Simplified Method for Simulating Ascent Configuration Changes of Large Scale Integrated Launch Vehicles in Thermal Desktop

Robert J. Kwas
Launch Services Program, Kennedy Space Flight Center, Cape Canaveral FL, 32899

ABSTRACT

Integrated Launch Vehicles (ILV) typically undergo several major changes to their outer mold line configuration during their ascent mission phase. These changes are typically induced by events such as first stage separation, upper stage separation, and payload fairing jettison. In Thermal Desktop (TD), ILV ascent configuration changes are traditionally simulated by developing specific cases to represent each of the configurations attained by the ILV during its ascent mission phase. For each case, build statements and Radiation Analysis Groups (RAG) are utilized in an effort to build only those portions of the SINDA network associated with the current case configuration. All cases comprising the ascent mission phase are then thermally sequenced by setting their initial temperatures to the final temperatures predicted from the previous case. While feasible, this method presents significant challenges when used in conjunction with large-scale ILV models, including manual verification/management of lengthy user – written build statements and complex RAG.

Recently, a new method has been developed for simulating ILV ascent configuration changes in TD. This method utilizes TD assemblies to physically translate specific parts of the ILV geometry during the ascent mission phase. This method avoids the need to use build statements, multiple RAG's, and multiple ascent cases. The method is shown to be quite simple to implement, verify, and very portable. This method also proves quite effective in handling ILV attitude control maneuvers such as Passive Thermal Control (PTC) rolls.

LIST OF ACRONYMS AND ABBREVIATIONS

ELV	Expendable Launch Vehicle
hrTime	Orbit Time

ILV	Integrated Launch Vehicle
MECO	Main Engine Cut OFF
PLF	Payload Fairing
PTC	Passive Thermal Control
RadCAD	Radiation Computer Aided Design
RAG	Radiation Analysis Group
RadK	Radiation Conductor
SINDA	Systems Improved Numerical Differencing Analyzer
SC	Spacecraft
TD	Thermal Desktop

INTRODUCTION

Due to their relatively low cost and high reliability, most spacecraft today are delivered to orbit using Expendable Launch Vehicles (ELV). Available from numerous launch service providers, ELV are typically two stage rockets consisting of a booster and an upper stage. Prior to being launched, the spacecraft is encapsulated in a protective covering known as a Payload Fairing (PLF). The encapsulated spacecraft is then mated to the forward end of the ELV using a structural fitting known as a payload adapter. In the mated configuration, the ELV and spacecraft are commonly referred to as the ILV. The basic components of an ILV are shown graphically in Figure 1.

Following launch, an ILV undergoes several separation events including booster separation, PLF jettison, and upper stage separation. As shown in Figure 2, these ascent separation events induce significant changes to the ILV configuration. Consequently, thermal models used for simulating ILV ascent mission phases must employ effective methods for simulating ILV ascent configuration changes.

TRADITIONAL METHOD FOR SIMULATING CONFIGURATION CHANGES

In TD, changes to model geometry are traditionally simulated by developing a specific case to represent each unique geometric configuration. For each case, build statements and RAG are utilized to build the specific portion of the SINDA network (linear and radiation) that the current case represents.

Written by the user in the operations block, a build statement is a SINDA command that determines what portion of the linear thermal network is built by the SINDA pre-processor. A build statement contains a list of user specified submodels. Defined by the user, submodels divide the linear thermal network into distinct subsections. When a TD case is run, the SINDA pre-processor builds only those submodels listed in the build statement written for that case. As this implies, build statements are case specific; a unique built statement, containing a unique list of submodels, can be written by the user for each case in the TD case manager.

Defined in the analysis group manager, RAG's determine what portion of the radiation network is built by RadCAD. A RAG is a user defined set of TD surfaces. Each RAG is given a unique name that can be referenced from within the radiation task tab. When a RadCAD case is run, radiation exchange factors and/or heating rates are generated only for those surfaces associated with the RAG(s) listed in the radiation task tab for that case.

