Quick Start Guide for Using Jet Engine Model Library for Orbiter 2016
By Thunder Chicken
August 14, 2022
PM me through Orbiter Forums with questions or problems.

1. Copy JetEngineLibrary.lib and JetEngineLibrary.h to your add-on project directory and add them to your solution.

2. In your main add-on code, include JetEngineLibrary.h:

```
#include "JetEngineLibrary.h"
```

3. Open JetEngineLibrary.h and edit the following lines to provide information for the engine you wish to implement. The provided default values are for a SNECMA-M52-P2 turbofan. These parameters are commonly available for a number of engines at various internet sources such as Wikipedia, or other sites. Watch the units!

```
// Minimum engine parameters that must be provided for model at 100% throttle at sea level
// Note that maximum thrust is needed to calculate the specific fuel consumption (sfc) and
// specific impulse (Isp), but is not used to calculate thrust. However, the maximum
// thrust calculated by the model should be close to these input values.
// SNECMA-M53-P2 Turbofan

const double air_mass_flow_max = 92.0; //maximum air flow rate through engine [kg/s]
const double inlet_diameter = 0.796;       //inlet diameter of engine [m]
const double max_thrust = 65.0;            //maximum thrust without afterburner [kN]
const double rp_max = 9.8;                 //pressure ratio at full throttle [-]
const double sfc = 32.4;                   //specific fuel consumption at full throttle [g/kN s]
const double bpr = 0.36;                   //turbofan bypass ratio (0 for pure turbojets)
const double fpr = 2.0;                    //turbofan pressure ratio (0 for pure turbojets)

const double max_thrust_ab = 95.0;         //maximum thrust with afterburner [kN]
const double sfc_ab = 53.0;                //specific fuel consumption with afterburner [g/kN s]
```

4. In the main code of your add-on, enter the following global code:

```
const double air_volume_flow_max =
JetEngineLibrary::JetEngine::GetJetEngineMaxAirFlow(inlet_diameter, air_mass_flow_max);
```

This runs a one-time iterative solver to calculate the maximum air volume flow rate at full throttle accounting for compressibility at the inlet of the engine.

5. In clbkSetClassCaps, make a propellant tank:

```
PROPELLANT_HANDLE main_fuel_tank = CreatePropellantResource(main_fuel_tank_max);
```

6. This engine model is implemented using a hack to the usual Orbiter thruster implementation. In contrast to rocket engines, the thrust of air-breathing engines depends not only on the throttle setting (which establishes the propellant flow), but also the ambient atmospheric conditions and flight speed (represented by the Mach number) as they affect the momentum and mass flow of the air which is the primary reaction mass. Full-throttle thrust generally decreases with airspeed and altitude. This model should enforce maximum speed and ceiling performance *when coupled with an appropriate aerodynamic model for your add-on.*

   To allow for this behavior, you must define a user-defined "main" thruster that applies the thrust calculated from the engine model, and a "dummy" thruster that is linked to the main engine thruster group simply to take the user's throttle input to pass it to the "main" thruster. Additionally, if your engine incorporates an afterburner, that will require a third thruster definition.

   The code below is an example implementation that would go into clbkSetClassCaps after the propellant tank definition for a single engine with afterburner. Note that the maximum thrust of the main engine (max_thrust) and afterburner (max_thrust_ab) are user-supplied in kN, and so they are multiplied by 1000 to convert to Newtons.

```
THRUSTER_HANDLE th_main =
        CreateThruster(_V(0, 0, 0), _V(0, 0, 1), max_thrust*1000, main_fuel_tank, Isp_max);
THGROUP_HANDLE thg_main =
        CreateThrusterGroup(&th_main, 1,THGROUP_USER);

THRUSTER_HANDLE th_dummy =
        CreateThruster(_V(0, 0, 0), _V(0, 0, 1), 1.0, main_fuel_tank, INFINITY);
THGROUP_HANDLE thg_dummy =
        CreateThrusterGroup(&th_dummy, 1, THGROUP_MAIN);

THRUSTER_HANDLE th_ab =
        CreateThruster(_V(0, 0, -4.3), _V(0, 0, 1), (max_thrust_ab - max_thrust) * 1000,
        main_fuel_tank, (Isp_max_ab - Isp_max));
THGROUP_HANDLE thg_ab =
        CreateThrusterGroup(&th_ab, 1, THGROUP_USER);
```

   The dummy thruster has a maximum thrust of 1 N (about the weight of an apple) so it will provide a sensible 0 – 1 indication of the throttle position on the main thruster readouts in Orbiter, and has infinite Isp so it will not consume any fuel. This small thrust is negligible compared to typical engine thrusts of tens or hundreds of kN.

7. To update the thrust and Isp of the engine at each time step, enter the following code into either clbkPreStep or clbkPostStep. The afterburner is toggled on by setting boolean afterburner = true or off by setting it = false.

It should be possible to implement multiple engines, but you will have to make main and dummy versions of each engine, define different throttles, etc.

```
//----JET ENGINE IMPLEMENTATION

//Get throttle input from user from dummy thruster

double throttle = GetThrusterGroupLevel(thg_dummy);

// Get local flight conditions from Orbiter

double M_0 = GetMachNumber();          //flight Mach number [-]
double T_0 = GetAtmTemperature();      //ambient static temperature [K]
double P_0 = GetAtmPressure();         //ambient static pressure [Pa absolute]
double rho = GetAtmDensity();          //ambient atmospheric density [kg/cubic meter]

//Use inputs and flight conditions to calculate jet engine thrust and Isp

double thrust =
        JetEngineLibrary::JetEngine::GetJetEngineThrust(throttle, afterburner,
        air_volume_flow_max, M_0, T_0, P_0, rho);

//Calculate Isp of main engine and afterburner

double Isp_main =
        JetEngineLibrary::JetEngine::GetJetEngineIsp(throttle, afterburner, thrust);
double Isp_ab   =
        JetEngineLibrary::JetEngine::GetJetEngineAfterburnerIsp(throttle, afterburner, thrust);

//Update Isp for user-defined main engine and afterburner

        SetThrusterIsp(th_main, Isp_main);
        SetThrusterIsp(th_ab, Isp_ab);

//Update thruster level for user-defined main engine and afterburner

SetThrusterGroupLevel(thg_main, fmin(thrust, max_thrust*1000) /(max_thrust*1000));
SetThrusterGroupLevel(thg_ab, (thrust - (max_thrust*1000)) / ((max_thrust_ab - max_thrust)
* 1000));

//----END JET ENGINE IMPLEMENTATION
```

Additional documentation on this model is in the works, but this should allow add-on developers a chance to implement the model and help beta test it. A JetEngineCodeSnippets.cpp file is included with much of the code described here and some additional implementation features that will hopefully help you get flying.