLASMA_zgesv_Tile

138.1 Purpose

PLASMA_zgesv_Tile - Computes the solution to a system of linear equations A **X = B, where A is an N-by-N matrix and X and B are N-by-NRHS matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A. The factored form of A is then used to solve the system of equations A** X = B. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

138.2 Arguments

```
A         PLASMA_Complex64_t* (INOUT)
          On entry, the N-by-N coefficient matrix A.
          On exit, the tile L and U factors from the factorization (not equivalent to LAPACK ↩
              ).

L         PLASMA_Complex64_t* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          necessary to solve the system of equations.

IPIV      int* (OUT)
          On exit, the pivot indices that define the permutations (not equivalent to LAPACK) ↩
              .

B         PLASMA_Complex64_t* (INOUT)
          On entry, the N-by-NRHS matrix of right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

138.3 Return Value:

```
           = 0: successful exit
           > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
                but the factor U is exactly singular, so the solution could not be computed.
```

138.4 C Bindings

```
int PLASMA_zgesv_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

138.5 Fortran Bindings

```
PLASMA_ZGESV_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

138.6 Online Browsing

Dive into PLASMA_zgesv_Tile

139 PLASMA_zgetrf

139.1 Purpose

PLASMA_zgetrf - Computes an LU factorization of a general M-by-N matrix A using the tile LU algorithm with partial tile pivoting with row interchanges.

139.2 Arguments

```
M         int (IN)
          The number of rows of the matrix A. M >= 0.

N         int (IN)
          The number of columns of the matrix A. N >= 0.

A         PLASMA_Complex64_t* (INOUT)
          On entry, the M-by-N matrix to be factored.
          On exit, the tile factors L and U from the factorization.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,M).

L         PLASMA_Complex64_t* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          required by PLASMA_zgetrs to solve the system of equations.

IPIV      int* (OUT)
          The pivot indices that define the permutations (not equivalent to LAPACK).
```

139.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
           > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
                but the factor U is exactly singular, and division by zero will occur
                if it is used to solve a system of equations.
```

139.4 C Bindings

```
int PLASMA_zgetrf(int M, int N, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *L, int  ↩
    *IPIV)
```

139.5 Fortran Bindings

```
PLASMA_ZGETRF(INTEGER M, INTEGER N, COMPLEX*16 A, INTEGER LDA, INTEGER LH, INTEGER IPIVH,  ↩
    INTEGER INFO)
```

139.6 Online Browsing

Dive into PLASMA_zgetrf

140 PLASMA_zgetrf_Tile

140.1 Purpose

PLASMA_zgetrf_Tile - Computes an LU factorization of a general M-by-N matrix A using the tile LU algorithm with partial tile pivoting with row interchanges. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

140.2 Arguments

```
A         PLASMA_Complex64_t* (INOUT)
          On entry, the M-by-N matrix to be factored.
          On exit, the tile factors L and U from the factorization.

L         PLASMA_Complex64_t* (OUT)
          On exit, auxiliary factorization data, related to the tile L factor,
          required by PLASMA_zgetrs to solve the system of equations.

IPIV      int* (OUT)
          The pivot indices that define the permutations (not equivalent to LAPACK).
```

140.3 Return Value:

```
           = 0: successful exit
           > 0: if i, U(i,i) is exactly zero. The factorization has been completed,
                but the factor U is exactly singular, and division by zero will occur
                if it is used to solve a system of equations.
```

140.4 C Bindings

```
int PLASMA_zgetrf_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV)
```

140.5 Fortran Bindings

```
PLASMA_ZGETRF_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER INFO)
```

140.6 Online Browsing

Dive into PLASMA_zgetrf_Tile

141 PLASMA_zgetrs

141.1 Purpose

PLASMA_zgetrs - Solves a system of linear equations A * X = B, with a general N-by-N matrix A using the tile LU factorization computed by PLASMA_zgetrf.

141.2 Arguments

```
trans     PLASMA_enum (IN)
          Intended to specify the the form of the system of equations:
          = PlasmaNoTrans:   A * X = B     (No transpose)
          = PlasmaTrans:     A**T * X = B  (Transpose)
          = PlasmaConjTrans: A**H * X = B  (Conjugate transpose)
          Currently only PlasmaNoTrans is supported.

N         int (IN)
          The order of the matrix A.  N >= 0.

NRHS      int (IN)
          The number of right hand sides, i.e., the number of columns of the matrix B.
          NRHS >= 0.

A         PLASMA_Complex64_t* (IN)
          The tile factors L and U from the factorization, computed by PLASMA_zgetrf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,N).

L         PLASMA_Complex64_t* (IN)
          Auxiliary factorization data, related to the tile L factor, computed by  ↩
              PLASMA_zgetrf.

IPIV      int* (IN)
          The pivot indices from PLASMA_zgetrf (not equivalent to LAPACK).

B         PLASMA_Complex64_t* (INOUT)
          On entry, the N-by-NRHS matrix of right hand side matrix B.
          On exit, the solution matrix X.

LDB       int (IN)
          The leading dimension of the array B. LDB >= max(1,N).
```