Using the traditional method, the user develops a TD model such that each ILV ascent configuration is represented by a distinct group of submodels and surfaces. By constructing the appropriate build statements and RAG's within the case manager, specific cases can be generated to represent each ILV ascent configuration. The cases are then thermally sequenced using the appropriate case temperature initializations. Specifically, the initial temperatures for each case are set to the final temperatures predicted by the previous case.

While seemingly straightforward, this method can be very challenging to apply large-scale ILV models. Large – scale ILV models can contain hundreds of submodels making build statements very lengthy. Likewise, RAG's may contain thousands of surfaces. Users must meticulously verify that each RAG and build statement contains the correct surfaces and submodels, respectively, for the case they represent. Unfortunately, much of this verification must be done manually. Note that running a model with malformed build statements and/or RAG will not invoke a run – time error or warning and, consequently, the user is often unaware that these errors even exist. Another unappealing aspect of this method has to do with implementing modifications to the model. Any modification that adds/deletes surfaces or changes the submodel architecture (i.e., adds deletes or renames submodels) will necessitate that the user make the appropriate updates (manually) to all build statements and RAG in the model. This process can be very time consuming and difficult to verify. Finally, the case temperature initializations required for this method can be very tedious to

implement. Each case typically contains a unique set of nodes and, consequently, the user must initialize the nodal temperatures of each case manually.

USING ASSEMBLIES TO SIMULATE CONFIGURATION CHANGES

Used in conjunction with articulating orbits, assemblies provide an effective method for simulating dynamic motion associated with model geometry. This method only requires that the user embed a relatively simple line of code within the assemblies' expression editor and, subsequently, synchronize the timing of the motion with the associated orbit time - line. This method proves equally effective for simulating both translational and rotational motion.

An assembly is an articulator that can linearly transform model geometry by either translation and/or rotation. The user first attaches the assembly to the specific surfaces in the model they desire to transform. From within the assembly translation/rotation input fields, the user specifies the direction and magnitude of the transformation. Assembly transformations can be expressed as constant values or as functions. When expressed as a constant, the transformation will be executed prior to running the case (provided the assembly is active). However, if expressed as a function of orbit time, the transformation can be *simulated* as occurring at some specific time(s) during the SINDA run. As will be shown, this dynamic transformation of model geometry can be effectively utilized to simulate ILV ascent configuration changes.

An example of a line of code that can be used to translate model geometry as a function of orbit time is shown below:

```
((hrTime >= 164) ? 200: 0)
```

Automatically calculated by RadCAD, hrTime is a symbol that represents the time associated with the current orbital position. As shown in Figure 3, by inserting this line of code into the "Translation X" expression editor, the assembly will translate the attached model geometry 200 meters in the positive X direction once the orbit time (hrTime) is greater or equal to 164 seconds. If the magnitude of this translation (i.e., 200 meters) is sufficiently large, then the radiant heat transfer between the translated geometry and the rest of the model will be broken (after the translation is executed). This is because rays shot from the translated surfaces will no longer intersect with the other surfaces in the model, and vice

versa. The translation distance required to break the radiation connection between the two halves of the model depends on the size of the surfaces being translated and the number of rays being shot in the RadCAD solution. However, it relatively easy for the user to determine this distance by reviewing the RadCAD output files.[1]

Note that in the expression editor, the user can specify the desired translation time and distance. Thus, the method for using assemblies to simulate ILV separation events is straightforward. The user first creates an assembly and attaches it to the surfaces associated with a particular ILV stage. Next, the user inserts the appropriate code into the expression editor such that the assembly translates the aforementioned surfaces at the appropriate (separation) time. Finally, the user ensures that the magnitude of the translation is sufficiently large as to break the radiation connection between the two halves of the model.

