

gomotion

Generated by Doxygen 1.8.6

Thu Oct 27 2016 12:31:54

Contents

| | | |
|----------|--|-----------|
| 1 | Module Index | 1 |
| 1.1 | Modules | 1 |
| 2 | Namespace Index | 3 |
| 2.1 | Namespace List | 3 |
| 3 | Data Structure Index | 5 |
| 3.1 | Data Structures | 5 |
| 4 | File Index | 7 |
| 4.1 | File List | 7 |
| 5 | Module Documentation | 9 |
| 5.1 | The Go Motion Pose Mathematics Library | 9 |
| 5.2 | Trajectory Planning Algorithms | 13 |
| 6 | Namespace Documentation | 15 |
| 6.1 | gomotion Namespace Reference | 15 |
| 6.1.1 | Typedef Documentation | 21 |
| 6.1.1.1 | go_vector | 21 |
| 6.1.1.2 | JointState | 21 |
| 6.1.2 | Enumeration Type Documentation | 21 |
| 6.1.2.1 | anonymous enum | 21 |
| 6.1.2.2 | anonymous enum | 21 |
| 6.1.2.3 | anonymous enum | 22 |
| 6.1.3 | Function Documentation | 22 |
| 6.1.3.1 | go_acoses | 22 |
| 6.1.3.2 | go_asines | 22 |
| 6.1.3.3 | go_atans | 22 |
| 6.1.3.4 | go_cart_cart_add | 22 |

| | | |
|----------|--|----|
| 6.1.3.5 | go_cart_cart_angle | 22 |
| 6.1.3.6 | go_cart_cart_compare | 22 |
| 6.1.3.7 | go_cart_cart_cross | 22 |
| 6.1.3.8 | go_cart_cart_disp | 22 |
| 6.1.3.9 | go_cart_cart_dot | 22 |
| 6.1.3.10 | go_cart_cart_par | 22 |
| 6.1.3.11 | go_cart_cart_perp | 22 |
| 6.1.3.12 | go_cart_cart_pose | 22 |
| 6.1.3.13 | go_cart_cart_proj | 23 |
| 6.1.3.14 | go_cart_cart_rot | 23 |
| 6.1.3.15 | go_cart_cart_sub | 23 |
| 6.1.3.16 | go_cart_centroid | 23 |
| 6.1.3.17 | go_cart_centroidize | 23 |
| 6.1.3.18 | go_cart_cyl_convert | 23 |
| 6.1.3.19 | go_cart_is_norm | 23 |
| 6.1.3.20 | go_cart_mag | 23 |
| 6.1.3.21 | go_cart_magsq | 23 |
| 6.1.3.22 | go_cart_neg | 23 |
| 6.1.3.23 | go_cart_normal | 23 |
| 6.1.3.24 | go_cart_plane_proj | 23 |
| 6.1.3.25 | go_cart_rvec_convert | 23 |
| 6.1.3.26 | go_cart_scale_mult | 23 |
| 6.1.3.27 | go_cart_sph_convert | 23 |
| 6.1.3.28 | go_cart_trilaterate | 24 |
| 6.1.3.29 | go_cart_unit | 24 |
| 6.1.3.30 | go_cart_vector_convert | 24 |
| 6.1.3.31 | go_cart_zero | 24 |
| 6.1.3.32 | go_cbrt | 24 |
| 6.1.3.33 | go_complex_add | 24 |
| 6.1.3.34 | go_complex_arg | 24 |
| 6.1.3.35 | go_complex_cbrt | 24 |
| 6.1.3.36 | go_complex_div | 24 |
| 6.1.3.37 | go_complex_inv | 24 |
| 6.1.3.38 | go_complex_mag | 24 |
| 6.1.3.39 | go_complex_magsq | 24 |
| 6.1.3.40 | go_complex_mult | 24 |
| 6.1.3.41 | go_complex_scale | 24 |

| | | |
|----------|--|----|
| 6.1.3.42 | go_complex_sq | 24 |
| 6.1.3.43 | go_complex_sqrt | 24 |
| 6.1.3.44 | go_complex_sub | 24 |
| 6.1.3.45 | go_cubic_solve | 24 |
| 6.1.3.46 | go_cyl_cart_convert | 24 |
| 6.1.3.47 | go_cyl_sph_convert | 24 |
| 6.1.3.48 | go_dh_hom_convert | 25 |
| 6.1.3.49 | go_dh_pose_convert | 25 |
| 6.1.3.50 | go_get_singular_epsilon | 25 |
| 6.1.3.51 | go_hom_dh_convert | 25 |
| 6.1.3.52 | go_hom_hom_mult | 25 |
| 6.1.3.53 | go_hom_inv | 25 |
| 6.1.3.54 | go_hom_pose_convert | 25 |
| 6.1.3.55 | go_line_evaluate | 25 |
| 6.1.3.56 | go_line_from_planes | 25 |
| 6.1.3.57 | go_line_from_point_direction | 25 |
| 6.1.3.58 | go_line_from_points | 25 |
| 6.1.3.59 | go_line_line_compare | 25 |
| 6.1.3.60 | go_line_plane_intersect | 25 |
| 6.1.3.61 | go_line_plane_proj | 26 |
| 6.1.3.62 | go_linear_cos_sin_solve | 26 |
| 6.1.3.63 | go_link_joint_set | 26 |
| 6.1.3.64 | go_link_pose_build | 26 |
| 6.1.3.65 | go_mat3_inv | 26 |
| 6.1.3.66 | go_mat3_mat3_mult | 26 |
| 6.1.3.67 | go_mat3_vec3_mult | 26 |
| 6.1.3.68 | go_mat4_inv | 26 |
| 6.1.3.69 | go_mat4_mat4_mult | 26 |
| 6.1.3.70 | go_mat4_vec4_mult | 26 |
| 6.1.3.71 | go_mat6_inv | 26 |
| 6.1.3.72 | go_mat6_mat6_mult | 26 |
| 6.1.3.73 | go_mat6_transpose | 26 |
| 6.1.3.74 | go_mat6_vec6_mult | 26 |
| 6.1.3.75 | go_mat_cart_mult | 27 |
| 6.1.3.76 | go_mat_inv | 27 |
| 6.1.3.77 | go_mat_is_norm | 27 |
| 6.1.3.78 | go_mat_mat_mult | 27 |

| | | |
|-----------|--|----|
| 6.1.3.79 | go_mat_matrix_convert | 27 |
| 6.1.3.80 | go_mat_norm | 27 |
| 6.1.3.81 | go_mat_quat_convert | 27 |
| 6.1.3.82 | go_mat_rpy_convert | 27 |
| 6.1.3.83 | go_mat_rvec_convert | 27 |
| 6.1.3.84 | go_mat_uxz_convert | 27 |
| 6.1.3.85 | go_mat_xyz_convert | 27 |
| 6.1.3.86 | go_mat_zyx_convert | 27 |
| 6.1.3.87 | go_mat_zyz_convert | 27 |
| 6.1.3.88 | go_matrix_inv | 27 |
| 6.1.3.89 | go_matrix_matrix_add | 27 |
| 6.1.3.90 | go_matrix_matrix_copy | 27 |
| 6.1.3.91 | go_matrix_matrix_mult | 27 |
| 6.1.3.92 | go_matrix_transpose | 27 |
| 6.1.3.93 | go_matrix_vector_cross | 27 |
| 6.1.3.94 | go_matrix_vector_mult | 27 |
| 6.1.3.95 | go_plane_evaluate | 27 |
| 6.1.3.96 | go_plane_from_abcd | 28 |
| 6.1.3.97 | go_plane_from_point_line | 28 |
| 6.1.3.98 | go_plane_from_point_normal | 28 |
| 6.1.3.99 | go_plane_from_points | 28 |
| 6.1.3.100 | go_plane_plane_compare | 28 |
| 6.1.3.101 | go_point_line_distance | 28 |
| 6.1.3.102 | go_point_line_proj | 28 |
| 6.1.3.103 | go_point_plane_distance | 28 |
| 6.1.3.104 | go_point_plane_proj | 28 |
| 6.1.3.105 | go_pose_cart_mult | 28 |
| 6.1.3.106 | go_pose_dh_convert | 28 |
| 6.1.3.107 | go_pose_hom_convert | 29 |
| 6.1.3.108 | go_pose_identity | 29 |
| 6.1.3.109 | go_pose_inv | 29 |
| 6.1.3.110 | go_pose_pose_compare | 29 |
| 6.1.3.111 | go_pose_pose_interp | 29 |
| 6.1.3.112 | go_pose_pose_mult | 29 |
| 6.1.3.113 | go_pose_scale_mult | 29 |
| 6.1.3.114 | go_pose_this | 29 |
| 6.1.3.115 | go_pose_vel_mult | 29 |

| | |
|---|----|
| 6.1.3.116 go_quadratic_solve | 29 |
| 6.1.3.117 go_quartic_solve | 29 |
| 6.1.3.118 go_quat_cart_mult | 29 |
| 6.1.3.119 go_quat_identity | 29 |
| 6.1.3.120 go_quat_inv | 29 |
| 6.1.3.121 go_quat_is_norm | 29 |
| 6.1.3.122 go_quat_mag | 30 |
| 6.1.3.123 go_quat_mat_convert | 30 |
| 6.1.3.124 go_quat_matrix_convert | 30 |
| 6.1.3.125 go_quat_norm | 30 |
| 6.1.3.126 go_quat_quat_compare | 30 |
| 6.1.3.127 go_quat_quat_mult | 30 |
| 6.1.3.128 go_quat_rpy_convert | 30 |
| 6.1.3.129 go_quat_rvec_convert | 30 |
| 6.1.3.130 go_quat_scale_mult | 30 |
| 6.1.3.131 go_quat_unit | 30 |
| 6.1.3.132 go_quat_xyz_convert | 30 |
| 6.1.3.133 go_quat_zyx_convert | 30 |
| 6.1.3.134 go_quat_zyz_convert | 30 |
| 6.1.3.135 go_rpy_cart_mult | 30 |
| 6.1.3.136 go_rpy_mat_convert | 30 |
| 6.1.3.137 go_rpy_quat_convert | 30 |
| 6.1.3.138 go_rpy_rvec_convert | 30 |
| 6.1.3.139 go_rpy_xyz_convert | 30 |
| 6.1.3.140 go_rpy_zyx_convert | 30 |
| 6.1.3.141 go_rpy_zyz_convert | 30 |
| 6.1.3.142 go_rvec_cart_convert | 30 |
| 6.1.3.143 go_rvec_mat_convert | 30 |
| 6.1.3.144 go_rvec_quat_convert | 30 |
| 6.1.3.145 go_rvec_rpy_convert | 30 |
| 6.1.3.146 go_rvec_rvec_compare | 30 |
| 6.1.3.147 go_rvec_scale_mult | 31 |
| 6.1.3.148 go_rvec_xyz_convert | 31 |
| 6.1.3.149 go_rvec_zyx_convert | 31 |
| 6.1.3.150 go_rvec_zyz_convert | 31 |
| 6.1.3.151 go_set_singular_epsilon | 31 |
| 6.1.3.152 go_sincos | 31 |

| | |
|---|----|
| 6.1.3.153 go_sph_cart_convert | 31 |
| 6.1.3.154 go_sph_cyl_convert | 31 |
| 6.1.3.155 go_traj_ca_compute | 31 |
| 6.1.3.156 go_traj_ca_extend | 31 |
| 6.1.3.157 go_traj_ca_generate | 31 |
| 6.1.3.158 go_traj_ca_interp | 32 |
| 6.1.3.159 go_traj_ca_scale | 32 |
| 6.1.3.160 go_traj_ca_stop | 32 |
| 6.1.3.161 go_traj_cj_compute | 33 |
| 6.1.3.162 go_traj_cj_compute_tees_a_c | 33 |
| 6.1.3.163 go_traj_cj_compute_tees_a_nc | 33 |
| 6.1.3.164 go_traj_cj_compute_tees_na_c | 33 |
| 6.1.3.165 go_traj_cj_compute_tees_na_nc | 34 |
| 6.1.3.166 go_traj_cj_compute_the_rest | 34 |
| 6.1.3.167 go_traj_cj_extend | 34 |
| 6.1.3.168 go_traj_cj_generate | 34 |
| 6.1.3.169 go_traj_cj_interp | 34 |
| 6.1.3.170 go_traj_cj_scale | 35 |
| 6.1.3.171 go_traj_cj_stop | 35 |
| 6.1.3.172 go_tridiag_ql | 36 |
| 6.1.3.173 go_tridiag_reduce | 37 |
| 6.1.3.174 go_uxz_mat_convert | 37 |
| 6.1.3.175 go_vector_cart_convert | 37 |
| 6.1.3.176 go_xyz_mat_convert | 37 |
| 6.1.3.177 go_xyz_quat_convert | 37 |
| 6.1.3.178 go_xyz_rpy_convert | 37 |
| 6.1.3.179 go_xyz_rvec_convert | 37 |
| 6.1.3.180 go_xyz_zyx_convert | 37 |
| 6.1.3.181 go_xyz_zyz_convert | 37 |
| 6.1.3.182 go_zyx_mat_convert | 37 |
| 6.1.3.183 go_zyx_quat_convert | 37 |
| 6.1.3.184 go_zyx_rpy_convert | 37 |
| 6.1.3.185 go_zyx_rvec_convert | 37 |
| 6.1.3.186 go_zyx_xyz_convert | 37 |
| 6.1.3.187 go_zyx_zyz_convert | 37 |
| 6.1.3.188 go_zyz_mat_convert | 37 |
| 6.1.3.189 go_zyz_quat_convert | 37 |

| | | |
|-----------|---|-----------|
| 6.1.3.190 | go_zyz_rpy_convert | 37 |
| 6.1.3.191 | go_zyz_rvec_convert | 37 |
| 6.1.3.192 | go_zyz_xyz_convert | 37 |
| 6.1.3.193 | go_zyz_zyx_convert | 37 |
| 6.1.3.194 | lubksb | 38 |
| 6.1.3.195 | ludcmp | 38 |
| 6.1.3.196 | sincos | 38 |
| 7 | Data Structure Documentation | 39 |
| 7.1 | gomotion::coeff Struct Reference | 39 |
| 7.1.1 | Detailed Description | 39 |
| 7.1.2 | Field Documentation | 39 |
| 7.1.2.1 | a | 39 |
| 7.2 | gomotion::go_body Struct Reference | 39 |
| 7.2.1 | Detailed Description | 40 |
| 7.2.2 | Field Documentation | 40 |
| 7.2.2.1 | inertia | 40 |
| 7.2.2.2 | mass | 40 |
| 7.3 | gomotion::go_cart Struct Reference | 40 |
| 7.3.1 | Detailed Description | 40 |
| 7.3.2 | Field Documentation | 40 |
| 7.3.2.1 | x | 40 |
| 7.3.2.2 | y | 40 |
| 7.3.2.3 | z | 40 |
| 7.4 | gomotion::go_complex Struct Reference | 40 |
| 7.4.1 | Field Documentation | 41 |
| 7.4.1.1 | im | 41 |
| 7.4.1.2 | re | 41 |
| 7.5 | gomotion::go_cubic Struct Reference | 41 |
| 7.5.1 | Field Documentation | 41 |
| 7.5.1.1 | a | 41 |
| 7.5.1.2 | b | 41 |
| 7.5.1.3 | c | 41 |
| 7.6 | gomotion::go_cyl Struct Reference | 41 |
| 7.6.1 | Detailed Description | 42 |
| 7.6.2 | Field Documentation | 42 |
| 7.6.2.1 | r | 42 |

| | | |
|----------|-------------------------------------|----|
| 7.6.2.2 | theta | 42 |
| 7.6.2.3 | z | 42 |
| 7.7 | gomotion::go_dh Struct Reference | 42 |
| 7.7.1 | Detailed Description | 42 |
| 7.7.2 | Field Documentation | 42 |
| 7.7.2.1 | a | 42 |
| 7.7.2.2 | alpha | 42 |
| 7.7.2.3 | d | 42 |
| 7.7.2.4 | theta | 43 |
| 7.8 | gomotion::go_hom Struct Reference | 43 |
| 7.8.1 | Field Documentation | 43 |
| 7.8.1.1 | rot | 43 |
| 7.8.1.2 | tran | 43 |
| 7.9 | gomotion::go_interp Class Reference | 44 |
| 7.9.1 | Detailed Description | 45 |
| 7.9.2 | Member Typedef Documentation | 45 |
| 7.9.2.1 | add_func | 45 |
| 7.9.2.2 | eval_func | 45 |
| 7.9.3 | Member Function Documentation | 45 |
| 7.9.3.1 | add_constant | 45 |
| 7.9.3.2 | add_cubic_pdv | 45 |
| 7.9.3.3 | add_cubic_pf | 45 |
| 7.9.3.4 | add_cubic_pv | 45 |
| 7.9.3.5 | add_linear | 45 |
| 7.9.3.6 | add_quintic_pdva | 45 |
| 7.9.3.7 | add_quintic_pf | 45 |
| 7.9.3.8 | add_quintic_pva | 45 |
| 7.9.3.9 | add_quintic_pvda | 45 |
| 7.9.3.10 | calc_coeff_bc | 45 |
| 7.9.3.11 | calc_coeff_constant | 45 |
| 7.9.3.12 | calc_coeff_cubic_bc | 46 |
| 7.9.3.13 | calc_coeff_cubic_pf | 46 |
| 7.9.3.14 | calc_coeff_linear | 46 |
| 7.9.3.15 | calc_coeff_pf | 46 |
| 7.9.3.16 | calc_coeff_quintic_bc | 46 |
| 7.9.3.17 | calc_coeff_quintic_pf | 46 |
| 7.9.3.18 | eval_constant | 46 |

| | | |
|----------|--|----|
| 7.9.3.19 | eval_cubic | 46 |
| 7.9.3.20 | eval_linear | 46 |
| 7.9.3.21 | eval_quintic | 46 |
| 7.9.3.22 | init | 46 |
| 7.9.3.23 | set_here | 46 |
| 7.9.4 | Field Documentation | 46 |
| 7.9.4.1 | a | 46 |
| 7.9.4.2 | p | 46 |
| 7.10 | gomotion::go_line Struct Reference | 46 |
| 7.10.1 | Detailed Description | 47 |
| 7.10.2 | Field Documentation | 47 |
| 7.10.2.1 | direction | 47 |
| 7.10.2.2 | point | 47 |
| 7.11 | gomotion::go_link Struct Reference | 47 |
| 7.11.1 | Detailed Description | 48 |
| 7.11.2 | Field Documentation | 48 |
| 7.11.2.1 | body | 48 |
| 7.11.2.2 | dh | 48 |
| 7.11.2.3 | pk | 49 |
| 7.11.2.4 | pp | 49 |
| 7.11.2.5 | quantity | 49 |
| 7.11.2.6 | type | 49 |
| 7.11.2.7 | u | 49 |
| 7.12 | gomotion::go_mat Struct Reference | 49 |
| 7.12.1 | Detailed Description | 50 |
| 7.12.2 | Field Documentation | 50 |
| 7.12.2.1 | x | 50 |
| 7.12.2.2 | y | 50 |
| 7.12.2.3 | z | 50 |
| 7.13 | gomotion::go_matrix Struct Reference | 50 |
| 7.13.1 | Field Documentation | 50 |
| 7.13.1.1 | cols | 50 |
| 7.13.1.2 | el | 50 |
| 7.13.1.3 | elcpy | 50 |
| 7.13.1.4 | index | 51 |
| 7.13.1.5 | rows | 51 |
| 7.13.1.6 | v | 51 |

| | | |
|-----------|--|----|
| 7.14 | gomotion::go_motion_circular_params Struct Reference | 51 |
| 7.14.1 | Field Documentation | 52 |
| 7.14.1.1 | center | 52 |
| 7.14.1.2 | normal | 52 |
| 7.14.1.3 | qnormal | 52 |
| 7.14.1.4 | rstart | 52 |
| 7.14.1.5 | rtot | 52 |
| 7.14.1.6 | stotinv | 52 |
| 7.14.1.7 | thtot | 52 |
| 7.14.1.8 | turns | 52 |
| 7.14.1.9 | urcsp | 52 |
| 7.14.1.10 | zstart | 52 |
| 7.14.1.11 | ztot | 53 |
| 7.15 | gomotion::go_motion_interface Struct Reference | 53 |
| 7.15.1 | Field Documentation | 53 |
| 7.15.1.1 | _deltat | 53 |
| 7.15.1.2 | _gmq | 53 |
| 7.15.1.3 | _gms | 53 |
| 7.15.1.4 | _id | 53 |
| 7.15.1.5 | _position | 53 |
| 7.15.1.6 | _queuesize | 54 |
| 7.15.1.7 | _space | 54 |
| 7.15.1.8 | _type | 54 |
| 7.16 | gomotion::go_motion_linear_params Struct Reference | 54 |
| 7.16.1 | Detailed Description | 54 |
| 7.16.2 | Field Documentation | 54 |
| 7.16.2.1 | uquat | 54 |
| 7.16.2.2 | uvec | 55 |
| 7.17 | gomotion::go_motion_params Struct Reference | 55 |
| 7.17.1 | Field Documentation | 55 |
| 7.17.1.1 | acc | 55 |
| 7.17.1.2 | jerk | 55 |
| 7.17.1.3 | vel | 55 |
| 7.18 | gomotion::go_motion_queue Struct Reference | 55 |
| 7.18.1 | Member Function Documentation | 57 |
| 7.18.1.1 | append | 57 |
| 7.18.1.2 | append_joint | 57 |

| | | |
|-----------|---------------------|----|
| 7.18.1.3 | append_ujoint | 57 |
| 7.18.1.4 | append_world | 57 |
| 7.18.1.5 | drop_pending | 57 |
| 7.18.1.6 | erase | 57 |
| 7.18.1.7 | get_joint_number | 57 |
| 7.18.1.8 | get_type | 57 |
| 7.18.1.9 | head | 58 |
| 7.18.1.10 | here | 58 |
| 7.18.1.11 | init | 58 |
| 7.18.1.12 | interp | 58 |
| 7.18.1.13 | interp_joint | 58 |
| 7.18.1.14 | interp_world | 58 |
| 7.18.1.15 | interp_world | 58 |
| 7.18.1.16 | is_empty | 58 |
| 7.18.1.17 | last_id | 58 |
| 7.18.1.18 | number | 58 |
| 7.18.1.19 | reset | 58 |
| 7.18.1.20 | set_cycle_time | 58 |
| 7.18.1.21 | set_here | 58 |
| 7.18.1.22 | set_id | 58 |
| 7.18.1.23 | set_joint_number | 58 |
| 7.18.1.24 | set_scale | 58 |
| 7.18.1.25 | set_type | 58 |
| 7.18.1.26 | size | 58 |
| 7.18.1.27 | stop | 58 |
| 7.18.1.28 | stop | 58 |
| 7.18.1.29 | there | 58 |
| 7.18.2 | Field Documentation | 58 |
| 7.18.2.1 | _deltat | 58 |
| 7.18.2.2 | _end | 58 |
| 7.18.2.3 | _endptr | 59 |
| 7.18.2.4 | _here | 59 |
| 7.18.2.5 | _joint_num | 59 |
| 7.18.2.6 | _last_id | 59 |
| 7.18.2.7 | _number | 59 |
| 7.18.2.8 | _size | 59 |
| 7.18.2.9 | _start | 59 |

| | | |
|-----------|--|----|
| 7.18.2.10 | <code>_startptr</code> | 59 |
| 7.18.2.11 | <code>_there</code> | 59 |
| 7.18.2.12 | <code>_time</code> | 59 |
| 7.18.2.13 | <code>_timescale</code> | 59 |
| 7.18.2.14 | <code>_type</code> | 60 |
| 7.19 | <code>gomotion::go_motion_spec</code> Struct Reference | 60 |
| 7.19.1 | Member Function Documentation | 61 |
| 7.19.1.1 | <code>get_id</code> | 61 |
| 7.19.1.2 | <code>get_type</code> | 61 |
| 7.19.1.3 | <code>init</code> | 61 |
| 7.19.1.4 | <code>set_cpar</code> | 61 |
| 7.19.1.5 | <code>set_end_pose</code> | 61 |
| 7.19.1.6 | <code>set_end_position</code> | 61 |
| 7.19.1.7 | <code>set_id</code> | 61 |
| 7.19.1.8 | <code>set_lpar</code> | 61 |
| 7.19.1.9 | <code>set_rpar</code> | 61 |
| 7.19.1.10 | <code>set_time</code> | 61 |
| 7.19.1.11 | <code>set_tpar</code> | 61 |
| 7.19.1.12 | <code>set_type</code> | 61 |
| 7.19.2 | Field Documentation | 61 |
| 7.19.2.1 | <code>cj</code> | 61 |
| 7.19.2.2 | <code>cpar</code> | 61 |
| 7.19.2.3 | <code>end</code> | 62 |
| 7.19.2.4 | <code>id</code> | 62 |
| 7.19.2.5 | <code>lpar</code> | 62 |
| 7.19.2.6 | <code>par</code> | 62 |
| 7.19.2.7 | <code>start</code> | 62 |
| 7.19.2.8 | <code>totalt</code> | 62 |
| 7.19.2.9 | <code>type</code> | 62 |
| 7.19.2.10 | <code>u</code> | 62 |
| 7.19.2.11 | <code>uquat</code> | 62 |
| 7.20 | <code>gomotion::go_pk</code> Struct Reference | 62 |
| 7.20.1 | Detailed Description | 63 |
| 7.20.2 | Field Documentation | 63 |
| 7.20.2.1 | <code>base</code> | 63 |
| 7.20.2.2 | <code>d</code> | 63 |
| 7.20.2.3 | <code>platform</code> | 63 |

| | | |
|----------|---|----|
| 7.21 | gomotion::go_plane Struct Reference | 64 |
| 7.21.1 | Detailed Description | 64 |
| 7.21.2 | Field Documentation | 64 |
| 7.21.2.1 | d | 64 |
| 7.21.2.2 | normal | 64 |
| 7.22 | gomotion::go_pose Struct Reference | 64 |
| 7.22.1 | Detailed Description | 65 |
| 7.22.2 | Field Documentation | 65 |
| 7.22.2.1 | rot | 65 |
| 7.22.2.2 | tran | 65 |
| 7.23 | gomotion::go_position Struct Reference | 65 |
| 7.23.1 | Detailed Description | 66 |
| 7.23.2 | Member Function Documentation | 66 |
| 7.23.2.1 | zero_joints | 66 |
| 7.23.2.2 | zero_pose | 66 |
| 7.23.3 | Field Documentation | 66 |
| 7.23.3.1 | joint | 67 |
| 7.23.3.2 | pose | 67 |
| 7.23.3.3 | u | 67 |
| 7.24 | gomotion::go_pp Struct Reference | 67 |
| 7.24.1 | Detailed Description | 67 |
| 7.24.2 | Field Documentation | 67 |
| 7.24.2.1 | pose | 68 |
| 7.25 | gomotion::go_quadratic Struct Reference | 68 |
| 7.25.1 | Field Documentation | 68 |
| 7.25.1.1 | a | 68 |
| 7.25.1.2 | b | 68 |
| 7.26 | gomotion::go_quartic Struct Reference | 68 |
| 7.26.1 | Field Documentation | 68 |
| 7.26.1.1 | a | 68 |
| 7.26.1.2 | b | 68 |
| 7.26.1.3 | c | 68 |
| 7.26.1.4 | d | 69 |
| 7.27 | gomotion::go_quat Struct Reference | 69 |
| 7.27.1 | Detailed Description | 69 |
| 7.27.2 | Field Documentation | 69 |
| 7.27.2.1 | s | 69 |

| | | |
|-----------|--|----|
| 7.27.2.2 | x | 69 |
| 7.27.2.3 | y | 69 |
| 7.27.2.4 | z | 69 |
| 7.28 | gomotion::go_rpy Struct Reference | 69 |
| 7.28.1 | Detailed Description | 70 |
| 7.28.2 | Field Documentation | 70 |
| 7.28.2.1 | p | 70 |
| 7.28.2.2 | r | 70 |
| 7.28.2.3 | y | 70 |
| 7.29 | gomotion::go_rvec Struct Reference | 70 |
| 7.29.1 | Detailed Description | 70 |
| 7.29.2 | Field Documentation | 70 |
| 7.29.2.1 | x | 70 |
| 7.29.2.2 | y | 70 |
| 7.29.2.3 | z | 70 |
| 7.30 | gomotion::go_scale_spec Struct Reference | 71 |
| 7.30.1 | Member Function Documentation | 72 |
| 7.30.1.1 | eval | 72 |
| 7.30.1.2 | init | 72 |
| 7.30.1.3 | set | 72 |
| 7.30.2 | Field Documentation | 72 |
| 7.30.2.1 | scale | 72 |
| 7.30.2.2 | scale_a_next | 72 |
| 7.30.2.3 | scale_b | 72 |
| 7.30.2.4 | scale_dir | 72 |
| 7.30.2.5 | scale_isneg | 72 |
| 7.30.2.6 | scale_next | 72 |
| 7.30.2.7 | scale_spec | 72 |
| 7.30.2.8 | scale_t | 72 |
| 7.30.2.9 | scale_v_next | 72 |
| 7.30.2.10 | scaling | 73 |
| 7.31 | gomotion::go_sph Struct Reference | 73 |
| 7.31.1 | Detailed Description | 73 |
| 7.31.2 | Field Documentation | 73 |
| 7.31.2.1 | phi | 73 |
| 7.31.2.2 | r | 73 |
| 7.31.2.3 | theta | 73 |

| | |
|---|----|
| 7.32 gomotion::go_traj_ca_spec Struct Reference | 73 |
| 7.32.1 Field Documentation | 74 |
| 7.32.1.1 at0 | 74 |
| 7.32.1.2 dt1 | 74 |
| 7.32.1.3 dt2 | 74 |
| 7.32.1.4 dtend | 74 |
| 7.32.1.5 invd | 74 |
| 7.32.1.6 t1 | 74 |
| 7.32.1.7 t2 | 74 |
| 7.32.1.8 tend | 75 |
| 7.32.1.9 vt1 | 75 |
| 7.33 gomotion::go_traj_cj_spec Struct Reference | 75 |
| 7.33.1 Field Documentation | 76 |
| 7.33.1.1 at1 | 76 |
| 7.33.1.2 dt1 | 76 |
| 7.33.1.3 dt2 | 76 |
| 7.33.1.4 dt3 | 76 |
| 7.33.1.5 dt4 | 76 |
| 7.33.1.6 dt5 | 76 |
| 7.33.1.7 dt6 | 76 |
| 7.33.1.8 dtend | 76 |
| 7.33.1.9 invd | 77 |
| 7.33.1.10 jt0 | 77 |
| 7.33.1.11 t1 | 77 |
| 7.33.1.12 t2 | 77 |
| 7.33.1.13 t3 | 77 |
| 7.33.1.14 t4 | 77 |
| 7.33.1.15 t5 | 77 |
| 7.33.1.16 t6 | 77 |
| 7.33.1.17 tend | 77 |
| 7.33.1.18 vt1 | 77 |
| 7.33.1.19 vt2 | 77 |
| 7.33.1.20 vt3 | 78 |
| 7.34 gomotion::go_traj_interp_spec Struct Reference | 78 |
| 7.34.1 Field Documentation | 78 |
| 7.34.1.1 a | 78 |
| 7.34.1.2 d | 78 |

| | | |
|----------|---|----|
| 7.34.1.3 | j | 78 |
| 7.34.1.4 | s | 78 |
| 7.34.1.5 | v | 78 |
| 7.35 | gomotion::go_uxz Struct Reference | 78 |
| 7.35.1 | Detailed Description | 79 |
| 7.35.2 | Field Documentation | 79 |
| 7.35.2.1 | x | 79 |
| 7.35.2.2 | z | 79 |
| 7.36 | gomotion::go_vel Struct Reference | 79 |
| 7.36.1 | Detailed Description | 80 |
| 7.36.2 | Field Documentation | 80 |
| 7.36.2.1 | v | 80 |
| 7.36.2.2 | w | 80 |
| 7.37 | gomotion::go_xyz Struct Reference | 80 |
| 7.37.1 | Detailed Description | 81 |
| 7.37.2 | Field Documentation | 81 |
| 7.37.2.1 | x | 81 |
| 7.37.2.2 | y | 81 |
| 7.37.2.3 | z | 81 |
| 7.38 | gomotion::go_zyx Struct Reference | 81 |
| 7.38.1 | Detailed Description | 81 |
| 7.38.2 | Field Documentation | 81 |
| 7.38.2.1 | x | 81 |
| 7.38.2.2 | y | 81 |
| 7.38.2.3 | z | 81 |
| 7.39 | gomotion::go_zyz Struct Reference | 81 |
| 7.39.1 | Detailed Description | 82 |
| 7.39.2 | Field Documentation | 82 |
| 7.39.2.1 | y | 82 |
| 7.39.2.2 | z | 82 |
| 7.39.2.3 | zp | 82 |
| 7.40 | gomotion::GoMotion Class Reference | 82 |
| 7.40.1 | Field Documentation | 82 |
| 7.40.1.1 | pgm | 82 |
| 7.41 | gomotion::GoMotionParams Struct Reference | 82 |
| 7.41.1 | Field Documentation | 83 |
| 7.41.1.1 | acc | 83 |

| | | |
|----------|--|-----------|
| 7.41.1.2 | jerk | 83 |
| 7.41.1.3 | vel | 83 |
| 7.42 | gomotion::points Struct Reference | 83 |
| 7.42.1 | Detailed Description | 83 |
| 7.42.2 | Field Documentation | 83 |
| 7.42.2.1 | p | 83 |
| 8 | File Documentation | 85 |
| 8.1 | /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gointerp.h File Reference | 85 |
| 8.2 | /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h File Reference | 86 |
| 8.2.1 | Detailed Description | 92 |
| 8.2.2 | Macro Definition Documentation | 93 |
| 8.2.2.1 | GO_2_PI | 93 |
| 8.2.2.2 | go_body_copy | 93 |
| 8.2.2.3 | go_body_init | 93 |
| 8.2.2.4 | GO_CLOSE | 93 |
| 8.2.2.5 | go_cub | 93 |
| 8.2.2.6 | go_link_to_string | 93 |
| 8.2.2.7 | GO_MATRIX_DECLARE | 94 |
| 8.2.2.8 | go_matrix_init | 94 |
| 8.2.2.9 | GO_PI | 94 |
| 8.2.2.10 | GO_PI_2 | 94 |
| 8.2.2.11 | GO_PI_4 | 94 |
| 8.2.2.12 | go_qua | 94 |
| 8.2.2.13 | GO_ROT_CLOSE | 94 |
| 8.2.2.14 | GO_ROT_SMALL | 94 |
| 8.2.2.15 | GO_SMALL | 95 |
| 8.2.2.16 | go_sq | 95 |
| 8.2.2.17 | GO_TO_DEG | 95 |
| 8.2.2.18 | GO_TO_RAD | 95 |
| 8.2.2.19 | GO_TRAN_CLOSE | 95 |
| 8.2.2.20 | GO_TRAN_SMALL | 95 |
| 8.2.2.21 | ISIZEOF | 95 |
| 8.3 | /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomotion.h File Reference | 95 |
| 8.3.1 | Detailed Description | 97 |
| 8.3.2 | Macro Definition Documentation | 97 |
| 8.3.2.1 | GO_MOTION_JOINT_NUM | 97 |

| | | |
|----------|---|-----|
| 8.4 | /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomove.h File Reference | 97 |
| 8.5 | /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gotraj.h File Reference | 99 |
| 8.5.1 | Macro Definition Documentation | 101 |
| 8.5.1.1 | reciprocate | 101 |
| 8.6 | /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gotypes.h File Reference | 101 |
| 8.6.1 | Macro Definition Documentation | 103 |
| 8.6.1.1 | GO_FLAG | 103 |
| 8.6.1.2 | GO_FLAG_UCHAR | 103 |
| 8.6.1.3 | GO_INF | 103 |
| 8.6.1.4 | GO_INTEGER | 103 |
| 8.6.1.5 | GO_INTEGER_INT | 103 |
| 8.6.1.6 | GO_INTEGER_MAX | 103 |
| 8.6.1.7 | go_quantity_to_string | 103 |
| 8.6.1.8 | GO_REAL | 103 |
| 8.6.1.9 | GO_REAL_DOUBLE | 103 |
| 8.6.1.10 | GO_REAL_EPSILON | 103 |
| 8.6.1.11 | GO_REAL_MAX | 103 |
| 8.6.1.12 | GO_REAL_MIN | 103 |
| 8.6.1.13 | GO_RESULT | 103 |
| 8.6.1.14 | GO_RESULT_INT | 103 |
| 8.6.1.15 | go_result_to_string | 103 |
| 8.6.2 | Typedef Documentation | 104 |
| 8.6.2.1 | go_flag | 104 |
| 8.6.2.2 | go_integer | 104 |
| 8.6.2.3 | go_real | 104 |
| 8.6.2.4 | go_result | 104 |
| 8.6.3 | Enumeration Type Documentation | 104 |
| 8.6.3.1 | anonymous enum | 104 |
| 8.6.3.2 | anonymous enum | 104 |
| 8.6.4 | Variable Documentation | 105 |
| 8.6.4.1 | go_flag_uchar | 105 |
| 8.6.4.2 | go_integer_int | 105 |
| 8.6.4.3 | go_real_double | 105 |
| 8.6.4.4 | go_result_int | 105 |
| 8.6.4.5 | gocode | 105 |
| 8.7 | /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gointerp.cpp File Reference | 105 |
| 8.8 | /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gomath.cpp File Reference | 106 |

| | | |
|----------|---|-----|
| 8.8.1 | Macro Definition Documentation | 110 |
| 8.8.1.1 | CBRT_IS_POW | 110 |
| 8.8.1.2 | COL_IS_UNIT | 110 |
| 8.8.1.3 | MAG2 | 110 |
| 8.8.1.4 | SIGN | 110 |
| 8.9 | /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gomotion.cpp File Reference | 110 |
| 8.9.1 | Macro Definition Documentation | 111 |
| 8.9.1.1 | maxit | 111 |
| 8.10 | /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gomove.cpp File Reference | 112 |
| 8.11 | /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gotraj.cpp File Reference | 112 |
| 8.11.1 | Function Documentation | 113 |
| 8.11.1.1 | go_exit | 113 |
| 8.11.1.2 | go_init | 113 |
| 8.11.2 | Variable Documentation | 113 |
| 8.11.2.1 | go_flag_uchar | 114 |
| 8.11.2.2 | go_integer_int | 114 |
| 8.11.2.3 | go_real_double | 114 |
| 8.11.2.4 | go_result_int | 114 |
| 8.11.2.5 | gocode | 114 |
| 8.12 | /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gotypes.c File Reference | 114 |
| 8.12.1 | Variable Documentation | 114 |
| 8.12.1.1 | go_flag_uchar | 114 |
| 8.12.1.2 | go_integer_int | 115 |
| 8.12.1.3 | go_real_double | 115 |
| 8.12.1.4 | go_result_int | 115 |
| 8.12.1.5 | gocode | 115 |

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

| | |
|--|----|
| The Go Motion Pose Mathematics Library | ?? |
| Trajectory Planning Algorithms | ?? |

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

[gomotion](#) ??