141.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

141.4 C Bindings

```
int PLASMA_zgetrs(PLASMA_enum uplo, int N, int NRHS, PLASMA_Complex64_t *A, int LDA,  ↩
    PLASMA_Complex64_t *L, int *IPIV, PLASMA_Complex64_t *B, int LDB)
```

141.5 Fortran Bindings

```
PLASMA_ZGETRS(INTEGER uplo, INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA, INTEGER LH, ↩
     INTEGER IPIVH, COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

141.6 Online Browsing

Dive into PLASMA_zgetrs

142 PLASMA_zgetrs_Tile

142.1 Purpose

PLASMA_zgetrs_Tile - Solves a system of linear equations A * X = B, with a general N-by-N matrix A using the tile LU factorization computed by PLASMA_zgetrf. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

142.2 Arguments

```
A         PLASMA_Complex64_t* (IN)
          The tile factors L and U from the factorization, computed by PLASMA_zgetrf.

L         PLASMA_Complex64_t* (IN)
          Auxiliary factorization data, related to the tile L factor, computed by  ↩
              PLASMA_zgetrf.

IPIV      int* (IN)
          The pivot indices from PLASMA_zgetrf (not equivalent to LAPACK).

B         PLASMA_Complex64_t* (INOUT)
          On entry, the N-by-NRHS matrix of right hand side matrix B.
          On exit, the solution matrix X.
```

142.3 Return Value:

```
           = 0: successful exit
```

142.4 C Bindings

```
int PLASMA_zgetrs_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

142.5 Fortran Bindings

```
PLASMA_ZGETRS_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

142.6 Online Browsing

Dive into PLASMA_zgetrs_Tile

143 PLASMA_zposv

143.1 Purpose

PLASMA_zposv - Computes the solution to a system of linear equations A * X = B, where A is an N-by-N symmetric positive definite (or Hermitian positive definite in the complex case) matrix and X and B are N-by-NRHS matrices. The Cholesky decomposition is used to factor A as

```
A = U**H * U, if uplo = PlasmaUpper, or
A = L * L**H, if uplo =  PlasmaLower,
```

where U is an upper triangular matrix and L is a lower triangular matrix. The factored form of A is then used to solve the system of equations A * X = B.

143.2 Arguments

```
uplo     PLASMA_enum (IN)
         Specifies whether the matrix A is upper triangular or lower triangular:
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

N        int (IN)
         The number of linear equations, i.e., the order of the matrix A. N >= 0.

NRHS     int (IN)
         The number of right hand sides, i.e., the number of columns of the matrix B. NRHS  ↩
             >= 0.

A        PLASMA_Complex64_t* (INOUT)
         On entry, the symmetric positive definite (or Hermitian) matrix A.
         If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A
         contains the upper triangular part of the matrix A, and the strictly lower  ↩
             triangular
         part of A is not referenced.
         If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower
         triangular part of the matrix A, and the strictly upper triangular part of A is  ↩
             not
         referenced.
         On exit, if return value = 0, the factor U or L from the Cholesky factorization
         A = U**H*U or A = L*L**H.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,N).

B        PLASMA_Complex64_t* (INOUT)
         On entry, the N-by-NRHS right hand side matrix B.
         On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB      int (IN)
         The leading dimension of the array B. LDB >= max(1,N).
```

143.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
          > 0: if i, the leading minor of order i of A is not positive definite, so the
               factorization could not be completed, and the solution has not been computed ↩
                   .
```

143.4 C Bindings

```
int PLASMA_zposv(PLASMA_enum uplo, int N, int NRHS, PLASMA_Complex64_t *A, int LDA,  ↩
    PLASMA_Complex64_t *B, int LDB)
```

143.5 Fortran Bindings

```
PLASMA_ZPOSV(INTEGER uplo, INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA, COMPLEX*16 B ↩
    , INTEGER LDB, INTEGER INFO)
```

143.6 Online Browsing

Dive into PLASMA_zposv

144 PLASMA_zposv_Tile

144.1 Purpose

PLASMA_zposv_Tile - Computes the solution to a system of linear equations A * X = B, where A is an N-by-N symmetric positive definite (or Hermitian positive definite in the complex case) matrix and X and B are N-by-NRHS matrices. The Cholesky decomposition is used to factor A as

```
A = U**H * U, if uplo = PlasmaUpper, or
A = L * L**H, if uplo =  PlasmaLower,
```

where U is an upper triangular matrix and L is a lower triangular matrix. The factored form of A is then used to solve the system of equations A * X = B. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