In order to implement this method properly, the user must shoot articulating Radiation Conductors (RadK's). Articulating RadK's are linked to an orbit, which in this case simulates discrete time points associated with the ILV ascent trajectory. Since the ILV geometry is changing during ascent, there needs to be at least two orbital positions shot for each configuration. These orbital positions will define the temporal extents of each configuration. Specifically, the orbital time (hrTime) associated with the first position will correspond to the initial time associated with that configuration. Likewise, the orbital time associated with the second position will correspond to the final time associated with that configuration. Shown graphically in Figure 4, this method ensures that the correct RadK's are loaded into the SINDA network at the correct time.

Note that there are several advantages to using assemblies for simulating ascent configuration changes. First, the entire ascent trajectory can be simulated using a single TD case. This precludes the need to perform manual case temperature initializations and, further, simplifies post processing as all results are written to a single output file. As the linear and radiation networks are manipulated using assembly induced translations, there is no need to employ user – written build statements or multiple RAG's. Consequently, all submodels can be managed (i.e., added, deleted, and renamed) using the SINDA submodel manager and, further, all surfaces can be put into a single RAG. Finally, the entire process can be verified simply by visualizing the orbit associated with the ascent trajectory. To

1 If the translation distance is sufficiently large to break the radiation connection, then there will be no radiation conductors between nodes of the translated and non – translated surfaces.

demonstrate this, the animation located in Appendix A was created. This animation (i.e., staging events) visualizes an orbit that was used to simulate the ILV ascent trajectory illustrated in Figure 2. The trajectory begins at launch and extends to spacecraft separation. By playing the animation, the reader can easily verify that the assemblies used to translate the various surfaces of the model are working correctly (i.e., the correct surfaces are translated at the correct times).

ATTITUDE CONTROL MANEUVERS

While effective for simulating staging events using translation operations, assemblies can also be utilized to simulate roll maneuvers. One such roll maneuver commonly employed during an ELV mission is a PTC roll.

A PTC roll prevents certain ILV components/systems from becoming too hot or cold by continually changing which side of the ILV is subjected to eclipse/illumination. A PTC roll is typically implemented during the upper stage coast period, which begins shortly after Main Engine Cut Off (MECO). The PTC roll is typically implemented about the longitudinal access of the ILV while it is flying broad side to the sun. Roll reversals are often utilized to mitigate any potential errors imparted to the ILV attitude control system.

PTC rolls can be simulated by defining a "vector – list" type orbit in the TD orbit manager and then mathematically rotating the appropriate solar and earth vectors using tensor transformations. While this method is effective, it is generally simpler to utilize an assembly to rotate the model geometry. As with translations, assemblies can rotate model geometry as a function of orbit time. This is accomplished by inserting the appropriate code in the assembly's rotation expression editor. An example of a line of code that can be used to rotate model geometry as a function of orbit time is shown below:

```
((hrTime < 800) && (hrTime > 500)) ? hrTime : 0
```

As shown in Figure 5, by inserting this line of code into the "Rotation 3" expression editor, the assembly will begin to rotate the attached model geometry about the ILV's positive Z-axis once orbit time (hrTime) exceeds 500 seconds. The rotation will stop once hrTime exceeds 800 seconds.

DISCUSSION

Assemblies provide the user with a powerful method for simulating dynamic motion in a TD model. Assemblies can be used to simulate translational motion (i.e., ILV staging events) as well as rotational motion (i.e., PTC rolls). Assemblies preclude the need to employ cumbersome user – written build statements and multiple RAG's. This greatly facilitates model verification, updates, and post processing of results.

As it requires articulating RadK's, the assembly method will increase the computational time associated with the RadCAD solution. However, this is generally trivial in comparison to the computational time associated with other parts of the radiation solution (i.e., FMH, Solar, Earth IR and Albedo). In addition, because geometric translations expand the model domain, the assembly method can distort the Oct Cells associated with the RadCAD solution and, consequently, increase the overall RadCAD solution time. However, this problem is easily mitigated by increasing oct tree parameters used in the RadCAD control criteria.