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

| | |
|---|----|
| <code>gomotion::coeff</code> | |
| This structure holds the polynomial coefficients | ?? |
| <code>gomotion::go_body</code> | ?? |
| <code>gomotion::go_cart</code> | ?? |
| <code>gomotion::go_complex</code> | ?? |
| <code>gomotion::go_cubic</code> | ?? |
| <code>gomotion::go_cyl</code> | ?? |
| <code>gomotion::go_dh</code> | ?? |
| <code>gomotion::go_hom</code> | ?? |
| <code>gomotion::go_interp</code> | |
| The interpolator structure | ?? |
| <code>gomotion::go_line</code> | ?? |
| <code>gomotion::go_link</code> | ?? |
| <code>gomotion::go_mat</code> | ?? |
| <code>gomotion::go_matrix</code> | ?? |
| <code>gomotion::go_motion_circular_params</code> | ?? |
| <code>gomotion::go_motion_interface</code> | ?? |
| <code>gomotion::go_motion_linear_params</code> | |
| In the following comments, LIN, CIR and ALL refer to linear moves, circular moves and both, respectively | ?? |
| <code>gomotion::go_motion_params</code> | ?? |
| <code>gomotion::go_motion_queue</code> | ?? |
| <code>gomotion::go_motion_spec</code> | ?? |
| <code>gomotion::go_pk</code> | ?? |
| <code>gomotion::go_plane</code> | ?? |
| <code>gomotion::go_pose</code> | ?? |
| <code>gomotion::go_position</code> | |
| Depending upon whether you are doing joint interpolation or world coordinate interpolation (as specified by your call to <code>go_motion_queue_set_type()</code>), fill in <code>joint[]</code> or <code>pose</code> accordingly | ?? |
| <code>gomotion::go_pp</code> | ?? |
| <code>gomotion::go_quadratic</code> | ?? |
| <code>gomotion::go_quartic</code> | ?? |
| <code>gomotion::go_quat</code> | ?? |
| <code>gomotion::go_rpy</code> | ?? |

| | |
|--|----|
| gomotion::go_rvec | ?? |
| gomotion::go_scale_spec | ?? |
| gomotion::go_sph | ?? |
| gomotion::go_traj_ca_spec | ?? |
| gomotion::go_traj_cj_spec | ?? |
| gomotion::go_traj_interp_spec | ?? |
| gomotion::go_uxz | ?? |
| gomotion::go_vel | ?? |
| gomotion::go_xyz | ?? |
| gomotion::go_zyx | ?? |
| gomotion::go_zyz | ?? |
| gomotion::GoMotion | ?? |
| gomotion::GoMotionParams | ?? |
| gomotion::points | |
| This structure holds the points to be interpolated | ?? |

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

| | |
|--|----|
| /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/ gointerp.h | ?? |
| /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/ gomath.h | |
| Declarations for pose math functions | ?? |
| /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/ gomotion.h | |
| Declarations for motion queue manipulation | ?? |
| /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/ gomove.h | ?? |
| /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/ gotraj.h | ?? |
| /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/ gotypes.h | ?? |
| /usr/local/michalos/nistfanuc_ws/src/gomotion/src/ gointerp.cpp | ?? |
| /usr/local/michalos/nistfanuc_ws/src/gomotion/src/ gomath.cpp | ?? |
| /usr/local/michalos/nistfanuc_ws/src/gomotion/src/ gomotion.cpp | ?? |
| /usr/local/michalos/nistfanuc_ws/src/gomotion/src/ gomove.cpp | ?? |
| /usr/local/michalos/nistfanuc_ws/src/gomotion/src/ gotraj.cpp | ?? |
| /usr/local/michalos/nistfanuc_ws/src/gomotion/src/ gotypes.c | ?? |

Chapter 5

Module Documentation

5.1 The Go Motion Pose Mathematics Library

Go Math Representations

Positions are vectors that indicate where something is. In the three-dimensional world, three numbers are necessary to indicate position. Go Motion supports position vectors in several representations: Cartesian, cylindrical and spherical.

The Cartesian representation uses three numbers **x**, **y** and **z** to represent distances from the origin along three perpendicular axes. The cylindrical representation uses three numbers **r**, **theta** and **z** to represent radial distance away from the origin, angle around the origin and distance up and down from the origin respectively. The spherical representation uses three numbers **theta**, **phi** and **r** to represent angle down from the zenith, angle around the origin and radius from the origin respectively. Go Motion has functions that convert position in one representation position in another representation, so the choice of which representation to use can be made for convenience. Cartesian representations will be assumed unless otherwise specified.

Orientations are vectors that indicate how something is rotated. In the three-dimensional world, three numbers are necessary to indicate orientation. Go Motion supports orientation vectors in several representations: roll, pitch and yaw; Euler angles; quaternions; rotation vectors and rotation matrices. Some of these representations use more than three numbers, exploiting redundancy to make calculations with these representations more efficient. For example, a quaternion uses four numbers, and a rotation matrix uses nine numbers.

Vectors are usually written as a column of numbers enclosed in vertical bars, like this:

```
| 1 |  
| 2 | - a vector depicted in its column form  
| 3 |
```

This can be unwieldy in text documentation, so vectors may also be written as a row of numbers enclosed in parentheses, like this:

```
( 1 2 3 ) - a vector depicted in its row form
```

The interpretation of a vector depends on the quantity it represents. The vector shown above could mean a translation of 1, 2 and 3 units in the x, y and z directions if the vector were a Cartesian position, or a rotation of 1, 2 and 3 units around the x, y and z directions if the vector were an orientation in roll, pitch and yaw.

Both position and orientation are needed to fully describe where something is and how it is rotated. The combination of position and orientation is called a 'pose'. Poses can be shown in row form like this pose representing a Cartesian position of (1 2 3) and an orientation in roll, pitch and yaw of (30 -30 90):

```
( 1 2 3 ; 30 -30 90 )
```

A semicolon is used to separate the position from the orientation.

Go Math Reference Frames

Regardless of the representation chosen, the numbers that indicate position and orientation of an object depend on the established origin. Several origins may be established for convenience, for example one fixed on the world and one that moves with a tool. These origins may differ from each other in both position and orientation. The establishment of the position and orientation of an origin is a 'reference frame'. The term 'coordinate frame' is used interchangeably with 'reference frame'.

When several reference frames are being used, they are denoted as identifiers in braces, for example,

{A} - a reference frame called 'A'.

{world} - the world reference frame.

{tool} - the world reference frame.

To convert the representation of a pose in one reference frame to its representation in another, one needs to know the position and orientation of one origin with respect to the other. This difference between the two origins is called a 'transform'.

Poses and transforms are similar things; both include position and orientation. Whether something is a pose or a transform depends on how one is using it. Poses are used to indicate the position and orientation of things with respect to an established reference frame. Transforms are used to indicate the position and orientation of reference frames with respect to other reference frames. If a 'thing' happens to be a reference frame, its pose is its transform.

Go Math Nomenclature

The letter **P** is used to denote positions and the letter **R** is used to denote orientations. As usual, trailing subscripts denote the identity of quantities, for example,

P_{hand} - the position of the hand.

R_{head} - the orientation of the head.

Leading superscripts denote the reference frame in which the quantity is expressed, for example,

$^A P_{hand}$ - the position of the hand with respect to the {A} reference frame.

$^B R_{head}$ - the orientation of the head with respect to the {B} reference frame.

The figure below shows a rotation of reference frame {B} with respect to reference frame {A} by an angle \mathbf{u} . Note that the rotation can be viewed as a rotation of {B} with respect to {A}, denoted $^A \mathbf{u}_B$, or as a rotation of {A} with respect to {B}, denoted $^B \mathbf{u}_A$. The heads of arrows are attached to the 'of the' frames, while the tails of arrows are based on the 'with respect to' frames. Angles are taken as positive according to the right hand rule, so in this figure $^A \mathbf{u}_B$ is a positive number of about 45 degrees, while $^B \mathbf{u}_A$ is a negative number of the same magnitude.

Figure 1.

Transforms from one reference frame to another are denoted with leading subscripts and superscripts. The leading subscript denotes the original frame, and the leading superscript denotes the new frame. If a transform is purely rotation, it is denoted with an **R**, for example,

$^B_A R$ - a rotation from the {A} frame to the {B} frame.

If a transform includes both a rotation and a translation, it is denoted with a **T**, for example,

$^{world}_{tool} T$ - a transform from the {tool} frame to the {world} frame.

To convert a quantity from one frame to another, pre-multiply the quantity by the transform. Algebraically, the leading subscript of the transform must match the leading superscript of the quantity, for example,

$$^B P = ^B_A R * ^A P$$

This is read, 'the position with respect to the B frame is the rotation from the A frame to the B frame times the position

with respect to the A frame.' Another example is,

$$\text{world } \mathbf{p} = \text{world}_{\text{tool}} \mathbf{T} * \text{tool } \mathbf{p}$$

This is read, 'the position with respect to the world frame is the transform from the tool frame to the world frame times the position with respect to the tool frame.'

One can generate a rotational transform by defining an orientation vector with the appropriate angles. If using a roll, pitch and yaw representation, the rotational transforms in the figure above (each about the **z** axis) is a yaw:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = {}^A R_B$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{bmatrix} = {}^B R_A$$

and negative angles work as well, for

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = {}^A R_B$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix} = {}^B R_A$$

Go Math Examples

Given a reference frame {B} rotated 30 degrees with respect to the **z** axis of reference frame {A}, as in Figure 1 above, transform from points in the {B} frame to points in the {A} frame like this:

```
go_rpy rot;
go_cart pt_in_b;
go_cart pt_in_a;

// The angle of {B} with respect to {A} is 30 degrees.
// Go uses angles in radians.
rot.r = 0, rot.p = 0, rot.y = GO_TO_RAD(30);

pt_in_b.x = 1, pt_in_b.y = 2, pt_in_b.z = 3;

// Multiply a transform and a point to get a new point.
go_rpy_cart_mult(&rot, &pt_in_b, &pt_in_a);

go_cart_print(&pt_in_a);
```

will print the position of the point in the {A} frame:

```
-0.133975 2.232051 3.000000
```

Here is a more complex example of a full transform, including both translation and rotation.

Figure 2.

The frame {B} is translated and rotated with respect to {A}. ${}^A x_B$ is the amount of translation of {B} in the **x** direction of {A}. Likewise, ${}^A y_B$ is the amount of translation of {B} in the **y** direction of {A}. ${}^A u_B$ is the rotation of the {B} frame about the **z** axis of the {A} frame.

To convert points in the {B} frame to points in the {A} frame, do this:

```
go_pose pose;
go_rpy rot;
go_cart pt_in_b;
```

```
go_cart pt_in_a;

// The translation of {B} with respect to {A} is about (2 1 0).
pose.tran.x = 2, pose.tran.y = 1, pose.tran.z = 0;

// The angle of {B} wrt {A} is about 30 degrees, made into radians.
rot.r = 0, rot.p = 0, rot.y = GO_TO_RAD(30);

// A 'go_pose' uses quaternions for rotations, so we have to
// convert a roll-pitch-yaw to a quaternion.
go_rpy_quat_convert(&rot, &pose.rot);

pt_in_b.x = 1, pt_in_b.y = 2, pt_in_b.z = 3;

// Multiply a transform and a point to get a new point.
go_pose_cart_mult(&pose, &pt_in_b, &pt_in_a);

go_cart_print(&pt_in_a);
```

This gives the transformed point in the {A} frame as

```
1.866025 3.232051 3.000000
```

5.2 Trajectory Planning Algorithms

gotraj.c

gotraj.c Trajectory planning functions

CV means constant velocity, CA means constant acceleration, CJ means constant jerk. !

The Go Motion trajectory planning algorithms are based on smooth velocity profiling with bounded speed, acceleration and jerk, called "constant jerk" or "S-curve" velocity profiling. This gives smoother control than "trapezoidal" velocity profiling, which transitions instantaneously between acceleration and no acceleration and incurs spikes in unbounded jerk.

Constant-jerk (CJ) profiling is shown in Figure 1, a plot of the speed versus time. There are 7 phases to the motion. Phase 1 is a jerk phase, where the acceleration varies smoothly from 0 at time 0 to $a1$ at time $t1$ following the jerk (change in acceleration per unit time) $j0$. Phase 2 is an acceleration phase, with constant acceleration $a1$ throughout. Phase 3 is a jerk phase (or de-jerk phase) with constant (negative) jerk slowing down the acceleration from $a1$ to 0. Phase 4 is a constant speed phase at speed $v3$. Phase 5 is a constant-jerk counterpart to phase 3, where the deceleration varies smoothly from 0 to $-a1$. Phase 6 is a constant-acceleration counterpart to phase 2. Phase 7 is a constant-jerk counterpart to phase 1, where the deceleration varies smoothly from $-a1$ to 0 and motion stops. Figure 1. Constant jerk velocity profiling.

Chapter 6

Namespace Documentation

6.1 gomotion Namespace Reference

Data Structures

- struct [go_motion_interface](#)
- struct [points](#)

This structure holds the points to be interpolated.

- struct [coeff](#)

This structure holds the polynomial coefficients.

- class [go_interp](#)

The interpolator structure.

- struct [go_cart](#)
- struct [go_sph](#)
- struct [go_cyl](#)
- struct [go_rvec](#)
- struct [go_mat](#)
- struct [go_quat](#)
- struct [go_zyz](#)
- struct [go_zyx](#)
- struct [go_xyz](#)
- struct [go_rpy](#)
- struct [go_uxz](#)
- struct [go_pose](#)
- struct [go_vel](#)
- struct [go_hom](#)
- struct [go_line](#)
- struct [go_plane](#)
- struct [go_matrix](#)
- struct [go_dh](#)
- struct [go_pk](#)
- struct [go_pp](#)
- struct [go_body](#)
- struct [go_link](#)
- struct [go_complex](#)

- struct [go_quadratic](#)
- struct [go_cubic](#)
- struct [go_quartic](#)
- struct [go_position](#)

Depending upon whether you are doing joint interpolation or world coordinate interpolation (as specified by your call to [go_motion_queue_set_type\(\)](#)), fill in [joint\[\]](#) or [pose](#) accordingly.

- struct [go_motion_params](#)
- struct [go_motion_linear_params](#)

In the following comments, LIN, CIR and ALL refer to linear moves, circular moves and both, respectively.

- struct [go_motion_circular_params](#)
- struct [go_motion_spec](#)
- struct [go_scale_spec](#)
- struct [go_motion_queue](#)
- struct [GoMotionParams](#)
- class [GoMotion](#)
- struct [go_traj_ca_spec](#)
- struct [go_traj_cj_spec](#)
- struct [go_traj_interp_spec](#)

Typedefs

- typedef [go_real](#) [go_vector](#)
- typedef [sensor_msgs::JointState_](#)
< std::allocator< void > > [JointState](#)

Enumerations

- enum { [COEFF_MAX](#) = 6 }
How many coefficients we support, one more than max polynomial degree.
- enum { [GO_LINK_DH](#) = 1, [GO_LINK_PK](#), [GO_LINK_PP](#) }
- enum {
 [GO_MOTION_NONE](#), [GO_MOTION_JOINT](#), [GO_MOTION_UJOINT](#), [GO_MOTION_WORLD](#),
 [GO_MOTION_LINEAR](#), [GO_MOTION_CIRCULAR](#) }

Functions

- void [sincos](#) (double x, double *sx, double *cx)
- void [go_sincos](#) ([go_real](#) t, [go_real](#) *s, [go_real](#) *c)
- [go_result](#) [go_asines](#) ([go_real](#) s, [go_real](#) *asp, [go_real](#) *asn)
- [go_result](#) [go_acoses](#) ([go_real](#) c, [go_real](#) *acp, [go_real](#) *acn)
- [go_result](#) [go_atans](#) ([go_real](#) t, [go_real](#) *atp, [go_real](#) *atn)
- [go_real](#) [go_cbrt](#) ([go_real](#) x)
- [go_result](#) [go_cart_sph_convert](#) (const [go_cart](#) *v, [go_sph](#) *s)
- [go_result](#) [go_cart_cyl_convert](#) (const [go_cart](#) *v, [go_cyl](#) *c)
- [go_result](#) [go_sph_cart_convert](#) (const [go_sph](#) *s, [go_cart](#) *v)
- [go_result](#) [go_sph_cyl_convert](#) (const [go_sph](#) *s, [go_cyl](#) *c)
- [go_result](#) [go_cyl_cart_convert](#) (const [go_cyl](#) *c, [go_cart](#) *v)
- [go_result](#) [go_cyl_sph_convert](#) (const [go_cyl](#) *c, [go_sph](#) *s)

- [go_result go_rvec_quat_convert](#) (const [go_rvec](#) *r, [go_quat](#) *q)
- [go_result go_rvec_mat_convert](#) (const [go_rvec](#) *r, [go_mat](#) *m)
- [go_result go_rvec_zyz_convert](#) (const [go_rvec](#) *rvec, [go_zyz](#) *zyz)
- [go_result go_rvec_zyx_convert](#) (const [go_rvec](#) *rvec, [go_zyx](#) *zyx)
- [go_result go_rvec_rpy_convert](#) (const [go_rvec](#) *r, [go_rpy](#) *rpy)
- [go_result go_quat_rvec_convert](#) (const [go_quat](#) *q, [go_rvec](#) *r)
- [go_result go_quat_mat_convert](#) (const [go_quat](#) *q, [go_mat](#) *m)
- [go_result go_quat_zyz_convert](#) (const [go_quat](#) *q, [go_zyz](#) *zyz)
- [go_result go_quat_zyx_convert](#) (const [go_quat](#) *q, [go_zyx](#) *zyx)
- [go_result go_quat_rpy_convert](#) (const [go_quat](#) *q, [go_rpy](#) *rpy)
- [go_result go_mat_rvec_convert](#) (const [go_mat](#) *m, [go_rvec](#) *r)
- [go_result go_mat_quat_convert](#) (const [go_mat](#) *m, [go_quat](#) *q)
- [go_result go_mat_zyz_convert](#) (const [go_mat](#) *m, [go_zyz](#) *zyz)
- [go_result go_mat_zyx_convert](#) (const [go_mat](#) *m, [go_zyx](#) *zyx)
- [go_result go_mat_xyz_convert](#) (const [go_mat](#) *m, [go_xyz](#) *xyz)
- [go_result go_mat_rpy_convert](#) (const [go_mat](#) *m, [go_rpy](#) *rpy)
- [go_result go_zyz_rvec_convert](#) (const [go_zyz](#) *zyz, [go_rvec](#) *r)
- [go_result go_zyz_quat_convert](#) (const [go_zyz](#) *zyz, [go_quat](#) *q)
- [go_result go_zyz_mat_convert](#) (const [go_zyz](#) *zyz, [go_mat](#) *m)
- [go_result go_zyz_zyx_convert](#) (const [go_zyz](#) *zyz, [go_zyx](#) *zyx)
- [go_result go_zyz_rpy_convert](#) (const [go_zyz](#) *zyz, [go_rpy](#) *rpy)
- [go_result go_zyx_rvec_convert](#) (const [go_zyx](#) *zyx, [go_rvec](#) *r)
- [go_result go_zyx_quat_convert](#) (const [go_zyx](#) *zyx, [go_quat](#) *q)
- [go_result go_zyx_mat_convert](#) (const [go_zyx](#) *zyx, [go_mat](#) *m)
- [go_result go_zyx_zyz_convert](#) (const [go_zyx](#) *zyx, [go_zyz](#) *zyz)
- [go_result go_zyx_rpy_convert](#) (const [go_zyx](#) *zyx, [go_rpy](#) *rpy)
- [go_result go_xyz_mat_convert](#) (const [go_xyz](#) *xyz, [go_mat](#) *m)
- [go_result go_rpy_rvec_convert](#) (const [go_rpy](#) *rpy, [go_rvec](#) *rvec)
- [go_result go_rpy_quat_convert](#) (const [go_rpy](#) *rpy, [go_quat](#) *quat)
- [go_result go_rpy_mat_convert](#) (const [go_rpy](#) *rpy, [go_mat](#) *m)
- [go_result go_rpy_zyz_convert](#) (const [go_rpy](#) *rpy, [go_zyz](#) *zyz)
- [go_result go_rpy_zyx_convert](#) (const [go_rpy](#) *rpy, [go_zyx](#) *zyx)
- [go_result go_uxz_mat_convert](#) (const [go_uxz](#) *uxz, [go_mat](#) *mat)
- [go_result go_mat_uxz_convert](#) (const [go_mat](#) *mat, [go_uxz](#) *uxz)
- [go_pose go_pose_this](#) ([go_real](#) x, [go_real](#) y, [go_real](#) z, [go_real](#) rs, [go_real](#) rx, [go_real](#) ry, [go_real](#) rz)
- [go_cart go_cart_zero](#) (void)
- [go_quat go_quat_identity](#) (void)
- [go_pose go_pose_identity](#) (void)
- [go_result go_pose_hom_convert](#) (const [go_pose](#) *p, [go_hom](#) *h)
- [go_result go_hom_pose_convert](#) (const [go_hom](#) *h, [go_pose](#) *p)
- [go_result go_cart_rvec_convert](#) (const [go_cart](#) *cart, [go_rvec](#) *rvec)
- [go_result go_rvec_cart_convert](#) (const [go_rvec](#) *rvec, [go_cart](#) *cart)
- [go_flag go_cart_cart_compare](#) (const [go_cart](#) *v1, const [go_cart](#) *v2)
- [go_result go_cart_cart_dot](#) (const [go_cart](#) *v1, const [go_cart](#) *v2, [go_real](#) *d)
- [go_result go_cart_cart_cross](#) (const [go_cart](#) *v1, const [go_cart](#) *v2, [go_cart](#) *vout)
- [go_result go_cart_mag](#) (const [go_cart](#) *v, [go_real](#) *d)
- [go_result go_cart_magsq](#) (const [go_cart](#) *v, [go_real](#) *d)
- [go_flag go_cart_cart_par](#) (const [go_cart](#) *v1, const [go_cart](#) *v2)
- [go_flag go_cart_cart_perp](#) (const [go_cart](#) *v1, const [go_cart](#) *v2)
- [go_result go_cart_cart_disp](#) (const [go_cart](#) *v1, const [go_cart](#) *v2, [go_real](#) *d)
- [go_result go_cart_cart_add](#) (const [go_cart](#) *v1, const [go_cart](#) *v2, [go_cart](#) *vout)

- [go_result go_cart_cart_sub](#) (const [go_cart](#) *v1, const [go_cart](#) *v2, [go_cart](#) *vout)
- [go_result go_cart_scale_mult](#) (const [go_cart](#) *v1, [go_real](#) d, [go_cart](#) *vout)
- [go_result go_cart_neg](#) (const [go_cart](#) *v1, [go_cart](#) *vout)
- [go_result go_cart_unit](#) (const [go_cart](#) *v, [go_cart](#) *vout)
- [go_result go_cart_is_norm](#) (const [go_cart](#) *v)
- [go_result go_cart_cart_rot](#) (const [go_cart](#) *v1, const [go_cart](#) *v2, [go_quat](#) *quat)
- [go_result go_cart_cart_proj](#) (const [go_cart](#) *v1, const [go_cart](#) *v2, [go_cart](#) *vout)
- [go_result go_cart_plane_proj](#) (const [go_cart](#) *v, const [go_cart](#) *normal, [go_cart](#) *vout)
- [go_result go_cart_cart_angle](#) (const [go_cart](#) *v1, const [go_cart](#) *v2, [go_real](#) *a)
- [go_result go_cart_normal](#) (const [go_cart](#) *v, [go_cart](#) *vout)
- [go_result go_cart_centroid](#) (const [go_cart](#) *varray, [go_integer](#) num, [go_cart](#) *centroid)
- [go_result go_cart_centroidize](#) (const [go_cart](#) *vinarray, [go_integer](#) num, [go_cart](#) *centroid, [go_cart](#) *voutarray)
- [go_complex go_complex_add](#) ([go_complex](#) z1, [go_complex](#) z2)
- [go_complex go_complex_sub](#) ([go_complex](#) z1, [go_complex](#) z2)
- [go_complex go_complex_mult](#) ([go_complex](#) z1, [go_complex](#) z2)
- [go_complex go_complex_inv](#) ([go_complex](#) z, [go_result](#) *result)
- [go_complex go_complex_div](#) ([go_complex](#) z1, [go_complex](#) z2, [go_result](#) *result)
- [go_complex go_complex_sq](#) ([go_complex](#) z)
- [go_complex go_complex_scale](#) ([go_complex](#) z, [go_real](#) scale)
- [go_real go_complex_mag](#) ([go_complex](#) z)
- [go_real go_complex_magsq](#) ([go_complex](#) z)
- [go_real go_complex_arg](#) ([go_complex](#) z)
- [void go_complex_sqrt](#) ([go_complex](#) z, [go_complex](#) *z1, [go_complex](#) *z2)
- [void go_complex_cbrt](#) ([go_complex](#) z, [go_complex](#) *z1, [go_complex](#) *z2, [go_complex](#) *z3)
- [go_result go_quadratic_solve](#) (const [go_quadratic](#) *quad, [go_complex](#) *z1, [go_complex](#) *z2)
- [go_result go_cubic_solve](#) (const [go_cubic](#) *cub, [go_complex](#) *z1, [go_complex](#) *z2, [go_complex](#) *z3)
- [go_result go_quartic_solve](#) (const [go_quartic](#) *quart, [go_complex](#) *z1, [go_complex](#) *z2, [go_complex](#) *z3, [go_complex](#) *z4)
- [go_result go_tridiag_reduce](#) ([go_real](#) **a, [go_integer](#) n, [go_real](#) *d, [go_real](#) *e)
- [go_result go_tridiag_ql](#) ([go_real](#) *d, [go_real](#) *e, [go_integer](#) n, [go_real](#) **z)
- [go_result go_cart_cart_pose](#) (const [go_cart](#) *v1, const [go_cart](#) *v2, [go_cart](#) *v1c, [go_cart](#) *v2c, [go_integer](#) num, [go_pose](#) *pout)
- [go_result go_cart_trilaterate](#) (const [go_cart](#) *c1, const [go_cart](#) *c2, const [go_cart](#) *c3, [go_real](#) l1, [go_real](#) l2, [go_real](#) l3, [go_cart](#) *out1, [go_cart](#) *out2)
- [go_flag go_rvec_rvec_compare](#) (const [go_rvec](#) *r1, const [go_rvec](#) *r2)
- [go_result go_rvec_scale_mult](#) (const [go_rvec](#) *r, [go_real](#) s, [go_rvec](#) *rout)
- [go_result go_rpy_cart_mult](#) (const [go_rpy](#) *rpy, const [go_cart](#) *in, [go_cart](#) *out)
- [go_result go_mat_norm](#) (const [go_mat](#) *mat, [go_mat](#) *mout)
- [go_flag go_mat_is_norm](#) (const [go_mat](#) *m)
- [go_result go_mat_inv](#) (const [go_mat](#) *m, [go_mat](#) *mout)
- [go_result go_mat_cart_mult](#) (const [go_mat](#) *m, const [go_cart](#) *v, [go_cart](#) *vout)
- [go_result go_mat_mat_mult](#) (const [go_mat](#) *m1, const [go_mat](#) *m2, [go_mat](#) *mout)
- [go_flag go_quat_quat_compare](#) (const [go_quat](#) *q1, const [go_quat](#) *q2)
- [go_result go_quat_mag](#) (const [go_quat](#) *quat, [go_real](#) *d)
- [go_result go_quat_unit](#) (const [go_quat](#) *q1, [go_quat](#) *qout)
- [go_result go_quat_norm](#) (const [go_quat](#) *q1, [go_quat](#) *qout)
- [go_result go_quat_inv](#) (const [go_quat](#) *q1, [go_quat](#) *qout)
- [go_flag go_quat_is_norm](#) (const [go_quat](#) *q1)
- [go_result go_quat_scale_mult](#) (const [go_quat](#) *q, [go_real](#) s, [go_quat](#) *qout)
- [go_result go_quat_quat_mult](#) (const [go_quat](#) *q1, const [go_quat](#) *q2, [go_quat](#) *qout)
- [go_result go_quat_cart_mult](#) (const [go_quat](#) *q1, const [go_cart](#) *v2, [go_cart](#) *vout)

- [go_flag go_pose_pose_compare](#) (const [go_pose](#) *p1, const [go_pose](#) *p2)
- [go_result go_pose_inv](#) (const [go_pose](#) *p1, [go_pose](#) *p2)
- [go_result go_pose_cart_mult](#) (const [go_pose](#) *p1, const [go_cart](#) *v2, [go_cart](#) *vout)
- [go_result go_pose_pose_mult](#) (const [go_pose](#) *p1, const [go_pose](#) *p2, [go_pose](#) *pout)
- [go_result go_pose_scale_mult](#) (const [go_pose](#) *p1, [go_real](#) s, [go_pose](#) *pout)
- [go_result go_pose_pose_interp](#) ([go_real](#) t1, const [go_pose](#) *p1, [go_real](#) t2, const [go_pose](#) *p2, [go_real](#) t3, [go_pose](#) *p3)
- [go_result go_hom_inv](#) (const [go_hom](#) *h1, [go_hom](#) *h2)
- [go_result go_hom_hom_mult](#) (const [go_hom](#) *h1, const [go_hom](#) *h2, [go_hom](#) *hout)
- [go_result go_pose_vel_mult](#) (const [go_pose](#) *pose, const [go_vel](#) *vel, [go_vel](#) *out)
- [go_result go_line_from_point_direction](#) (const [go_cart](#) *point, const [go_cart](#) *direction, [go_line](#) *line)
- [go_result go_line_from_points](#) (const [go_cart](#) *point1, const [go_cart](#) *point2, [go_line](#) *line)
- [go_result go_line_from_planes](#) (const [go_plane](#) *plane1, const [go_plane](#) *plane2, [go_line](#) *line)
- [go_flag go_line_line_compare](#) (const [go_line](#) *line1, const [go_line](#) *line2)
- [go_result go_line_evaluate](#) (const [go_line](#) *line, [go_real](#) d, [go_cart](#) *point)
- [go_result go_point_line_distance](#) (const [go_cart](#) *point, const [go_line](#) *line, [go_real](#) *distance)
- [go_result go_point_line_proj](#) (const [go_cart](#) *point, const [go_line](#) *line, [go_cart](#) *pout)
- [go_result go_point_plane_proj](#) (const [go_cart](#) *point, const [go_plane](#) *plane, [go_cart](#) *proj)
- [go_result go_line_plane_proj](#) (const [go_line](#) *line, const [go_plane](#) *plane, [go_line](#) *proj)
- [go_result go_plane_from_point_normal](#) (const [go_cart](#) *point, const [go_cart](#) *normal, [go_plane](#) *plane)
- [go_result go_plane_from_abcd](#) ([go_real](#) A, [go_real](#) B, [go_real](#) C, [go_real](#) D, [go_plane](#) *plane)
- [go_result go_plane_from_points](#) (const [go_cart](#) *point1, const [go_cart](#) *point2, const [go_cart](#) *point3, [go_plane](#) *plane)
- [go_result go_plane_from_point_line](#) (const [go_cart](#) *point, const [go_line](#) *line, [go_plane](#) *plane)
- [go_flag go_plane_plane_compare](#) (const [go_plane](#) *plane1, const [go_plane](#) *plane2)
- [go_result go_point_plane_distance](#) (const [go_cart](#) *point, const [go_plane](#) *plane, [go_real](#) *distance)
- [go_result go_plane_evaluate](#) (const [go_plane](#) *plane, [go_real](#) u, [go_real](#) v, [go_cart](#) *point)
- [go_result go_line_plane_intersect](#) (const [go_line](#) *line, const [go_plane](#) *plane, [go_cart](#) *point, [go_real](#) *distance)
- [go_real go_get_singular_epsilon](#) (void)
- [go_result go_set_singular_epsilon](#) ([go_real](#) epsilon)
- [go_result ludcmp](#) ([go_real](#) **a, [go_real](#) *scratchrow, [go_integer](#) n, [go_integer](#) *indx, [go_real](#) *d)
- [go_result lubksb](#) ([go_real](#) **a, [go_integer](#) n, [go_integer](#) *indx, [go_real](#) *b)
- [go_result go_cart_vector_convert](#) (const [go_cart](#) *c, [go_real](#) *v)
- [go_result go_vector_cart_convert](#) (const [go_real](#) *v, [go_cart](#) *c)
- [go_result go_quat_matrix_convert](#) (const [go_quat](#) *quat, [go_matrix](#) *matrix)
- [go_result go_mat_matrix_convert](#) (const [go_mat](#) *mat, [go_matrix](#) *matrix)
- [go_result go_matrix_matrix_add](#) (const [go_matrix](#) *a, const [go_matrix](#) *b, [go_matrix](#) *apb)
- [go_result go_matrix_matrix_copy](#) (const [go_matrix](#) *src, [go_matrix](#) *dst)
- [go_result go_matrix_matrix_mult](#) (const [go_matrix](#) *a, const [go_matrix](#) *b, [go_matrix](#) *ab)
- [go_result go_matrix_vector_mult](#) (const [go_matrix](#) *a, const [go_vector](#) *v, [go_vector](#) *axv)
- [go_result go_matrix_vector_cross](#) (const [go_matrix](#) *a, const [go_vector](#) *v, [go_matrix](#) *axv)
- [go_result go_matrix_transpose](#) (const [go_matrix](#) *a, [go_matrix](#) *at)
- [go_result go_matrix_inv](#) (const [go_matrix](#) *m, [go_matrix](#) *minv)
- [go_result go_mat3_inv](#) ([go_real](#) a[3][3], [go_real](#) ainv[3][3])
- [go_result go_mat3_mat3_mult](#) ([go_real](#) a[3][3], [go_real](#) b[3][3], [go_real](#) axb[3][3])
- [go_result go_mat3_vec3_mult](#) ([go_real](#) a[3][3], [go_real](#) v[3], [go_real](#) axv[3])
- [go_result go_mat4_inv](#) ([go_real](#) a[4][4], [go_real](#) ainv[4][4])
- [go_result go_mat4_mat4_mult](#) ([go_real](#) a[4][4], [go_real](#) b[4][4], [go_real](#) axb[4][4])
- [go_result go_mat4_vec4_mult](#) ([go_real](#) a[4][4], [go_real](#) v[4], [go_real](#) axv[4])
- [go_result go_mat6_inv](#) ([go_real](#) a[6][6], [go_real](#) ainv[6][6])
- [go_result go_mat6_transpose](#) ([go_real](#) a[6][6], [go_real](#) at[6][6])

- `go_result go_mat6_mat6_mult (go_real a[6][6], go_real b[6][6], go_real axb[6][6])`
- `go_result go_mat6_vec6_mult (go_real a[6][6], go_real v[6], go_real axv[6])`
- `go_result go_dh_hom_convert (const go_dh *dh, go_hom *h)`
- `go_result go_dh_pose_convert (const go_dh *dh, go_pose *p)`
- `go_result go_hom_dh_convert (const go_hom *h, go_dh *dh)`
- `go_result go_pose_dh_convert (const go_pose *p, go_dh *dh)`
- `go_result go_link_joint_set (const go_link *link, go_real joint, go_link *linkout)`
- `go_result go_link_pose_build (const go_link *link_params, go_integer num, go_pose *pose)`
- `go_result go_linear_cos_sin_solve (go_real a, go_real b, go_real *th1, go_real *th2)`
- `go_result go_rvec_xyz_convert (const go_rvec *, go_xyz *)`
- `go_result go_quat_xyz_convert (const go_quat *, go_xyz *)`
- `go_result go_zyz_xyz_convert (const go_zyz *, go_xyz *)`
- `go_result go_zyx_xyz_convert (const go_zyx *, go_xyz *)`
- `go_result go_xyz_rvec_convert (const go_xyz *, go_rvec *)`
- `go_result go_xyz_quat_convert (const go_xyz *, go_quat *)`
- `go_result go_xyz_zyz_convert (const go_xyz *, go_zyz *)`
- `go_result go_xyz_zyx_convert (const go_xyz *, go_zyx *)`
- `go_result go_xyz_rpy_convert (const go_xyz *, go_rpy *)`
- `go_result go_rpy_xyz_convert (const go_rpy *, go_xyz *)`
- `go_result go_traj_ca_generate (go_real acc, go_real deltacc, go_real deltvel, go_traj_ca_spec *pts)`
go_traj_ca_generate() takes an accel value, and intervals for the accel period and cruise period, and fills in the *go_traj_ca_spec* with the interval parameters.
- `go_result go_traj_ca_compute (go_real d, go_real v, go_real a, go_traj_ca_spec *pts)`
go_traj_ca_compute() takes values for distance 'd' to move, max velocity 'v' to limit if necessary, and constant accel 'a', and fills in the *go_traj_ca_spec* with the interval parameters.
- `go_result go_traj_ca_scale (const go_traj_ca_spec *ts, go_real t, go_traj_ca_spec *pts)`
go_traj_ca_scale() takes a time 't' for the desired time of the motion, and scales the times and motion params so that the total distance remains the same and everything else is in proportion
- `go_result go_traj_ca_stop (const go_traj_ca_spec *ts, go_real t, go_traj_ca_spec *pts)`
go_traj_ca_stop() takes a time 't' for the desired time to begin stopping, and recomputes the times so that the move will stop as soon as possible during subsequent interps.
- `go_result go_traj_ca_extend (const go_traj_ca_spec *ts, go_real t, go_traj_ca_spec *pts)`
go_traj_ca_extend() takes a time 't' for the desired time to finish the motion, and extends the constant-speed section so that it stops then.
- `go_result go_traj_ca_interp (const go_traj_ca_spec *ts, go_real t, go_traj_interp_spec *ti)`
go_traj_ca_interp() takes a *go_traj_ca_spec* and interpolates the d-v-a values for the given time t, storing the d-v-a values in ti.
- `go_result go_traj_cj_generate (go_real jrk, go_real deltjrk, go_real deltacc, go_real deltvel, go_traj_cj_spec *pts)`
go_traj_cj_generate() takes a jerk value, and intervals for the jerk period, accel period and cruise period, and fills in the *go_traj_cj_spec* with the interval parameters.
- `go_result go_traj_cj_compute (go_real d, go_real v, go_real a, go_real j, go_traj_cj_spec *pts)`
go_traj_cj_compute() takes values for distance 'd' to move, max velocity 'v' to limit if necessary, max accel 'a' to limit if necessary, and constant jerk 'j', and fills in the *go_traj_cj_spec* with the interval parameters.
- `go_result go_traj_cj_scale (const go_traj_cj_spec *ts, go_real t, go_traj_cj_spec *pts)`
go_traj_cj_scale() takes a time 't' for the desired time of the motion, and scales the times and motion params so that the total distance remains the same and everything else is in proportion
- `go_result go_traj_cj_stop (const go_traj_cj_spec *ts, go_real t, go_traj_cj_spec *pts)`
go_traj_cj_stop() takes a time 't' for the desired time to begin stopping, and recomputes the times so that the move will stop as soon as possible during subsequent interps.
- `go_result go_traj_cj_extend (const go_traj_cj_spec *ts, go_real t, go_traj_cj_spec *pts)`

`go_traj_cj_extend()` takes a time 't' for the desired time to finish the motion, and extends the constant-speed section so that it stops then.