144.2 Arguments

```
uplo     PLASMA_enum (IN)
         Specifies whether the matrix A is upper triangular or lower triangular:
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

A        PLASMA_Complex64_t* (INOUT)
         On entry, the symmetric positive definite (or Hermitian) matrix A.
         If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A
         contains the upper triangular part of the matrix A, and the strictly lower  ↩
             triangular
         part of A is not referenced.
         If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower
         triangular part of the matrix A, and the strictly upper triangular part of A is  ↩
             not
         referenced.
         On exit, if return value = 0, the factor U or L from the Cholesky factorization
         A = U**H*U or A = L*L**H.

B        PLASMA_Complex64_t* (INOUT)
         On entry, the N-by-NRHS right hand side matrix B.
         On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

144.3 Return Value:

```
          = 0: successful exit
          > 0: if i, the leading minor of order i of A is not positive definite, so the
               factorization could not be completed, and the solution has not been computed ↩
                   .
```

144.4 C Bindings

```
int PLASMA_zposv_Tile(PLASMA_enum uplo, PLASMA_desc *A, PLASMA_desc *B)
```

144.5 Fortran Bindings

```
PLASMA_ZPOSV_TILE(INTEGER uplo, INTEGER*4 A, INTEGER*4 B, INTEGER INFO)
```

144.6 Online Browsing

Dive into PLASMA_zposv_Tile

145 PLASMA_zpotrf

145.1 Purpose

PLASMA_zpotrf - Computes the Cholesky factorization of a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A. The factorization has the form

```
A = U**H * U, if uplo = PlasmaUpper, or
A = L * L**H, if uplo =  PlasmaLower,
```

where U is an upper triangular matrix and L is a lower triangular matrix.

145.2 Arguments

```
uplo     PLASMA_enum (IN)
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

N        int (IN)
         The order of the matrix A. N >= 0.

A        PLASMA_Complex64_t* (INOUT)
         On entry, the symmetric positive definite (or Hermitian) matrix A.
         If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A
         contains the upper triangular part of the matrix A, and the strictly lower  ↩
             triangular
         part of A is not referenced.
         If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower
         triangular part of the matrix A, and the strictly upper triangular part of A is  ↩
             not
         referenced.
         On exit, if return value = 0, the factor U or L from the Cholesky factorization
         A = U**H*U or A = L*L**H.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,N).
```

145.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
          > 0: if i, the leading minor of order i of A is not positive definite, so the
               factorization could not be completed, and the solution has not been computed ↩
                   .
```

145.4 C Bindings

```
int PLASMA_zpotrf(PLASMA_enum uplo, int N, PLASMA_Complex64_t *A, int LDA)
```

145.5 Fortran Bindings

```
PLASMA_ZPOTRF(INTEGER uplo, INTEGER N, COMPLEX*16 A, INTEGER LDA, INTEGER INFO)
```

145.6 Online Browsing

Dive into PLASMA_zpotrf

146 PLASMA_zpotrf_Tile

146.1 Purpose

PLASMA_zpotrf_Tile - Computes the Cholesky factorization of a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A. The factorization has the form

```
A = U**H * U, if uplo = PlasmaUpper, or
A = L * L**H, if uplo =  PlasmaLower,
```

where U is an upper triangular matrix and L is a lower triangular matrix. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

146.2 Arguments

```
uplo     PLASMA_enum (IN)
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

A        PLASMA_Complex64_t* (INOUT)
         On entry, the symmetric positive definite (or Hermitian) matrix A.
         If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A
         contains the upper triangular part of the matrix A, and the strictly lower  ↩
             triangular
         part of A is not referenced.
         If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower
         triangular part of the matrix A, and the strictly upper triangular part of A is  ↩
             not
         referenced.
         On exit, if return value = 0, the factor U or L from the Cholesky factorization
         A = U**H*U or A = L*L**H.
```

146.3 Return Value:

```
          = 0: successful exit
          > 0: if i, the leading minor of order i of A is not positive definite, so the
               factorization could not be completed, and the solution has not been computed ↩
                   .
```

146.4 C Bindings

```
int PLASMA_zpotrf_Tile(PLASMA_enum uplo, PLASMA_desc *A)
```

146.5 Fortran Bindings

```
PLASMA_ZPOTRF_TILE(INTEGER uplo, INTEGER*4 A, INTEGER INFO)
```

146.6 Online Browsing

Dive into PLASMA_zpotrf_Tile

147 PLASMA_zpotrs

147.1 Purpose

PLASMA_zpotrs - Solves a system of linear equations A * X = B with a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A using the Cholesky factorization A = U**H*U or A = L*LH computed by PLASMA_zpotrf.