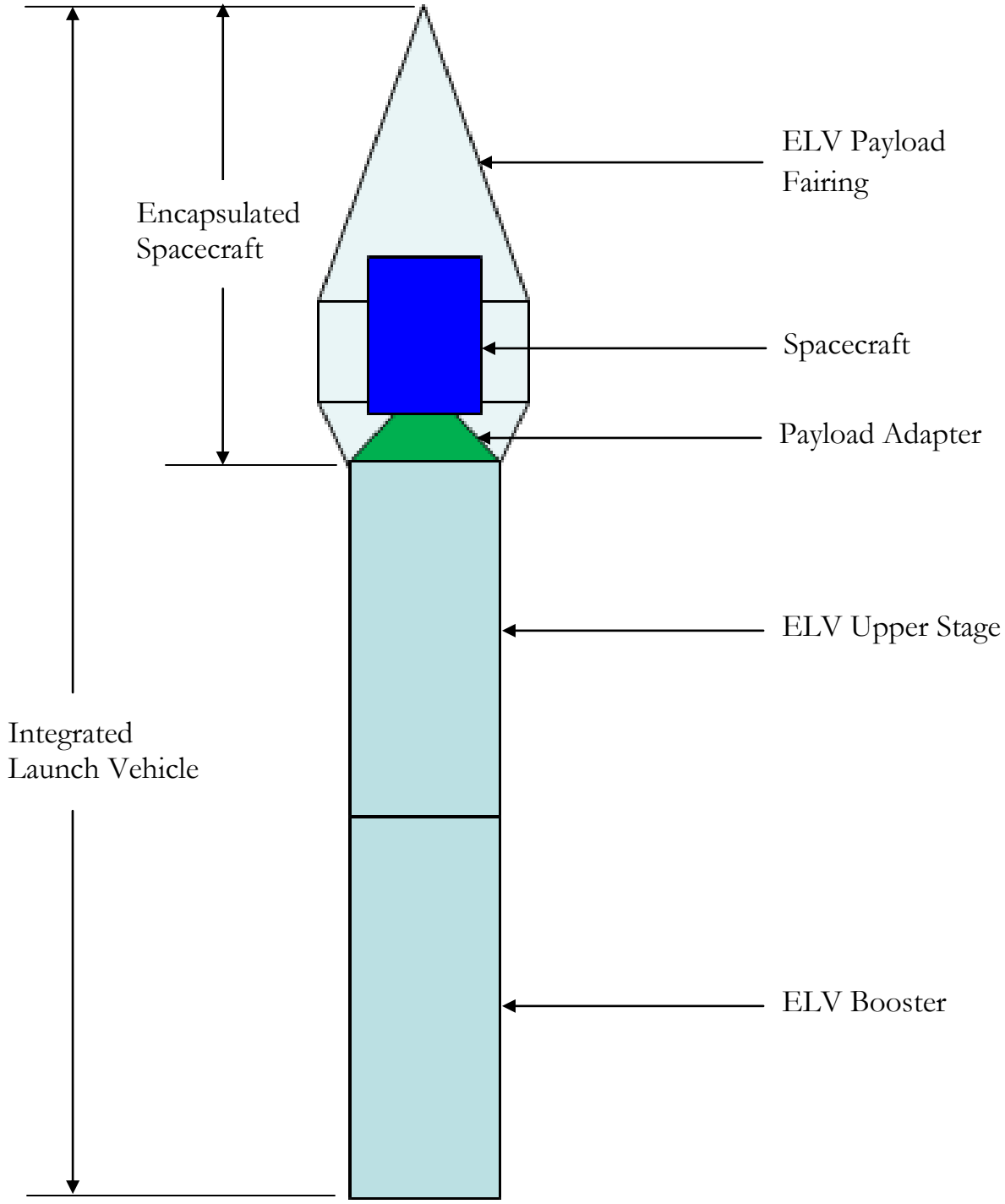


Figure 1 Typical Components Comprising an Integrated Launch Vehicle