- `go_result go_traj_cj_interp` (const `go_traj_cj_spec` *ts, `go_real` t, `go_traj_interp_spec` *ti)

`go_traj_cj_interp()` takes a `go_traj_cj_spec` and interpolates the d-v-a-j values for the given time t, storing the d-v-a-j values in ti.

- `go_result go_traj_cj_compute_the_rest` (`go_traj_cj_spec` *pts)

! Expects pts attributes dtend, jt0, at1, vt3, t1, t2 and t3 to be already filled in from previous calls to one of the four `go_traj_cj_compute_tees` functions and `go_traj_cj_compute`.

- `go_result go_traj_cj_compute_tees_na_nc` (`go_traj_cj_spec` *pts)

! Computes the trajectory defining times t1, t2 and t4 for a motion with no acceleration or cruise phase, given pts filled in with dtend and jt0.

- `go_result go_traj_cj_compute_tees_a_nc` (`go_traj_cj_spec` *pts)

! Computes the trajectory defining times t1, t2 and t4 for a motion with acceleration but no cruise phase, given pts filled in with dtend, jt0 and at1.

- `go_result go_traj_cj_compute_tees_na_c` (`go_traj_cj_spec` *pts)

! Computes the trajectory defining times t1, t2 and t4 for a motion with no acceleration but a cruise phase, given pts filled in with dtend, jt0 and vt3.

- `go_result go_traj_cj_compute_tees_a_c` (`go_traj_cj_spec` *pts)

! Computes the trajectory defining times t1, t2 and t4 for a motion with no acceleration but a cruise phase, given pts filled in with dtend, jt0, at1 and vt3.

6.1.1 Typedef Documentation

6.1.1.1 typedef go_real gomotion::go_vector

Declare a matrix variable *m* with *rows* rows and *cols* columns. Allocates *rows* X *columns* of space in *mspace*.

6.1.1.2 typedef sensor_msgs::JointState_<std::allocator<void>> > gomotion::JointState

6.1.2 Enumeration Type Documentation

6.1.2.1 anonymous enum

How many coefficients we support, one more than max polynomial degree.

Enumerator

COEFF_MAX

6.1.2.2 anonymous enum

Types of link parameter representations

Enumerator

GO_LINK_DH for Denavit-Hartenberg params

GO_LINK_PK for parallel kinematics

GO_LINK_PP for serial kinematics

6.1.2.3 anonymous enum

Enumerator

GO_MOTION_NONE types of motion queueing done
GO_MOTION_JOINT
GO_MOTION_UJOINT
GO_MOTION_WORLD
GO_MOTION_LINEAR these are used to further specify types of world motion
GO_MOTION_CIRCULAR

6.1.3 Function Documentation

6.1.3.1 **go_result gomotion::go_acoses (go_real c, go_real * acp, go_real * acn)**

Double-valued inverse cosine, giving both solutions

6.1.3.2 **go_result gomotion::go_asines (go_real s, go_real * asp, go_real * asn)**

Double-valued inverse sine, giving both solutions

6.1.3.3 **go_result gomotion::go_atans (go_real t, go_real * atp, go_real * atn)**

Double-valued inverse tangent, giving both solutions

6.1.3.4 **go_result gomotion::go_cart_cart_add (const go_cart * v1, const go_cart * v2, go_cart * vout)**

6.1.3.5 **go_result gomotion::go_cart_cart_angle (const go_cart * v1, const go_cart * v2, go_real * a)**

6.1.3.6 **go_flag gomotion::go_cart_cart_compare (const go_cart * v1, const go_cart * v2)**

6.1.3.7 **go_result gomotion::go_cart_cart_cross (const go_cart * v1, const go_cart * v2, go_cart * vout)**

6.1.3.8 **go_result gomotion::go_cart_cart_disp (const go_cart * v1, const go_cart * v2, go_real * disp)**

Places the Cartesian displacement between two vectors *v1* and *v2* in *disp*, returning *GO_RESULT_OK*.

6.1.3.9 **go_result gomotion::go_cart_cart_dot (const go_cart * v1, const go_cart * v2, go_real * d)**

6.1.3.10 **go_flag gomotion::go_cart_cart_par (const go_cart * v1, const go_cart * v2)**

6.1.3.11 **go_flag gomotion::go_cart_cart_perp (const go_cart * v1, const go_cart * v2)**

6.1.3.12 **go_result gomotion::go_cart_cart_pose (const go_cart * v1, const go_cart * v2, go_cart * v1c, go_cart * v2c, go_integer num, go_pose * pout)**

Given an array of Cartesian points measured in one coordinate system, and an associated array measured in another coordinate system, returns the best-fit transform that takes points in the first coordinate system to their representation in the second coordinate system.

Parameters

| | |
|-------------|--|
| <i>v1</i> | An array of Cartesian points in one coordinate system |
| <i>v2</i> | An array of the same Cartesian points in another coordinate system |
| <i>v1c</i> | A scratch array of the same size as <i>v1</i> needed for calculation. |
| <i>v2c</i> | A scratch array of the same size as <i>v2</i> needed for calculation. |
| <i>num</i> | The number of points in the <i>v1</i> and <i>v2</i> arrays, which must be the same size. |
| <i>pout</i> | A pointer to the calculated best-fit pose filled in by this function. |

6.1.3.13 **go_result** gomotion::go_cart_cart_proj (const go_cart * *v1*, const go_cart * *v2*, go_cart * *vout*)

Project vector *v1* onto *v2*, with the resulting vector placed into *vout*. Returns GO_RESULT_OK if it can be done, otherwise something else.

6.1.3.14 **go_result** gomotion::go_cart_cart_rot (const go_cart * *v1*, const go_cart * *v2*, go_quat * *quat*)

Given two non-zero vectors *v1* and *v2*, fill in *quat* with the minimum rotation that brings *v1* to *v2*.

6.1.3.15 **go_result** gomotion::go_cart_cart_sub (const go_cart * *v1*, const go_cart * *v2*, go_cart * *vout*)

6.1.3.16 **go_result** gomotion::go_cart_centroid (const go_cart * *varray*, go_integer *num*, go_cart * *centroid*)

6.1.3.17 **go_result** gomotion::go_cart_centroidize (const go_cart * *vinarray*, go_integer *num*, go_cart * *centroid*, go_cart * *voutarray*)

6.1.3.18 **go_result** gomotion::go_cart_cyl_convert (const go_cart * *v*, go_cyl * *c*)

6.1.3.19 **go_result** gomotion::go_cart_is_norm (const go_cart * *v*)

6.1.3.20 **go_result** gomotion::go_cart_mag (const go_cart * *v*, go_real * *d*)

6.1.3.21 **go_result** gomotion::go_cart_magsq (const go_cart * *v*, go_real * *d*)

6.1.3.22 **go_result** gomotion::go_cart_neg (const go_cart * *v1*, go_cart * *vout*)

6.1.3.23 **go_result** gomotion::go_cart_normal (const go_cart * *v*, go_cart * *vout*)

go_cart_normal finds one of the infinite vectors perpendicular to *v*, putting the result in *vout*.

6.1.3.24 **go_result** gomotion::go_cart_plane_proj (const go_cart * *v*, const go_cart * *normal*, go_cart * *vout*)

6.1.3.25 **go_result** gomotion::go_cart_rvec_convert (const go_cart * *cart*, go_rvec * *rvec*)

go_cart_rvec_convert and go_rvec_cart_convert convert between Cartesian vectors and rotation vectors. The conversion is trivial but keeps types distinct.

6.1.3.26 **go_result** gomotion::go_cart_scale_mult (const go_cart * *v1*, go_real *d*, go_cart * *vout*)

6.1.3.27 **go_result** gomotion::go_cart_sph_convert (const go_cart * *v*, go_sph * *s*)

6.1.3.28 **go_result gomotion::go_cart_trilaterate** (const go_cart * *c1*, const go_cart * *c2*, const go_cart * *c3*, go_real *l1*, go_real *l2*, go_real *l3*, go_cart * *out1*, go_cart * *out2*)

Returns the Cartesian point *p* whose distances from three other points *c1*, *c2* and *c3* are *l1*, *l2* and *l3*, respectively. In general there are 0, 1 or two points possible. If no point is possible, this returns GO_RESULT_ERROR, otherwise the points are returned in *p1* and *p2*, which may be the same point.

6.1.3.29 **go_result gomotion::go_cart_unit** (const go_cart * *v*, go_cart * *vout*)

6.1.3.30 **go_result gomotion::go_cart_vector_convert** (const go_cart * *c*, go_real * *v*)

6.1.3.31 **go_cart gomotion::go_cart_zero** (void)

Returns the zero vector.

6.1.3.32 **go_real gomotion::go_cbrt** (go_real *x*)

Returns the cube root of *x*.

6.1.3.33 **go_complex gomotion::go_complex_add** (go_complex *z1*, go_complex *z2*)

6.1.3.34 **go_real gomotion::go_complex_arg** (go_complex *z*)

6.1.3.35 **void gomotion::go_complex_cbrt** (go_complex *z*, go_complex * *z1*, go_complex * *z2*, go_complex * *z3*)

6.1.3.36 **go_complex gomotion::go_complex_div** (go_complex *z1*, go_complex *z2*, go_result * *result*)

6.1.3.37 **go_complex gomotion::go_complex_inv** (go_complex *z*, go_result * *result*)

6.1.3.38 **go_real gomotion::go_complex_mag** (go_complex *z*)

6.1.3.39 **go_real gomotion::go_complex_magsq** (go_complex *z*)

6.1.3.40 **go_complex gomotion::go_complex_mult** (go_complex *z1*, go_complex *z2*)

6.1.3.41 **go_complex gomotion::go_complex_scale** (go_complex *z*, go_real *scale*)

6.1.3.42 **go_complex gomotion::go_complex_sq** (go_complex *z*)

6.1.3.43 **void gomotion::go_complex_sqrt** (go_complex *z*, go_complex * *z1*, go_complex * *z2*)

6.1.3.44 **go_complex gomotion::go_complex_sub** (go_complex *z1*, go_complex *z2*)

6.1.3.45 **go_result gomotion::go_cubic_solve** (const go_cubic * *cub*, go_complex * *z1*, go_complex * *z2*, go_complex * *z3*)

6.1.3.46 **go_result gomotion::go_cyl_cart_convert** (const go_cyl * *c*, go_cart * *v*)

6.1.3.47 **go_result gomotion::go_cyl_sph_convert** (const go_cyl * *c*, go_sph * *s*)

6.1.3.48 **go_result** gomotion::go_dh_hom_convert (const go_dh * *dh*, go_hom * *hom*)

Converts DH parameters in *dh* to their transform equivalent, stored in *hom*.

6.1.3.49 **go_result** gomotion::go_dh_pose_convert (const go_dh * *dh*, go_pose * *pose*)

Converts DH parameters in *dh* to their pose equivalent, stored in *pose*.

6.1.3.50 **go_real** gomotion::go_get_singular_epsilon (void)

6.1.3.51 **go_result** gomotion::go_hom_dh_convert (const go_hom * *hom*, go_dh * *dh*)

Counterpart to go_pose_dh_convert, for transforms.

6.1.3.52 **go_result** gomotion::go_hom_hom_mult (const go_hom * *h1*, const go_hom * *h2*, go_hom * *hout*)

6.1.3.53 **go_result** gomotion::go_hom_inv (const go_hom * *h1*, go_hom * *h2*)

6.1.3.54 **go_result** gomotion::go_hom_pose_convert (const go_hom * *h*, go_pose * *p*)

6.1.3.55 **go_result** gomotion::go_line_evaluate (const go_line * *line*, go_real *d*, go_cart * *point*)

Fills in *point* with the point located distance *d* along *line*

6.1.3.56 **go_result** gomotion::go_line_from_planes (const go_plane * *plane1*, const go_plane * *plane2*, go_line * *line*)

Fill in *line* with the intersection of the two planes. Returns GO_RESULT_OK if the planes are not parallel, otherwise GO_RESULT_ERROR.

6.1.3.57 **go_result** gomotion::go_line_from_point_direction (const go_cart * *point*, const go_cart * *direction*, go_line * *line*)

Fills in *line* given *point* and *direction*. Returns GO_RESULT_OK if *direction* is non-zero, otherwise GO_RESULT_ERROR.

6.1.3.58 **go_result** gomotion::go_line_from_points (const go_cart * *point1*, const go_cart * *point2*, go_line * *line*)

Fills in *line* given two points. Returns GO_RESULT_OK if the points are different, otherwise GO_RESULT_ERROR.

6.1.3.59 **go_flag** gomotion::go_line_line_compare (const go_line * *line1*, const go_line * *line2*)

Returns non-zero if the lines are the same, otherwise zero.

6.1.3.60 **go_result** gomotion::go_line_plane_intersect (const go_line * *line*, const go_plane * *plane*, go_cart * *point*, go_real * *distance*)

Fills in *point* with the intersection point of *line* with *plane*, and *distance* with the distance along the line to the intersection point. Returns GO_RESULT_ERROR if the line is parallel to the plane and not lying in the plane, otherwise GO_RESULT_OK.

6.1.3.61 **go_result** gomotion::go_line_plane_proj (const go_line * *line*, const go_plane * *plane*, go_line * *proj*)

Fills in *proj* with the projection of *line* onto *plane*

6.1.3.62 **go_result** gomotion::go_linear_cos_sin_solve (go_real *a*, go_real *b*, go_real * *th1*, go_real * *th2*)

Solves $\cos \theta + b \sin \theta = 1$ for θ , two solutions

6.1.3.63 **go_result** gomotion::go_link_joint_set (const go_link * *link*, go_real *joint*, go_link * *linkout*)

Fixes the link in *link* to its value when the joint variable is *joint*, storing the result in *linkout*.

6.1.3.64 **go_result** gomotion::go_link_pose_build (const go_link * *links*, go_integer *num*, go_pose * *pose*)

Takes the link description of the device in *links*, and the number of these in *num*, and builds the pose of the device and stores in *pose*. *links* should have the value of the free link parameter filled in with the current joint value, e.g., with a prior call to `go_link_joint_set`.

6.1.3.65 **go_result** gomotion::go_mat3_inv (go_real *a*[3][3], go_real *ainv*[3][3])

6.1.3.66 **go_result** gomotion::go_mat3_mat3_mult (go_real *a*[3][3], go_real *b*[3][3], go_real *axb*[3][3])

6.1.3.67 **go_result** gomotion::go_mat3_vec3_mult (go_real *a*[3][3], go_real *v*[3], go_real *axv*[3])

6.1.3.68 **go_result** gomotion::go_mat4_inv (go_real *a*[4][4], go_real *ainv*[4][4])

6.1.3.69 **go_result** gomotion::go_mat4_mat4_mult (go_real *a*[4][4], go_real *b*[4][4], go_real *axb*[4][4])

6.1.3.70 **go_result** gomotion::go_mat4_vec4_mult (go_real *a*[4][4], go_real *v*[4], go_real *axv*[4])

6.1.3.71 **go_result** gomotion::go_mat6_inv (go_real *a*[6][6], go_real *ainv*[6][6])

Given a 6x6 matrix *a*, computes the inverse and returns it in *ainv*. Leaves *a* untouched. Returns GO_RESULT_OK if there is an inverse, else GO_RESULT_SINGULAR if the matrix is singular.

6.1.3.72 **go_result** gomotion::go_mat6_mat6_mult (go_real *a*[6][6], go_real *b*[6][6], go_real *axb*[6][6])

Given two 6x6 matrices *a* and *b*, multiplies them and returns the result in *axb*. Leaves *a* and *b* untouched. Returns GO_RESULT_OK.

6.1.3.73 **go_result** gomotion::go_mat6_transpose (go_real *a*[6][6], go_real *at*[6][6])

Given a 6x6 matrix *a*, generates the transpose and returns it in *at*. Leaves *a* untouched. Returns GO_RESULT_OK.

6.1.3.74 **go_result** gomotion::go_mat6_vec6_mult (go_real *a*[6][6], go_real *v*[6], go_real *axv*[6])

Given a 6x6 matrix *a* and a 6x1 vector *v*, multiplies them and returns the result in *axv*. Leaves *a* and *v* untouched. Returns GO_RESULT_OK.

6.1.3.75 `go_result gomotion::go_mat_cart_mult (const go_mat * m, const go_cart * v, go_cart * vout)`

6.1.3.76 `go_result gomotion::go_mat_inv (const go_mat * m, go_mat * mout)`

6.1.3.77 `go_flag gomotion::go_mat_is_norm (const go_mat * m)`

6.1.3.78 `go_result gomotion::go_mat_mat_mult (const go_mat * m1, const go_mat * m2, go_mat * mout)`

6.1.3.79 `go_result gomotion::go_mat_matrix_convert (const go_mat * mat, go_matrix * matrix)`

6.1.3.80 `go_result gomotion::go_mat_norm (const go_mat * m, go_mat * mout)`

Normalizes rotation matrix *m* so that all columns are mutually perpendicular unit vectors, placing the result in *mout*.

6.1.3.81 `go_result gomotion::go_mat_quat_convert (const go_mat * m, go_quat * q)`

6.1.3.82 `go_result gomotion::go_mat_rpy_convert (const go_mat * m, go_rpy * rpy)`

6.1.3.83 `go_result gomotion::go_mat_rvec_convert (const go_mat * m, go_rvec * r)`

6.1.3.84 `go_result gomotion::go_mat_uxz_convert (const go_mat * mat, go_uxz * uxz)`

6.1.3.85 `go_result gomotion::go_mat_xyz_convert (const go_mat * m, go_xyz * xyz)`

6.1.3.86 `go_result gomotion::go_mat_zyx_convert (const go_mat * m, go_zyx * zyx)`

6.1.3.87 `go_result gomotion::go_mat_zyz_convert (const go_mat * m, go_zyz * zyz)`

6.1.3.88 `go_result gomotion::go_matrix_inv (const go_matrix * m, go_matrix * minv)`

6.1.3.89 `go_result gomotion::go_matrix_matrix_add (const go_matrix * a, const go_matrix * b, go_matrix * apb)`

6.1.3.90 `go_result gomotion::go_matrix_matrix_copy (const go_matrix * src, go_matrix * dst)`

6.1.3.91 `go_result gomotion::go_matrix_matrix_mult (const go_matrix * a, const go_matrix * b, go_matrix * ab)`

6.1.3.92 `go_result gomotion::go_matrix_transpose (const go_matrix * a, go_matrix * at)`

6.1.3.93 `go_result gomotion::go_matrix_vector_cross (const go_matrix * a, const go_vector * v, go_matrix * axv)`

The matrix-vector cross product is a matrix of the same dimension, whose columns are the column-wise cross products of the matrix and the vector. The matrices must be 3xN, the vector 3x1.

6.1.3.94 `go_result gomotion::go_matrix_vector_mult (const go_matrix * a, const go_vector * v, go_vector * axv)`

6.1.3.95 `go_result gomotion::go_plane_evaluate (const go_plane * plane, go_real u, go_real v, go_cart * point)`

Fills in *point* with the point located distances *u* and *v* along some othogonal planar coordinate system in *plane*

6.1.3.96 **go_result** gomotion::go_plane_from_abcd (go_real *A*, go_real *B*, go_real *C*, go_real *D*, go_plane * *plane*)

Fills in *plane* given the *A*, *B*, *C* and *D* values in the canonical form $Ax + By + Cz + D = 0$. Returns GO_RESULT_OK if not all of *A*, *B* and *C* are zero, otherwise GO_RESULT_ERROR.

6.1.3.97 **go_result** gomotion::go_plane_from_point_line (const go_cart * *point*, const go_line * *line*, go_plane * *plane*)

Fills in *plane* given a *point* on the plane and a *line* in the plane. Returns GO_RESULT_OK if the point is not on the line, GO_RESULT_ERROR otherwise.

6.1.3.98 **go_result** gomotion::go_plane_from_point_normal (const go_cart * *point*, const go_cart * *normal*, go_plane * *plane*)

Fills in *plane* give a *point* on the plane and the normal *direction*.

6.1.3.99 **go_result** gomotion::go_plane_from_points (const go_cart * *point1*, const go_cart * *point2*, const go_cart * *point3*, go_plane * *plane*)

Fills in *plane* given three points. Returns GO_RESULT_OK if the points are distinct, otherwise GO_RESULT_ERROR.

6.1.3.100 **go_flag** gomotion::go_plane_plane_compare (const go_plane * *plane1*, const go_plane * *plane2*)

Returns non-zero if the planes are coincident and have the same normal direction, otherwise zero.

6.1.3.101 **go_result** gomotion::go_point_line_distance (const go_cart * *point*, const go_line * *line*, go_real * *distance*)

Fills in *distance* with the distance from *point* to *line*

6.1.3.102 **go_result** gomotion::go_point_line_proj (const go_cart * *point*, const go_line * *line*, go_cart * *pout*)

Fills in *pout* with the nearest point on *line* to *point*

6.1.3.103 **go_result** gomotion::go_point_plane_distance (const go_cart * *point*, const go_plane * *plane*, go_real * *distance*)

Fills in the *distance* from the *point* to the *plane*.

6.1.3.104 **go_result** gomotion::go_point_plane_proj (const go_cart * *point*, const go_plane * *plane*, go_cart * *proj*)

Fills in *proj* with the projection of *point* onto *plane*

6.1.3.105 **go_result** gomotion::go_pose_cart_mult (const go_pose * *p1*, const go_cart * *v2*, go_cart * *vout*)

6.1.3.106 **go_result** gomotion::go_pose_dh_convert (const go_pose * *pose*, go_dh * *dh*)

Converts *pose* to the equivalent DH parameters, stored in *dh*. Warning! Conversion from these DH parameters back to a pose via *go_dh_pose_convert* will NOT in general result in the same pose. Poses have 6 degrees of freedom, DH parameters have 4, and conversion to DH parameters loses some information. The source of this information loss is the

convention imposed on DH parameters for choice of X-Y-Z axis directions. With poses, there is no such convention, and poses are thus freer than DH parameters.

6.1.3.107 `go_result gomotion::go_pose_hom_convert (const go_pose * p, go_hom * h)`

6.1.3.108 `go_pose gomotion::go_pose_identity (void)`

Returns the identity pose, no translation or rotation.

6.1.3.109 `go_result gomotion::go_pose_inv (const go_pose * p1, go_pose * p2)`

6.1.3.110 `go_flag gomotion::go_pose_pose_compare (const go_pose * p1, const go_pose * p2)`

6.1.3.111 `go_result gomotion::go_pose_pose_interp (go_real t1, const go_pose * p1, go_real t2, const go_pose * p2, go_real t3, go_pose * p3)`

Given two times $t1$ and $t2$, and associated poses $p1$ and $p2$, interpolates (or extrapolates) to find pose $p3$ at time $t3$. The result is stored in $p3$. Returns `GO_RESULT_OK` if $t1$ and $t2$ are distinct and $p1$ and $p2$ are valid poses, otherwise it can't interpolate and returns a relevant error.

6.1.3.112 `go_result gomotion::go_pose_pose_mult (const go_pose * p1, const go_pose * p2, go_pose * pout)`

6.1.3.113 `go_result gomotion::go_pose_scale_mult (const go_pose * p1, go_real s, go_pose * pout)`

6.1.3.114 `go_pose gomotion::go_pose_this (go_real x, go_real y, go_real z, go_real rs, go_real rx, go_real ry, go_real rz)`

Convenience function that returns a [go_pose](#) given individual elements.

6.1.3.115 `go_result gomotion::go_pose_vel_mult (const go_pose * pose, const go_vel * vel, go_vel * out)`

Given $pose$ transformation from frame A to B, and a velocity vel in frame A, transform the velocity into frame B and place in out .

6.1.3.116 `go_result gomotion::go_quadratic_solve (const go_quadratic * quad, go_complex * z1, go_complex * z2)`

6.1.3.117 `go_result gomotion::go_quartic_solve (const go_quartic * quart, go_complex * z1, go_complex * z2, go_complex * z3, go_complex * z4)`

6.1.3.118 `go_result gomotion::go_quat_cart_mult (const go_quat * q1, const go_cart * v2, go_cart * vout)`

6.1.3.119 `go_quat gomotion::go_quat_identity (void)`

Returns the identity (zero) quaternion, i.e., no rotation.

6.1.3.120 `go_result gomotion::go_quat_inv (const go_quat * q1, go_quat * qout)`

6.1.3.121 `go_flag gomotion::go_quat_is_norm (const go_quat * q1)`

- 6.1.3.122 `go_result gomotion::go_quat_mag (const go_quat * quat, go_real * d)`
- 6.1.3.123 `go_result gomotion::go_quat_mat_convert (const go_quat * q, go_mat * m)`
- 6.1.3.124 `go_result gomotion::go_quat_matrix_convert (const go_quat * quat, go_matrix * matrix)`
- 6.1.3.125 `go_result gomotion::go_quat_norm (const go_quat * q1, go_quat * qout)`
- 6.1.3.126 `go_flag gomotion::go_quat_quat_compare (const go_quat * q1, const go_quat * q2)`
- 6.1.3.127 `go_result gomotion::go_quat_quat_mult (const go_quat * q1, const go_quat * q2, go_quat * qout)`
- 6.1.3.128 `go_result gomotion::go_quat_rpy_convert (const go_quat * q, go_rpy * rpy)`
- 6.1.3.129 `go_result gomotion::go_quat_rvec_convert (const go_quat * q, go_rvec * r)`
- 6.1.3.130 `go_result gomotion::go_quat_scale_mult (const go_quat * q, go_real s, go_quat * qout)`
- 6.1.3.131 `go_result gomotion::go_quat_unit (const go_quat * q, go_quat * qout)`

`go_quat_unit` takes a quaternion rotation *q* and converts it into a unit rotation about the same axis, *qout*.

- 6.1.3.132 `go_result gomotion::go_quat_xyz_convert (const go_quat * , go_xyz *)`
- 6.1.3.133 `go_result gomotion::go_quat_zyx_convert (const go_quat * q, go_zyx * zyx)`
- 6.1.3.134 `go_result gomotion::go_quat_zyz_convert (const go_quat * q, go_zyz * zyz)`
- 6.1.3.135 `go_result gomotion::go_rpy_cart_mult (const go_rpy * rpy, const go_cart * in, go_cart * out)`
- 6.1.3.136 `go_result gomotion::go_rpy_mat_convert (const go_rpy * rpy, go_mat * m)`
- 6.1.3.137 `go_result gomotion::go_rpy_quat_convert (const go_rpy * rpy, go_quat * quat)`
- 6.1.3.138 `go_result gomotion::go_rpy_rvec_convert (const go_rpy * rpy, go_rvec * rvec)`
- 6.1.3.139 `go_result gomotion::go_rpy_xyz_convert (const go_rpy * , go_xyz *)`
- 6.1.3.140 `go_result gomotion::go_rpy_zyx_convert (const go_rpy * rpy, go_zyx * zyx)`
- 6.1.3.141 `go_result gomotion::go_rpy_zyz_convert (const go_rpy * rpy, go_zyz * zyz)`
- 6.1.3.142 `go_result gomotion::go_rvec_cart_convert (const go_rvec * rvec, go_cart * cart)`
- 6.1.3.143 `go_result gomotion::go_rvec_mat_convert (const go_rvec * r, go_mat * m)`
- 6.1.3.144 `go_result gomotion::go_rvec_quat_convert (const go_rvec * r, go_quat * q)`
- 6.1.3.145 `go_result gomotion::go_rvec_rpy_convert (const go_rvec * r, go_rpy * rpy)`
- 6.1.3.146 `go_flag gomotion::go_rvec_rvec_compare (const go_rvec * r1, const go_rvec * r2)`

6.1.3.147 `go_result gomotion::go_rvec_scale_mult (const go_rvec * r, go_real s, go_rvec * rout)`

6.1.3.148 `go_result gomotion::go_rvec_xyz_convert (const go_rvec * , go_xyz *)`

6.1.3.149 `go_result gomotion::go_rvec_zyx_convert (const go_rvec * rvec, go_zyx * zyx)`

6.1.3.150 `go_result gomotion::go_rvec_zyz_convert (const go_rvec * rvec, go_zyz * zyz)`

6.1.3.151 `go_result gomotion::go_set_singular_epsilon (go_real epsilon)`

6.1.3.152 `void gomotion::go_sincos (go_real x, go_real * s, go_real * c)`

Returns the sine and cosine of x (in radians) in s and c , respectively. Implemented as a single call when supported, to speed up the calculation of the two values.

6.1.3.153 `go_result gomotion::go_sph_cart_convert (const go_sph * s, go_cart * v)`

6.1.3.154 `go_result gomotion::go_sph_cyl_convert (const go_sph * s, go_cyl * c)`

6.1.3.155 `go_result gomotion::go_traj_ca_compute (go_real d, go_real v, go_real a, go_traj_ca_spec * pts)` `[inline]`

`go_traj_ca_compute()` takes values for distance 'd' to move, max velocity 'v' to limit if necessary, and constant accel 'a', and fills in the `go_traj_ca_spec` with the interval parameters.

cruise phase

no cruise phase

cruise phase

no cruise phase

6.1.3.156 `go_result gomotion::go_traj_ca_extend (const go_traj_ca_spec * ts, go_real t, go_traj_ca_spec * pts)`
`[inline]`

`go_traj_ca_extend()` takes a time 't' for the desired time to finish the motion, and extends the constant-speed section so that it stops then.

The time must be shorter than the original time, and longer than the fastest stopping time.

6.1.3.157 `go_result gomotion::go_traj_ca_generate (go_real acc, go_real deltacc, go_real deltvel, go_traj_ca_spec * pts)`
`[inline]`

`go_traj_ca_generate()` takes an accel value, and intervals for the accel period and cruise period, and fills in the `go_traj_ca_spec` with the interval parameters.

constant acceleration functions

This is useful for generating test cases. any time intervals less than 0. mean 0.

compute intermediate values for end of each phase

any time intervals less than 0. mean 0.

compute intermediate values for end of each phase

6.1.3.158 `go_result gomotion::go_traj_ca_interp (const go_traj_ca_spec * ts, go_real t, go_traj_interp_spec * ti)`
`[inline]`

`go_traj_ca_interp()` takes a `go_traj_ca_spec` and interpolates the d-v-a values for the given time t, storing the d-v-a values in ti.

run interpolation on each phase for each motion parameter, using

$$d = d_0 + v_0 t + 1/2 a_0 t^2 \quad v = v_0 + a_0 t \quad a = a_0 \quad j = 0$$

where d0, v0, a0 are the initial values for each phase

6.1.3.159 `go_result gomotion::go_traj_ca_scale (const go_traj_ca_spec * ts, go_real t, go_traj_ca_spec * pts)` `[inline]`

`go_traj_ca_scale()` takes a time 't' for the desired time of the motion, and scales the times and motion params so that the total distance remains the same and everything else is in proportion

can't shrink motion

this just adds a cruise phase to stretch the time, and reduces

the acc value accordingly

can't shrink motion

this just adds a cruise phase to stretch the time, and reduces

the acc value accordingly

6.1.3.160 `go_result gomotion::go_traj_ca_stop (const go_traj_ca_spec * ts, go_real t, go_traj_ca_spec * pts)` `[inline]`

`go_traj_ca_stop()` takes a time 't' for the desired time to begin stopping, and recomputes the times so that the move will stop as soon as possible during subsequent interps.

stage I

null stage II, same as I

stage III

stage I unchanged

stage II

stage III

already in stage III, do nothing

stage I

null stage II, same as I

stage III

stage I unchanged

stage II

stage III

already in stage III, do nothing

6.1.3.161 `go_result gomotion::go_traj_cj_compute (go_real d, go_real v, go_real a, go_real j, go_traj_cj_spec * pts)`
`[inline]`

`go_traj_cj_compute()` takes values for distance 'd' to move, max velocity 'v' to limit if necessary, max accel 'a' to limit if necessary, and constant jerk 'j', and fills in the `go_traj_cj_spec` with the interval parameters.

! Given a total move distance *d*, maximum peak speed *v*, maximum peak acceleration *a* and jerk *j*, compute the times for each of the seven phases (jerk, accel, dejerk, cruise, dejerk, decel and jerk), and the associated cumulative distances, instantaneous speeds and accelerations.

The resulting structure *pts* will be used by the constant-jerk interpolater `go_traj_cj_interp` repeatedly when evaluating the motion along this trajectory. fill the targets in, and they will be recomputed if needed

accel phase

cruise phase

no cruise phase

no accel phase

cruise phase

no cruise phase

fill the targets in, and they will be recomputed if needed

accel phase

cruise phase

no cruise phase

no accel phase

cruise phase

no cruise phase

6.1.3.162 `go_result gomotion::go_traj_cj_compute_tees_a_c (go_traj_cj_spec * pts)` `[inline]`

! Computes the trajectory defining times *t1*, *t2* and *t4* for a motion with no acceleration but a cruise phase, given *pts* filled in with *dtend*, *jt0*, *at1* and *vt3*.

A subsequent call to `go_traj_cj_compute_the_rest` will fill in the remaining parameters which will be used repeatedly by `go_traj_cj_interp`.

6.1.3.163 `go_result gomotion::go_traj_cj_compute_tees_a_nc (go_traj_cj_spec * pts)` `[inline]`

! Computes the trajectory defining times *t1*, *t2* and *t4* for a motion with acceleration but no cruise phase, given *pts* filled in with *dtend*, *jt0* and *at1*.

A subsequent call to `go_traj_cj_compute_the_rest` will fill in the remaining parameters which will be used repeatedly by `go_traj_cj_interp`.

6.1.3.164 `go_result gomotion::go_traj_cj_compute_tees_na_c (go_traj_cj_spec * pts)` `[inline]`

! Computes the trajectory defining times *t1*, *t2* and *t4* for a motion with no acceleration but a cruise phase, given *pts* filled in with *dtend*, *jt0* and *vt3*.

A subsequent call to `go_traj_cj_compute_the_rest` will fill in the remaining parameters which will be used repeatedly by `go_traj_cj_interp`.

6.1.3.165 `go_result gomotion::go_traj_cj_compute_tees_na_nc (go_traj_cj_spec * pts) [inline]`

! Computes the trajectory defining times $t1$, $t2$ and $t4$ for a motion with no acceleration or cruise phase, given pts filled in with $dtend$ and $jt0$.

A subsequent call to `go_traj_cj_compute_the_rest` will fill in the remaining parameters which will be used repeatedly by `go_traj_cj_interp`.

6.1.3.166 `go_result gomotion::go_traj_cj_compute_the_rest (go_traj_cj_spec * pts) [inline]`

! Expects pts attributes $dtend$, $jt0$, $at1$, $vt3$, $t1$, $t2$ and $t3$ to be already filled in from previous calls to one of the four `go_traj_cj_compute_tees` functions and `go_traj_cj_compute`.

Fills in all the remaining values in pts : the times for each of the seven phases, the associated cumulative distances, and the instantaneous speeds and accelerations.

6.1.3.167 `go_result gomotion::go_traj_cj_extend (const go_traj_cj_spec * ts, go_real t, go_traj_cj_spec * pts) [inline]`

`go_traj_cj_extend()` takes a time 't' for the desired time to finish the motion, and extends the constant-speed section so that it stops then.

The time must be shorter than the original time, and longer than the fastest stopping time.

6.1.3.168 `go_result gomotion::go_traj_cj_generate (go_real jrk, go_real deltjrk, go_real deltacc, go_real deltvel, go_traj_cj_spec * pts) [inline]`

`go_traj_cj_generate()` takes a jerk value, and intervals for the jerk period, accel period and cruise period, and fills in the `go_traj_cj_spec` with the interval parameters.

constant jerk functions

This is useful for generating test cases. any time intervals less than 0. mean 0.

compute intermediate values for end of each phase

any time intervals less than 0. mean 0.

compute intermediate values for end of each phase

6.1.3.169 `go_result gomotion::go_traj_cj_interp (const go_traj_cj_spec * ts, go_real t, go_traj_interp_spec * ti) [inline]`

`go_traj_cj_interp()` takes a `go_traj_cj_spec` and interpolates the d-v-a-j values for the given time t, storing the d-v-a-j values in ti .

run interpolation on each phase for each motion parameter, using

$$d = d_0 + v_0 t + 1/2 a_0 t^2 + 1/6 j_0 t^3 \quad v = v_0 + a_0 t + 1/2 j_0 t^2 \quad a = a_0 + j_0 t \quad j = j_0$$

where d_0 , v_0 , a_0 , j_0 are the initial values for each phase

6.1.3.170 `go_result gomotion::go_traj_cj_scale (const go_traj_cj_spec * ts, go_real t, go_traj_cj_spec * pts)` `[inline]`

`go_traj_cj_scale()` takes a time '*t*' for the desired time of the motion, and scales the times and motion params so that the total distance remains the same and everything else is in proportion

can't shrink shorter

can't shrink shorter

6.1.3.171 `go_result gomotion::go_traj_cj_stop (const go_traj_cj_spec * ts, go_real t, go_traj_cj_spec * pts)` `[inline]`

`go_traj_cj_stop()` takes a time '*t*' for the desired time to begin stopping, and recomputes the times so that the move will stop as soon as possible during subsequent interps.

incremental phase II distance

incremental phase III distance

incremental phase IV time

incremental phase IV distance

stopping in stage I

adjust stage I

null stage II, same as I

stage III

null stage IV, same as III

stage V

do earlier

so we can use it here

null stage VI, same as V

stage VII

stopping in stage II

stage I unchanged

adjust stage II

stage III

null stage IV, same as III

stage V

stage VI

stage VII

stopping in stage III

stages I, II and III unchanged

null stage IV

take off original stage IV for remaining stages

stopping in stage IV

stages I, II and III unchanged

adjusted stage IV
 stopping in stage V, VI or VII– leave alone
 incremental phase II distance
 incremental phase III distance
 incremental phase IV time
 incremental phase IV distance
 stopping in stage I
 adjust stage I
 null stage II, same as I
 stage III
 null stage IV, same as III
 stage V
 do earlier
 so we can use it here
 null stage VI, same as V
 stage VII
 stopping in stage II
 stage I unchanged
 adjust stage II
 stage III
 null stage IV, same as III
 stage V
 stage VI
 stage VII
 stopping in stage III
 stages I, II and III unchanged
 null stage IV
 take off original stage IV for remaining stages
 stopping in stage IV
 stages I, II and III unchanged
 adjusted stage IV
 stopping in stage V, VI or VII– leave alone

6.1.3.172 `go_result gomotion::go_tridiag_ql (go_real * d, go_real * e, go_integer n, go_real ** z)`

This computes the eigenvalues and eigenvectors of the tridiagonalized matrix with diagonal *d* and off-diagonal *e*, using QL reduction with implicit shifts. *d*, *e* and *z* should have been generated by a prior call to `go_tridiag_reduce`. *d* will contain the eigenvalues, and *z* will contain the eigenvectors.