147.2 Arguments

```
uplo      PLASMA_enum (IN)
          = PlasmaUpper: Upper triangle of A is stored;
          = PlasmaLower: Lower triangle of A is stored.

N         int (IN)
          The order of the matrix A. N >= 0.

NRHS      int (IN)
          The number of right hand sides, i.e., the number of columns of the matrix B. NRHS  ↩
              >= 0.

A         PLASMA_Complex64_t* (IN)
          The triangular factor U or L from the Cholesky factorization A = U**H*U or A = L*L ↩
              **H,
          computed by PLASMA_zpotrf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,N).

B         PLASMA_Complex64_t* (INOUT)
          On entry, the N-by-NRHS right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB       int (IN)
          The leading dimension of the array B. LDB >= max(1,N).
```

147.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
```

147.4 C Bindings

```
int PLASMA_zpotrs(PLASMA_enum uplo, int N, int NRHS, PLASMA_Complex64_t *A, int LDA,  ↩
    PLASMA_Complex64_t *B, int LDB)
```

147.5 Fortran Bindings

```
PLASMA_ZPOTRS(INTEGER uplo, INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA, COMPLEX*16  ↩
    B, INTEGER  LDB, INTEGER  INFO)
```

147.6 Online Browsing

Dive into PLASMA_zpotrs

148 PLASMA_zpotrs_Tile

148.1 Purpose

PLASMA_zpotrs_Tile - Solves a system of linear equations A * X = B with a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A using the Cholesky factorization A = U**H*U or A = L*LH computed by PLASMA_zpotrf. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

148.2 Arguments

```
uplo     PLASMA_enum (IN)
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

A        PLASMA_Complex64_t* (IN)
         The triangular factor U or L from the Cholesky factorization A = U**H*U or A = L*L ↩
             **H,
         computed by PLASMA_zpotrf.

B        PLASMA_Complex64_t* (INOUT)
         On entry, the N-by-NRHS right hand side matrix B.
         On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

148.3 Return Value:

```
          = 0: successful exit
```

148.4 C Bindings

```
int PLASMA_zpotrs_Tile(PLASMA_enum uplo, PLASMA_desc *A, PLASMA_desc *B)
```

148.5 Fortran Bindings

```
PLASMA_ZPOTRS_TILE(INTEGER uplo, INTEGER*4 A, INTEGER*4 B, INTEGER INFO)
```

148.6 Online Browsing

Dive into PLASMA_zpotrs_Tile

149 PLASMA_ztrsm

149.1 Purpose

PLASMA_ztrsm - Computes triangular solve A*X = B or X*A = B

149.2 Arguments

```
side     PLASMA_enum (IN)
         Specifies whether A appears on the left or on the right of X:
         = PlasmaLeft:  A*X = B
         = PlasmaRight: X*A = B

uplo     PLASMA_enum (IN)
         Specifies whether the matrix A is upper triangular or lower triangular:
         = PlasmaUpper: Upper triangle of A is stored;
         = PlasmaLower: Lower triangle of A is stored.

transA   PLASMA_enum (IN)
         Specifies whether the matrix A is transposed, not transposed or conjugate  ↩
             transposed:
         = PlasmaNoTrans:   A is transposed;
         = PlasmaTrans:     A is not transposed;
         = PlasmaConjTrans: A is conjugate transposed.

diag     PLASMA_enum (IN)
         Specifies whether or not A is unit triangular:
         = PlasmaNonUnit: A is non unit;
         = PlasmaUnit:    A us unit.

N        int (IN)
         The order of the matrix A. N >= 0.

NRHS     int (IN)
         The number of right hand sides, i.e., the number of columns of the matrix B. NRHS  ↩
             >= 0.

A        PLASMA_Complex64_t* (IN)
         The triangular matrix A. If uplo = PlasmaUpper, the leading N-by-N upper  ↩
             triangular
         part of the array A contains the upper triangular matrix, and the strictly lower
         triangular part of A is not referenced. If uplo = PlasmaLower, the leading N-by-N
```

```
          lower triangular part of the array A contains the lower triangular matrix, and the
          strictly upper triangular part of A is not referenced. If diag = PlasmaUnit, the
          diagonal elements of A are also not referenced and are assumed to be 1.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,N).

B         PLASMA_Complex64_t* (INOUT)
          On entry, the N-by-NRHS right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB       PLASMA_Complex64_t* (IN)
          The leading dimension of the array B. LDB >= max(1,N).
```

149.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

149.4 C Bindings

```
int PLASMA_ztrsm(PLASMA_enum side, PLASMA_enum uplo, PLASMA_enum transA, PLASMA_enum diag,  ↩
    int N, int NRHS, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *B, int LDB)
```

149.5 Fortran Bindings

```
PLASMA_ZTRSM(INTEGER side, INTEGER uplo, INTEGER transA, INTEGER diag, INTEGER N, INTEGER  ↩
    NRHS, COMPLEX*16 A, INTEGER LDA, COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

149.6 Online Browsing

Dive into PLASMA_ztrsm

150 PLASMA_ztrsmpl

150.1 Purpose

PLASMA_ztrsmpl - Performs the forward substitution step of solving a system of linear equations after the tile LU factorization of the matrix.

150.2 Arguments

```
N         int (IN)
          The order of the matrix A. N >= 0.

