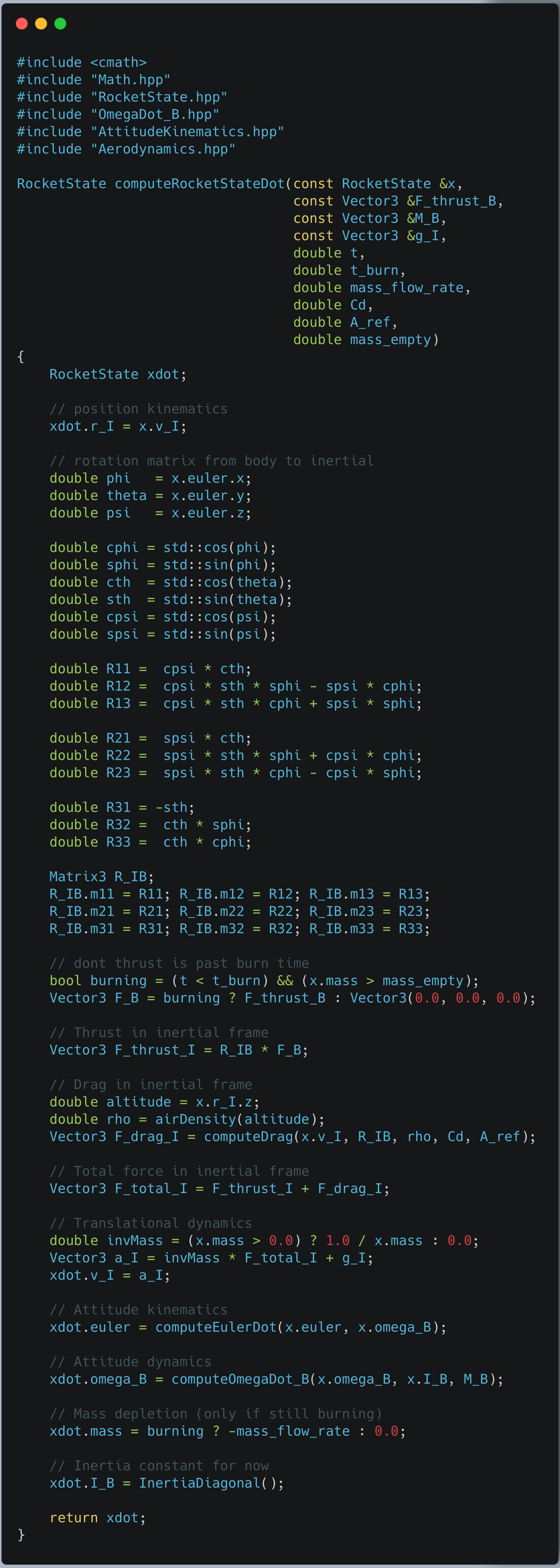

```cpp
#include <cmath>
#include "Math.hpp"
#include "RocketState.hpp"
#include "OmegaDot_B.hpp"
#include "AttitudeKinematics.hpp"
#include "Aerodynamics.hpp"

RocketState computeRocketStateDot(const RocketState &x,
                                  const Vector3 &F_thrust_B,
                                  const Vector3 &M_B,
                                  const Vector3 &g_I,
                                  double t,
                                  double t_burn,
                                  double mass_flow_rate,
                                  double Cd,
                                  double A_ref,
                                  double mass_empty)
{
    RocketState xdot;

    // position kinematics
    xdot.r_I = x.v_I;

    // rotation matrix from body to inertial
    double phi   = x.euler.x;
    double theta = x.euler.y;
    double psi   = x.euler.z;

    double cphi = std::cos(phi);
    double sphi = std::sin(phi);
    double cth  = std::cos(theta);
    double sth  = std::sin(theta);
    double cpsi = std::cos(psi);
    double spsi = std::sin(psi);

    double R11 =  cpsi * cth;
    double R12 =  cpsi * sth * sphi - spsi * cphi;
    double R13 =  cpsi * sth * cphi + spsi * sphi;

    double R21 =  spsi * cth;
    double R22 =  spsi * sth * sphi + cpsi * cphi;
    double R23 =  spsi * sth * cphi - cpsi * sphi;

    double R31 = -sth;
    double R32 =  cth * sphi;
    double R33 =  cth * cphi;

    Matrix3 R_IB;
    R_IB.m11 = R11; R_IB.m12 = R12; R_IB.m13 = R13;
    R_IB.m21 = R21; R_IB.m22 = R22; R_IB.m23 = R23;
    R_IB.m31 = R31; R_IB.m32 = R32; R_IB.m33 = R33;

    // dont thrust is past burn time
    bool burning = (t < t_burn) && (x.mass > mass_empty);
    Vector3 F_B = burning ? F_thrust_B : Vector3(0.0, 0.0, 0.0);

    // Thrust in inertial frame
    Vector3 F_thrust_I = R_IB * F_B;

    // Drag in inertial frame
    double altitude = x.r_I.z;
    double rho = airDensity(altitude);
    Vector3 F_drag_I = computeDrag(x.v_I, R_IB, rho, Cd, A_ref);

    // Total force in inertial frame
    Vector3 F_total_I = F_thrust_I + F_drag_I;

    // Translational dynamics
    double invMass = (x.mass > 0.0) ? 1.0 / x.mass : 0.0;
    Vector3 a_I = invMass * F_total_I + g_I;
    xdot.v_I = a_I;

    // Attitude kinematics
    xdot.euler = computeEulerDot(x.euler, x.omega_B);

    // Attitude dynamics
    xdot.omega_B = computeOmegaDot_B(x.omega_B, x.I_B, M_B);

    // Mass depletion (only if still burning)
    xdot.mass = burning ? -mass_flow_rate : 0.0;

    // Inertia constant for now
    xdot.I_B = InertiaDiagonal();

    return xdot;
}
```