Calling example:

```
go_tridiag_reduce(matrix, 3, d, e); go_tridiag_ql(d, e, 3, matrix);
```

d will contain the eigenvalues of the original *matrix*, and the new *matrix* will contain the eigenvectors.

6.1.3.173 `go_result gomotion::go_tridiag_reduce (go_real ** a, go_integer n, go_real * d, go_real * e)`

This computes the Householder reduction to tridiagonal form of a real symmetric matrix *a*. *a* will contain the orthogonal matrix that effects the transformation. *d* will contain the diagonal elements, and *e* will contain the off-diagonal elements. This is used by `go_tridiag_ql` to find the eigenvalues and eigenvectors of a real symmetric matrix.

6.1.3.174 `go_result gomotion::go_uxz_mat_convert (const go_uxz * uxz, go_mat * mat)`

6.1.3.175 `go_result gomotion::go_vector_cart_convert (const go_real * v, go_cart * c)`

6.1.3.176 `go_result gomotion::go_xyz_mat_convert (const go_xyz * xyz, go_mat * m)`

6.1.3.177 `go_result gomotion::go_xyz_quat_convert (const go_xyz * , go_quat *)`

6.1.3.178 `go_result gomotion::go_xyz_rpy_convert (const go_xyz * , go_rpy *)`

6.1.3.179 `go_result gomotion::go_xyz_rvec_convert (const go_xyz * , go_rvec *)`

6.1.3.180 `go_result gomotion::go_xyz_zyx_convert (const go_xyz * , go_zyx *)`

6.1.3.181 `go_result gomotion::go_xyz_zyz_convert (const go_xyz * , go_zyz *)`

6.1.3.182 `go_result gomotion::go_zyx_mat_convert (const go_zyx * zyx, go_mat * m)`

6.1.3.183 `go_result gomotion::go_zyx_quat_convert (const go_zyx * zyx, go_quat * q)`

6.1.3.184 `go_result gomotion::go_zyx_rpy_convert (const go_zyx * zyx, go_rpy * rpy)`

6.1.3.185 `go_result gomotion::go_zyx_rvec_convert (const go_zyx * zyx, go_rvec * r)`

6.1.3.186 `go_result gomotion::go_zyx_xyz_convert (const go_zyx * , go_xyz *)`

6.1.3.187 `go_result gomotion::go_zyx_zyz_convert (const go_zyx * zyx, go_zyz * zyz)`

6.1.3.188 `go_result gomotion::go_zyz_mat_convert (const go_zyz * zyz, go_mat * m)`

6.1.3.189 `go_result gomotion::go_zyz_quat_convert (const go_zyz * zyz, go_quat * q)`

6.1.3.190 `go_result gomotion::go_zyz_rpy_convert (const go_zyz * zyz, go_rpy * rpy)`

6.1.3.191 `go_result gomotion::go_zyz_rvec_convert (const go_zyz * zyz, go_rvec * r)`

6.1.3.192 `go_result gomotion::go_zyz_xyz_convert (const go_zyz * , go_xyz *)`

6.1.3.193 `go_result gomotion::go_zyz_zyx_convert (const go_zyz * zyz, go_zyx * zyx)`

6.1.3.194 `go_result gomotion::lubksb (go_real ** a, go_integer n, go_integer * indx, go_real * b)`

6.1.3.195 `go_result gomotion::ludcmp (go_real ** a, go_real * scratchrow, go_integer n, go_integer * indx, go_real * d)`

6.1.3.196 `void gomotion::sincos (double x, double * sx, double * cx)`

Chapter 7

Data Structure Documentation

7.1 gomotion::coeff Struct Reference

This structure holds the polynomial coefficients.

```
#include <gointerp.h>
```

Data Fields

- [go_real a \[COEFF_MAX\]](#)

7.1.1 Detailed Description

This structure holds the polynomial coefficients.

In the derivations below, coefficients a,b,c,d,... are used. Here, they correspond to the a[] array like this:

a[0] = 0th order coeff, e.g., 'd' for cubic a[1] = 1st order coeff, e.g., 'c' for cubic ...

7.1.2 Field Documentation

7.1.2.1 [go_real gomotion::coeff::a\[COEFF_MAX\]](#)

The documentation for this struct was generated from the following file:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gointerp.h](#)

7.2 gomotion::go_body Struct Reference

```
#include <gomath.h>
```

Data Fields

- [go_real mass](#)
- [go_real inertia \[3\]\[3\]](#)

7.2.1 Detailed Description

Rigid body

7.2.2 Field Documentation

7.2.2.1 `go_real gomotion::go_body::inertia[3][3]`

The *inertia* matrix is the 3x3 matrix of moments of inertia with respect to the body's origin.

7.2.2.2 `go_real gomotion::go_body::mass`

total mass of the rigid body

The documentation for this struct was generated from the following file:

- `/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h`

7.3 `gomotion::go_cart` Struct Reference

```
#include <gomath.h>
```

Data Fields

- `go_real x`
- `go_real y`
- `go_real z`

7.3.1 Detailed Description

A point or vector in Cartesian coordinates.

7.3.2 Field Documentation

7.3.2.1 `go_real gomotion::go_cart::x`

7.3.2.2 `go_real gomotion::go_cart::y`

7.3.2.3 `go_real gomotion::go_cart::z`

The documentation for this struct was generated from the following file:

- `/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h`

7.4 `gomotion::go_complex` Struct Reference

```
#include <gomath.h>
```

Data Fields

- [go_real re](#)
- [go_real im](#)

7.4.1 Field Documentation

7.4.1.1 [go_real gomotion::go_complex::im](#)

7.4.1.2 [go_real gomotion::go_complex::re](#)

The documentation for this struct was generated from the following file:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h](#)

7.5 gomotion::go_cubic Struct Reference

```
#include <gomath.h>
```

Data Fields

- [go_real a](#)
- [go_real b](#)
- [go_real c](#)

7.5.1 Field Documentation

7.5.1.1 [go_real gomotion::go_cubic::a](#)

7.5.1.2 [go_real gomotion::go_cubic::b](#)

7.5.1.3 [go_real gomotion::go_cubic::c](#)

The documentation for this struct was generated from the following file:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h](#)

7.6 gomotion::go_cyl Struct Reference

```
#include <gomath.h>
```

Data Fields

- [go_real theta](#)
- [go_real r](#)
- [go_real z](#)

7.6.1 Detailed Description

A point or vector in cylindrical coordinates.

7.6.2 Field Documentation

7.6.2.1 `go_real gomotion::go_cyl::r`

7.6.2.2 `go_real gomotion::go_cyl::theta`

7.6.2.3 `go_real gomotion::go_cyl::z`

The documentation for this struct was generated from the following file:

- `/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h`

7.7 gomotion::go_dh Struct Reference

```
#include <gomath.h>
```

Data Fields

- `go_real a`
- `go_real alpha`
- `go_real d`
- `go_real theta`

7.7.1 Detailed Description

These DH parameters follow the convention in John J. Craig, *Introduction to Robotics: Mechanics and Control*.

7.7.2 Field Documentation

7.7.2.1 `go_real gomotion::go_dh::a`

`a[i-1]`

7.7.2.2 `go_real gomotion::go_dh::alpha`

`alpha[i-1]`

7.7.2.3 `go_real gomotion::go_dh::d`

`d[i]`

7.7.2.4 go_real gomotion::go_dh::theta

theta[i]

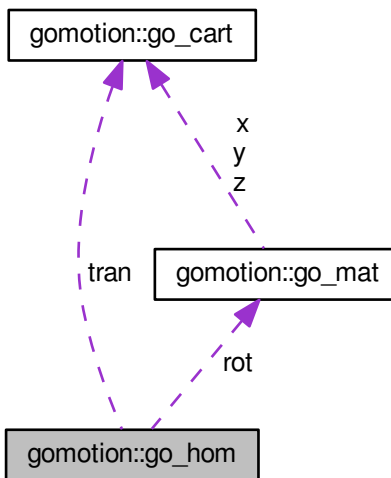
The documentation for this struct was generated from the following file:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h](#)

7.8 gomotion::go_hom Struct Reference

```
#include <gomath.h>
```

Collaboration diagram for gomotion::go_hom:



Data Fields

- [go_cart tran](#)
- [go_mat rot](#)

7.8.1 Field Documentation

7.8.1.1 go_mat gomotion::go_hom::rot

7.8.1.2 go_cart gomotion::go_hom::tran

The documentation for this struct was generated from the following file:

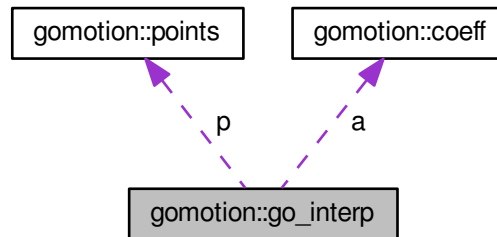
- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h](#)

7.9 gomotion::go_interp Class Reference

The interpolator structure.

```
#include <gointerp.h>
```

Collaboration diagram for gomotion::go_interp:



Public Types

- typedef `go_result`(* `add_func`)(`go_real` pos)
- typedef `go_real`(* `eval_func`)(const `go_real` t)

Public Member Functions

- `go_result` init ()
- `go_result` set_here (`go_real` here)
- `go_result` calc_coeff_constant (const `points` *p, `coeff` *a)
- `go_result` calc_coeff_linear (const `points` *p, `coeff` *a)
- `go_result` calc_coeff_cubic_bc (const `points` *p, `coeff` *a)
- `go_result` calc_coeff_cubic_pf (const `points` *p, `coeff` *a)
- `go_result` calc_coeff_quintic_bc (const `points` *p, `coeff` *a)
- `go_result` calc_coeff_quintic_pf (const `points` *p, `coeff` *a)
- `go_result` calc_coeff_bc (`go_integer` order, const `points` *p, `coeff` *a)
- `go_result` calc_coeff_pf (`go_integer` order, const `points` *p, `coeff` *a)
- `go_real` eval_constant (`go_real` t)
- `go_real` eval_linear (`go_real` t)
- `go_real` eval_cubic (`go_real` t)
- `go_real` eval_quintic (`go_real` t)
- `go_result` add_constant (`go_real` pos)
- `go_result` add_linear (`go_real` pos)
- `go_result` add_cubic_pv (`go_real` pos, `go_real` vel)
- `go_result` add_cubic_pdv (`go_real` pos)
- `go_result` add_cubic_pf (`go_real` pos)
- `go_result` add_quintic_pva (`go_real` pos, `go_real` vel, `go_real` acc)
- `go_result` add_quintic_pvda (`go_real` pos, `go_real` vel)

- [go_result add_quintic_pdva \(go_real pos\)](#)
- [go_result add_quintic_pf \(go_real pos\)](#)

Data Fields

- [coeff a](#)
- [points p](#)

7.9.1 Detailed Description

The interpolator structure.

For constant- and linear interpolation, exact-fit and boundary interpolation is the same, so we only have one type for these. Here we pass single points in succession, and interpolation is done at/between them.

For higher-order interpolation, we have two types, exact-fit and boundary. For exact-fit, we pass single points in succession, and interpolation is done in the middle interval. For boundary, we pass n-tuples of pos, vel, accel, etc. and interpolation is done between each tuple.

A variation on boundary interpolation is possible if the derivative values are not known but are estimated by differencing. The functions below suffixed `_est` are used for this.

7.9.2 Member Typedef Documentation

7.9.2.1 `typedef go_result(* gomotion::go_interp::add_func)(go_real pos)`

7.9.2.2 `typedef go_real(* gomotion::go_interp::eval_func)(const go_real t)`

7.9.3 Member Function Documentation

7.9.3.1 `go_result gomotion::go_interp::add_constant (go_real pos)`

7.9.3.2 `go_result gomotion::go_interp::add_cubic_pdv (go_real pos)`

7.9.3.3 `go_result gomotion::go_interp::add_cubic_pf (go_real pos)`

7.9.3.4 `go_result gomotion::go_interp::add_cubic_pv (go_real pos, go_real vel)`

7.9.3.5 `go_result gomotion::go_interp::add_linear (go_real pos)`

7.9.3.6 `go_result gomotion::go_interp::add_quintic_pdva (go_real pos)`

7.9.3.7 `go_result gomotion::go_interp::add_quintic_pf (go_real pos)`

7.9.3.8 `go_result gomotion::go_interp::add_quintic_pva (go_real pos, go_real vel, go_real acc)`

7.9.3.9 `go_result gomotion::go_interp::add_quintic_pvda (go_real pos, go_real vel)`

7.9.3.10 `go_result gomotion::go_interp::calc_coeff_bc (go_integer order, const points * p, coeff * a)`

7.9.3.11 `go_result gomotion::go_interp::calc_coeff_constant (const points * p, coeff * a)`

7.9.3.12 `go_result gomotion::go_interp::calc_coeff_cubic_bc (const points * p, coeff * a)`

7.9.3.13 `go_result gomotion::go_interp::calc_coeff_cubic_pf (const points * p, coeff * a)`

7.9.3.14 `go_result gomotion::go_interp::calc_coeff_linear (const points * p, coeff * a)`

7.9.3.15 `go_result gomotion::go_interp::calc_coeff_pf (go_integer order, const points * p, coeff * a)`

7.9.3.16 `go_result gomotion::go_interp::calc_coeff_quintic_bc (const points * p, coeff * a)`

7.9.3.17 `go_result gomotion::go_interp::calc_coeff_quintic_pf (const points * p, coeff * a)`

7.9.3.18 `go_real gomotion::go_interp::eval_constant (go_real t)`

7.9.3.19 `go_real gomotion::go_interp::eval_cubic (go_real t)`

7.9.3.20 `go_real gomotion::go_interp::eval_linear (go_real t)`

7.9.3.21 `go_real gomotion::go_interp::eval_quintic (go_real t)`

7.9.3.22 `go_result gomotion::go_interp::init ()`

7.9.3.23 `go_result gomotion::go_interp::set_here (go_real here)`

7.9.4 Field Documentation

7.9.4.1 `coeff gomotion::go_interp::a`

7.9.4.2 `points gomotion::go_interp::p`

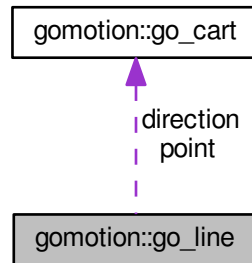
The documentation for this class was generated from the following files:

- `/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gointerp.h`
- `/usr/local/michalos/nistfanuc_ws/src/gomotion/src/gointerp.cpp`

7.10 gomotion::go_line Struct Reference

```
#include <gomath.h>
```

Collaboration diagram for gomotion::go_line:



Data Fields

- [go_cart point](#)
- [go_cart direction](#)

7.10.1 Detailed Description

Lines are represented in point-direction form (point p, direction v) as $(x - px)/vx = (y - py)/vy = (z - pz)/vz$

7.10.2 Field Documentation

7.10.2.1 [go_cart](#) gomotion::go_line::direction

7.10.2.2 [go_cart](#) gomotion::go_line::point

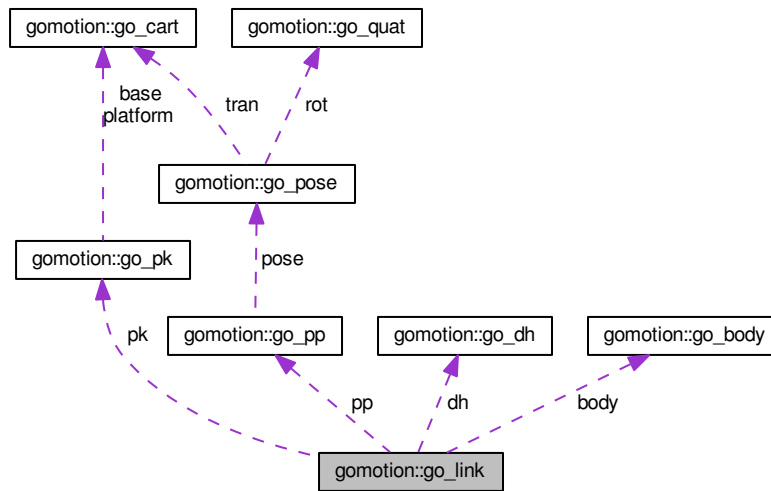
The documentation for this struct was generated from the following file:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h](#)

7.11 gomotion::go_link Struct Reference

```
#include <gomath.h>
```

Collaboration diagram for gomotion::go_link:



Data Fields

- union {
 go_dh dh
 go_pk pk
 go_pp pp
 } u
- go_body body
- go_flag type
- go_flag quantity

7.11.1 Detailed Description

This is the generic link structure for PKM sliding/cable links and serial revolute/prismatic links.

7.11.2 Field Documentation

7.11.2.1 go_body gomotion::go_link::body

the link's rigid body parameters

7.11.2.2 go_dh gomotion::go_link::dh

if you have DH params and don't want to convert to PP

7.11.2.3 go_pk gomotion::go_link::pk

if you have a parallel machine, e.g., hexapod or robot crane

7.11.2.4 go_pp gomotion::go_link::pp

if you have a serial machine, e.g., an industrial robot

7.11.2.5 go_flag gomotion::go_link::quantity

one of GO_QUANTITY_LENGTH,ANGLE

7.11.2.6 go_flag gomotion::go_link::type

one of GO_LINK_DH,PK,PP

7.11.2.7 union { ... } gomotion::go_link::u

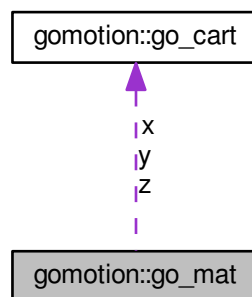
The documentation for this struct was generated from the following file:

- /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/[gomath.h](#)

7.12 gomotion::go_mat Struct Reference

```
#include <gomath.h>
```

Collaboration diagram for gomotion::go_mat:



Data Fields

- [go_cart x](#)

- [go_cart y](#)
- [go_cart z](#)

7.12.1 Detailed Description

A rotation matrix.

7.12.2 Field Documentation

7.12.2.1 `go_cart gomotion::go_mat::x`

X unit vector

7.12.2.2 `go_cart gomotion::go_mat::y`

Y unit vector

7.12.2.3 `go_cart gomotion::go_mat::z`

Z unit vector

The documentation for this struct was generated from the following file:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h](#)

7.13 `gomotion::go_matrix` Struct Reference

```
#include <gomath.h>
```

Data Fields

- [go_integer rows](#)
- [go_integer cols](#)
- [go_real ** el](#)
- [go_real ** elcpy](#)
- [go_real * v](#)
- [go_integer * index](#)

7.13.1 Field Documentation

7.13.1.1 `go_integer gomotion::go_matrix::cols`

7.13.1.2 `go_real** gomotion::go_matrix::el`

7.13.1.3 `go_real** gomotion::go_matrix::elcpy`

7.13.1.4 `go_integer*` `gomotion::go_matrix::index`

7.13.1.5 `go_integer` `gomotion::go_matrix::rows`

7.13.1.6 `go_real*` `gomotion::go_matrix::v`

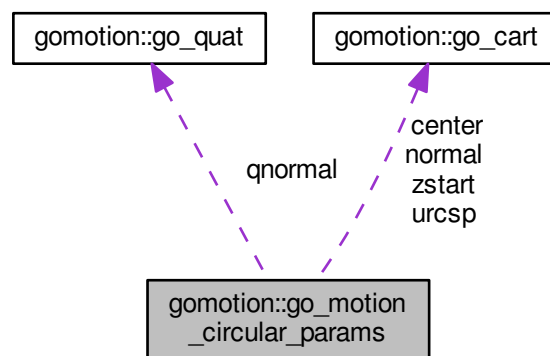
The documentation for this struct was generated from the following file:

- `/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h`

7.14 gomotion::go_motion_circular_params Struct Reference

```
#include <gomotion.h>
```

Collaboration diagram for `gomotion::go_motion_circular_params`:



Data Fields

- `go_cart` `center`
- `go_cart` `normal`
- `go_quat` `qnormal`
- `go_cart` `urcsp`
- `go_real` `rstart`
- `go_cart` `zstart`
- `go_real` `thtot`
- `go_real` `rtot`
- `go_real` `ztot`
- `go_real` `stotinv`
- `go_integer` `turns`

7.14.1 Field Documentation

7.14.1.1 `go_cart gomotion::go_motion_circular_params::center`

The vector to the circle center.

7.14.1.2 `go_cart gomotion::go_motion_circular_params::normal`

The normal vector that defines the plane of the circle.

7.14.1.3 `go_quat gomotion::go_motion_circular_params::qnormal`

The normal vector expressed as a unit rotation.

7.14.1.4 `go_real gomotion::go_motion_circular_params::rstart`

The starting radius.

7.14.1.5 `go_real gomotion::go_motion_circular_params::rtot`

The signed displacement from start to end projected radii

7.14.1.6 `go_real gomotion::go_motion_circular_params::stotinv`

The inverse of total approximate arc length. If negative, no translation motion is taking place.

7.14.1.7 `go_real gomotion::go_motion_circular_params::thtot`

The signed total angular displacement around normal vector

7.14.1.8 `go_integer gomotion::go_motion_circular_params::turns`

The number of turns in the circle. 0 means partial CCW, -1 means partial CW, otherwise more turns are added in each direction.

7.14.1.9 `go_cart gomotion::go_motion_circular_params::urcsp`

The unit vector from center to start, projected onto the normal plane.

7.14.1.10 `go_cart gomotion::go_motion_circular_params::zstart`

The vector from the normal plane to the start.

7.14.1.11 go_real gomotion::go_motion_circular_params::ztot

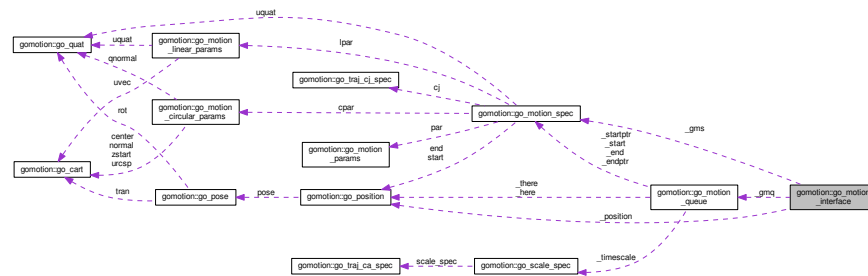
The signed displacement from start to end z off-normals

The documentation for this struct was generated from the following file:

- /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomotion.h

7.15 gomotion::go_motion_interface Struct Reference

Collaboration diagram for gomotion::go_motion_interface:



Data Fields

- int `_type`
- double `_deltat`
- `go_motion_spec` `_gms`
- `std::vector< go_motion_spec >` `_space`
- `go_motion_queue` `_gmq`
- `go_position` `_position`
- `size_t` `_queuesize`

Static Public Attributes

- static `size_t` `_id = 0`

7.15.1 Field Documentation

7.15.1.1 double gomotion::go_motion_interface::_deltat

7.15.1.2 go_motion_queue gomotion::go_motion_interface::_gmq

7.15.1.3 go_motion_spec gomotion::go_motion_interface::_gms

7.15.1.4 size_t gomotion::go_motion_interface::_id = 0 [static]

7.15.1.5 go_position gomotion::go_motion_interface::_position

7.15.1.6 `size_t gomotion::go_motion_interface::_queuesize`

7.15.1.7 `std::vector<go_motion_spec> gomotion::go_motion_interface::_space`

7.15.1.8 `int gomotion::go_motion_interface::_type`

The documentation for this struct was generated from the following file:

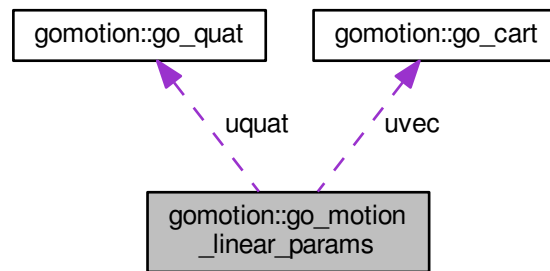
- [/usr/local/michalos/nistfanuc_ws/src/gomotion/src/gomove.cpp](#)

7.16 gomotion::go_motion_linear_params Struct Reference

In the following comments, LIN, CIR and ALL refer to linear moves, circular moves and both, respectively.

```
#include <gomotion.h>
```

Collaboration diagram for gomotion::go_motion_linear_params:



Data Fields

- [go_cart uvec](#)
- [go_quat uquat](#)

7.16.1 Detailed Description

In the following comments, LIN, CIR and ALL refer to linear moves, circular moves and both, respectively.

If they are not in parens, e.g., ALL:, you need to provide them for that type of motion. If they are in parens, e.g., (CIR), they are computed for you.

7.16.2 Field Documentation

7.16.2.1 `go_quat gomotion::go_motion_linear_params::uquat`

7.16.2.2 go_cart gomotion::go_motion_linear_params::uvec

The documentation for this struct was generated from the following file:

- /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/[gomotion.h](#)

7.17 gomotion::go_motion_params Struct Reference

```
#include <gomotion.h>
```

Data Fields

- [go_real vel](#)
max vel for each motion
- [go_real acc](#)
max accel for each motion
- [go_real jerk](#)
max jerk for each motion

7.17.1 Field Documentation

7.17.1.1 go_real gomotion::go_motion_params::acc

max accel for each motion

7.17.1.2 go_real gomotion::go_motion_params::jerk

max jerk for each motion

7.17.1.3 go_real gomotion::go_motion_params::vel

max vel for each motion

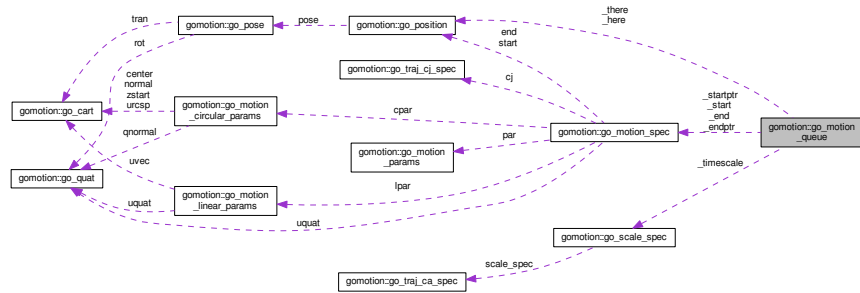
The documentation for this struct was generated from the following file:

- /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/[gomotion.h](#)

7.18 gomotion::go_motion_queue Struct Reference

```
#include <gomotion.h>
```

Collaboration diagram for gomotion::go_motion_queue:



Public Member Functions

- `go_result init (go_motion_spec *space, go_integer size, go_real deltat)`
- `go_result reset ()`
- `go_result set_type (go_flag type)`
- `go_flag get_type ()`
- `go_result set_joint_number (go_integer joints)`
- `go_integer get_joint_number ()`
- `go_result set_here (const go_position *here)`
- `go_result set_cycle_time (go_real deltat)`
- `go_result set_scale (go_real scale, go_real scale_v, go_real scale_a)`
- `go_result append (const go_motion_spec *motion)`
- `go_result number (go_integer *number)`
- `go_result size (go_integer *size)`
- `go_result head (go_motion_spec *motion)`
- `go_result here (go_position *position)`
- `go_result there (go_position *position)`
- `go_result interp (go_position *position)`
- `go_result stop (go_motion_queue *queue)`
- `go_result set_id (go_integer id)`
- `go_integer last_id ()`
- `go_flag is_empty ()`
- `go_result erase ()`
- `go_result drop_pending ()`
- `go_result stop ()`

Data Fields

- `go_flag _type`
go_MOTION_JOINT for joint, otherwise world
- `go_position _here`
initial starting point
- `go_position _there`
where the end is
- `go_motion_spec * _startptr`

- ptr to original start*
- [go_motion_spec * _endptr](#)
- the original end*
- [go_motion_spec * _start](#)
- ptr to first entry of the queue*
- [go_motion_spec * _end](#)
- ptr to last (empty) entry*
- [go_integer _size](#)
- size of whole queue*
- [go_integer _joint_num](#)
- number of joints to be interpolated*
- [go_integer _number](#)
- number of motions on queue*
- [go_integer _last_id](#)
- id of last motion appended*
- [go_real _deltat](#)
- cycle time*
- [go_real _time](#)
- time into the current spec*
- [go_scale_spec _timescale](#)
- walked-in time scale factor*

Protected Member Functions

- [go_result append_joint](#) (const [go_motion_spec](#) *motion)
- [go_result append_ujoint](#) (const [go_motion_spec](#) *motion)
- [go_result append_world](#) (const [go_motion_spec](#) *motion)
- [go_result interp_joint](#) ([go_real](#) *joint)
- [go_result interp_world](#) ([go_pose](#) *pose)
- [go_result interp_world](#) ([go_motion_spec](#) *motion, [go_real](#) time, [go_pose](#) *pose)

7.18.1 Member Function Documentation

- 7.18.1.1 [go_result gomotion::go_motion_queue::append \(const \[go_motion_spec\]\(#\) * *motion* \)](#)
- 7.18.1.2 [go_result gomotion::go_motion_queue::append_joint \(const \[go_motion_spec\]\(#\) * *motion* \)](#) [protected]
- 7.18.1.3 [go_result gomotion::go_motion_queue::append_ujoint \(const \[go_motion_spec\]\(#\) * *motion* \)](#) [protected]
- 7.18.1.4 [go_result gomotion::go_motion_queue::append_world \(const \[go_motion_spec\]\(#\) * *motion* \)](#) [protected]
- 7.18.1.5 [go_result gomotion::go_motion_queue::drop_pending \(\)](#)
- 7.18.1.6 [go_result gomotion::go_motion_queue::erase \(\)](#)
- 7.18.1.7 [go_integer gomotion::go_motion_queue::get_joint_number \(\)](#)
- 7.18.1.8 [go_flag gomotion::go_motion_queue::get_type \(\)](#)

- 7.18.1.9 `go_result gomotion::go_motion_queue::head (go_motion_spec * motion)`
- 7.18.1.10 `go_result gomotion::go_motion_queue::here (go_position * position)`
- 7.18.1.11 `go_result gomotion::go_motion_queue::init (go_motion_spec * space, go_integer size, go_real deltat)`
- 7.18.1.12 `go_result gomotion::go_motion_queue::interp (go_position * position)`
- 7.18.1.13 `go_result gomotion::go_motion_queue::interp_joint (go_real * joint)` [protected]
- 7.18.1.14 `go_result gomotion::go_motion_queue::interp_world (go_pose * pose)` [protected]
- 7.18.1.15 `go_result gomotion::go_motion_queue::interp_world (go_motion_spec * motion, go_real time, go_pose * pose)` [protected]
- 7.18.1.16 `go_flag gomotion::go_motion_queue::is_empty ()`
- 7.18.1.17 `go_integer gomotion::go_motion_queue::last_id ()`
- 7.18.1.18 `go_result gomotion::go_motion_queue::number (go_integer * number)`
- 7.18.1.19 `go_result gomotion::go_motion_queue::reset ()`
- 7.18.1.20 `go_result gomotion::go_motion_queue::set_cycle_time (go_real deltat)`
- 7.18.1.21 `go_result gomotion::go_motion_queue::set_here (const go_position * here)`
- 7.18.1.22 `go_result gomotion::go_motion_queue::set_id (go_integer id)`
- 7.18.1.23 `go_result gomotion::go_motion_queue::set_joint_number (go_integer joints)`
- 7.18.1.24 `go_result gomotion::go_motion_queue::set_scale (go_real scale, go_real scale_v, go_real scale_a)`
- 7.18.1.25 `go_result gomotion::go_motion_queue::set_type (go_flag type)`
- 7.18.1.26 `go_result gomotion::go_motion_queue::size (go_integer * size)`
- 7.18.1.27 `go_result gomotion::go_motion_queue::stop (go_motion_queue * queue)`
- 7.18.1.28 `go_result gomotion::go_motion_queue::stop ()`
- 7.18.1.29 `go_result gomotion::go_motion_queue::there (go_position * position)`

7.18.2 Field Documentation

7.18.2.1 `go_real gomotion::go_motion_queue::_deltat`

cycle time

7.18.2.2 `go_motion_spec* gomotion::go_motion_queue::_end`

ptr to last (empty) entry

7.18.2.3 **go_motion_spec*** gomotion::go_motion_queue::_endptr

the original end

7.18.2.4 **go_position** gomotion::go_motion_queue::_here

initial starting point

7.18.2.5 **go_integer** gomotion::go_motion_queue::_joint_num

number of joints to be interpolated

7.18.2.6 **go_integer** gomotion::go_motion_queue::_last_id

id of last motion appended

7.18.2.7 **go_integer** gomotion::go_motion_queue::_number

number of motions on queue

7.18.2.8 **go_integer** gomotion::go_motion_queue::_size

size of whole queue

7.18.2.9 **go_motion_spec*** gomotion::go_motion_queue::_start

ptr to first entry of the queue

7.18.2.10 **go_motion_spec*** gomotion::go_motion_queue::_startptr

ptr to original start

7.18.2.11 **go_position** gomotion::go_motion_queue::_there

where the end is

7.18.2.12 **go_real** gomotion::go_motion_queue::_time

time into the current spec

7.18.2.13 **go_scale_spec** gomotion::go_motion_queue::_timescale

walked-in time scale factor

7.18.2.14 go_flag gomotion::go_motion_queue::_type

go_MOTION_JOINT for joint, otherwise world

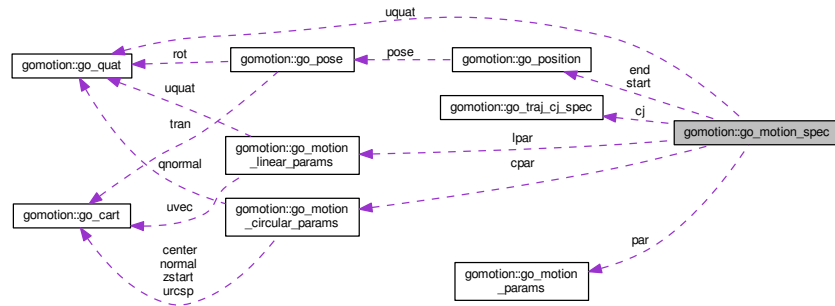
The documentation for this struct was generated from the following files:

- /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomotion.h
- /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gomotion.cpp

7.19 gomotion::go_motion_spec Struct Reference

```
#include <gomotion.h>
```

Collaboration diagram for gomotion::go_motion_spec:



Public Member Functions

- go_result init ()
- go_result set_type (go_flag type)
- go_flag get_type ()
- go_result set_id (go_integer id)
- go_integer get_id ()
- go_result set_jpar (go_integer i, go_real vel, go_real acc, go_real jerk)
- go_result set_tpar (go_real vel, go_real acc, go_real jerk)
- go_result set_rpar (go_real vel, go_real acc, go_real jerk)
- go_result set_cpar (go_cart *center, go_cart *normal, go_integer turns)
- go_result set_time (go_real time)
- go_result set_end_position (go_position *end)
- go_result set_end_pose (go_pose *end)

Data Fields

- go_flag type
ALL: GO_MOTION_JOINT, LINEAR, CIRCULAR.
- go_integer id
ALL: id echoed as current move.

- [go_real totalt](#)
(ALL) total planned time for the motion
- [go_position start](#)
(ALL) start pose wrt world; prev end
- [go_position end](#)
ALL: target position, joints or pose.
- [go_quat uquat](#)
(ALL) unit rotation from start to end
- [go_motion_params par](#) [GO_MOTION_JOINT_NUM]
- union {
 [go_motion_linear_params lpar](#)
 (LIN) linear params
 [go_motion_circular_params cpar](#)
 CIR: some circular params.
} u
- [go_traj_cj_spec cj](#) [GO_MOTION_JOINT_NUM]

7.19.1 Member Function Documentation

- 7.19.1.1 [go_integer gomotion::go_motion_spec::get_id \(\)](#)
- 7.19.1.2 [go_flag gomotion::go_motion_spec::get_type \(\)](#)
- 7.19.1.3 [go_result gomotion::go_motion_spec::init \(\)](#)
- 7.19.1.4 [go_result gomotion::go_motion_spec::set_cpar \(go_cart * center, go_cart * normal, go_integer turns \)](#)
- 7.19.1.5 [go_result gomotion::go_motion_spec::set_end_pose \(go_pose * end \)](#)
- 7.19.1.6 [go_result gomotion::go_motion_spec::set_end_position \(go_position * end \)](#)
- 7.19.1.7 [go_result gomotion::go_motion_spec::set_id \(go_integer id \)](#)
- 7.19.1.8 [go_result gomotion::go_motion_spec::set_jpar \(go_integer i, go_real vel, go_real acc, go_real jerk \)](#)
- 7.19.1.9 [go_result gomotion::go_motion_spec::set_rpar \(go_real vel, go_real acc, go_real jerk \)](#)
- 7.19.1.10 [go_result gomotion::go_motion_spec::set_time \(go_real time \)](#)
- 7.19.1.11 [go_result gomotion::go_motion_spec::set_tpar \(go_real vel, go_real acc, go_real jerk \)](#)
- 7.19.1.12 [go_result gomotion::go_motion_spec::set_type \(go_flag type \)](#)

7.19.2 Field Documentation

- 7.19.2.1 [go_traj_cj_spec gomotion::go_motion_spec::cj\[GO_MOTION_JOINT_NUM\]](#)
- 7.19.2.2 [go_motion_circular_params gomotion::go_motion_spec::cpar](#)

CIR: some circular params.

7.19.2.3 go_position gomotion::go_motion_spec::end

ALL: target position, joints or pose.

7.19.2.4 go_integer gomotion::go_motion_spec::id

ALL: id echoed as current move.