NRHS      int (IN)
          The number of right hand sides, i.e., the number of columns of the matrix B. NRHS  ↩
              >= 0.

A         PLASMA_Complex64_t* (IN)
```

```
          The tile factor L from the factorization, computed by PLASMA_zgetrf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,N).

L         PLASMA_Complex64_t* (IN)
          Auxiliary factorization data, related to the tile L factor, computed by  ↩
              PLASMA_zgetrf.

IPIV      int* (IN)
          The pivot indices from PLASMA_zgetrf (not equivalent to LAPACK).

B         PLASMA_Complex64_t* (INOUT)
          On entry, the N-by-NRHS right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB       PLASMA_Complex64_t* (IN)
          The leading dimension of the array B. LDB >= max(1,N).
```

150.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

150.4 C Bindings

```
int PLASMA_ztrsmpl(int N, int NRHS, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *L,  ↩
    int *IPIV, PLASMA_Complex64_t *B, int LDB)
```

150.5 Fortran Bindings

```
PLASMA_ZTRSMPL(INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA, INTEGER LH, INTEGER  ↩
    IPIVH, COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

150.6 Online Browsing

Dive into PLASMA_ztrsmpl

151 PLASMA_ztrsmpl_Tile

151.1 Purpose

PLASMA_ztrsmpl_Tile - Performs the forward substitution step of solving a system of linear equations after the tile LU factorization of the matrix. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

151.2 Arguments

```
A         PLASMA_Complex64_t* (IN)
          The tile factor L from the factorization, computed by PLASMA_zgetrf.

L         PLASMA_Complex64_t* (IN)
          Auxiliary factorization data, related to the tile L factor, computed by  ↩
              PLASMA_zgetrf.

IPIV      int* (IN)
          The pivot indices from PLASMA_zgetrf (not equivalent to LAPACK).

B         PLASMA_Complex64_t* (INOUT)
          On entry, the N-by-NRHS right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

151.3 Return Value:

```
           = 0: successful exit
```

151.4 C Bindings

```
int PLASMA_ztrsmpl_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

151.5 Fortran Bindings

```
PLASMA_ZTRSMPL_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

151.6 Online Browsing

Dive into PLASMA_ztrsmpl_Tile

152 PLASMA_ztrsm_Tile

152.1 Purpose

PLASMA_ztrsm_Tile - Computes triangular solve A*X = B or X*A = B All matrices are passed through descriptors. All dimensions are taken from the descriptors.

152.2 Arguments

```
side      PLASMA_enum (IN)
          Specifies whether A appears on the left or on the right of X:
          = PlasmaLeft:  A*X = B
          = PlasmaRight: X*A = B

uplo      PLASMA_enum (IN)
          Specifies whether the matrix A is upper triangular or lower triangular:
          = PlasmaUpper: Upper triangle of A is stored;
```

```
          = PlasmaLower: Lower triangle of A is stored.

transA    PLASMA_enum (IN)
          Specifies whether the matrix A is transposed, not transposed or conjugate  ↩
              transposed:
          = PlasmaNoTrans:   A is transposed;
          = PlasmaTrans:     A is not transposed;
          = PlasmaConjTrans: A is conjugate transposed.

diag      PLASMA_enum (IN)
          Specifies whether or not A is unit triangular:
          = PlasmaNonUnit: A is non unit;
          = PlasmaUnit:    A us unit.

A         PLASMA_Complex64_t* (IN)
          The triangular matrix A. If uplo = PlasmaUpper, the leading N-by-N upper  ↩
              triangular
          part of the array A contains the upper triangular matrix, and the strictly lower
          triangular part of A is not referenced. If uplo = PlasmaLower, the leading N-by-N
          lower triangular part of the array A contains the lower triangular matrix, and the
          strictly upper triangular part of A is not referenced. If diag = PlasmaUnit, the
          diagonal elements of A are also not referenced and are assumed to be 1.

B         PLASMA_Complex64_t* (INOUT)
          On entry, the N-by-NRHS right hand side matrix B.
          On exit, if return value = 0, the N-by-NRHS solution matrix X.
```

152.3 Return Value:

```
           = 0: successful exit
```

152.4 C Bindings

```
int PLASMA_ztrsm_Tile(PLASMA_enum side, PLASMA_enum uplo, PLASMA_enum transA, PLASMA_enum  ↩
    diag, PLASMA_desc *A, PLASMA_desc *B)
```

152.5 Fortran Bindings

```
PLASMA_ZTRSM_TILE(INTEGER side, INTEGER uplo, INTEGER transA, INTEGER diag, INTEGER*4 A,  ↩
    INTEGER*4 B, INTEGER INFO)
```

152.6 Online Browsing

Dive into PLASMA_ztrsm_Tile

153 PLASMA_zunglq

153.1 Purpose

PLASMA_zunglq - Generates an M-by-N matrix Q with orthonormal rows, which is defined as the first M rows of a product of the elementary reflectors returned by PLASMA_zgelqf.

153.2 Arguments

```
M         int (IN)
          The number of rows of the matrix Q. M >= 0.

N         int (IN)
          The number of columns of the matrix Q. N >= M.

K         int (IN)
          The number of rows of elementary tile reflectors whose product defines the matrix  ↩
              Q.
          M >= K >= 0.

A         PLASMA_Complex64_t* (IN)
          Details of the LQ factorization of the original matrix A as returned by  ↩
              PLASMA_zgelqf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,M).

T         PLASMA_Complex64_t* (IN)
          Auxiliary factorization data, computed by PLASMA_zgelqf.

B         PLASMA_Complex64_t* (OUT)
          On exit, the M-by-N matrix Q.

LDA       int (IN)
          The leading dimension of the array B. LDB >= max(1,M).
```

153.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

153.4 C Bindings

```
int PLASMA_zunglq(int M, int N, int K, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t * ↩
    T, PLASMA_Complex64_t *B, int LDB)
```

153.5 Fortran Bindings

```
PLASMA_ZUNGLQ(INTEGER M, INTEGER N, INTEGER K, COMPLEX*16 A, INTEGER LDA, INTEGER T,  ↩
    COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

153.6 Online Browsing

Dive into PLASMA_zunglq

154 PLASMA_zunglq_Tile

154.1 Purpose

PLASMA_zunglq_Tile - Generates an M-by-N matrix Q with orthonormal rows, which is defined as the first M rows of a product of the elementary reflectors returned by PLASMA_zgelqf_Tile. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

154.2 Arguments

```
A         PLASMA_Complex64_t* (IN)
          Details of the LQ factorization of the original matrix A as returned by  ↩
              PLASMA_zgelqf.

T         PLASMA_Complex64_t* (IN)
          Auxiliary factorization data, computed by PLASMA_zgelqf.

B         PLASMA_Complex64_t* (OUT)
          On exit, the M-by-N matrix Q.
```

154.3 Return Value:

```
           = 0: successful exit
```

154.4 C Bindings

```
int PLASMA_zunglq_Tile(PLASMA_desc *A, PLASMA_desc *T, PLASMA_desc *B)
```

154.5 Fortran Bindings

```

```

154.6 Online Browsing

Dive into PLASMA_zunglq_Tile

155 PLASMA_zungqr

155.1 Purpose

PLASMA_zungqr - Generates an M-by-N matrix Q with orthonormal columns, which is defined as the first N columns of a product of the elementary reflectors returned by PLASMA_zgeqrf.

155.2 Arguments

```
M         int (IN)
          The number of rows of the matrix Q. M >= 0.

N         int (IN)
          The number of columns of the matrix Q. N >= M.

K         int (IN)
          The number of columns of elementary tile reflectors whose product defines the  ↩
              matrix Q.
          M >= K >= 0.

A         PLASMA_Complex64_t* (IN)
          Details of the QR factorization of the original matrix A as returned by  ↩
              PLASMA_zgeqrf.

LDA       int (IN)
          The leading dimension of the array A. LDA >= max(1,M).

T         PLASMA_Complex64_t* (IN)
          Auxiliary factorization data, computed by PLASMA_zgeqrf.

B         PLASMA_Complex64_t* (OUT)
          On exit, the M-by-N matrix Q.

LDB       int (IN)
          The leading dimension of the array B. LDB >= max(1,M).
```

155.3 Return Value:

```
           = 0: successful exit
           < 0: if -i, the i-th argument had an illegal value
```

155.4 C Bindings

```
int PLASMA_zungqr(int M, int N, int K, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t * ↩
    T, PLASMA_Complex64_t *B, int LDB)
```

155.5 Fortran Bindings

```
PLASMA_ZUNGQR(INTEGER M, INTEGER N, INTEGER K, COMPLEX*16 A, INTEGER LDA, INTEGER T,  ↩
    COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

155.6 Online Browsing

Dive into PLASMA_zungqr

156 PLASMA_zungqr_Tile

156.1 Purpose

PLASMA_zungqr_Tile - Generates an M-by-N matrix Q with orthonormal columns, which is defined as the first N columns of a product of the elementary reflectors returned by PLASMA_zgeqrf_Tile. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

156.2 Arguments

```
A         PLASMA_Complex64_t* (IN)
          Details of the QR factorization of the original matrix A as returned by  ↩
              PLASMA_zgeqrf.