7.19.2.5 go_motion_linear_params gomotion::go_motion_spec::lpar

(LIN) linear params

7.19.2.6 go_motion_params gomotion::go_motion_spec::par[GO_MOTION_JOINT_NUM]**7.19.2.7 go_position** gomotion::go_motion_spec::start

(ALL) start pose wrt world; prev end

7.19.2.8 go_real gomotion::go_motion_spec::totalt

(ALL) total planned time for the motion

7.19.2.9 go_flag gomotion::go_motion_spec::type

ALL: GO_MOTION_JOINT, LINEAR, CIRCULAR.

7.19.2.10 union { ... } gomotion::go_motion_spec::u**7.19.2.11 go_quat** gomotion::go_motion_spec::uquat

(ALL) unit rotation from start to end

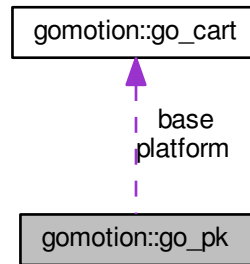
The documentation for this struct was generated from the following files:

- /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/[gomotion.h](#)
- /usr/local/michalos/nistfanuc_ws/src/gomotion/src/[gomotion.cpp](#)

7.20 gomotion::go_pk Struct Reference

```
#include <gomath.h>
```

Collaboration diagram for gomotion::go_pk:



Data Fields

- [go_cart base](#)
- [go_cart platform](#)
- [go_real d](#)

7.20.1 Detailed Description

PK parameters are used for parallel kinematic mechanisms, and represent the Cartesian positions of the ends of the link in the stationary base frame and the moving platform frame. Currently this only supports prismatic links.

7.20.2 Field Documentation

7.20.2.1 `go_cart` `gomotion::go_pk::base`

position of fixed end in base frame

7.20.2.2 `go_real` `gomotion::go_pk::d`

the length of the link

7.20.2.3 `go_cart` `gomotion::go_pk::platform`

position of moving end in platform frame

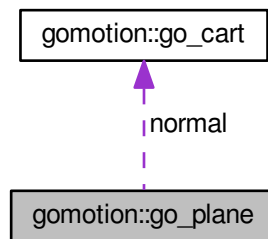
The documentation for this struct was generated from the following file:

- `/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h`

7.21 gomotion::go_plane Struct Reference

```
#include <gomath.h>
```

Collaboration diagram for gomotion::go_plane:



Data Fields

- [go_cart normal](#)
- [go_real d](#)

7.21.1 Detailed Description

Given a plane as $Ax + By + Cz + D = 0$, the normal vector *normal* is the Cartesian vector (A,B,C), and the number *d* is the value D.

Planes have a handedness, given by the direction of the normal vector, so two planes that appear coincident may be different by the direction of their anti-parallel normal vectors.

7.21.2 Field Documentation

7.21.2.1 [go_real gomotion::go_plane::d](#)

7.21.2.2 [go_cart gomotion::go_plane::normal](#)

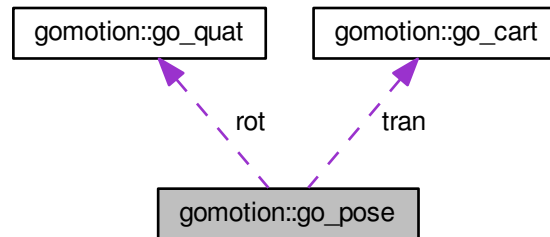
The documentation for this struct was generated from the following file:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h](#)

7.22 gomotion::go_pose Struct Reference

```
#include <gomath.h>
```

Collaboration diagram for gomotion::go_pose:



Data Fields

- [go_cart tran](#)
- [go_quat rot](#)

7.22.1 Detailed Description

A [go_pose](#) represents the Cartesian position vector and quaternion orientation of a frame.

7.22.2 Field Documentation

7.22.2.1 [go_quat](#) gomotion::go_pose::rot

7.22.2.2 [go_cart](#) gomotion::go_pose::tran

The documentation for this struct was generated from the following file:

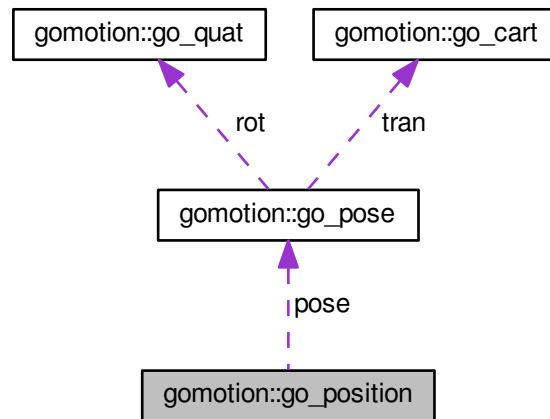
- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h](#)

7.23 gomotion::go_position Struct Reference

Depending upon whether you are doing joint interpolation or world coordinate interpolation (as specified by your call to `go_motion_queue_set_type()`), fill in `joint[]` or `pose` accordingly.

```
#include <gomotion.h>
```

Collaboration diagram for gomotion::go_position:



Public Member Functions

- void [zero_joints](#) ()
- void [zero_pose](#) ()

Data Fields

- union {
[go_real joint](#) [[GO_MOTION_JOINT_NUM](#)]
[go_pose pose](#)
 } [u](#)

7.23.1 Detailed Description

Depending upon whether you are doing joint interpolation or world coordinate interpolation (as specified by your call to `go_motion_queue_set_type()`), fill in `joint[]` or `pose` accordingly.

7.23.2 Member Function Documentation

7.23.2.1 void `gomotion::go_position::zero_joints` ()

7.23.2.2 void `gomotion::go_position::zero_pose` ()

7.23.3 Field Documentation

7.23.3.1 `go_real gomotion::go_position::joint[GO_MOTION_JOINT_NUM]`

7.23.3.2 `go_pose gomotion::go_position::pose`

7.23.3.3 `union { ... } gomotion::go_position::u`

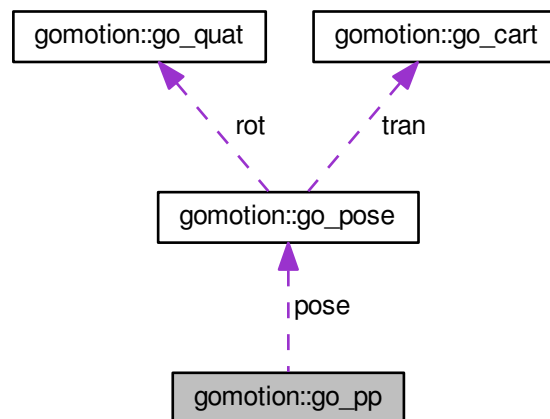
The documentation for this struct was generated from the following files:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomotion.h](#)
- [/usr/local/michalos/nistfanuc_ws/src/gomotion/src/gomotion.cpp](#)

7.24 gomotion::go_pp Struct Reference

```
#include <gomath.h>
```

Collaboration diagram for gomotion::go_pp:



Data Fields

- [go_pose pose](#)

7.24.1 Detailed Description

PP parameters represent the pose of the link with respect to the previous link. Revolute joints rotate about the Z axis, prismatic joints slide along the Z axis.

7.24.2 Field Documentation

7.24.2.1 `go_pose` `gomotion::go_pp::pose`

the pose of the link wrt to the previous link

The documentation for this struct was generated from the following file:

- `/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h`

7.25 `gomotion::go_quadratic` Struct Reference

```
#include <gomath.h>
```

Data Fields

- `go_real a`
- `go_real b`

7.25.1 Field Documentation

7.25.1.1 `go_real` `gomotion::go_quadratic::a`

7.25.1.2 `go_real` `gomotion::go_quadratic::b`

The documentation for this struct was generated from the following file:

- `/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h`

7.26 `gomotion::go_quartic` Struct Reference

```
#include <gomath.h>
```

Data Fields

- `go_real a`
- `go_real b`
- `go_real c`
- `go_real d`

7.26.1 Field Documentation

7.26.1.1 `go_real` `gomotion::go_quartic::a`

7.26.1.2 `go_real` `gomotion::go_quartic::b`

7.26.1.3 `go_real` `gomotion::go_quartic::c`

7.26.1.4 go_real gomotion::go_quartic::d

The documentation for this struct was generated from the following file:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h](#)

7.27 gomotion::go_quat Struct Reference

```
#include <gomath.h>
```

Data Fields

- [go_real s](#)
- [go_real x](#)
- [go_real y](#)
- [go_real z](#)

7.27.1 Detailed Description

A quaternion. *s* is the cosine of the half angle of rotation, and the *xyz* elements comprise the vector that points in the direction of positive rotation and whose magnitude is the sine of the half angle of rotation.

7.27.2 Field Documentation

7.27.2.1 go_real gomotion::go_quat::s

7.27.2.2 go_real gomotion::go_quat::x

7.27.2.3 go_real gomotion::go_quat::y

7.27.2.4 go_real gomotion::go_quat::z

The documentation for this struct was generated from the following file:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h](#)

7.28 gomotion::go_rpy Struct Reference

```
#include <gomath.h>
```

Data Fields

- [go_real r](#)
- [go_real p](#)
- [go_real y](#)

7.28.1 Detailed Description

Roll-pitch-yaw angles. r is the amount of the first rotation (roll) around the X axis. p is the amount of the second rotation (pitch) around the original Y axis. y is the amount of the third rotation (yaw) around the original Z axis.

7.28.2 Field Documentation

7.28.2.1 `go_real gomotion::go_rpy::p`

7.28.2.2 `go_real gomotion::go_rpy::r`

7.28.2.3 `go_real gomotion::go_rpy::y`

The documentation for this struct was generated from the following file:

- `/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h`

7.29 gomotion::go_rvec Struct Reference

```
#include <gomath.h>
```

Data Fields

- `go_real x`
- `go_real y`
- `go_real z`

7.29.1 Detailed Description

A rotation vector, whose direction points along the axis of positive rotation, and whose magnitude is the amount of rotation around this axis, in radians.

7.29.2 Field Documentation

7.29.2.1 `go_real gomotion::go_rvec::x`

7.29.2.2 `go_real gomotion::go_rvec::y`

7.29.2.3 `go_real gomotion::go_rvec::z`

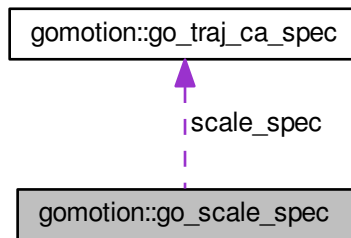
The documentation for this struct was generated from the following file:

- `/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h`

7.30 gomotion::go_scale_spec Struct Reference

```
#include <gomotion.h>
```

Collaboration diagram for gomotion::go_scale_spec:



Public Member Functions

- [go_result init \(go_real scale\)](#)
- [go_result set \(go_real scale, go_real scale_v, go_real scale_a\)](#)
- [go_result eval \(go_real deltat, go_real *scale\)](#)

Data Fields

- [go_traj_ca_spec scale_spec](#)
- [go_flag scaling](#)
non-zero means we're scaling time
- [go_flag scale_dir](#)
non-zero means add scale to scale_b
- [go_flag scale_isneg](#)
non-zero means negative scale
- [go_real scale_b](#)
original base scale factor
- [go_real scale](#)
current scale factor
- [go_real scale_next](#)
pending scale factor
- [go_real scale_v_next](#)
pending $d(\text{scale})/dt$
- [go_real scale_a_next](#)
pending $d^2(\text{scale})/dt^2$
- [go_real scale_t](#)
time into scaling

7.30.1 Member Function Documentation

7.30.1.1 **go_result** gomotion::go_scale_spec::eval (*go_real deltat*, *go_real * scale*)

7.30.1.2 **go_result** gomotion::go_scale_spec::init (*go_real scale*)

7.30.1.3 **go_result** gomotion::go_scale_spec::set (*go_real scale*, *go_real scale_v*, *go_real scale_a*)

7.30.2 Field Documentation

7.30.2.1 **go_real** gomotion::go_scale_spec::scale

current scale factor

7.30.2.2 **go_real** gomotion::go_scale_spec::scale_a_next

pending $d^2(\text{scale})/dt^2$

7.30.2.3 **go_real** gomotion::go_scale_spec::scale_b

original base scale factor

7.30.2.4 **go_flag** gomotion::go_scale_spec::scale_dir

non-zero means add scale to scale_b

7.30.2.5 **go_flag** gomotion::go_scale_spec::scale_isneg

non-zero means negative scale

7.30.2.6 **go_real** gomotion::go_scale_spec::scale_next

pending scale factor

7.30.2.7 **go_traj_ca_spec** gomotion::go_scale_spec::scale_spec

7.30.2.8 **go_real** gomotion::go_scale_spec::scale_t

time into scaling

7.30.2.9 **go_real** gomotion::go_scale_spec::scale_v_next

pending $d(\text{scale})/dt$

7.30.2.10 go_flag gomotion::go_scale_spec::scaling

non-zero means we're scaling time

The documentation for this struct was generated from the following files:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomotion.h](#)
- [/usr/local/michalos/nistfanuc_ws/src/gomotion/src/gomotion.cpp](#)

7.31 gomotion::go_sph Struct Reference

```
#include <gomath.h>
```

Data Fields

- [go_real theta](#)
- [go_real phi](#)
- [go_real r](#)

7.31.1 Detailed Description

A point or vector in spherical coordinates, with *phi* as the angle down from the zenith, not up from the XY plane.

7.31.2 Field Documentation

7.31.2.1 [go_real gomotion::go_sph::phi](#)

7.31.2.2 [go_real gomotion::go_sph::r](#)

7.31.2.3 [go_real gomotion::go_sph::theta](#)

The documentation for this struct was generated from the following file:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h](#)

7.32 gomotion::go_traj_ca_spec Struct Reference

```
#include <gotraj.h>
```

Data Fields

- [go_real at0](#)
accel for Phase I and III
- [go_real t1](#)
cumulative time at end of Phase I
- [go_real dt1](#)

- cumulative distance at end of Phase I*
 - [go_real vt1](#)
 - speed at end of Phase I/in Phase II*
 - [go_real t2](#)
 - cumulative time at end of Phase II*
 - [go_real dt2](#)
 - cumulative distance at end of Phase II*
 - [go_real tend](#)
 - total time for motion*
 - [go_real dtend](#)
 - total distance for motion*
 - [go_real invd](#)
 - inverse of total distance*

7.32.1 Field Documentation

7.32.1.1 [go_real gomotion::go_traj_ca_spec::at0](#)

accel for Phase I and III

7.32.1.2 [go_real gomotion::go_traj_ca_spec::dt1](#)

cumulative distance at end of Phase I

7.32.1.3 [go_real gomotion::go_traj_ca_spec::dt2](#)

cumulative distance at end of Phase II

7.32.1.4 [go_real gomotion::go_traj_ca_spec::dtend](#)

total distance for motion

7.32.1.5 [go_real gomotion::go_traj_ca_spec::invd](#)

inverse of total distance

7.32.1.6 [go_real gomotion::go_traj_ca_spec::t1](#)

cumulative time at end of Phase I

7.32.1.7 [go_real gomotion::go_traj_ca_spec::t2](#)

cumulative time at end of Phase II

7.32.1.8 go_real gomotion::go_traj_ca_spec::tend

total time for motion

7.32.1.9 go_real gomotion::go_traj_ca_spec::vt1

speed at end of Phase I/in Phase II

The documentation for this struct was generated from the following file:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gotraj.h](#)

7.33 gomotion::go_traj_cj_spec Struct Reference

```
#include <gotraj.h>
```

Data Fields

- [go_real jt0](#)
jerk in Phases I, III, V, VII
- [go_real t1](#)
cumulative time at end of Phase I
- [go_real dt1](#)
cumulative distance at end of Phase I
- [go_real vt1](#)
speed at end of Phase I
- [go_real at1](#)
accel at end of Phase I
- [go_real t2](#)
cumulative time at end of Phase II
- [go_real dt2](#)
cumulative distance at end of Phase II
- [go_real vt2](#)
speed at end of Phase II
- [go_real t3](#)
cumulative time at end of Phase III
- [go_real dt3](#)
cumulative distance at end of Phase III
- [go_real vt3](#)
speed at end of Phase III/in Phase IV
- [go_real t4](#)
cumulative time at end of Phase IV
- [go_real dt4](#)
cumulative distance at end of Phase IV
- [go_real t5](#)
cumulative time at end of Phase V
- [go_real dt5](#)

- cumulative distance at end of Phase V*
- [go_real t6](#)
 - cumulative time at end of Phase VI*
- [go_real dt6](#)
 - cumulative distance at end of Phase VI*
- [go_real tend](#)
 - total time for motion*
- [go_real dtend](#)
 - total distance for motion*
- [go_real invd](#)
 - inverse of total distance*

7.33.1 Field Documentation

7.33.1.1 [go_real gomotion::go_traj_cj_spec::at1](#)

accel at end of Phase I

7.33.1.2 [go_real gomotion::go_traj_cj_spec::dt1](#)

cumulative distance at end of Phase I

7.33.1.3 [go_real gomotion::go_traj_cj_spec::dt2](#)

cumulative distance at end of Phase II

7.33.1.4 [go_real gomotion::go_traj_cj_spec::dt3](#)

cumulative distance at end of Phase III

7.33.1.5 [go_real gomotion::go_traj_cj_spec::dt4](#)

cumulative distance at end of Phase IV

7.33.1.6 [go_real gomotion::go_traj_cj_spec::dt5](#)

cumulative distance at end of Phase V

7.33.1.7 [go_real gomotion::go_traj_cj_spec::dt6](#)

cumulative distance at end of Phase VI

7.33.1.8 [go_real gomotion::go_traj_cj_spec::dtend](#)

total distance for motion

7.33.1.9 **go_real** gomotion::go_traj_cj_spec::invd

inverse of total distance

7.33.1.10 **go_real** gomotion::go_traj_cj_spec::jt0

jerk in Phases I, III, V, VII

7.33.1.11 **go_real** gomotion::go_traj_cj_spec::t1

cumulative time at end of Phase I

7.33.1.12 **go_real** gomotion::go_traj_cj_spec::t2

cumulative time at end of Phase II

7.33.1.13 **go_real** gomotion::go_traj_cj_spec::t3

cumulative time at end of Phase III

7.33.1.14 **go_real** gomotion::go_traj_cj_spec::t4

cumulative time at end of Phase IV

7.33.1.15 **go_real** gomotion::go_traj_cj_spec::t5

cumulative time at end of Phase V

7.33.1.16 **go_real** gomotion::go_traj_cj_spec::t6

cumulative time at end of Phase VI

7.33.1.17 **go_real** gomotion::go_traj_cj_spec::tend

total time for motion

7.33.1.18 **go_real** gomotion::go_traj_cj_spec::vt1

speed at end of Phase I

7.33.1.19 **go_real** gomotion::go_traj_cj_spec::vt2

speed at end of Phase II

7.33.1.20 `go_real gomotion::go_traj_cj_spec::vt3`

speed at end of Phase III/in Phase IV

The documentation for this struct was generated from the following file:

- `/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gotraj.h`

7.34 `gomotion::go_traj_interp_spec` Struct Reference

```
#include <gotraj.h>
```

Data Fields

- `go_real s`
- `go_real d`
- `go_real v`
- `go_real a`
- `go_real j`

7.34.1 Field Documentation

7.34.1.1 `go_real gomotion::go_traj_interp_spec::a`

7.34.1.2 `go_real gomotion::go_traj_interp_spec::d`

7.34.1.3 `go_real gomotion::go_traj_interp_spec::j`

7.34.1.4 `go_real gomotion::go_traj_interp_spec::s`

7.34.1.5 `go_real gomotion::go_traj_interp_spec::v`

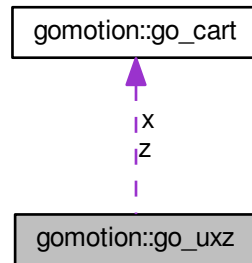
The documentation for this struct was generated from the following file:

- `/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gotraj.h`

7.35 `gomotion::go_uxz` Struct Reference

```
#include <gomath.h>
```

Collaboration diagram for gomotion::go_uxz:



Data Fields

- [go_cart x](#)
- [go_cart z](#)

7.35.1 Detailed Description

X-Z vector format. *x* is a unit vector in the X direction, and *z* is a unit vector in the Z direction.

7.35.2 Field Documentation

7.35.2.1 go_cart gomotion::go_uxz::x

7.35.2.2 go_cart gomotion::go_uxz::z

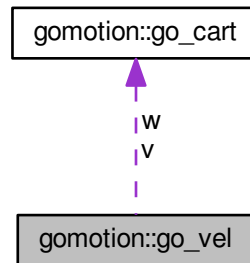
The documentation for this struct was generated from the following file:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h](#)

7.36 gomotion::go_vel Struct Reference

```
#include <gomath.h>
```

Collaboration diagram for gomotion::go_vel:



Data Fields

- [go_cart v](#)
- [go_cart w](#)

7.36.1 Detailed Description

A [go_vel](#) represents the linear- and angular velocity vectors of a frame. v is the Cartesian linear velocity vector. w is the Cartesian angular velocity vector, the instantaneous vector about which the frame is rotating.

7.36.2 Field Documentation

7.36.2.1 [go_cart gomotion::go_vel::v](#)

7.36.2.2 [go_cart gomotion::go_vel::w](#)

The documentation for this struct was generated from the following file:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h](#)

7.37 gomotion::go_xyz Struct Reference

```
#include <gomath.h>
```

Data Fields

- [go_real x](#)
- [go_real y](#)
- [go_real z](#)

7.37.1 Detailed Description

XYZ Euler angles. x is the amount of the first rotation around the X axis. y is the amount of the second rotation around the new Y axis. z is the amount of the third rotation around the new Z axis.

7.37.2 Field Documentation

7.37.2.1 `go_real gomotion::go_xyz::x`

7.37.2.2 `go_real gomotion::go_xyz::y`

7.37.2.3 `go_real gomotion::go_xyz::z`

The documentation for this struct was generated from the following file:

- `/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h`

7.38 gomotion::go_zyx Struct Reference

```
#include <gomath.h>
```

Data Fields

- `go_real z`
- `go_real y`
- `go_real x`

7.38.1 Detailed Description

ZYX Euler angles. z is the amount of the first rotation around the Z axis. y is the amount of the second rotation around the new Y axis. x is the amount of the third rotation around the new X axis.

7.38.2 Field Documentation

7.38.2.1 `go_real gomotion::go_zyx::x`

7.38.2.2 `go_real gomotion::go_zyx::y`

7.38.2.3 `go_real gomotion::go_zyx::z`

The documentation for this struct was generated from the following file:

- `/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h`

7.39 gomotion::go_zyz Struct Reference

```
#include <gomath.h>
```

Data Fields

- [go_real z](#)
- [go_real y](#)
- [go_real zp](#)

7.39.1 Detailed Description

ZYZ Euler angles. *z* is the amount of the first rotation around the Z axis. *y* is the amount of the second rotation around the new Y axis. *zp* is the amount of the third rotation around the new Z axis.

7.39.2 Field Documentation

7.39.2.1 [go_real gomotion::go_zyz::y](#)

7.39.2.2 [go_real gomotion::go_zyz::z](#)

7.39.2.3 [go_real gomotion::go_zyz::zp](#)

The documentation for this struct was generated from the following file:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h](#)

7.40 gomotion::GoMotion Class Reference

```
#include <gomove.h>
```

Protected Attributes

- [boost::shared_ptr](#)
[< go_motion_interface > pgm](#)

7.40.1 Field Documentation

7.40.1.1 [boost::shared_ptr<go_motion_interface> gomotion::GoMotion::pgm](#) [protected]

The documentation for this class was generated from the following files:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomove.h](#)
- [/usr/local/michalos/nistfanuc_ws/src/gomotion/src/gomove.cpp](#)

7.41 gomotion::GoMotionParams Struct Reference

```
#include <gomove.h>
```


Data Fields

- double [vel](#)
- double [acc](#)
- double [jerk](#)

7.41.1 Field Documentation

7.41.1.1 double gomotion::GoMotionParams::acc

7.41.1.2 double gomotion::GoMotionParams::jerk

7.41.1.3 double gomotion::GoMotionParams::vel

The documentation for this struct was generated from the following file:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomove.h](#)

7.42 gomotion::points Struct Reference

This structure holds the points to be interpolated.

```
#include <gointerp.h>
```

Data Fields

- [go_real](#) p [COEFF_MAX]

7.42.1 Detailed Description

This structure holds the points to be interpolated.

See notes below for how to fill the p[] array for the particular type of interpolation.

7.42.2 Field Documentation

7.42.2.1 [go_real](#) gomotion::points::p[COEFF_MAX]

The documentation for this struct was generated from the following file:

- [/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gointerp.h](#)

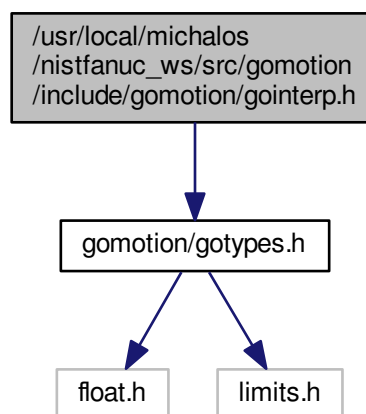
Chapter 8

File Documentation

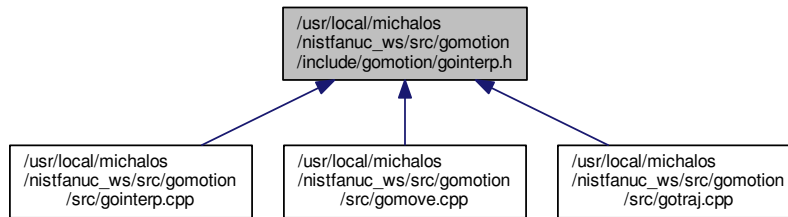
8.1 /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gointerp.h File Reference

```
#include "gomotion/gotypes.h"
```

Include dependency graph for gointerp.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gomotion::points](#)
This structure holds the points to be interpolated.
- struct [gomotion::coeff](#)
This structure holds the polynomial coefficients.
- class [gomotion::go_interp](#)
The interpolator structure.

Namespaces

- [gomotion](#)

Enumerations

- enum { [gomotion::COEFF_MAX](#) = 6 }
How many coefficients we support, one more than max polynomial degree.

8.2 /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomath.h File Reference

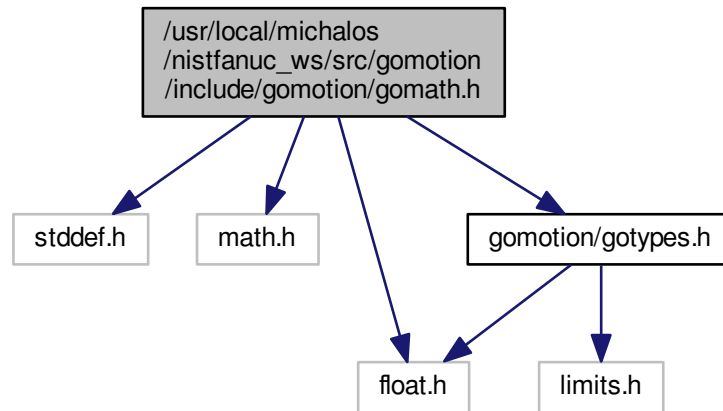
Declarations for pose math functions.

```

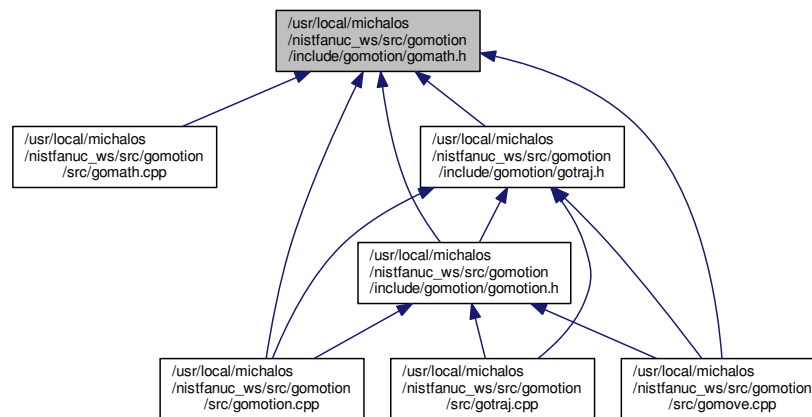
#include <stddef.h>
#include <math.h>
#include <float.h>
#include "gomotion/gotypes.h"

```

Include dependency graph for gomath.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `gomotion::go_cart`
- struct `gomotion::go_sph`
- struct `gomotion::go_cyl`
- struct `gomotion::go_rvec`
- struct `gomotion::go_mat`
- struct `gomotion::go_quat`

- struct [gomotion::go_zyz](#)
- struct [gomotion::go_zyx](#)
- struct [gomotion::go_xyz](#)
- struct [gomotion::go_rpy](#)
- struct [gomotion::go_uxz](#)
- struct [gomotion::go_pose](#)
- struct [gomotion::go_vel](#)
- struct [gomotion::go_hom](#)
- struct [gomotion::go_line](#)
- struct [gomotion::go_plane](#)
- struct [gomotion::go_matrix](#)
- struct [gomotion::go_dh](#)
- struct [gomotion::go_pk](#)
- struct [gomotion::go_pp](#)
- struct [gomotion::go_body](#)
- struct [gomotion::go_link](#)
- struct [gomotion::go_complex](#)
- struct [gomotion::go_quadratic](#)
- struct [gomotion::go_cubic](#)
- struct [gomotion::go_quartic](#)

Namespaces

- [gomotion](#)

Macros

- `#define go_sq(x) ((x)*(x))`
- `#define go_cub(x) ((x)*(x)*(x))`
- `#define go_qua(x) ((x)*(x)*(x)*(x))`
- `#define GO_PI 3.14159265358979323846`
- `#define GO_2_PI (2.0*GO_PI)`
- `#define GO_PI_2 1.57079632679489661923`
- `#define GO_PI_4 0.78539816339744830962`
- `#define GO_TO_DEG(rad) ((rad)*57.295779513082323)`
- `#define GO_TO_RAD(deg) ((deg)*0.0174532925199432952)`
- `#define GO_TRAN_CLOSE(x, y) (fabs((x)-(y)) < GO_REAL_EPSILON)`
- `#define GO_TRAN_SMALL(x) (fabs(x) < GO_REAL_EPSILON)`
- `#define GO_ROT_CLOSE(x, y) (fabs((x)-(y)) < GO_REAL_EPSILON)`
- `#define GO_ROT_SMALL(x) (fabs(x) < GO_REAL_EPSILON)`
- `#define GO_CLOSE(x, y) (fabs((x)-(y)) < GO_REAL_EPSILON)`
- `#define GO_SMALL(x) (fabs(x) < GO_REAL_EPSILON)`
- `#define GO_MATRIX_DECLARE(M, Mspace, _rows, _cols)`
- `#define ISIZEOF(x) ((int) sizeof(x))`
- `#define go_matrix_init(M, Mspace, _rows, _cols)`
- `#define go_body_init(b)`
- `#define go_body_copy(src, dst)`
- `#define go_link_to_string(L)`

Typedefs

- typedef [go_real](#) gomotion::go_vector

Enumerations

- enum { [gomotion::GO_LINK_DH](#) = 1, [gomotion::GO_LINK_PK](#), [gomotion::GO_LINK_PP](#) }

Functions

- void [gomotion::go_sincos](#) ([go_real](#) t, [go_real](#) *s, [go_real](#) *c)
- [go_real](#) [gomotion::go_cbrt](#) ([go_real](#) x)
- [go_result](#) [gomotion::go_asines](#) ([go_real](#) s, [go_real](#) *asp, [go_real](#) *asn)
- [go_result](#) [gomotion::go_acoses](#) ([go_real](#) c, [go_real](#) *acp, [go_real](#) *acn)
- [go_result](#) [gomotion::go_atans](#) ([go_real](#) t, [go_real](#) *atp, [go_real](#) *atn)
- [go_pose](#) [gomotion::go_pose_this](#) ([go_real](#) x, [go_real](#) y, [go_real](#) z, [go_real](#) rs, [go_real](#) rx, [go_real](#) ry, [go_real](#) rz)
- [go_cart](#) [gomotion::go_cart_zero](#) (void)
- [go_quat](#) [gomotion::go_quat_identity](#) (void)
- [go_pose](#) [gomotion::go_pose_identity](#) (void)
- [go_result](#) [gomotion::go_line_from_point_direction](#) (const [go_cart](#) *point, const [go_cart](#) *direction, [go_line](#) *line)
- [go_result](#) [gomotion::go_line_from_points](#) (const [go_cart](#) *point1, const [go_cart](#) *point2, [go_line](#) *line)
- [go_result](#) [gomotion::go_line_from_planes](#) (const [go_plane](#) *plane1, const [go_plane](#) *plane2, [go_line](#) *line)
- [go_flag](#) [gomotion::go_line_line_compare](#) (const [go_line](#) *line1, const [go_line](#) *line2)
- [go_result](#) [gomotion::go_line_evaluate](#) (const [go_line](#) *line, [go_real](#) d, [go_cart](#) *point)
- [go_result](#) [gomotion::go_point_line_distance](#) (const [go_cart](#) *point, const [go_line](#) *line, [go_real](#) *distance)
- [go_result](#) [gomotion::go_point_line_proj](#) (const [go_cart](#) *point, const [go_line](#) *line, [go_cart](#) *pout)
- [go_result](#) [gomotion::go_point_plane_proj](#) (const [go_cart](#) *point, const [go_plane](#) *plane, [go_cart](#) *proj)
- [go_result](#) [gomotion::go_line_plane_proj](#) (const [go_line](#) *line, const [go_plane](#) *plane, [go_line](#) *proj)
- [go_result](#) [gomotion::go_plane_from_point_normal](#) (const [go_cart](#) *point, const [go_cart](#) *normal, [go_plane](#) *plane)
- [go_result](#) [gomotion::go_plane_from_abcd](#) ([go_real](#) A, [go_real](#) B, [go_real](#) C, [go_real](#) D, [go_plane](#) *plane)
- [go_result](#) [gomotion::go_plane_from_points](#) (const [go_cart](#) *point1, const [go_cart](#) *point2, const [go_cart](#) *point3, [go_plane](#) *plane)
- [go_result](#) [gomotion::go_plane_from_point_line](#) (const [go_cart](#) *point, const [go_line](#) *line, [go_plane](#) *plane)
- [go_flag](#) [gomotion::go_plane_plane_compare](#) (const [go_plane](#) *plane1, const [go_plane](#) *plane2)
- [go_result](#) [gomotion::go_point_plane_distance](#) (const [go_cart](#) *point, const [go_plane](#) *plane, [go_real](#) *distance)
- [go_result](#) [gomotion::go_plane_evaluate](#) (const [go_plane](#) *plane, [go_real](#) u, [go_real](#) v, [go_cart](#) *point)
- [go_result](#) [gomotion::go_line_plane_intersect](#) (const [go_line](#) *line, const [go_plane](#) *plane, [go_cart](#) *point, [go_real](#) *distance)
- [go_result](#) [gomotion::go_cart_sph_convert](#) (const [go_cart](#) *v, [go_sph](#) *s)
- [go_result](#) [gomotion::go_cart_cyl_convert](#) (const [go_cart](#) *v, [go_cyl](#) *c)
- [go_result](#) [gomotion::go_sph_cart_convert](#) (const [go_sph](#) *s, [go_cart](#) *v)
- [go_result](#) [gomotion::go_sph_cyl_convert](#) (const [go_sph](#) *s, [go_cyl](#) *c)
- [go_result](#) [gomotion::go_cyl_cart_convert](#) (const [go_cyl](#) *c, [go_cart](#) *v)
- [go_result](#) [gomotion::go_cyl_sph_convert](#) (const [go_cyl](#) *c, [go_sph](#) *s)
- [go_result](#) [gomotion::go_rvec_quat_convert](#) (const [go_rvec](#) *r, [go_quat](#) *q)
- [go_result](#) [gomotion::go_rvec_mat_convert](#) (const [go_rvec](#) *r, [go_mat](#) *m)
- [go_result](#) [gomotion::go_rvec_zyz_convert](#) (const [go_rvec](#) *rvec, [go_zyz](#) *zyz)
- [go_result](#) [gomotion::go_rvec_zyx_convert](#) (const [go_rvec](#) *rvec, [go_zyx](#) *zyx)
- [go_result](#) [gomotion::go_rvec_xyz_convert](#) (const [go_rvec](#) *, [go_xyz](#) *)

- [go_result gomotion::go_rvec_rpy_convert](#) (const go_rvec *r, go_rpy *rpy)
- [go_result gomotion::go_quat_rvec_convert](#) (const go_quat *q, go_rvec *r)
- [go_result gomotion::go_quat_mat_convert](#) (const go_quat *q, go_mat *m)
- [go_result gomotion::go_quat_zyz_convert](#) (const go_quat *q, go_zyz *zyz)
- [go_result gomotion::go_quat_zyx_convert](#) (const go_quat *q, go_zyx *zyx)
- [go_result gomotion::go_quat_xyz_convert](#) (const go_quat *, go_xyz *)
- [go_result gomotion::go_quat_rpy_convert](#) (const go_quat *q, go_rpy *rpy)
- [go_result gomotion::go_mat_rvec_convert](#) (const go_mat *m, go_rvec *r)
- [go_result gomotion::go_mat_quat_convert](#) (const go_mat *m, go_quat *q)
- [go_result gomotion::go_mat_zyz_convert](#) (const go_mat *m, go_zyz *zyz)
- [go_result gomotion::go_mat_zyx_convert](#) (const go_mat *m, go_zyx *zyx)
- [go_result gomotion::go_mat_xyz_convert](#) (const go_mat *m, go_xyz *xyz)
- [go_result gomotion::go_mat_rpy_convert](#) (const go_mat *m, go_rpy *rpy)
- [go_result gomotion::go_zyz_rvec_convert](#) (const go_zyz *zyz, go_rvec *r)
- [go_result gomotion::go_zyz_quat_convert](#) (const go_zyz *zyz, go_quat *q)
- [go_result gomotion::go_zyz_mat_convert](#) (const go_zyz *zyz, go_mat *m)
- [go_result gomotion::go_zyz_zyx_convert](#) (const go_zyz *zyz, go_zyx *zyx)
- [go_result gomotion::go_zyz_xyz_convert](#) (const go_zyz *, go_xyz *)
- [go_result gomotion::go_zyz_rpy_convert](#) (const go_zyz *zyz, go_rpy *rpy)
- [go_result gomotion::go_zyx_rvec_convert](#) (const go_zyx *zyx, go_rvec *r)
- [go_result gomotion::go_zyx_quat_convert](#) (const go_zyx *zyx, go_quat *q)
- [go_result gomotion::go_zyx_mat_convert](#) (const go_zyx *zyx, go_mat *m)
- [go_result gomotion::go_zyx_zyz_convert](#) (const go_zyx *zyx, go_zyz *zyz)
- [go_result gomotion::go_zyx_xyz_convert](#) (const go_zyx *, go_xyz *)
- [go_result gomotion::go_zyx_rpy_convert](#) (const go_zyx *zyx, go_rpy *rpy)
- [go_result gomotion::go_xyz_rvec_convert](#) (const go_xyz *, go_rvec *)
- [go_result gomotion::go_xyz_quat_convert](#) (const go_xyz *, go_quat *)
- [go_result gomotion::go_xyz_mat_convert](#) (const go_xyz *xyz, go_mat *m)
- [go_result gomotion::go_xyz_zyz_convert](#) (const go_xyz *, go_zyz *)
- [go_result gomotion::go_xyz_zyx_convert](#) (const go_xyz *, go_zyx *)
- [go_result gomotion::go_xyz_rpy_convert](#) (const go_xyz *, go_rpy *)
- [go_result gomotion::go_rpy_rvec_convert](#) (const go_rpy *rpy, go_rvec *rvec)
- [go_result gomotion::go_rpy_quat_convert](#) (const go_rpy *rpy, go_quat *quat)
- [go_result gomotion::go_rpy_mat_convert](#) (const go_rpy *rpy, go_mat *m)
- [go_result gomotion::go_rpy_zyz_convert](#) (const go_rpy *rpy, go_zyz *zyz)
- [go_result gomotion::go_rpy_zyx_convert](#) (const go_rpy *rpy, go_zyx *zyx)
- [go_result gomotion::go_rpy_xyz_convert](#) (const go_rpy *, go_xyz *)
- [go_result gomotion::go_uxz_mat_convert](#) (const go_uxz *uxz, go_mat *mat)
- [go_result gomotion::go_mat_uxz_convert](#) (const go_mat *mat, go_uxz *uxz)
- [go_result gomotion::go_pose_hom_convert](#) (const go_pose *p, go_hom *h)
- [go_result gomotion::go_hom_pose_convert](#) (const go_hom *h, go_pose *p)
- [go_result gomotion::go_cart_rvec_convert](#) (const go_cart *cart, go_rvec *rvec)
- [go_result gomotion::go_rvec_cart_convert](#) (const go_rvec *rvec, go_cart *cart)
- [go_flag gomotion::go_cart_cart_compare](#) (const go_cart *v1, const go_cart *v2)
- [go_result gomotion::go_cart_cart_dot](#) (const go_cart *v1, const go_cart *v2, [go_real](#) *d)
- [go_result gomotion::go_cart_cart_cross](#) (const go_cart *v1, const go_cart *v2, go_cart *vout)
- [go_result gomotion::go_cart_mag](#) (const go_cart *v, [go_real](#) *d)
- [go_result gomotion::go_cart_magsq](#) (const go_cart *v, [go_real](#) *d)
- [go_flag gomotion::go_cart_cart_par](#) (const go_cart *v1, const go_cart *v2)
- [go_flag gomotion::go_cart_cart_perp](#) (const go_cart *v1, const go_cart *v2)
- [go_result gomotion::go_cart_cart_disp](#) (const go_cart *v1, const go_cart *v2, [go_real](#) *d)

- [go_result gomotion::go_cart_cart_add](#) (const go_cart *v1, const go_cart *v2, go_cart *vout)
- [go_result gomotion::go_cart_cart_sub](#) (const go_cart *v1, const go_cart *v2, go_cart *vout)
- [go_result gomotion::go_cart_scale_mult](#) (const go_cart *v1, [go_real](#) d, go_cart *vout)
- [go_result gomotion::go_cart_neg](#) (const go_cart *v1, go_cart *vout)
- [go_result gomotion::go_cart_unit](#) (const go_cart *v, go_cart *vout)
- [go_result gomotion::go_cart_cart_rot](#) (const go_cart *v1, const go_cart *v2, go_quat *quat)
- [go_result gomotion::go_cart_cart_proj](#) (const go_cart *v1, const go_cart *v2, go_cart *vout)
- [go_result gomotion::go_cart_plane_proj](#) (const go_cart *v, const go_cart *normal, go_cart *vout)
- [go_result gomotion::go_cart_cart_angle](#) (const go_cart *v1, const go_cart *v2, [go_real](#) *a)
- [go_result gomotion::go_cart_normal](#) (const go_cart *v, go_cart *vout)
- [go_result gomotion::go_cart_centroid](#) (const go_cart *varray, [go_integer](#) num, go_cart *centroid)
- [go_result gomotion::go_cart_centroidize](#) (const go_cart *vinarray, [go_integer](#) num, go_cart *centroid, go_cart *voutarray)
- [go_result gomotion::go_cart_cart_pose](#) (const go_cart *v1, const go_cart *v2, go_cart *v1c, go_cart *v2c, [go_integer](#) num, go_pose *pout)
- [go_result gomotion::go_cart_trilaterate](#) (const go_cart *c1, const go_cart *c2, const go_cart *c3, [go_real](#) l1, [go_real](#) l2, [go_real](#) l3, go_cart *out1, go_cart *out2)
- [go_flag gomotion::go_quat_quat_compare](#) (const go_quat *q1, const go_quat *q2)
- [go_result gomotion::go_quat_mag](#) (const go_quat *quat, [go_real](#) *d)
- [go_result gomotion::go_quat_unit](#) (const go_quat *q1, go_quat *qout)
- [go_result gomotion::go_quat_norm](#) (const go_quat *q1, go_quat *qout)
- [go_result gomotion::go_quat_inv](#) (const go_quat *q1, go_quat *qout)
- [go_flag gomotion::go_quat_is_norm](#) (const go_quat *q1)
- [go_result gomotion::go_quat_scale_mult](#) (const go_quat *q, [go_real](#) s, go_quat *qout)
- [go_result gomotion::go_quat_quat_mult](#) (const go_quat *q1, const go_quat *q2, go_quat *qout)
- [go_result gomotion::go_quat_cart_mult](#) (const go_quat *q1, const go_cart *v2, go_cart *vout)
- [go_flag gomotion::go_rvec_rvec_compare](#) (const go_rvec *r1, const go_rvec *r2)
- [go_result gomotion::go_rvec_scale_mult](#) (const go_rvec *r, [go_real](#) s, go_rvec *rout)
- [go_result gomotion::go_rpy_cart_mult](#) (const go_rpy *rpy, const go_cart *in, go_cart *out)
- [go_result gomotion::go_mat_norm](#) (const go_mat *mat, go_mat *mout)
- [go_flag gomotion::go_mat_is_norm](#) (const go_mat *m)
- [go_result gomotion::go_mat_inv](#) (const go_mat *m, go_mat *mout)
- [go_result gomotion::go_mat_cart_mult](#) (const go_mat *m, const go_cart *v, go_cart *vout)
- [go_result gomotion::go_mat_mat_mult](#) (const go_mat *m1, const go_mat *m2, go_mat *mout)
- [go_flag gomotion::go_pose_pose_compare](#) (const go_pose *p1, const go_pose *p2)
- [go_result gomotion::go_pose_inv](#) (const go_pose *p1, go_pose *p2)
- [go_result gomotion::go_pose_cart_mult](#) (const go_pose *p1, const go_cart *v2, go_cart *vout)
- [go_result gomotion::go_pose_pose_mult](#) (const go_pose *p1, const go_pose *p2, go_pose *pout)
- [go_result gomotion::go_pose_scale_mult](#) (const go_pose *p1, [go_real](#) s, go_pose *pout)
- [go_result gomotion::go_pose_pose_interp](#) ([go_real](#) t1, const go_pose *p1, [go_real](#) t2, const go_pose *p2, [go_real](#) t3, go_pose *p3)
- [go_result gomotion::go_hom_inv](#) (const go_hom *h1, go_hom *h2)
- [go_result gomotion::go_hom_hom_mult](#) (const go_hom *h1, const go_hom *h2, go_hom *hout)
- [go_result gomotion::go_pose_vel_mult](#) (const go_pose *pose, const go_vel *vel, go_vel *out)
- [go_real gomotion::go_get_singular_epsilon](#) (void)
- [go_result gomotion::go_set_singular_epsilon](#) ([go_real](#) epsilon)
- [go_result gomotion::ludcmp](#) ([go_real](#) **a, [go_real](#) *scratchrow, [go_integer](#) n, [go_integer](#) *indx, [go_real](#) *d)
- [go_result gomotion::lubksb](#) ([go_real](#) **a, [go_integer](#) n, [go_integer](#) *indx, [go_real](#) *b)
- [go_result gomotion::go_cart_vector_convert](#) (const go_cart *c, [go_real](#) *v)
- [go_result gomotion::go_vector_cart_convert](#) (const [go_real](#) *v, go_cart *c)
- [go_result gomotion::go_quat_matrix_convert](#) (const go_quat *quat, go_matrix *matrix)

- [go_result gomotion::go_mat_matrix_convert](#) (const go_mat *mat, go_matrix *matrix)
- [go_result gomotion::go_matrix_matrix_add](#) (const go_matrix *a, const go_matrix *b, go_matrix *apb)
- [go_result gomotion::go_matrix_matrix_copy](#) (const go_matrix *src, go_matrix *dst)
- [go_result gomotion::go_matrix_matrix_mult](#) (const go_matrix *a, const go_matrix *b, go_matrix *ab)
- [go_result gomotion::go_matrix_vector_mult](#) (const go_matrix *a, const go_vector *v, go_vector *axv)
- [go_result gomotion::go_matrix_vector_cross](#) (const go_matrix *a, const go_vector *v, go_matrix *axv)
- [go_result gomotion::go_matrix_transpose](#) (const go_matrix *a, go_matrix *at)
- [go_result gomotion::go_matrix_inv](#) (const go_matrix *m, go_matrix *minv)
- [go_result gomotion::go_mat3_inv](#) (go_real a[3][3], go_real ainv[3][3])
- [go_result gomotion::go_mat3_mat3_mult](#) (go_real a[3][3], go_real b[3][3], go_real axb[3][3])
- [go_result gomotion::go_mat3_vec3_mult](#) (go_real a[3][3], go_real v[3], go_real axv[3])
- [go_result gomotion::go_mat4_inv](#) (go_real a[4][4], go_real ainv[4][4])
- [go_result gomotion::go_mat4_mat4_mult](#) (go_real a[4][4], go_real b[4][4], go_real axb[4][4])
- [go_result gomotion::go_mat4_vec4_mult](#) (go_real a[4][4], go_real v[4], go_real axv[4])
- [go_result gomotion::go_mat6_inv](#) (go_real a[6][6], go_real ainv[6][6])
- [go_result gomotion::go_mat6_transpose](#) (go_real a[6][6], go_real at[6][6])
- [go_result gomotion::go_mat6_mat6_mult](#) (go_real a[6][6], go_real b[6][6], go_real axb[6][6])
- [go_result gomotion::go_mat6_vec6_mult](#) (go_real a[6][6], go_real v[6], go_real axv[6])
- [go_result gomotion::go_dh_hom_convert](#) (const go_dh *dh, go_hom *h)
- [go_result gomotion::go_dh_pose_convert](#) (const go_dh *dh, go_pose *p)
- [go_result gomotion::go_pose_dh_convert](#) (const go_pose *p, go_dh *dh)
- [go_result gomotion::go_hom_dh_convert](#) (const go_hom *h, go_dh *dh)
- [go_result gomotion::go_link_joint_set](#) (const go_link *link, go_real joint, go_link *linkout)
- [go_result gomotion::go_link_pose_build](#) (const go_link *link_params, go_integer num, go_pose *pose)
- [go_complex gomotion::go_complex_add](#) (go_complex z1, go_complex z2)
- [go_complex gomotion::go_complex_sub](#) (go_complex z1, go_complex z2)
- [go_complex gomotion::go_complex_mult](#) (go_complex z1, go_complex z2)
- [go_complex gomotion::go_complex_inv](#) (go_complex z, go_result *result)
- [go_complex gomotion::go_complex_div](#) (go_complex z1, go_complex z2, go_result *result)
- [go_complex gomotion::go_complex_sq](#) (go_complex z)
- [go_complex gomotion::go_complex_scale](#) (go_complex z, go_real scale)
- [go_real gomotion::go_complex_mag](#) (go_complex z)
- [go_real gomotion::go_complex_magsq](#) (go_complex z)
- [go_real gomotion::go_complex_arg](#) (go_complex z)
- [void gomotion::go_complex_sqrt](#) (go_complex z, go_complex *z1, go_complex *z2)
- [void gomotion::go_complex_cbrt](#) (go_complex z, go_complex *z1, go_complex *z2, go_complex *z3)
- [go_result gomotion::go_quadratic_solve](#) (const go_quadratic *quad, go_complex *z1, go_complex *z2)
- [go_result gomotion::go_cubic_solve](#) (const go_cubic *cub, go_complex *z1, go_complex *z2, go_complex *z3)
- [go_result gomotion::go_quartic_solve](#) (const go_quartic *quart, go_complex *z1, go_complex *z2, go_complex *z3, go_complex *z4)
- [go_result gomotion::go_tridiag_reduce](#) (go_real **a, go_integer n, go_real *d, go_real *e)
- [go_result gomotion::go_tridiag_ql](#) (go_real *d, go_real *e, go_integer n, go_real **z)
- [go_result gomotion::go_linear_cos_sin_solve](#) (go_real a, go_real b, go_real *th1, go_real *th2)

8.2.1 Detailed Description

Declarations for pose math functions.

8.2.2 Macro Definition Documentation

8.2.2.1 #define GO_2_PI (2.0*GO_PI)

The value of twice Pi.

8.2.2.2 #define go_body_copy(src, dst)

Value:

```
(dst)->mass = (src)->mass;
(dst)->inertia[0][0] = (src)->inertia[0][0];
(dst)->inertia[0][1] = (src)->inertia[0][1];
(dst)->inertia[0][2] = (src)->inertia[0][2];
(dst)->inertia[1][0] = (src)->inertia[1][0];
(dst)->inertia[1][1] = (src)->inertia[1][1];
(dst)->inertia[1][2] = (src)->inertia[1][2];
(dst)->inertia[2][0] = (src)->inertia[2][0];
(dst)->inertia[2][1] = (src)->inertia[2][1];
(dst)->inertia[2][2] = (src)->inertia[2][2];
```

8.2.2.3 #define go_body_init(b)

Value:

```
(b)->mass = 1;
(b)->inertia[0][0] = 1;
(b)->inertia[0][1] = 0;
(b)->inertia[0][2] = 0;
(b)->inertia[1][0] = 0;
(b)->inertia[1][1] = 1;
(b)->inertia[1][2] = 0;
(b)->inertia[2][0] = 0;
(b)->inertia[2][1] = 0;
(b)->inertia[2][2] = 1;
```

8.2.2.4 #define GO_CLOSE(x, y) (fabs((x)-(y)) < GO_REAL_EPSILON)

How close general quantities must be to be equal. Use this when you have something other than translational or rotational quantities, otherwise use one of *GO_TRAN,ROT_CLOSE*.

8.2.2.5 #define go_cub(x) ((x)*(x)*(x))

Returns the cube of x.

8.2.2.6 #define go_link_to_string(L)

Value:

```
(L) == GO_LINK_DH ? "DH" :
(L) == GO_LINK_PK ? "PK" :
(L) == GO_LINK_PP ? "PP" : "None"
```

8.2.2.7 #define GO_MATRIX_DECLARE(M, Mspace, _rows, _cols)

Value:

```
go_matrix M = {0, 0, 0, 0, 0, 0}; \
struct { \
    go_real *el[_rows]; \
    go_real *elcpy[_rows]; \
    go_real stg[_rows][_cols]; \
    go_real stgcpy[_rows][_cols]; \
    go_real v[_rows]; \
    go_integer index[_rows]; \
} Mspace
```

8.2.2.8 #define go_matrix_init(M, Mspace, _rows, _cols)

Value:

```
M.el = Mspace.el; \
M.elcpy = Mspace.elcpy; \
for (M.rows = 0; M.rows < (_rows) && M.rows < ISIZEOF(Mspace.el)/ISIZEOF(*Mspace.el); M. \
    rows++) { \
    M.el[M.rows] = Mspace.stg[M.rows]; \
    M.elcpy[M.rows] = Mspace.stgcpy[M.rows]; \
} \
M.cols = (_cols) < ISIZEOF(Mspace.stg[0])/ISIZEOF(*Mspace.stg[0]) ? (_cols) : \
    ISIZEOF(Mspace.stg[0])/ISIZEOF(*Mspace.stg[0]); \
M.v = Mspace.v; \
M.index = Mspace.index
```

8.2.2.9 #define GO_PI 3.14159265358979323846

The value of Pi.

8.2.2.10 #define GO_PI_2 1.57079632679489661923

The value of half of Pi.

8.2.2.11 #define GO_PI_4 0.78539816339744830962

The value of one-fourth of Pi.

8.2.2.12 #define go_qua(x) ((x)*(x)*(x)*(x))

Returns x to the fourth power.

8.2.2.13 #define GO_ROT_CLOSE(x, y) (fabs((x)-(y)) < GO_REAL_EPSILON)

How close rotational quantities must be to be equal.

8.2.2.14 #define GO_ROT_SMALL(x) (fabs(x) < GO_REAL_EPSILON)

How small a rotational quantity must be to be zero.

8.2.2.15 `#define GO_SMALL(x) (fabs(x) < GO_REAL_EPSILON)`

How small a general quantity must be to be zero. Use this when you have something other than a translational or rotational quantity, otherwise use one of `GO_TRAN`, `ROT_SMALL`.

8.2.2.16 `#define go_sq(x) ((x)*(x))`

Returns the square of *x*.

8.2.2.17 `#define GO_TO_DEG(rad) ((rad)*57.295779513082323)`

Returns *rad* in radians as its value in degrees.

8.2.2.18 `#define GO_TO_RAD(deg) ((deg)*0.0174532925199432952)`

Returns *deg* in degrees as its value in radians.

8.2.2.19 `#define GO_TRAN_CLOSE(x, y) (fabs((x)-(y)) < GO_REAL_EPSILON)`

How close translational quantities must be to be equal.

8.2.2.20 `#define GO_TRAN_SMALL(x) (fabs(x) < GO_REAL_EPSILON)`

How small a translational quantity must be to be zero.

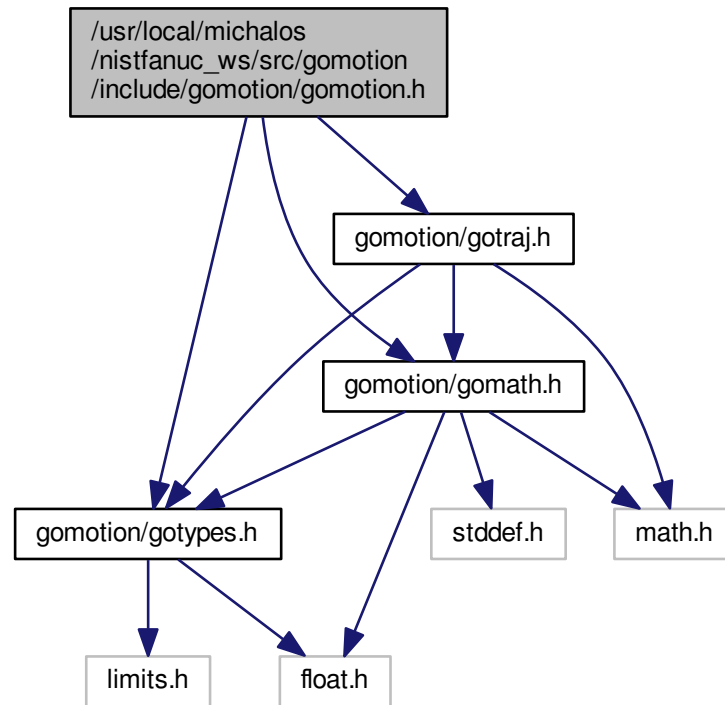
8.2.2.21 `#define ISIZEOF(x) ((int) sizeof(x))`

8.3 /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomotion.h File Reference

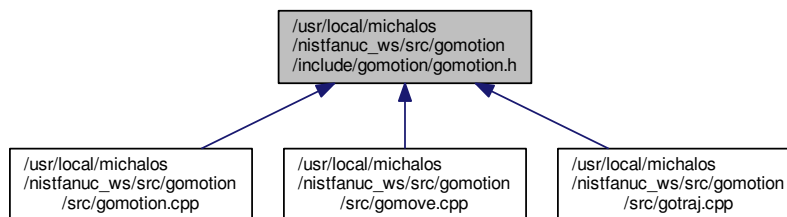
Declarations for motion queue manipulation.

```
#include "gomotion/gotypes.h"
#include "gomotion/gomath.h"
#include "gomotion/gotraj.h"
```

Include dependency graph for gomotion.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `gomotion::go_position`

Depending upon whether you are doing joint interpolation or world coordinate interpolation (as specified by your call to `go_motion_queue_set_type()`), fill in `joint[]` or `pose` accordingly.

- struct [gomotion::go_motion_params](#)
- struct [gomotion::go_motion_linear_params](#)

In the following comments, LIN, CIR and ALL refer to linear moves, circular moves and both, respectively.

- struct [gomotion::go_motion_circular_params](#)
- struct [gomotion::go_motion_spec](#)
- struct [gomotion::go_scale_spec](#)
- struct [gomotion::go_motion_queue](#)

Namespaces

- [gomotion](#)

Macros

- `#define GO_MOTION_JOINT_NUM 8 /** number of joints supported */`

Enumerations

- enum {
[gomotion::GO_MOTION_NONE](#), [gomotion::GO_MOTION_JOINT](#), [gomotion::GO_MOTION_UJOINT](#), [gomotion::GO_MOTION_WORLD](#),
[gomotion::GO_MOTION_LINEAR](#), [gomotion::GO_MOTION_CIRCULAR](#) }

8.3.1 Detailed Description

Declarations for motion queue manipulation.

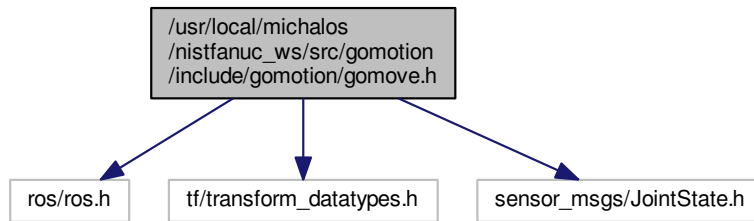
8.3.2 Macro Definition Documentation

8.3.2.1 `#define GO_MOTION_JOINT_NUM 8 /** number of joints supported */`

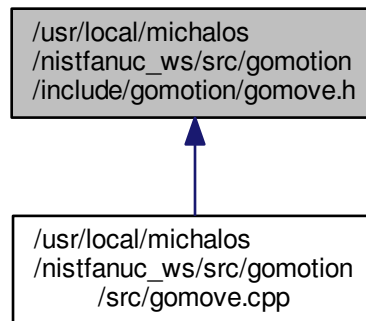
8.4 /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gomove.h File Reference

```
#include <ros/ros.h>
#include <tf/transform_datatypes.h>
#include <sensor_msgs/JointState.h>
```

Include dependency graph for gomove.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gomotion::GoMotionParams](#)
- class [gomotion::GoMotion](#)

Namespaces

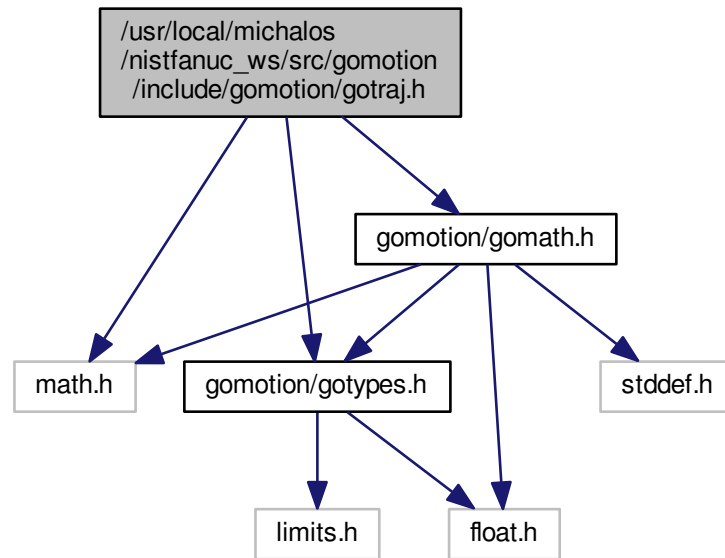
- [gomotion](#)

Typedefs

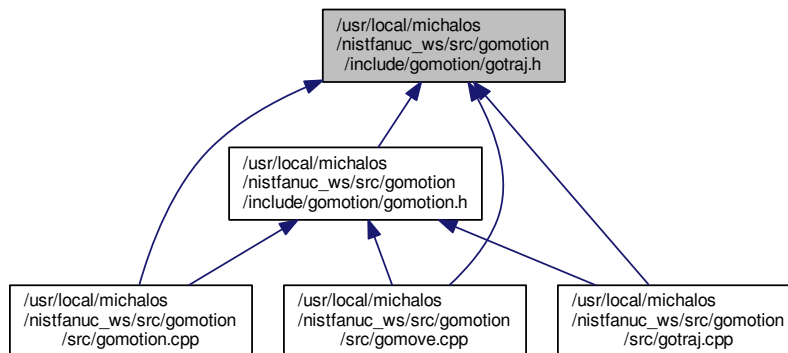
- typedef
sensor_msgs::JointState_
< std::allocator< void > > [gomotion::JointState](#)

8.5 /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gotraj.h File Reference

```
#include <math.h>
#include "gomotion/gotypes.h"
#include "gomotion/gomath.h"
Include dependency graph for gotraj.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gomotion::go_traj_ca_spec](#)
- struct [gomotion::go_traj_cj_spec](#)
- struct [gomotion::go_traj_interp_spec](#)

Namespaces

- [gomotion](#)

Macros

- `#define reciprocate(x) (x) <= 0.0 ? GO_INF : 1.0 / (x)`

Functions

- [go_result gomotion::go_traj_ca_generate](#) ([go_real](#) acc, [go_real](#) deltacc, [go_real](#) deltvel, [go_traj_ca_spec](#) *pts)
[go_traj_ca_generate\(\)](#) takes an accel value, and intervals for the accel period and cruise period, and fills in the [go_traj_ca_spec](#) with the interval parameters.
- [go_result gomotion::go_traj_ca_compute](#) ([go_real](#) d, [go_real](#) v, [go_real](#) a, [go_traj_ca_spec](#) *pts)
[go_traj_ca_compute\(\)](#) takes values for distance 'd' to move, max velocity 'v' to limit if necessary, and constant accel 'a', and fills in the [go_traj_ca_spec](#) with the interval parameters.
- [go_result gomotion::go_traj_ca_scale](#) (const [go_traj_ca_spec](#) *ts, [go_real](#) t, [go_traj_ca_spec](#) *pts)
[go_traj_ca_scale\(\)](#) takes a time 't' for the desired time of the motion, and scales the times and motion params so that the total distance remains the same and everything else is in proportion
- [go_result gomotion::go_traj_ca_stop](#) (const [go_traj_ca_spec](#) *ts, [go_real](#) t, [go_traj_ca_spec](#) *pts)
[go_traj_ca_stop\(\)](#) takes a time 't' for the desired time to begin stopping, and recomputes the times so that the move will stop as soon as possible during subsequent interps.
- [go_result gomotion::go_traj_ca_extend](#) (const [go_traj_ca_spec](#) *ts, [go_real](#) t, [go_traj_ca_spec](#) *pts)
[go_traj_ca_extend\(\)](#) takes a time 't' for the desired time to finish the motion, and extends the constant-speed section so that it stops then.
- [go_result gomotion::go_traj_ca_interp](#) (const [go_traj_ca_spec](#) *ts, [go_real](#) t, [go_traj_interp_spec](#) *ti)
[go_traj_ca_interp\(\)](#) takes a [go_traj_ca_spec](#) and interpolates the d-v-a values for the given time t, storing the d-v-a values in ti.
- [go_result gomotion::go_traj_cj_generate](#) ([go_real](#) jrk, [go_real](#) deltjrk, [go_real](#) deltacc, [go_real](#) deltvel, [go_traj_cj_spec](#) *pts)
[go_traj_cj_generate\(\)](#) takes a jerk value, and intervals for the jerk period, accel period and cruise period, and fills in the [go_traj_cj_spec](#) with the interval parameters.
- [go_result gomotion::go_traj_cj_compute](#) ([go_real](#) d, [go_real](#) v, [go_real](#) a, [go_real](#) j, [go_traj_cj_spec](#) *pts)
[go_traj_cj_compute\(\)](#) takes values for distance 'd' to move, max velocity 'v' to limit if necessary, max accel 'a' to limit if necessary, and constant jerk 'j', and fills in the [go_traj_cj_spec](#) with the interval parameters.
- [go_result gomotion::go_traj_cj_scale](#) (const [go_traj_cj_spec](#) *ts, [go_real](#) t, [go_traj_cj_spec](#) *pts)
[go_traj_cj_scale\(\)](#) takes a time 't' for the desired time of the motion, and scales the times and motion params so that the total distance remains the same and everything else is in proportion
- [go_result gomotion::go_traj_cj_stop](#) (const [go_traj_cj_spec](#) *ts, [go_real](#) t, [go_traj_cj_spec](#) *pts)
[go_traj_cj_stop\(\)](#) takes a time 't' for the desired time to begin stopping, and recomputes the times so that the move will stop as soon as possible during subsequent interps.
- [go_result gomotion::go_traj_cj_extend](#) (const [go_traj_cj_spec](#) *ts, [go_real](#) t, [go_traj_cj_spec](#) *pts)
[go_traj_cj_extend\(\)](#) takes a time 't' for the desired time to finish the motion, and extends the constant-speed section so that it stops then.

- `go_result gomotion::go_traj_cj_interp` (const `go_traj_cj_spec` *ts, `go_real` t, `go_traj_interp_spec` *ti)
`go_traj_cj_interp()` takes a `go_traj_cj_spec` and interpolates the d-v-a-j values for the given time t, storing the d-v-a-j values in ti.
- `go_result gomotion::go_traj_cj_compute_the_rest` (`go_traj_cj_spec` *pts)
! Expects pts attributes dtend, jt0, at1, vt3, t1, t2 and t3 to be already filled in from previous calls to one of the four `go_traj_cj_compute_tees` functions and `go_traj_cj_compute`.
- `go_result gomotion::go_traj_cj_compute_tees_na_nc` (`go_traj_cj_spec` *pts)
! Computes the trajectory defining times t1, t2 and t4 for a motion with no acceleration or cruise phase, given pts filled in with dtend and jt0.
- `go_result gomotion::go_traj_cj_compute_tees_a_nc` (`go_traj_cj_spec` *pts)
! Computes the trajectory defining times t1, t2 and t4 for a motion with acceleration but no cruise phase, given pts filled in with dtend, jt0 and at1.
- `go_result gomotion::go_traj_cj_compute_tees_na_c` (`go_traj_cj_spec` *pts)
! Computes the trajectory defining times t1, t2 and t4 for a motion with no acceleration but a cruise phase, given pts filled in with dtend, jt0 and vt3.
- `go_result gomotion::go_traj_cj_compute_tees_a_c` (`go_traj_cj_spec` *pts)
! Computes the trajectory defining times t1, t2 and t4 for a motion with no acceleration but a cruise phase, given pts filled in with dtend, jt0, at1 and vt3.

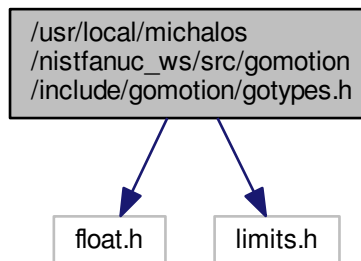
8.5.1 Macro Definition Documentation

8.5.1.1 `#define reciprocate(x)(x) <= 0.0 ? GO_INF : 1.0 / (x)`

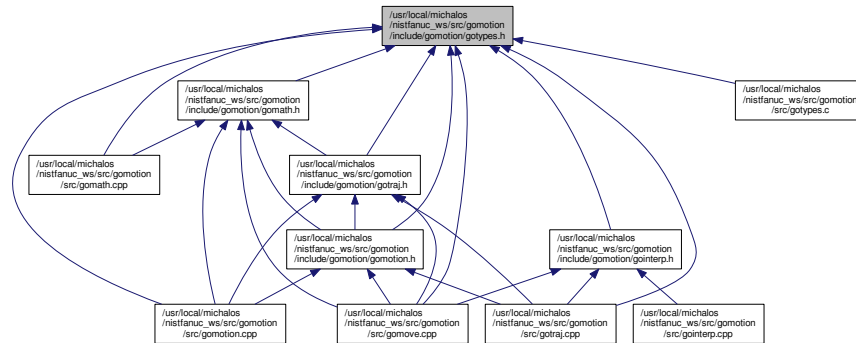
8.6 /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gotypes.h File Reference

```
#include <float.h>
#include <limits.h>
```

Include dependency graph for gotypes.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define GO_RESULT_INT`
- `#define GO_RESULT go_result_int`
- `#define go_result_to_string(r)`
- `#define go_quantity_to_string(q)`
- `#define GO_REAL_DOUBLE`
- `#define GO_REAL go_real_double`
- `#define GO_REAL_MIN DBL_MIN`
- `#define GO_REAL_MAX DBL_MAX`
- `#define GO_REAL_EPSILON (1.0e-7)`
- `#define GO_INF DBL_MAX`
- `#define GO_INTEGER_INT`
- `#define GO_INTEGER go_integer_int`
- `#define GO_INTEGER_MAX INT_MAX`
- `#define GO_FLAG_UCHAR`
- `#define GO_FLAG go_flag_uchar`

Typedefs

- `typedef int go_result`
- `typedef double go_real`
- `typedef int go_integer`
- `typedef unsigned char go_flag`

Enumerations

- `enum {`
`GO_RESULT_OK = 0, GO_RESULT_IGNORED, GO_RESULT_BAD_ARGS, GO_RESULT_RANGE_ERROR,`
`GO_RESULT_DOMAIN_ERROR, GO_RESULT_ERROR, GO_RESULT_IMPL_ERROR, GO_RESULT_NORM-`
`ERROR,`
`GO_RESULT_DIV_ERROR, GO_RESULT_SINGULAR, GO_RESULT_NO_SPACE, GO_RESULT_EMPTY,`
`GO_RESULT_BUG }`
- `enum { GO_QUANTITY_NONE = 0, GO_QUANTITY_LENGTH, GO_QUANTITY_ANGLE }`

Variables

- [int go_result_int](#)
- [int go_real_double](#)
- [int go_integer_int](#)
- [int go_flag_uchar](#)
- [go_flag gocode](#)

8.6.1 Macro Definition Documentation

8.6.1.1 `#define GO_FLAG go_flag_uchar`

8.6.1.2 `#define GO_FLAG_UCHAR`

8.6.1.3 `#define GO_INF DBL_MAX`

8.6.1.4 `#define GO_INTEGER go_integer_int`

8.6.1.5 `#define GO_INTEGER_INT`

8.6.1.6 `#define GO_INTEGER_MAX INT_MAX`

8.6.1.7 `#define go_quantity_to_string(q)`

Value:

```
(q) == GO_QUANTITY_LENGTH ? "Length" : \
(q) == GO_QUANTITY_ANGLE ? "Angle" : "None"
```

8.6.1.8 `#define GO_REAL go_real_double`

8.6.1.9 `#define GO_REAL_DOUBLE`

8.6.1.10 `#define GO_REAL_EPSILON (1.0e-7)`

8.6.1.11 `#define GO_REAL_MAX DBL_MAX`

8.6.1.12 `#define GO_REAL_MIN DBL_MIN`

8.6.1.13 `#define GO_RESULT go_result_int`

8.6.1.14 `#define GO_RESULT_INT`

8.6.1.15 `#define go_result_to_string(r)`

Value:

```
(r) == GO_RESULT_OK ? "Ok" : \
(r) == GO_RESULT_IGNORED ? "Ignored" : \
(r) == GO_RESULT_BAD_ARGS ? "Bad Args" : \
(r) == GO_RESULT_RANGE_ERROR ? "Range Error" : \
(r) == GO_RESULT_DOMAIN_ERROR ? "Domain Error" : \
(r) == GO_RESULT_ERROR ? "General Error" :
```

```
(r) == GO_RESULT_IMPL_ERROR ? "Implementation Error" :
(r) == GO_RESULT_NORM_ERROR ? "Norm Error" :
(r) == GO_RESULT_DIV_ERROR ? "Div Error" :
(r) == GO_RESULT_SINGULAR ? "Singular" :
(r) == GO_RESULT_NO_SPACE ? "No Space" :
(r) == GO_RESULT_EMPTY ? "Empty" :
(r) == GO_RESULT_BUG ? "Bug" : "?"
```

8.6.2 Typedef Documentation

8.6.2.1 typedef unsigned char go_flag

8.6.2.2 typedef int go_integer

8.6.2.3 typedef double go_real

8.6.2.4 typedef int go_result

8.6.3 Enumeration Type Documentation

8.6.3.1 anonymous enum

GO_RESULT symbols run through a small range of values, on the order of tens, suitable for a byte. GO_RESULT_OK is zero for easy detection of error conditions, e.g., if (result) { handle error }

Enumerator

GO_RESULT_OK
GO_RESULT_IGNORED
GO_RESULT_BAD_ARGS
GO_RESULT_RANGE_ERROR
GO_RESULT_DOMAIN_ERROR
GO_RESULT_ERROR
GO_RESULT_IMPL_ERROR
GO_RESULT_NORM_ERROR
GO_RESULT_DIV_ERROR
GO_RESULT_SINGULAR
GO_RESULT_NO_SPACE
GO_RESULT_EMPTY
GO_RESULT_BUG

8.6.3.2 anonymous enum

Joints are characterized by the quantities they affect, such as length for linear joints and angle for rotary joints.

Enumerator

GO_QUANTITY_NONE
GO_QUANTITY_LENGTH
GO_QUANTITY_ANGLE

8.6.4 Variable Documentation

8.6.4.1 `int go_flag_uchar`

8.6.4.2 `int go_integer_int`

8.6.4.3 `int go_real_double`

8.6.4.4 `int go_result_int`

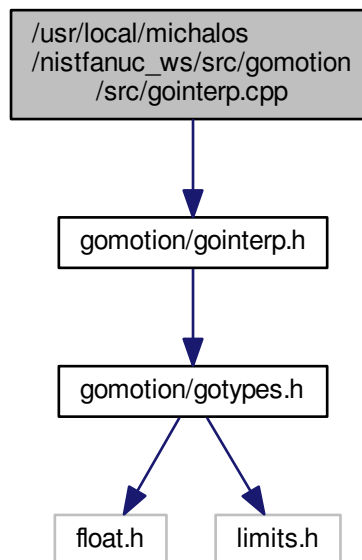
8.6.4.5 `go_flag` `gocode`

An ad-hoc flag set by various functions that can be used to indicate various code paths taken.

8.7 /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gointerp.cpp File Reference