T         PLASMA_Complex64_t* (IN)
          Auxiliary factorization data, computed by PLASMA_zgeqrf.

B         PLASMA_Complex64_t* (OUT)
          On exit, the M-by-N matrix Q.
```

156.3 Return Value:

```
           = 0: successful exit
```

156.4 C Bindings

```
int PLASMA_zungqr_Tile(PLASMA_desc *A, PLASMA_desc *T, PLASMA_desc *B)
```

156.5 Fortran Bindings

```
PLASMA_ZUNGQR_TILE(INTEGER*4 A, INTEGER*4 T, INTEGER*4 B, INTEGER INFO)
```

156.6 Online Browsing

Dive into PLASMA_zungqr_Tile

157 PLASMA_zunmlq

157.1 Purpose

PLASMA_zunmlq - overwrites the general M-by-N matrix C with Q*C, where Q is an orthogonal matrix (unitary in the complex case) defined as the product of elementary reflectors returned by PLASMA_zgelqf. Q is of order M.

157.2 Arguments

```
side     PLASMA_enum (IN)
         Intended usage:
         = PlasmaLeft:  apply Q or Q**H from the left;
         = PlasmaRight: apply Q or Q**H from the right.
         Currently only PlasmaLeft is supported.

trans    PLASMA_enum (IN)
         Intended usage:
         = PlasmaNoTrans:   no transpose, apply Q;
         = PlasmaConjTrans: conjugate transpose, apply Q**H.
         Currently only PlasmaConjTrans is supported.

M        int (IN)
         The number of rows of the matrix C. M >= 0.

N        int (IN)
         The number of columns of the matrix C. N >= 0.

K        int (IN)
         The number of rows of elementary tile reflectors whose product defines the matrix  ↩
             Q.
         M >= K >= 0.

A        PLASMA_Complex64_t* (IN)
         Details of the LQ factorization of the original matrix A as returned by  ↩
             PLASMA_zgelqf.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,K).

T        PLASMA_Complex64_t* (IN)
         Auxiliary factorization data, computed by PLASMA_zgelqf.

B        PLASMA_Complex64_t* (INOUT)
         On entry, the M-by-N matrix B.
         On exit, B is overwritten by Q*B or Q**H*B.

LDB      int (IN)
         The leading dimension of the array C. LDC >= max(1,M).
```

157.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
```

157.4 C Bindings

```
int PLASMA_zunmlq(PLASMA_enum side, PLASMA_enum trans, int M, int N, int K,  ↩
    PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *T, PLASMA_Complex64_t *B, int LDB)
```

157.5 Fortran Bindings

```
PLASMA_ZUNMLQ(INTEGER side, INTEGER trans, INTEGER M, INTEGER N, INTEGER K, COMPLEX*16 A,  ↩
    INTEGER LDA, INTEGER T, COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

157.6 Online Browsing

Dive into PLASMA_zunmlq

158 PLASMA_zunmlq_Tile

158.1 Purpose

PLASMA_zunmlq_Tile - overwrites the general M-by-N matrix C with Q*C, where Q is an orthogonal matrix (unitary in the complex case) defined as the product of elementary reflectors returned by PLASMA_zgelqf_Tile Q is of order M. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

158.2 Arguments

```
side     PLASMA_enum (IN)
         Intended usage:
         = PlasmaLeft:  apply Q or Q**H from the left;
         = PlasmaRight: apply Q or Q**H from the right.
         Currently only PlasmaLeft is supported.

trans    PLASMA_enum (IN)
         Intended usage:
         = PlasmaNoTrans:   no transpose, apply Q;
         = PlasmaConjTrans: conjugate transpose, apply Q**H.
         Currently only PlasmaConjTrans is supported.

A        PLASMA_Complex64_t* (IN)
         Details of the LQ factorization of the original matrix A as returned by  ↩
             PLASMA_zgelqf.

T        PLASMA_Complex64_t* (IN)
         Auxiliary factorization data, computed by PLASMA_zgelqf.