```
#include "gomotion/gointerp.h"
```

Include dependency graph for gointerp.cpp:

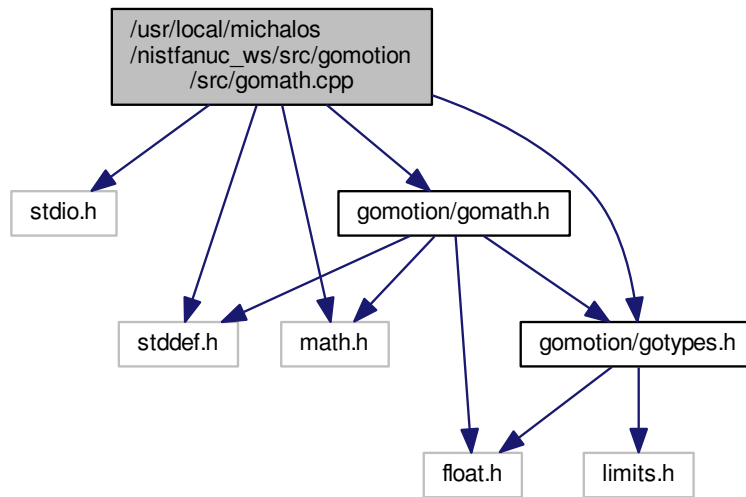


Namespaces

- [gomotion](#)

8.8 /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gomath.cpp File Reference

```
#include <stdio.h>
#include <stddef.h>
#include <math.h>
#include "gomotion/gotypes.h"
#include "gomotion/gomath.h"
Include dependency graph for gomath.cpp:
```



Namespaces

- [gomotion](#)

Macros

- `#define CBRT_IS_POW`
- `#define SIGN(a, b) ((b) >= 0.0 ? fabs(a) : -fabs(a))`
- `#define MAG2(a, b) sqrt((a)*(a)+(b)*(b))`
- `#define COL_IS_UNIT(r) GO_TRAN_CLOSE(go_sq((r).x) + go_sq((r).y) + go_sq((r).z), 1)`

Functions

- `void gomotion::sincos (double x, double *sx, double *cx)`
- `void gomotion::go_sincos (go_real t, go_real *s, go_real *c)`
- `go_result gomotion::go_asines (go_real s, go_real *asp, go_real *asn)`
- `go_result gomotion::go_acoses (go_real c, go_real *acp, go_real *acn)`
- `go_result gomotion::go_atans (go_real t, go_real *atp, go_real *atn)`

- [go_real gomotion::go_cbrt \(go_real x\)](#)
- [go_result gomotion::go_cart_sph_convert \(const go_cart *v, go_sph *s\)](#)
- [go_result gomotion::go_cart_cyl_convert \(const go_cart *v, go_cyl *c\)](#)
- [go_result gomotion::go_sph_cart_convert \(const go_sph *s, go_cart *v\)](#)
- [go_result gomotion::go_sph_cyl_convert \(const go_sph *s, go_cyl *c\)](#)
- [go_result gomotion::go_cyl_cart_convert \(const go_cyl *c, go_cart *v\)](#)
- [go_result gomotion::go_cyl_sph_convert \(const go_cyl *c, go_sph *s\)](#)
- [go_result gomotion::go_rvec_quat_convert \(const go_rvec *r, go_quat *q\)](#)
- [go_result gomotion::go_rvec_mat_convert \(const go_rvec *r, go_mat *m\)](#)
- [go_result gomotion::go_rvec_zyz_convert \(const go_rvec *rvec, go_zyz *zyz\)](#)
- [go_result gomotion::go_rvec_zyx_convert \(const go_rvec *rvec, go_zyx *zyx\)](#)
- [go_result gomotion::go_rvec_rpy_convert \(const go_rvec *r, go_rpy *rpy\)](#)
- [go_result gomotion::go_quat_rvec_convert \(const go_quat *q, go_rvec *r\)](#)
- [go_result gomotion::go_quat_mat_convert \(const go_quat *q, go_mat *m\)](#)
- [go_result gomotion::go_quat_zyz_convert \(const go_quat *q, go_zyz *zyz\)](#)
- [go_result gomotion::go_quat_zyx_convert \(const go_quat *q, go_zyx *zyx\)](#)
- [go_result gomotion::go_quat_rpy_convert \(const go_quat *q, go_rpy *rpy\)](#)
- [go_result gomotion::go_mat_rvec_convert \(const go_mat *m, go_rvec *r\)](#)
- [go_result gomotion::go_mat_quat_convert \(const go_mat *m, go_quat *q\)](#)
- [go_result gomotion::go_mat_zyz_convert \(const go_mat *m, go_zyz *zyz\)](#)
- [go_result gomotion::go_mat_zyx_convert \(const go_mat *m, go_zyx *zyx\)](#)
- [go_result gomotion::go_mat_xyz_convert \(const go_mat *m, go_xyz *xyz\)](#)
- [go_result gomotion::go_mat_rpy_convert \(const go_mat *m, go_rpy *rpy\)](#)
- [go_result gomotion::go_zyz_rvec_convert \(const go_zyz *zyz, go_rvec *r\)](#)
- [go_result gomotion::go_zyz_quat_convert \(const go_zyz *zyz, go_quat *q\)](#)
- [go_result gomotion::go_zyz_mat_convert \(const go_zyz *zyz, go_mat *m\)](#)
- [go_result gomotion::go_zyz_zyx_convert \(const go_zyz *zyz, go_zyx *zyx\)](#)
- [go_result gomotion::go_zyz_rpy_convert \(const go_zyz *zyz, go_rpy *rpy\)](#)
- [go_result gomotion::go_zyx_rvec_convert \(const go_zyx *zyx, go_rvec *r\)](#)
- [go_result gomotion::go_zyx_quat_convert \(const go_zyx *zyx, go_quat *q\)](#)
- [go_result gomotion::go_zyx_mat_convert \(const go_zyx *zyx, go_mat *m\)](#)
- [go_result gomotion::go_zyx_zyz_convert \(const go_zyx *zyx, go_zyz *zyz\)](#)
- [go_result gomotion::go_zyx_rpy_convert \(const go_zyx *zyx, go_rpy *rpy\)](#)
- [go_result gomotion::go_xyz_mat_convert \(const go_xyz *xyz, go_mat *m\)](#)
- [go_result gomotion::go_rpy_rvec_convert \(const go_rpy *rpy, go_rvec *rvec\)](#)
- [go_result gomotion::go_rpy_quat_convert \(const go_rpy *rpy, go_quat *quat\)](#)
- [go_result gomotion::go_rpy_mat_convert \(const go_rpy *rpy, go_mat *m\)](#)
- [go_result gomotion::go_rpy_zyz_convert \(const go_rpy *rpy, go_zyz *zyz\)](#)
- [go_result gomotion::go_rpy_zyx_convert \(const go_rpy *rpy, go_zyx *zyx\)](#)
- [go_result gomotion::go_uxz_mat_convert \(const go_uxz *uxz, go_mat *mat\)](#)
- [go_result gomotion::go_mat_uxz_convert \(const go_mat *mat, go_uxz *uxz\)](#)
- [go_pose gomotion::go_pose_this \(go_real x, go_real y, go_real z, go_real rs, go_real rx, go_real ry, go_real rz\)](#)
- [go_cart gomotion::go_cart_zero \(void\)](#)
- [go_quat gomotion::go_quat_identity \(void\)](#)
- [go_pose gomotion::go_pose_identity \(void\)](#)
- [go_result gomotion::go_pose_hom_convert \(const go_pose *p, go_hom *h\)](#)
- [go_result gomotion::go_hom_pose_convert \(const go_hom *h, go_pose *p\)](#)
- [go_result gomotion::go_cart_rvec_convert \(const go_cart *cart, go_rvec *rvec\)](#)
- [go_result gomotion::go_rvec_cart_convert \(const go_rvec *rvec, go_cart *cart\)](#)
- [go_flag gomotion::go_cart_cart_compare \(const go_cart *v1, const go_cart *v2\)](#)
- [go_result gomotion::go_cart_cart_dot \(const go_cart *v1, const go_cart *v2, go_real *d\)](#)

- [go_result gomotion::go_cart_cart_cross](#) (const go_cart *v1, const go_cart *v2, go_cart *vout)
- [go_result gomotion::go_cart_mag](#) (const go_cart *v, [go_real](#) *d)
- [go_result gomotion::go_cart_magsq](#) (const go_cart *v, [go_real](#) *d)
- [go_flag gomotion::go_cart_cart_par](#) (const go_cart *v1, const go_cart *v2)
- [go_flag gomotion::go_cart_cart_perp](#) (const go_cart *v1, const go_cart *v2)
- [go_result gomotion::go_cart_cart_disp](#) (const go_cart *v1, const go_cart *v2, [go_real](#) *d)
- [go_result gomotion::go_cart_cart_add](#) (const go_cart *v1, const go_cart *v2, go_cart *vout)
- [go_result gomotion::go_cart_cart_sub](#) (const go_cart *v1, const go_cart *v2, go_cart *vout)
- [go_result gomotion::go_cart_scale_mult](#) (const go_cart *v1, [go_real](#) d, go_cart *vout)
- [go_result gomotion::go_cart_neg](#) (const go_cart *v1, go_cart *vout)
- [go_result gomotion::go_cart_unit](#) (const go_cart *v, go_cart *vout)
- [go_result gomotion::go_cart_is_norm](#) (const go_cart *v)
- [go_result gomotion::go_cart_cart_rot](#) (const go_cart *v1, const go_cart *v2, go_quat *quat)
- [go_result gomotion::go_cart_cart_proj](#) (const go_cart *v1, const go_cart *v2, go_cart *vout)
- [go_result gomotion::go_cart_plane_proj](#) (const go_cart *v, const go_cart *normal, go_cart *vout)
- [go_result gomotion::go_cart_cart_angle](#) (const go_cart *v1, const go_cart *v2, [go_real](#) *a)
- [go_result gomotion::go_cart_normal](#) (const go_cart *v, go_cart *vout)
- [go_result gomotion::go_cart_centroid](#) (const go_cart *varray, [go_integer](#) num, go_cart *centroid)
- [go_result gomotion::go_cart_centroidize](#) (const go_cart *varray, [go_integer](#) num, go_cart *centroid, go_cart *voutarray)
- [go_complex gomotion::go_complex_add](#) (go_complex z1, go_complex z2)
- [go_complex gomotion::go_complex_sub](#) (go_complex z1, go_complex z2)
- [go_complex gomotion::go_complex_mult](#) (go_complex z1, go_complex z2)
- [go_complex gomotion::go_complex_inv](#) (go_complex z, [go_result](#) *result)
- [go_complex gomotion::go_complex_div](#) (go_complex z1, go_complex z2, [go_result](#) *result)
- [go_complex gomotion::go_complex_sq](#) (go_complex z)
- [go_complex gomotion::go_complex_scale](#) (go_complex z, [go_real](#) scale)
- [go_real gomotion::go_complex_mag](#) (go_complex z)
- [go_real gomotion::go_complex_magsq](#) (go_complex z)
- [go_real gomotion::go_complex_arg](#) (go_complex z)
- [void gomotion::go_complex_sqrt](#) (go_complex z, go_complex *z1, go_complex *z2)
- [void gomotion::go_complex_cbrt](#) (go_complex z, go_complex *z1, go_complex *z2, go_complex *z3)
- [go_result gomotion::go_quadratic_solve](#) (const go_quadratic *quad, go_complex *z1, go_complex *z2)
- [go_result gomotion::go_cubic_solve](#) (const go_cubic *cub, go_complex *z1, go_complex *z2, go_complex *z3)
- [go_result gomotion::go_quartic_solve](#) (const go_quartic *quart, go_complex *z1, go_complex *z2, go_complex *z3, go_complex *z4)
- [go_result gomotion::go_tridiag_reduce](#) ([go_real](#) **a, [go_integer](#) n, [go_real](#) *d, [go_real](#) *e)
- [go_result gomotion::go_tridiag_qi](#) ([go_real](#) *d, [go_real](#) *e, [go_integer](#) n, [go_real](#) **z)
- [go_result gomotion::go_cart_cart_pose](#) (const go_cart *v1, const go_cart *v2, go_cart *v1c, go_cart *v2c, [go_integer](#) num, go_pose *pout)
- [go_result gomotion::go_cart_trilaterate](#) (const go_cart *c1, const go_cart *c2, const go_cart *c3, [go_real](#) l1, [go_real](#) l2, [go_real](#) l3, go_cart *out1, go_cart *out2)
- [go_flag gomotion::go_rvec_rvec_compare](#) (const go_rvec *r1, const go_rvec *r2)
- [go_result gomotion::go_rvec_scale_mult](#) (const go_rvec *r, [go_real](#) s, go_rvec *rout)
- [go_result gomotion::go_rpy_cart_mult](#) (const go_rpy *rpy, const go_cart *in, go_cart *out)
- [go_result gomotion::go_mat_norm](#) (const go_mat *mat, go_mat *mout)
- [go_flag gomotion::go_mat_is_norm](#) (const go_mat *m)
- [go_result gomotion::go_mat_inv](#) (const go_mat *m, go_mat *mout)
- [go_result gomotion::go_mat_cart_mult](#) (const go_mat *m, const go_cart *v, go_cart *vout)
- [go_result gomotion::go_mat_mat_mult](#) (const go_mat *m1, const go_mat *m2, go_mat *mout)
- [go_flag gomotion::go_quat_quat_compare](#) (const go_quat *q1, const go_quat *q2)

- `go_result gomotion::go_quat_mag` (const go_quat *quat, go_real *d)
- `go_result gomotion::go_quat_unit` (const go_quat *q1, go_quat *qout)
- `go_result gomotion::go_quat_norm` (const go_quat *q1, go_quat *qout)
- `go_result gomotion::go_quat_inv` (const go_quat *q1, go_quat *qout)
- `go_flag gomotion::go_quat_is_norm` (const go_quat *q1)
- `go_result gomotion::go_quat_scale_mult` (const go_quat *q, go_real s, go_quat *qout)
- `go_result gomotion::go_quat_quat_mult` (const go_quat *q1, const go_quat *q2, go_quat *qout)
- `go_result gomotion::go_quat_cart_mult` (const go_quat *q1, const go_cart *v2, go_cart *vout)
- `go_flag gomotion::go_pose_pose_compare` (const go_pose *p1, const go_pose *p2)
- `go_result gomotion::go_pose_inv` (const go_pose *p1, go_pose *p2)
- `go_result gomotion::go_pose_cart_mult` (const go_pose *p1, const go_cart *v2, go_cart *vout)
- `go_result gomotion::go_pose_pose_mult` (const go_pose *p1, const go_pose *p2, go_pose *pout)
- `go_result gomotion::go_pose_scale_mult` (const go_pose *p1, go_real s, go_pose *pout)
- `go_result gomotion::go_pose_pose_interp` (go_real t1, const go_pose *p1, go_real t2, const go_pose *p2, go_real t3, go_pose *p3)
- `go_result gomotion::go_hom_inv` (const go_hom *h1, go_hom *h2)
- `go_result gomotion::go_hom_hom_mult` (const go_hom *h1, const go_hom *h2, go_hom *hout)
- `go_result gomotion::go_pose_vel_mult` (const go_pose *pose, const go_vel *vel, go_vel *out)
- `go_result gomotion::go_line_from_point_direction` (const go_cart *point, const go_cart *direction, go_line *line)
- `go_result gomotion::go_line_from_points` (const go_cart *point1, const go_cart *point2, go_line *line)
- `go_result gomotion::go_line_from_planes` (const go_plane *plane1, const go_plane *plane2, go_line *line)
- `go_flag gomotion::go_line_line_compare` (const go_line *line1, const go_line *line2)
- `go_result gomotion::go_line_evaluate` (const go_line *line, go_real d, go_cart *point)
- `go_result gomotion::go_point_line_distance` (const go_cart *point, const go_line *line, go_real *distance)
- `go_result gomotion::go_point_line_proj` (const go_cart *point, const go_line *line, go_cart *pout)
- `go_result gomotion::go_point_plane_proj` (const go_cart *point, const go_plane *plane, go_cart *proj)
- `go_result gomotion::go_line_plane_proj` (const go_line *line, const go_plane *plane, go_line *proj)
- `go_result gomotion::go_plane_from_point_normal` (const go_cart *point, const go_cart *normal, go_plane *plane)
- `go_result gomotion::go_plane_from_abcd` (go_real A, go_real B, go_real C, go_real D, go_plane *plane)
- `go_result gomotion::go_plane_from_points` (const go_cart *point1, const go_cart *point2, const go_cart *point3, go_plane *plane)
- `go_result gomotion::go_plane_from_point_line` (const go_cart *point, const go_line *line, go_plane *plane)
- `go_flag gomotion::go_plane_plane_compare` (const go_plane *plane1, const go_plane *plane2)
- `go_result gomotion::go_point_plane_distance` (const go_cart *point, const go_plane *plane, go_real *distance)
- `go_result gomotion::go_plane_evaluate` (const go_plane *plane, go_real u, go_real v, go_cart *point)
- `go_result gomotion::go_line_plane_intersect` (const go_line *line, const go_plane *plane, go_cart *point, go_real *distance)
- `go_real gomotion::go_get_singular_epsilon` (void)
- `go_result gomotion::go_set_singular_epsilon` (go_real epsilon)
- `go_result gomotion::ludcmp` (go_real **a, go_real *scratchrow, go_integer n, go_integer *indx, go_real *d)
- `go_result gomotion::lubksb` (go_real **a, go_integer n, go_integer *indx, go_real *b)
- `go_result gomotion::go_cart_vector_convert` (const go_cart *c, go_real *v)
- `go_result gomotion::go_vector_cart_convert` (const go_real *v, go_cart *c)
- `go_result gomotion::go_quat_matrix_convert` (const go_quat *quat, go_matrix *matrix)
- `go_result gomotion::go_mat_matrix_convert` (const go_mat *mat, go_matrix *matrix)
- `go_result gomotion::go_matrix_matrix_add` (const go_matrix *a, const go_matrix *b, go_matrix *apb)
- `go_result gomotion::go_matrix_matrix_copy` (const go_matrix *src, go_matrix *dst)
- `go_result gomotion::go_matrix_matrix_mult` (const go_matrix *a, const go_matrix *b, go_matrix *ab)
- `go_result gomotion::go_matrix_vector_mult` (const go_matrix *a, const go_vector *v, go_vector *avv)
- `go_result gomotion::go_matrix_vector_cross` (const go_matrix *a, const go_vector *v, go_matrix *avv)

- [go_result gomotion::go_matrix_transpose](#) (const go_matrix *a, go_matrix *at)
- [go_result gomotion::go_matrix_inv](#) (const go_matrix *m, go_matrix *minv)
- [go_result gomotion::go_mat3_inv](#) (go_real a[3][3], go_real ainv[3][3])
- [go_result gomotion::go_mat3_mat3_mult](#) (go_real a[3][3], go_real b[3][3], go_real axb[3][3])
- [go_result gomotion::go_mat3_vec3_mult](#) (go_real a[3][3], go_real v[3], go_real axv[3])
- [go_result gomotion::go_mat4_inv](#) (go_real a[4][4], go_real ainv[4][4])
- [go_result gomotion::go_mat4_mat4_mult](#) (go_real a[4][4], go_real b[4][4], go_real axb[4][4])
- [go_result gomotion::go_mat4_vec4_mult](#) (go_real a[4][4], go_real v[4], go_real axv[4])
- [go_result gomotion::go_mat6_inv](#) (go_real a[6][6], go_real ainv[6][6])
- [go_result gomotion::go_mat6_transpose](#) (go_real a[6][6], go_real at[6][6])
- [go_result gomotion::go_mat6_mat6_mult](#) (go_real a[6][6], go_real b[6][6], go_real axb[6][6])
- [go_result gomotion::go_mat6_vec6_mult](#) (go_real a[6][6], go_real v[6], go_real axv[6])
- [go_result gomotion::go_dh_hom_convert](#) (const go_dh *dh, go_hom *h)
- [go_result gomotion::go_dh_pose_convert](#) (const go_dh *dh, go_pose *p)
- [go_result gomotion::go_hom_dh_convert](#) (const go_hom *h, go_dh *dh)
- [go_result gomotion::go_pose_dh_convert](#) (const go_pose *p, go_dh *dh)
- [go_result gomotion::go_link_joint_set](#) (const go_link *link, go_real joint, go_link *linkout)
- [go_result gomotion::go_link_pose_build](#) (const go_link *link_params, go_integer num, go_pose *pose)
- [go_result gomotion::go_linear_cos_sin_solve](#) (go_real a, go_real b, go_real *th1, go_real *th2)

8.8.1 Macro Definition Documentation

8.8.1.1 `#define CBRT_IS_POW`

8.8.1.2 `#define COL_IS_UNIT(r) GO_TRAN_CLOSE(go_sq((r).x) + go_sq((r).y) + go_sq((r).z), 1)`

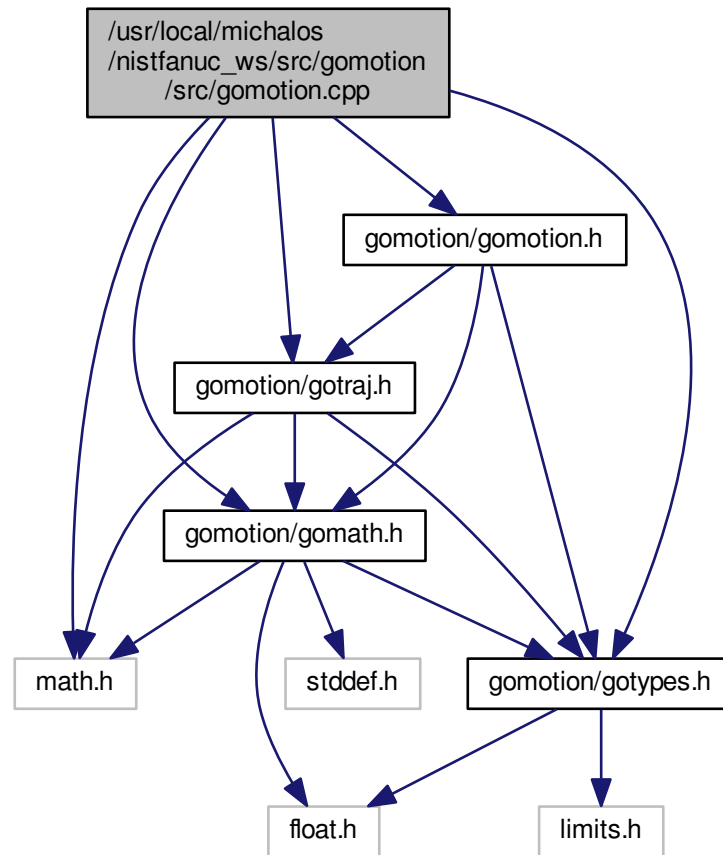
8.8.1.3 `#define MAG2(a, b) sqrt((a)*(a)+(b)*(b))`

8.8.1.4 `#define SIGN(a, b) ((b) >= 0.0 ? fabs(a) : -fabs(a))`

8.9 /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gomotion.cpp File Reference

```
#include <math.h>
#include "gomotion/gotypes.h"
#include "gomotion/gomath.h"
#include "gomotion/gotraaj.h"
#include "gomotion/gomotion.h"
```

Include dependency graph for gomotion.cpp:



Namespaces

- [gomotion](#)

Macros

- `#define maxit(a, m) {if ((a) > (m)) (m) = (a);}`

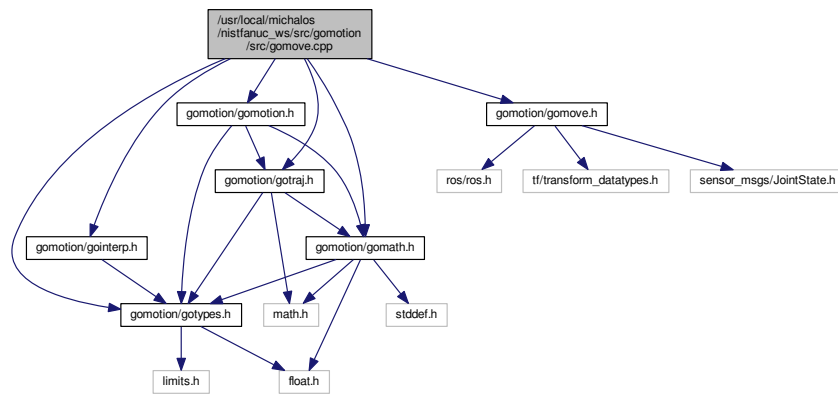
8.9.1 Macro Definition Documentation

8.9.1.1 `#define maxit(a, m) {if ((a) > (m)) (m) = (a);}`

8.10 /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gomove.cpp File Reference

```
#include "gomotion/gotypes.h"
#include "gomotion/gointerp.h"
#include "gomotion/gotraj.h"
#include "gomotion/gomotion.h"
#include "gomotion/gomove.h"
#include "gomotion/gomath.h"
```

Include dependency graph for gomove.cpp:



Data Structures

- struct [gomotion::go_motion_interface](#)

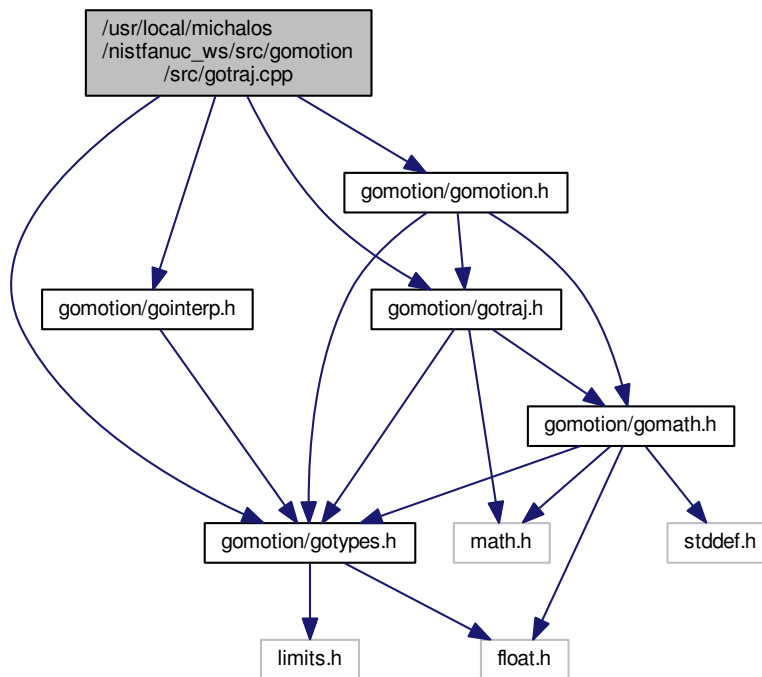
Namespaces

- [gomotion](#)

8.11 /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gotraj.cpp File Reference

```
#include "gomotion/gotypes.h"
#include "gomotion/gointerp.h"
#include "gomotion/gotraj.h"
#include "gomotion/gomotion.h"
```

Include dependency graph for gotraj.cpp:



Functions

- `int go_init ()`
- `int go_exit ()`

Variables

- `int go_result_int = 1`
- `int go_real_double = 1`
- `int go_integer_int = 1`
- `int go_flag_uchar = 1`
- `go_flag gocode = 0`

8.11.1 Function Documentation

8.11.1.1 `int go_exit ()`

8.11.1.2 `int go_init ()`

8.11.2 Variable Documentation

8.11.2.1 `int go_flag_uchar = 1`

8.11.2.2 `int go_integer_int = 1`

8.11.2.3 `int go_real_double = 1`

8.11.2.4 `int go_result_int = 1`

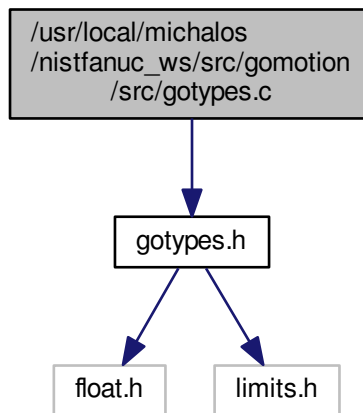
8.11.2.5 `go_flag gocode = 0`

An ad-hoc flag set by various functions that can be used to indicate various code paths taken.

8.12 /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gotypes.c File Reference

```
#include "gotypes.h"
```

Include dependency graph for gotypes.c:



Variables

- `int go_result_int = 1`
- `int go_real_double = 1`
- `int go_integer_int = 1`
- `int go_flag_uchar = 1`
- `go_flag gocode = 0`

8.12.1 Variable Documentation

8.12.1.1 `int go_flag_uchar = 1`

8.12.1.2 `int go_integer_int = 1`

8.12.1.3 `int go_real_double = 1`

8.12.1.4 `int go_result_int = 1`

8.12.1.5 `go_flag gocode = 0`

An ad-hoc flag set by various functions that can be used to indicate various code paths taken.

Index

/usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/gointerp.-
 h, 85
 /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/goqueue.-
 h, 86
 /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/goqueue.h
 h, 95
 /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/goqueue.h
 h, 97
 /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/goqueue.h
 h, 99
 /usr/local/michalos/nistfanuc_ws/src/gomotion/include/gomotion/goqueue.h
 h, 101
 /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gointerp.-
 cpp, 105
 /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gomath.-
 cpp, 106
 /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gomotion.-
 cpp, 110
 /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gomove.-
 cpp, 112
 /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gotraj.-
 cpp, 112
 /usr/local/michalos/nistfanuc_ws/src/gomotion/src/gotypes.-
 c, 114
 _deltat
 gomotion::go_motion_interface, 53
 gomotion::go_motion_queue, 58
 _end
 gomotion::go_motion_queue, 58
 _endptr
 gomotion::go_motion_queue, 58
 _gmq
 gomotion::go_motion_interface, 53
 _gms
 gomotion::go_motion_interface, 53
 _here
 gomotion::go_motion_queue, 59
 _id
 gomotion::go_motion_interface, 53
 _joint_num
 gomotion::go_motion_queue, 59
 _last_id
 gomotion::go_motion_queue, 59
 _number
 gomotion::go_motion_queue, 59
 a
 gomotion::coeff, 39
 gomotion::go_cubic, 41
 gomotion::go_dh, 42
 gomotion::go_interp, 46
 gomotion::go_quadratic, 68
 gomotion::go_quartic, 68
 gomotion::go_traj_interp_spec, 78
 acc
 gomotion::go_motion_params, 55
 gomotion::GoMotionParams, 83
 add_constant
 gomotion::go_interp, 45
 add_cubic_pdv
 gomotion::go_interp, 45
 add_cubic_pf
 gomotion::go_interp, 45
 add_cubic_pv
 gomotion::go_interp, 45
 add_func
 gomotion::go_interp, 45
 add_linear
 gomotion::go_interp, 45
 add_quintic_pdva

- gomotion::go_interp, 45
- add_quintic_pf
 - gomotion::go_interp, 45
- add_quintic_pva
 - gomotion::go_interp, 45
- add_quintic_pvda
 - gomotion::go_interp, 45
- alpha
 - gomotion::go_dh, 42
- append
 - gomotion::go_motion_queue, 57
- append_joint
 - gomotion::go_motion_queue, 57
- append_ujoint
 - gomotion::go_motion_queue, 57
- append_world
 - gomotion::go_motion_queue, 57
- at0
 - gomotion::go_traj_ca_spec, 74
- at1
 - gomotion::go_traj_cj_spec, 76
- b
 - gomotion::go_cubic, 41
 - gomotion::go_quadratic, 68
 - gomotion::go_quartic, 68
- base
 - gomotion::go_pk, 63
- body
 - gomotion::go_link, 48
- c
 - gomotion::go_cubic, 41
 - gomotion::go_quartic, 68
- COEFF_MAX
 - gomotion, 21
- CBRT_IS_POW
 - gomath.cpp, 110
- COL_IS_UNIT
 - gomath.cpp, 110
- calc_coeff_bc
 - gomotion::go_interp, 45
- calc_coeff_constant
 - gomotion::go_interp, 45
- calc_coeff_cubic_bc
 - gomotion::go_interp, 45
- calc_coeff_cubic_pf
 - gomotion::go_interp, 46
- calc_coeff_linear
 - gomotion::go_interp, 46
- calc_coeff_pf
 - gomotion::go_interp, 46
- calc_coeff_quintic_bc
 - gomotion::go_interp, 46
- calc_coeff_quintic_pf
 - gomotion::go_interp, 46
- center
 - gomotion::go_motion_circular_params, 52
- cj
 - gomotion::go_motion_spec, 61
- cols
 - gomotion::go_matrix, 50
- cpar
 - gomotion::go_motion_spec, 61
- d
 - gomotion::go_dh, 42
 - gomotion::go_pk, 63
 - gomotion::go_plane, 64
 - gomotion::go_quartic, 68
 - gomotion::go_traj_interp_spec, 78
- dh
 - gomotion::go_link, 48
- direction
 - gomotion::go_line, 47
- drop_pending
 - gomotion::go_motion_queue, 57
- dt1
 - gomotion::go_traj_ca_spec, 74
 - gomotion::go_traj_cj_spec, 76
- dt2
 - gomotion::go_traj_ca_spec, 74
 - gomotion::go_traj_cj_spec, 76
- dt3
 - gomotion::go_traj_cj_spec, 76
- dt4
 - gomotion::go_traj_cj_spec, 76
- dt5
 - gomotion::go_traj_cj_spec, 76
- dt6
 - gomotion::go_traj_cj_spec, 76
- dtend
 - gomotion::go_traj_ca_spec, 74
 - gomotion::go_traj_cj_spec, 76
- el
 - gomotion::go_matrix, 50
- elcpy
 - gomotion::go_matrix, 50
- end
 - gomotion::go_motion_spec, 61
- erase
 - gomotion::go_motion_queue, 57
- eval
 - gomotion::go_scale_spec, 72
- eval_constant
 - gomotion::go_interp, 46
- eval_cubic
 - gomotion::go_interp, 46
- eval_func

- gomotion::go_interp, 45
- eval_linear
 - gomotion::go_interp, 46
- eval_quintic
 - gomotion::go_interp, 46
- GO_LINK_DH
 - gomotion, 21
- GO_LINK_PK
 - gomotion, 21
- GO_LINK_PP
 - gomotion, 21
- GO_MOTION_CIRCULAR
 - gomotion, 22
- GO_MOTION_JOINT
 - gomotion, 22
- GO_MOTION_LINEAR
 - gomotion, 22
- GO_MOTION_NONE
 - gomotion, 22
- GO_MOTION_UJOINT
 - gomotion, 22
- GO_MOTION_WORLD
 - gomotion, 22
- GO_QUANTITY_ANGLE
 - gotypes.h, 104
- GO_QUANTITY_LENGTH
 - gotypes.h, 104
- GO_QUANTITY_NONE
 - gotypes.h, 104
- GO_RESULT_BAD_ARGS
 - gotypes.h, 104
- GO_RESULT_BUG
 - gotypes.h, 104
- GO_RESULT_DIV_ERROR
 - gotypes.h, 104
- GO_RESULT_DOMAIN_ERROR
 - gotypes.h, 104
- GO_RESULT_EMPTY
 - gotypes.h, 104
- GO_RESULT_ERROR
 - gotypes.h, 104
- GO_RESULT_IGNORED
 - gotypes.h, 104
- GO_RESULT_IMPL_ERROR
 - gotypes.h, 104
- GO_RESULT_NO_SPACE
 - gotypes.h, 104
- GO_RESULT_NORM_ERROR
 - gotypes.h, 104
- GO_RESULT_OK
 - gotypes.h, 104
- GO_RESULT_RANGE_ERROR
 - gotypes.h, 104
- GO_RESULT_SINGULAR
 - gotypes.h, 104
- GO_2_PI
 - gomath.h, 93
- GO_CLOSE
 - gomath.h, 93
- GO_FLAG
 - gotypes.h, 103
- GO_FLAG_UCHAR
 - gotypes.h, 103
- GO_INF
 - gotypes.h, 103
- GO_INTEGER
 - gotypes.h, 103
- GO_INTEGER_INT
 - gotypes.h, 103
- GO_INTEGER_MAX
 - gotypes.h, 103
- GO_MATRIX_DECLARE
 - gomath.h, 93
- GO_PI
 - gomath.h, 94
- GO_PI_2
 - gomath.h, 94
- GO_PI_4
 - gomath.h, 94
- GO_REAL
 - gotypes.h, 103
- GO_REAL_DOUBLE
 - gotypes.h, 103
- GO_REAL_EPSILON
 - gotypes.h, 103
- GO_REAL_MAX
 - gotypes.h, 103
- GO_REAL_MIN
 - gotypes.h, 103
- GO_RESULT
 - gotypes.h, 103
- GO_RESULT_INT
 - gotypes.h, 103
- GO_ROT_CLOSE
 - gomath.h, 94
- GO_ROT_SMALL
 - gomath.h, 94
- GO_SMALL
 - gomath.h, 94
- GO_TO_DEG
 - gomath.h, 95
- GO_TO_RAD
 - gomath.h, 95
- GO_TRAN_CLOSE
 - gomath.h, 95
- GO_TRAN_SMALL
 - gomath.h, 95

- get_id
 - gomotion::go_motion_spec, [61](#)
- get_joint_number
 - gomotion::go_motion_queue, [57](#)
- get_type
 - gomotion::go_motion_queue, [57](#)
 - gomotion::go_motion_spec, [61](#)
- go_acoses
 - gomotion, [22](#)
- go_asines
 - gomotion, [22](#)
- go_atans
 - gomotion, [22](#)
- go_body_copy
 - gomath.h, [93](#)
- go_body_init
 - gomath.h, [93](#)
- go_cart_cart_add
 - gomotion, [22](#)
- go_cart_cart_angle
 - gomotion, [22](#)
- go_cart_cart_compare
 - gomotion, [22](#)
- go_cart_cart_cross
 - gomotion, [22](#)
- go_cart_cart_disp
 - gomotion, [22](#)
- go_cart_cart_dot
 - gomotion, [22](#)
- go_cart_cart_par
 - gomotion, [22](#)
- go_cart_cart_perp
 - gomotion, [22](#)
- go_cart_cart_pose
 - gomotion, [22](#)
- go_cart_cart_proj
 - gomotion, [23](#)
- go_cart_cart_rot
 - gomotion, [23](#)
- go_cart_cart_sub
 - gomotion, [23](#)
- go_cart_centroid
 - gomotion, [23](#)
- go_cart_centroidize
 - gomotion, [23](#)
- go_cart_cyl_convert
 - gomotion, [23](#)
- go_cart_is_norm
 - gomotion, [23](#)
- go_cart_mag
 - gomotion, [23](#)
- go_cart_magsq
 - gomotion, [23](#)
- go_cart_neg
 - gomotion, [23](#)
- go_cart_normal
 - gomotion, [23](#)
- go_cart_plane_proj
 - gomotion, [23](#)
- go_cart_rvec_convert
 - gomotion, [23](#)
- go_cart_scale_mult
 - gomotion, [23](#)
- go_cart_sph_convert
 - gomotion, [23](#)
- go_cart_trilaterate
 - gomotion, [23](#)
- go_cart_unit
 - gomotion, [24](#)
- go_cart_vector_convert
 - gomotion, [24](#)
- go_cart_zero
 - gomotion, [24](#)
- go_cbrt
 - gomotion, [24](#)
- go_complex_add
 - gomotion, [24](#)
- go_complex_arg
 - gomotion, [24](#)
- go_complex_cbrt
 - gomotion, [24](#)
- go_complex_div
 - gomotion, [24](#)
- go_complex_inv
 - gomotion, [24](#)
- go_complex_mag
 - gomotion, [24](#)
- go_complex_magsq
 - gomotion, [24](#)
- go_complex_mult
 - gomotion, [24](#)
- go_complex_scale
 - gomotion, [24](#)
- go_complex_sq
 - gomotion, [24](#)
- go_complex_sqrt
 - gomotion, [24](#)
- go_complex_sub
 - gomotion, [24](#)
- go_cub
 - gomath.h, [93](#)
- go_cubic_solve
 - gomotion, [24](#)
- go_cyl_cart_convert
 - gomotion, [24](#)
- go_cyl_sph_convert
 - gomotion, [24](#)
- go_dh_hom_convert

- gomotion, 24
- go_dh_pose_convert
 - gomotion, 25
- go_exit
 - gotraj.cpp, 113
- go_flag
 - gotypes.h, 104
- go_flag_uchar
 - gotraj.cpp, 113
 - gotypes.c, 114
 - gotypes.h, 105
- go_get_singular_epsilon
 - gomotion, 25
- go_hom_dh_convert
 - gomotion, 25
- go_hom_hom_mult
 - gomotion, 25
- go_hom_inv
 - gomotion, 25
- go_hom_pose_convert
 - gomotion, 25
- go_init
 - gotraj.cpp, 113
- go_integer
 - gotypes.h, 104
- go_integer_int
 - gotraj.cpp, 114
 - gotypes.c, 114
 - gotypes.h, 105
- go_line_evaluate
 - gomotion, 25
- go_line_from_planes
 - gomotion, 25
- go_line_from_point_direction
 - gomotion, 25
- go_line_from_points
 - gomotion, 25
- go_line_line_compare
 - gomotion, 25
- go_line_plane_intersect
 - gomotion, 25
- go_line_plane_proj
 - gomotion, 25
- go_linear_cos_sin_solve
 - gomotion, 26
- go_link_joint_set
 - gomotion, 26
- go_link_pose_build
 - gomotion, 26
- go_link_to_string
 - gomath.h, 93
- go_mat3_inv
 - gomotion, 26
- go_mat3_mat3_mult
 - gomotion, 26
- go_mat3_vec3_mult
 - gomotion, 26
- go_mat4_inv
 - gomotion, 26
- go_mat4_mat4_mult
 - gomotion, 26
- go_mat4_vec4_mult
 - gomotion, 26
- go_mat6_inv
 - gomotion, 26
- go_mat6_mat6_mult
 - gomotion, 26
- go_mat6_transpose
 - gomotion, 26
- go_mat6_vec6_mult
 - gomotion, 26
- go_mat_cart_mult
 - gomotion, 26
- go_mat_inv
 - gomotion, 27
- go_mat_is_norm
 - gomotion, 27
- go_mat_mat_mult
 - gomotion, 27
- go_mat_matrix_convert
 - gomotion, 27
- go_mat_norm
 - gomotion, 27
- go_mat_quat_convert
 - gomotion, 27
- go_mat_rpy_convert
 - gomotion, 27
- go_mat_rvec_convert
 - gomotion, 27
- go_mat_uxz_convert
 - gomotion, 27
- go_mat_xyz_convert
 - gomotion, 27
- go_mat_zyx_convert
 - gomotion, 27
- go_mat_zyz_convert
 - gomotion, 27
- go_matrix_init
 - gomath.h, 94
- go_matrix_inv
 - gomotion, 27
- go_matrix_matrix_add
 - gomotion, 27
- go_matrix_matrix_copy
 - gomotion, 27
- go_matrix_matrix_mult
 - gomotion, 27
- go_matrix_transpose

- gomotion, [27](#)
- go_matrix_vector_cross
 - gomotion, [27](#)
- go_matrix_vector_mult
 - gomotion, [27](#)
- go_plane_evaluate
 - gomotion, [27](#)
- go_plane_from_abcd
 - gomotion, [27](#)
- go_plane_from_point_line
 - gomotion, [28](#)
- go_plane_from_point_normal
 - gomotion, [28](#)
- go_plane_from_points
 - gomotion, [28](#)
- go_plane_plane_compare
 - gomotion, [28](#)
- go_point_line_distance
 - gomotion, [28](#)
- go_point_line_proj
 - gomotion, [28](#)
- go_point_plane_distance
 - gomotion, [28](#)
- go_point_plane_proj
 - gomotion, [28](#)
- go_pose_cart_mult
 - gomotion, [28](#)
- go_pose_dh_convert
 - gomotion, [28](#)
- go_pose_hom_convert
 - gomotion, [29](#)
- go_pose_identity
 - gomotion, [29](#)
- go_pose_inv
 - gomotion, [29](#)
- go_pose_pose_compare
 - gomotion, [29](#)
- go_pose_pose_interp
 - gomotion, [29](#)
- go_pose_pose_mult
 - gomotion, [29](#)
- go_pose_scale_mult
 - gomotion, [29](#)
- go_pose_this
 - gomotion, [29](#)
- go_pose_vel_mult
 - gomotion, [29](#)
- go_qua
 - gomath.h, [94](#)
- go_quadratic_solve
 - gomotion, [29](#)
- go_quantity_to_string
 - gotypes.h, [103](#)
- go_quartic_solve
 - gomotion, [29](#)
- go_quat_cart_mult
 - gomotion, [29](#)
- go_quat_identity
 - gomotion, [29](#)
- go_quat_inv
 - gomotion, [29](#)
- go_quat_is_norm
 - gomotion, [29](#)
- go_quat_mag
 - gomotion, [29](#)
- go_quat_mat_convert
 - gomotion, [30](#)
- go_quat_matrix_convert
 - gomotion, [30](#)
- go_quat_norm
 - gomotion, [30](#)
- go_quat_quat_compare
 - gomotion, [30](#)
- go_quat_quat_mult
 - gomotion, [30](#)
- go_quat_rpy_convert
 - gomotion, [30](#)
- go_quat_rvec_convert
 - gomotion, [30](#)
- go_quat_scale_mult
 - gomotion, [30](#)
- go_quat_unit
 - gomotion, [30](#)
- go_quat_xyz_convert
 - gomotion, [30](#)
- go_quat_zyx_convert
 - gomotion, [30](#)
- go_quat_zyz_convert
 - gomotion, [30](#)
- go_real
 - gotypes.h, [104](#)
- go_real_double
 - gotraj.cpp, [114](#)
 - gotypes.c, [115](#)
 - gotypes.h, [105](#)
- go_result
 - gotypes.h, [104](#)
- go_result_int
 - gotraj.cpp, [114](#)
 - gotypes.c, [115](#)
 - gotypes.h, [105](#)
- go_result_to_string
 - gotypes.h, [103](#)
- go_rpy_cart_mult
 - gomotion, [30](#)
- go_rpy_mat_convert
 - gomotion, [30](#)
- go_rpy_quat_convert

gomotion, [30](#)
 go_rpy_rvec_convert
 gomotion, [30](#)
 go_rpy_xyz_convert
 gomotion, [30](#)
 go_rpy_zyx_convert
 gomotion, [30](#)
 go_rpy_zyz_convert
 gomotion, [30](#)
 go_rvec_cart_convert
 gomotion, [30](#)
 go_rvec_mat_convert
 gomotion, [30](#)
 go_rvec_quat_convert
 gomotion, [30](#)
 go_rvec_rpy_convert
 gomotion, [30](#)
 go_rvec_rvec_compare
 gomotion, [30](#)
 go_rvec_scale_mult
 gomotion, [30](#)
 go_rvec_xyz_convert
 gomotion, [31](#)
 go_rvec_zyx_convert
 gomotion, [31](#)
 go_rvec_zyz_convert
 gomotion, [31](#)
 go_set_singular_epsilon
 gomotion, [31](#)
 go_sincos
 gomotion, [31](#)
 go_sph_cart_convert
 gomotion, [31](#)
 go_sph_cyl_convert
 gomotion, [31](#)
 go_sq
 gomath.h, [95](#)
 go_traj_ca_compute
 gomotion, [31](#)
 go_traj_ca_extend
 gomotion, [31](#)
 go_traj_ca_generate
 gomotion, [31](#)
 go_traj_ca_interp
 gomotion, [31](#)
 go_traj_ca_scale
 gomotion, [32](#)
 go_traj_ca_stop
 gomotion, [32](#)
 go_traj_cj_compute
 gomotion, [32](#)
 go_traj_cj_compute_tees_a_c
 gomotion, [33](#)
 go_traj_cj_compute_tees_a_nc

gomotion, [33](#)
 go_traj_cj_compute_tees_na_c
 gomotion, [33](#)
 go_traj_cj_compute_tees_na_nc
 gomotion, [34](#)
 go_traj_cj_compute_the_rest
 gomotion, [34](#)
 go_traj_cj_extend
 gomotion, [34](#)
 go_traj_cj_generate
 gomotion, [34](#)
 go_traj_cj_interp
 gomotion, [34](#)
 go_traj_cj_scale
 gomotion, [34](#)
 go_traj_cj_stop
 gomotion, [35](#)
 go_tridiag_ql
 gomotion, [36](#)
 go_tridiag_reduce
 gomotion, [37](#)
 go_uxz_mat_convert
 gomotion, [37](#)
 go_vector
 gomotion, [21](#)
 go_vector_cart_convert
 gomotion, [37](#)
 go_xyz_mat_convert
 gomotion, [37](#)
 go_xyz_quat_convert
 gomotion, [37](#)
 go_xyz_rpy_convert
 gomotion, [37](#)
 go_xyz_rvec_convert
 gomotion, [37](#)
 go_xyz_zyx_convert
 gomotion, [37](#)
 go_xyz_zyz_convert
 gomotion, [37](#)
 go_zyx_mat_convert
 gomotion, [37](#)
 go_zyx_quat_convert
 gomotion, [37](#)
 go_zyx_rpy_convert
 gomotion, [37](#)
 go_zyx_rvec_convert
 gomotion, [37](#)
 go_zyx_xyz_convert
 gomotion, [37](#)
 go_zyx_zyz_convert
 gomotion, [37](#)
 go_zyz_mat_convert
 gomotion, [37](#)
 go_zyz_quat_convert

- gomotion, [37](#)
- go_yyz_rpy_convert
 - gomotion, [37](#)
- go_yyz_rvec_convert
 - gomotion, [37](#)
- go_yyz_xyz_convert
 - gomotion, [37](#)
- go_yyz_zyx_convert
 - gomotion, [37](#)
- gocode
 - gotraj.cpp, [114](#)
 - gotypes.c, [115](#)
 - gotypes.h, [105](#)
- gomath.cpp
 - CBRT_IS_POW, [110](#)
 - COL_IS_UNIT, [110](#)
 - MAG2, [110](#)
 - SIGN, [110](#)
- gomath.h
 - GO_2_PI, [93](#)
 - GO_CLOSE, [93](#)
 - GO_MATRIX_DECLARE, [93](#)
 - GO_PI, [94](#)
 - GO_PI_2, [94](#)
 - GO_PI_4, [94](#)
 - GO_ROT_CLOSE, [94](#)
 - GO_ROT_SMALL, [94](#)
 - GO_SMALL, [94](#)
 - GO_TO_DEG, [95](#)
 - GO_TO_RAD, [95](#)
 - GO_TRAN_CLOSE, [95](#)
 - GO_TRAN_SMALL, [95](#)
 - go_body_copy, [93](#)
 - go_body_init, [93](#)
 - go_cub, [93](#)
 - go_link_to_string, [93](#)
 - go_matrix_init, [94](#)
 - go_qua, [94](#)
 - go_sq, [95](#)
 - ISIZEOF, [95](#)
- gomotion, [15](#)
 - COEFF_MAX, [21](#)
 - GO_LINK_DH, [21](#)
 - GO_LINK_PK, [21](#)
 - GO_LINK_PP, [21](#)
 - GO_MOTION_CIRCULAR, [22](#)
 - GO_MOTION_JOINT, [22](#)
 - GO_MOTION_LINEAR, [22](#)
 - GO_MOTION_NONE, [22](#)
 - GO_MOTION_UJOINT, [22](#)
 - GO_MOTION_WORLD, [22](#)
 - go_acoses, [22](#)
 - go_asines, [22](#)
 - go_atans, [22](#)
 - go_cart_cart_add, [22](#)
 - go_cart_cart_angle, [22](#)
 - go_cart_cart_compare, [22](#)
 - go_cart_cart_cross, [22](#)
 - go_cart_cart_disp, [22](#)
 - go_cart_cart_dot, [22](#)
 - go_cart_cart_par, [22](#)
 - go_cart_cart_perp, [22](#)
 - go_cart_cart_pose, [22](#)
 - go_cart_cart_proj, [23](#)
 - go_cart_cart_rot, [23](#)
 - go_cart_cart_sub, [23](#)
 - go_cart_centroid, [23](#)
 - go_cart_centroidize, [23](#)
 - go_cart_cyl_convert, [23](#)
 - go_cart_is_norm, [23](#)
 - go_cart_mag, [23](#)
 - go_cart_magsq, [23](#)
 - go_cart_neg, [23](#)
 - go_cart_normal, [23](#)
 - go_cart_plane_proj, [23](#)
 - go_cart_rvec_convert, [23](#)
 - go_cart_scale_mult, [23](#)
 - go_cart_sph_convert, [23](#)
 - go_cart_trilaterate, [23](#)
 - go_cart_unit, [24](#)
 - go_cart_vector_convert, [24](#)
 - go_cart_zero, [24](#)
 - go_cbrt, [24](#)
 - go_complex_add, [24](#)
 - go_complex_arg, [24](#)
 - go_complex_cbrt, [24](#)
 - go_complex_div, [24](#)
 - go_complex_inv, [24](#)
 - go_complex_mag, [24](#)
 - go_complex_magsq, [24](#)
 - go_complex_mult, [24](#)
 - go_complex_scale, [24](#)
 - go_complex_sq, [24](#)
 - go_complex_sqrt, [24](#)
 - go_complex_sub, [24](#)
 - go_cubic_solve, [24](#)
 - go_cyl_cart_convert, [24](#)
 - go_cyl_sph_convert, [24](#)
 - go_dh_hom_convert, [24](#)
 - go_dh_pose_convert, [25](#)
 - go_get_singular_epsilon, [25](#)
 - go_hom_dh_convert, [25](#)
 - go_hom_hom_mult, [25](#)
 - go_hom_inv, [25](#)
 - go_hom_pose_convert, [25](#)
 - go_line_evaluate, [25](#)
 - go_line_from_planes, [25](#)
 - go_line_from_point_direction, [25](#)

go_line_from_points, 25
go_line_line_compare, 25
go_line_plane_intersect, 25
go_line_plane_proj, 25
go_linear_cos_sin_solve, 26
go_link_joint_set, 26
go_link_pose_build, 26
go_mat3_inv, 26
go_mat3_mat3_mult, 26
go_mat3_vec3_mult, 26
go_mat4_inv, 26
go_mat4_mat4_mult, 26
go_mat4_vec4_mult, 26
go_mat6_inv, 26
go_mat6_mat6_mult, 26
go_mat6_transpose, 26
go_mat6_vec6_mult, 26
go_mat_cart_mult, 26
go_mat_inv, 27
go_mat_is_norm, 27
go_mat_mat_mult, 27
go_mat_matrix_convert, 27
go_mat_norm, 27
go_mat_quat_convert, 27
go_mat_rpy_convert, 27
go_mat_rvec_convert, 27
go_mat_uxz_convert, 27
go_mat_xyz_convert, 27
go_mat_zyx_convert, 27
go_mat_zyz_convert, 27
go_matrix_inv, 27
go_matrix_matrix_add, 27
go_matrix_matrix_copy, 27
go_matrix_matrix_mult, 27
go_matrix_transpose, 27
go_matrix_vector_cross, 27
go_matrix_vector_mult, 27
go_plane_evaluate, 27
go_plane_from_abcd, 27
go_plane_from_point_line, 28
go_plane_from_point_normal, 28
go_plane_from_points, 28
go_plane_plane_compare, 28
go_point_line_distance, 28
go_point_line_proj, 28
go_point_plane_distance, 28
go_point_plane_proj, 28
go_pose_cart_mult, 28
go_pose_dh_convert, 28
go_pose_hom_convert, 29
go_pose_identity, 29
go_pose_inv, 29
go_pose_pose_compare, 29
go_pose_pose_interp, 29
go_pose_pose_mult, 29
go_pose_scale_mult, 29
go_pose_this, 29
go_pose_vel_mult, 29
go_quadratic_solve, 29
go_quartic_solve, 29
go_quat_cart_mult, 29
go_quat_identity, 29
go_quat_inv, 29
go_quat_is_norm, 29
go_quat_mag, 29
go_quat_mat_convert, 30
go_quat_matrix_convert, 30
go_quat_norm, 30
go_quat_quat_compare, 30
go_quat_quat_mult, 30
go_quat_rpy_convert, 30
go_quat_rvec_convert, 30
go_quat_scale_mult, 30
go_quat_unit, 30
go_quat_xyz_convert, 30
go_quat_zyx_convert, 30
go_quat_zyz_convert, 30
go_rpy_cart_mult, 30
go_rpy_mat_convert, 30
go_rpy_quat_convert, 30
go_rpy_rvec_convert, 30
go_rpy_xyz_convert, 30
go_rpy_zyx_convert, 30
go_rpy_zyz_convert, 30
go_rvec_cart_convert, 30
go_rvec_mat_convert, 30
go_rvec_quat_convert, 30
go_rvec_rpy_convert, 30
go_rvec_rvec_compare, 30
go_rvec_scale_mult, 30
go_rvec_xyz_convert, 31
go_rvec_zyx_convert, 31
go_rvec_zyz_convert, 31
go_set_singular_epsilon, 31
go_sincos, 31
go_sph_cart_convert, 31
go_sph_cyl_convert, 31
go_traj_ca_compute, 31
go_traj_ca_extend, 31
go_traj_ca_generate, 31
go_traj_ca_interp, 31
go_traj_ca_scale, 32
go_traj_ca_stop, 32
go_traj_cj_compute, 32
go_traj_cj_compute_teas_a_c, 33
go_traj_cj_compute_teas_a_nc, 33
go_traj_cj_compute_teas_na_c, 33
go_traj_cj_compute_teas_na_nc, 34

- go_traj_cj_compute_the_rest, 34
- go_traj_cj_extend, 34
- go_traj_cj_generate, 34
- go_traj_cj_interp, 34
- go_traj_cj_scale, 34
- go_traj_cj_stop, 35
- go_tridiag_q1, 36
- go_tridiag_reduce, 37
- go_uxz_mat_convert, 37
- go_vector, 21
- go_vector_cart_convert, 37
- go_xyz_mat_convert, 37
- go_xyz_quat_convert, 37
- go_xyz_rpy_convert, 37
- go_xyz_rvec_convert, 37
- go_xyz_zyx_convert, 37
- go_xyz_zyz_convert, 37
- go_zyx_mat_convert, 37
- go_zyx_quat_convert, 37
- go_zyx_rpy_convert, 37
- go_zyx_rvec_convert, 37
- go_zyx_xyz_convert, 37
- go_zyx_zyz_convert, 37
- go_zyz_mat_convert, 37
- go_zyz_quat_convert, 37
- go_zyz_rpy_convert, 37
- go_zyz_rvec_convert, 37
- go_zyz_xyz_convert, 37
- go_zyz_zyx_convert, 37
- JointState, 21
- lubksb, 37
- ludcmp, 38
- sincos, 38
- gomotion.cpp
 - maxit, 111
- gomotion::GoMotion, 82
 - pgm, 82
- gomotion::GoMotionParams, 82
 - acc, 83
 - jerk, 83
 - vel, 83
- gomotion::coeff, 39
 - a, 39
- gomotion::go_body, 39
 - inertia, 40
 - mass, 40
- gomotion::go_cart, 40
 - x, 40
 - y, 40
 - z, 40
- gomotion::go_complex, 40
 - im, 41
 - re, 41
- gomotion::go_cubic, 41
 - a, 41
 - b, 41
 - c, 41
- gomotion::go_cyl, 41
 - r, 42
 - theta, 42
 - z, 42
- gomotion::go_dh, 42
 - a, 42
 - alpha, 42
 - d, 42
 - theta, 42
- gomotion::go_hom, 43
 - rot, 43
 - tran, 43
- gomotion::go_interp, 44
 - a, 46
 - add_constant, 45
 - add_cubic_pdv, 45
 - add_cubic_pf, 45
 - add_cubic_pv, 45
 - add_func, 45
 - add_linear, 45
 - add_quintic_pdva, 45
 - add_quintic_pf, 45
 - add_quintic_pva, 45
 - add_quintic_pvda, 45
 - calc_coeff_bc, 45
 - calc_coeff_constant, 45
 - calc_coeff_cubic_bc, 45
 - calc_coeff_cubic_pf, 46
 - calc_coeff_linear, 46
 - calc_coeff_pf, 46
 - calc_coeff_quintic_bc, 46
 - calc_coeff_quintic_pf, 46
 - eval_constant, 46
 - eval_cubic, 46
 - eval_func, 45
 - eval_linear, 46
 - eval_quintic, 46
 - init, 46
 - p, 46
 - set_here, 46
- gomotion::go_line, 46
 - direction, 47
 - point, 47
- gomotion::go_link, 47
 - body, 48
 - dh, 48
 - pk, 48
 - pp, 49
 - quantity, 49
 - type, 49
 - u, 49

- gomotion::go_mat, 49
 - x, 50
 - y, 50
 - z, 50
- gomotion::go_matrix, 50
 - cols, 50
 - el, 50
 - elcpy, 50
 - index, 50
 - rows, 51
 - v, 51
- gomotion::go_motion_circular_params, 51
 - center, 52
 - normal, 52
 - qnormal, 52
 - rstart, 52
 - rtot, 52
 - stotinv, 52
 - thtot, 52
 - turns, 52
 - urcsp, 52
 - zstart, 52
 - ztot, 52
- gomotion::go_motion_interface, 53
 - _deltat, 53
 - _gmq, 53
 - _gms, 53
 - _id, 53
 - _position, 53
 - _queuesize, 53
 - _space, 54
 - _type, 54
- gomotion::go_motion_linear_params, 54
 - uquat, 54
 - uvec, 54
- gomotion::go_motion_params, 55
 - acc, 55
 - jerk, 55
 - vel, 55
- gomotion::go_motion_queue, 55
 - _deltat, 58
 - _end, 58
 - _endptr, 58
 - _here, 59
 - _joint_num, 59
 - _last_id, 59
 - _number, 59
 - _size, 59
 - _start, 59
 - _startptr, 59
 - _there, 59
 - _time, 59
 - _timescale, 59
 - _type, 59
- append, 57
- append_joint, 57
- append_ujoint, 57
- append_world, 57
- drop_pending, 57
- erase, 57
- get_joint_number, 57
- get_type, 57
- head, 57
- here, 58
- init, 58
- interp, 58
- interp_joint, 58
- interp_world, 58
- is_empty, 58
- last_id, 58
- number, 58
- reset, 58
- set_cycle_time, 58
- set_here, 58
- set_id, 58
- set_joint_number, 58
- set_scale, 58
- set_type, 58
- size, 58
- stop, 58
- there, 58
- gomotion::go_motion_spec, 60
 - cj, 61
 - cpar, 61
 - end, 61
 - get_id, 61
 - get_type, 61
 - id, 62
 - init, 61
 - lpar, 62
 - par, 62
 - set_cpar, 61
 - set_end_pose, 61
 - set_end_position, 61
 - set_id, 61
 - set_jpar, 61
 - set_rpar, 61
 - set_time, 61
 - set_tpar, 61
 - set_type, 61
 - start, 62
 - totalt, 62
 - type, 62
 - u, 62
 - uquat, 62
- gomotion::go_pk, 62
 - base, 63
 - d, 63

- platform, [63](#)
- gomotion::go_plane, [64](#)
 - d, [64](#)
 - normal, [64](#)
- gomotion::go_pose, [64](#)
 - rot, [65](#)
 - tran, [65](#)
- gomotion::go_position, [65](#)
 - joint, [66](#)
 - pose, [67](#)
 - u, [67](#)
 - zero_joints, [66](#)
 - zero_pose, [66](#)
- gomotion::go_pp, [67](#)
 - pose, [67](#)
- gomotion::go_quadratic, [68](#)
 - a, [68](#)
 - b, [68](#)
- gomotion::go_quartic, [68](#)
 - a, [68](#)
 - b, [68](#)
 - c, [68](#)
 - d, [68](#)
- gomotion::go_quat, [69](#)
 - s, [69](#)
 - x, [69](#)
 - y, [69](#)
 - z, [69](#)
- gomotion::go_rpy, [69](#)
 - p, [70](#)
 - r, [70](#)
 - y, [70](#)
- gomotion::go_rvec, [70](#)
 - x, [70](#)
 - y, [70](#)
 - z, [70](#)
- gomotion::go_scale_spec, [71](#)
 - eval, [72](#)
 - init, [72](#)
 - scale, [72](#)
 - scale_a_next, [72](#)
 - scale_b, [72](#)
 - scale_dir, [72](#)
 - scale_isneg, [72](#)
 - scale_next, [72](#)
 - scale_spec, [72](#)
 - scale_t, [72](#)
 - scale_v_next, [72](#)
 - scaling, [72](#)
 - set, [72](#)
- gomotion::go_sph, [73](#)
 - phi, [73](#)
 - r, [73](#)
 - theta, [73](#)
- gomotion::go_traj_ca_spec, [73](#)
 - at0, [74](#)
 - dt1, [74](#)
 - dt2, [74](#)
 - dtend, [74](#)
 - invd, [74](#)
 - t1, [74](#)
 - t2, [74](#)
 - tend, [74](#)
 - vt1, [75](#)
- gomotion::go_traj_cj_spec, [75](#)
 - at1, [76](#)
 - dt1, [76](#)
 - dt2, [76](#)
 - dt3, [76](#)
 - dt4, [76](#)
 - dt5, [76](#)
 - dt6, [76](#)
 - dtend, [76](#)
 - invd, [76](#)
 - jt0, [77](#)
 - t1, [77](#)
 - t2, [77](#)
 - t3, [77](#)
 - t4, [77](#)
 - t5, [77](#)
 - t6, [77](#)
 - tend, [77](#)
 - vt1, [77](#)
 - vt2, [77](#)
 - vt3, [77](#)
- gomotion::go_traj_interp_spec, [78](#)
 - a, [78](#)
 - d, [78](#)
 - j, [78](#)
 - s, [78](#)
 - v, [78](#)
- gomotion::go_uxz, [78](#)
 - x, [79](#)
 - z, [79](#)
- gomotion::go_vel, [79](#)
 - v, [80](#)
 - w, [80](#)
- gomotion::go_xyz, [80](#)
 - x, [81](#)
 - y, [81](#)
 - z, [81](#)
- gomotion::go_zyx, [81](#)
 - x, [81](#)
 - y, [81](#)
 - z, [81](#)
- gomotion::go_zyz, [81](#)
 - y, [82](#)
 - z, [82](#)

- zp, [82](#)
- gomotion::points, [83](#)
- p, [83](#)
- gotraj.cpp
 - go_exit, [113](#)
 - go_flag_uchar, [113](#)
 - go_init, [113](#)
 - go_integer_int, [114](#)
 - go_real_double, [114](#)
 - go_result_int, [114](#)
 - gocode, [114](#)
- gotraj.h
 - reciprocate, [101](#)
- gotypes.h
 - GO_QUANTITY_ANGLE, [104](#)
 - GO_QUANTITY_LENGTH, [104](#)
 - GO_QUANTITY_NONE, [104](#)
 - GO_RESULT_BAD_ARGS, [104](#)
 - GO_RESULT_BUG, [104](#)
 - GO_RESULT_DIV_ERROR, [104](#)
 - GO_RESULT_DOMAIN_ERROR, [104](#)
 - GO_RESULT_EMPTY, [104](#)
 - GO_RESULT_ERROR, [104](#)
 - GO_RESULT_IGNORED, [104](#)
 - GO_RESULT_IMPL_ERROR, [104](#)
 - GO_RESULT_NO_SPACE, [104](#)
 - GO_RESULT_NORM_ERROR, [104](#)
 - GO_RESULT_OK, [104](#)
 - GO_RESULT_RANGE_ERROR, [104](#)
 - GO_RESULT_SINGULAR, [104](#)
- gotypes.c
 - go_flag_uchar, [114](#)
 - go_integer_int, [114](#)
 - go_real_double, [115](#)
 - go_result_int, [115](#)
 - gocode, [115](#)
- gotypes.h
 - GO_FLAG, [103](#)
 - GO_FLAG_UCHAR, [103](#)
 - GO_INF, [103](#)
 - GO_INTEGER, [103](#)
 - GO_INTEGER_INT, [103](#)
 - GO_INTEGER_MAX, [103](#)
 - GO_REAL, [103](#)
 - GO_REAL_DOUBLE, [103](#)
 - GO_REAL_EPSILON, [103](#)
 - GO_REAL_MAX, [103](#)
 - GO_REAL_MIN, [103](#)
 - GO_RESULT, [103](#)
 - GO_RESULT_INT, [103](#)
 - go_flag, [104](#)
 - go_flag_uchar, [105](#)
 - go_integer, [104](#)
 - go_integer_int, [105](#)
 - go_quantity_to_string, [103](#)
 - go_real, [104](#)
 - go_real_double, [105](#)
 - go_result, [104](#)
 - go_result_int, [105](#)
 - go_result_to_string, [103](#)
 - gocode, [105](#)
- head
 - gomotion::go_motion_queue, [57](#)
- here
 - gomotion::go_motion_queue, [58](#)
- ISIZEOF
 - gomath.h, [95](#)
- id
 - gomotion::go_motion_spec, [62](#)
- im
 - gomotion::go_complex, [41](#)
- index
 - gomotion::go_matrix, [50](#)
- inertia
 - gomotion::go_body, [40](#)
- init
 - gomotion::go_interp, [46](#)
 - gomotion::go_motion_queue, [58](#)
 - gomotion::go_motion_spec, [61](#)
 - gomotion::go_scale_spec, [72](#)
- interp
 - gomotion::go_motion_queue, [58](#)
- interp_joint
 - gomotion::go_motion_queue, [58](#)
- interp_world
 - gomotion::go_motion_queue, [58](#)
- invd
 - gomotion::go_traj_ca_spec, [74](#)
 - gomotion::go_traj_cj_spec, [76](#)
- is_empty
 - gomotion::go_motion_queue, [58](#)
- j
 - gomotion::go_traj_interp_spec, [78](#)
- jerk
 - gomotion::go_motion_params, [55](#)
 - gomotion::GoMotionParams, [83](#)
- joint
 - gomotion::go_position, [66](#)
- JointState
 - gomotion, [21](#)
- jt0
 - gomotion::go_traj_cj_spec, [77](#)
- last_id
 - gomotion::go_motion_queue, [58](#)
- lpar

- gomotion::go_motion_spec, 62
- lubksb
 - gomotion, 37
- ludcmp
 - gomotion, 38
- MAG2
 - gomath.cpp, 110
- mass
 - gomotion::go_body, 40
- maxit
 - gomotion.cpp, 111
- normal
 - gomotion::go_motion_circular_params, 52
 - gomotion::go_plane, 64
- number
 - gomotion::go_motion_queue, 58
- p
 - gomotion::go_interp, 46
 - gomotion::go_rpy, 70
 - gomotion::points, 83
- par
 - gomotion::go_motion_spec, 62
- pgm
 - gomotion::GoMotion, 82
- phi
 - gomotion::go_sph, 73
- pk
 - gomotion::go_link, 48
- platform
 - gomotion::go_pk, 63
- point
 - gomotion::go_line, 47
- pose
 - gomotion::go_position, 67
 - gomotion::go_pp, 67
- pp
 - gomotion::go_link, 49
- qnormal
 - gomotion::go_motion_circular_params, 52
- quantity
 - gomotion::go_link, 49
- r
 - gomotion::go_cyl, 42
 - gomotion::go_rpy, 70
 - gomotion::go_sph, 73
- re
 - gomotion::go_complex, 41
- reciprocate
 - gotraj.h, 101
- reset
 - gomotion::go_motion_queue, 58
- rot
 - gomotion::go_hom, 43
 - gomotion::go_pose, 65
- rows
 - gomotion::go_matrix, 51
- rstart
 - gomotion::go_motion_circular_params, 52
- rtot
 - gomotion::go_motion_circular_params, 52
- s
 - gomotion::go_quat, 69
 - gomotion::go_traj_interp_spec, 78
- SIGN
 - gomath.cpp, 110
- scale
 - gomotion::go_scale_spec, 72
- scale_a_next
 - gomotion::go_scale_spec, 72
- scale_b
 - gomotion::go_scale_spec, 72
- scale_dir
 - gomotion::go_scale_spec, 72
- scale_isneg
 - gomotion::go_scale_spec, 72
- scale_next
 - gomotion::go_scale_spec, 72
- scale_spec
 - gomotion::go_scale_spec, 72
- scale_t
 - gomotion::go_scale_spec, 72
- scale_v_next
 - gomotion::go_scale_spec, 72
- scaling
 - gomotion::go_scale_spec, 72
- set
 - gomotion::go_scale_spec, 72
- set_cpar
 - gomotion::go_motion_spec, 61
- set_cycle_time
 - gomotion::go_motion_queue, 58
- set_end_pose
 - gomotion::go_motion_spec, 61
- set_end_position
 - gomotion::go_motion_spec, 61
- set_here
 - gomotion::go_interp, 46
 - gomotion::go_motion_queue, 58
- set_id
 - gomotion::go_motion_queue, 58
 - gomotion::go_motion_spec, 61
- set_joint_number
 - gomotion::go_motion_queue, 58

set_jpar
 gomotion::go_motion_spec, 61
 set_rpar
 gomotion::go_motion_spec, 61
 set_scale
 gomotion::go_motion_queue, 58
 set_time
 gomotion::go_motion_spec, 61
 set_tpar
 gomotion::go_motion_spec, 61
 set_type
 gomotion::go_motion_queue, 58
 gomotion::go_motion_spec, 61
 sincos
 gomotion, 38
 size
 gomotion::go_motion_queue, 58
 start
 gomotion::go_motion_spec, 62
 stop
 gomotion::go_motion_queue, 58
 stotinv
 gomotion::go_motion_circular_params, 52

 t1
 gomotion::go_traj_ca_spec, 74
 gomotion::go_traj_cj_spec, 77
 t2
 gomotion::go_traj_ca_spec, 74
 gomotion::go_traj_cj_spec, 77
 t3
 gomotion::go_traj_cj_spec, 77
 t4
 gomotion::go_traj_cj_spec, 77
 t5
 gomotion::go_traj_cj_spec, 77
 t6
 gomotion::go_traj_cj_spec, 77
 tend
 gomotion::go_traj_ca_spec, 74
 gomotion::go_traj_cj_spec, 77
 The Go Motion Pose Mathematics Library, 9
 there
 gomotion::go_motion_queue, 58
 theta
 gomotion::go_cyl, 42
 gomotion::go_dh, 42
 gomotion::go_sph, 73
 thtot
 gomotion::go_motion_circular_params, 52
 totalt
 gomotion::go_motion_spec, 62
 Trajectory Planning Algorithms, 13
 tran
 gomotion::go_hom, 43
 gomotion::go_pose, 65
 turns
 gomotion::go_motion_circular_params, 52
 type
 gomotion::go_link, 49
 gomotion::go_motion_spec, 62

 u
 gomotion::go_link, 49
 gomotion::go_motion_spec, 62
 gomotion::go_position, 67
 uquat
 gomotion::go_motion_linear_params, 54
 gomotion::go_motion_spec, 62
 urcsp
 gomotion::go_motion_circular_params, 52
 uvec
 gomotion::go_motion_linear_params, 54

 v
 gomotion::go_matrix, 51
 gomotion::go_traj_interp_spec, 78
 gomotion::go_vel, 80
 vel
 gomotion::go_motion_params, 55
 gomotion::GoMotionParams, 83
 vt1
 gomotion::go_traj_ca_spec, 75
 gomotion::go_traj_cj_spec, 77
 vt2
 gomotion::go_traj_cj_spec, 77
 vt3
 gomotion::go_traj_cj_spec, 77

 w
 gomotion::go_vel, 80

 x
 gomotion::go_cart, 40
 gomotion::go_mat, 50
 gomotion::go_quat, 69
 gomotion::go_rvec, 70
 gomotion::go_uxz, 79
 gomotion::go_xyz, 81
 gomotion::go_zyx, 81

 y
 gomotion::go_cart, 40
 gomotion::go_mat, 50
 gomotion::go_quat, 69
 gomotion::go_rpy, 70
 gomotion::go_rvec, 70
 gomotion::go_xyz, 81
 gomotion::go_zyx, 81

gomotion::go_zyz, [82](#)

z

gomotion::go_cart, [40](#)

gomotion::go_cyl, [42](#)

gomotion::go_mat, [50](#)

gomotion::go_quat, [69](#)

gomotion::go_rvec, [70](#)

gomotion::go_uxz, [79](#)

gomotion::go_xyz, [81](#)

gomotion::go_zyx, [81](#)

gomotion::go_zyz, [82](#)

zero_joints

gomotion::go_position, [66](#)

zero_pose

gomotion::go_position, [66](#)

zp

gomotion::go_zyz, [82](#)

zstart

gomotion::go_motion_circular_params, [52](#)

ztot

gomotion::go_motion_circular_params, [52](#)