B        PLASMA_Complex64_t* (INOUT)
         On entry, the M-by-N matrix B.
         On exit, B is overwritten by Q*B or Q**H*B.
```

158.3 Return Value:

```
          = 0: successful exit
```

158.4 C Bindings

```
int PLASMA_zunmlq_Tile(PLASMA_enum side, PLASMA_enum trans, PLASMA_desc *A, PLASMA_desc *T, ↩
     PLASMA_desc *B)
```

158.5 Fortran Bindings

```
PLASMA_ZUNMLQ_TILE(INTEGER side, INTEGER trans, INTEGER*4 A, INTEGER*4 T, INTEGER*4 B,  ↩
    INTEGER INFO)
```

158.6 Online Browsing

Dive into PLASMA_zunmlq_Tile

159 PLASMA_zunmqr

159.1 Purpose

PLASMA_zunmqr - overwrites the general M-by-N matrix C with Q*C, where Q is an orthogonal matrix (unitary in the complex case) defined as the product of elementary reflectors returned by PLASMA_zgeqrf. Q is of order M.

159.2 Arguments

```
side     PLASMA_enum (IN)
         Intended usage:
         = PlasmaLeft:  apply Q or Q**H from the left;
         = PlasmaRight: apply Q or Q**H from the right.
         Currently only PlasmaLeft is supported.

trans    PLASMA_enum (IN)
         Intended usage:
         = PlasmaNoTrans:   no transpose, apply Q;
         = PlasmaConjTrans: conjugate transpose, apply Q**H.
         Currently only PlasmaConjTrans is supported.

M        int (IN)
         The number of rows of the matrix C. M >= 0.

N        int (IN)
         The number of columns of the matrix C. N >= 0.

K        int (IN)
         The number of columns of elementary tile reflectors whose product defines the  ↩
             matrix Q.
         M >= K >= 0.

A        PLASMA_Complex64_t* (IN)
         Details of the QR factorization of the original matrix A as returned by  ↩
             PLASMA_zgeqrf.

LDA      int (IN)
         The leading dimension of the array A. LDA >= max(1,M);

T        PLASMA_Complex64_t* (IN)
         Auxiliary factorization data, computed by PLASMA_zgeqrf.

B        PLASMA_Complex64_t* (INOUT)
         On entry, the M-by-N matrix B.
         On exit, B is overwritten by Q*B or Q**H*B.

LDB      int (IN)
         The leading dimension of the array C. LDC >= max(1,M).
```

159.3 Return Value:

```
          = 0: successful exit
          < 0: if -i, the i-th argument had an illegal value
```

159.4 C Bindings

```
int PLASMA_zunmqr(PLASMA_enum side, PLASMA_enum trans, int M, int N, int K,  ↩
    PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *T, PLASMA_Complex64_t *B, int LDB)
```

159.5 Fortran Bindings

```
PLASMA_ZUNMQR(INTEGER side, INTEGER trans, INTEGER M, INTEGER N, INTEGER K, COMPLEX*16 A,  ↩
    INTEGER LDA, INTEGER T, COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

159.6 Online Browsing

Dive into PLASMA_zunmqr

160 PLASMA_zunmqr_Tile

160.1 Purpose

PLASMA_zunmqr_Tile - overwrites the general M-by-N matrix C with Q*C, where Q is an orthogonal matrix (unitary in the complex case) defined as the product of elementary reflectors returned by PLASMA_zgeqrf_Tile Q is of order M. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

160.2 Arguments

```
side     PLASMA_enum (IN)
         Intended usage:
         = PlasmaLeft:  apply Q or Q**H from the left;
         = PlasmaRight: apply Q or Q**H from the right.
         Currently only PlasmaLeft is supported.

trans    PLASMA_enum (IN)
         Intended usage:
         = PlasmaNoTrans:   no transpose, apply Q;
         = PlasmaConjTrans: conjugate transpose, apply Q**H.
         Currently only PlasmaConjTrans is supported.

A        PLASMA_Complex64_t* (IN)
         Details of the QR factorization of the original matrix A as returned by  ↩
             PLASMA_zgeqrf.

T        PLASMA_Complex64_t* (IN)
         Auxiliary factorization data, computed by PLASMA_zgeqrf.

B        PLASMA_Complex64_t* (INOUT)
         On entry, the M-by-N matrix B.
         On exit, B is overwritten by Q*B or Q**H*B.
```

160.3 Return Value:

```
          = 0: successful exit
```

160.4 C Bindings

```
int PLASMA_zunmqr_Tile(PLASMA_enum side, PLASMA_enum trans, PLASMA_desc *A, PLASMA_desc *T, ↩
     PLASMA_desc *B)
```

160.5 Fortran Bindings

```
PLASMA_ZUNMQR_TILE(INTEGER side, INTEGER trans, INTEGER*4 A, INTEGER*4 T, INTEGER*4 B,  ↩
    INTEGER INFO)
```

160.6 Online Browsing

Dive into PLASMA_zunmqr_Tile