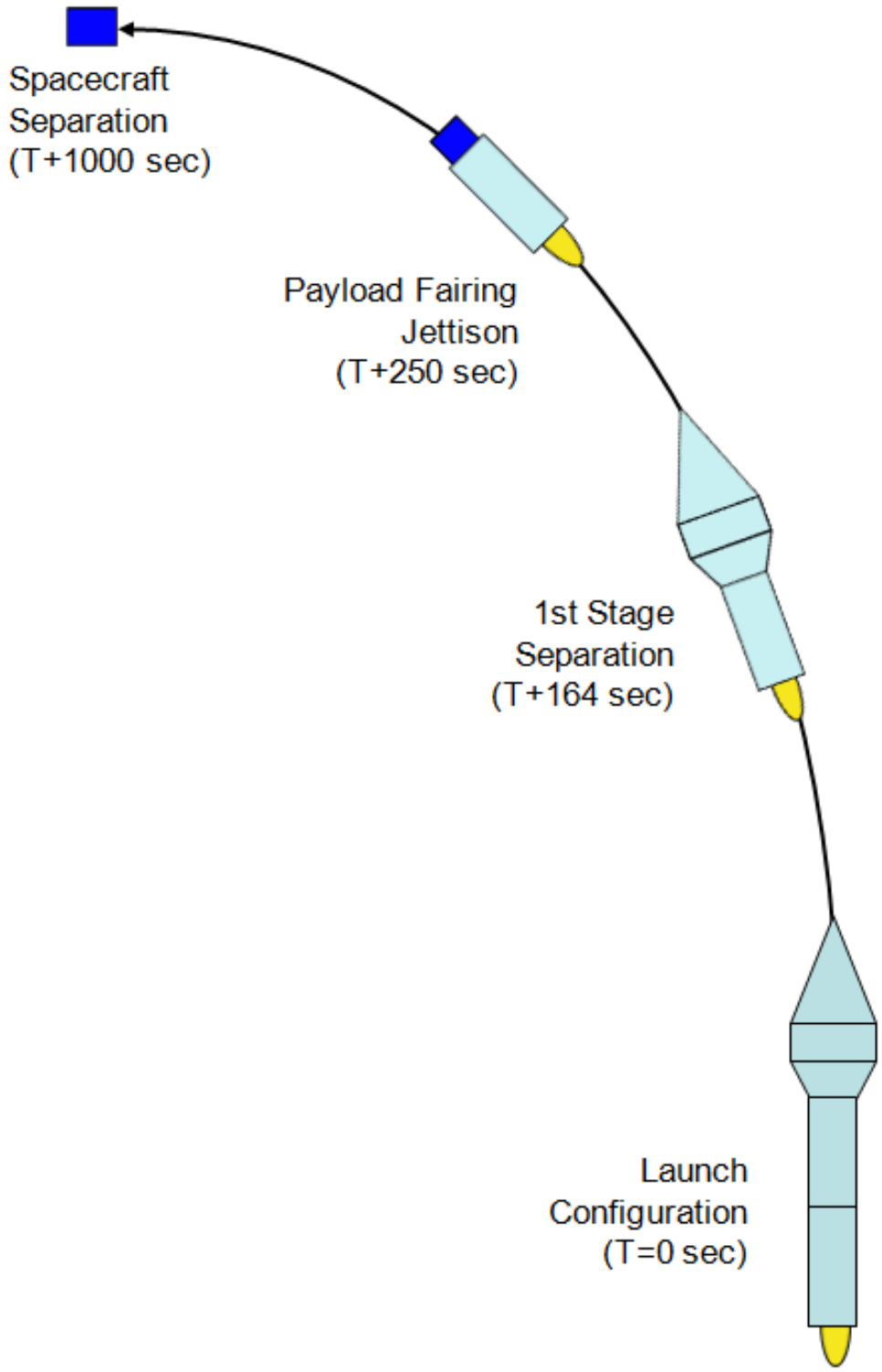


Figure 2 Typical ILV Ascent Configuration Changes

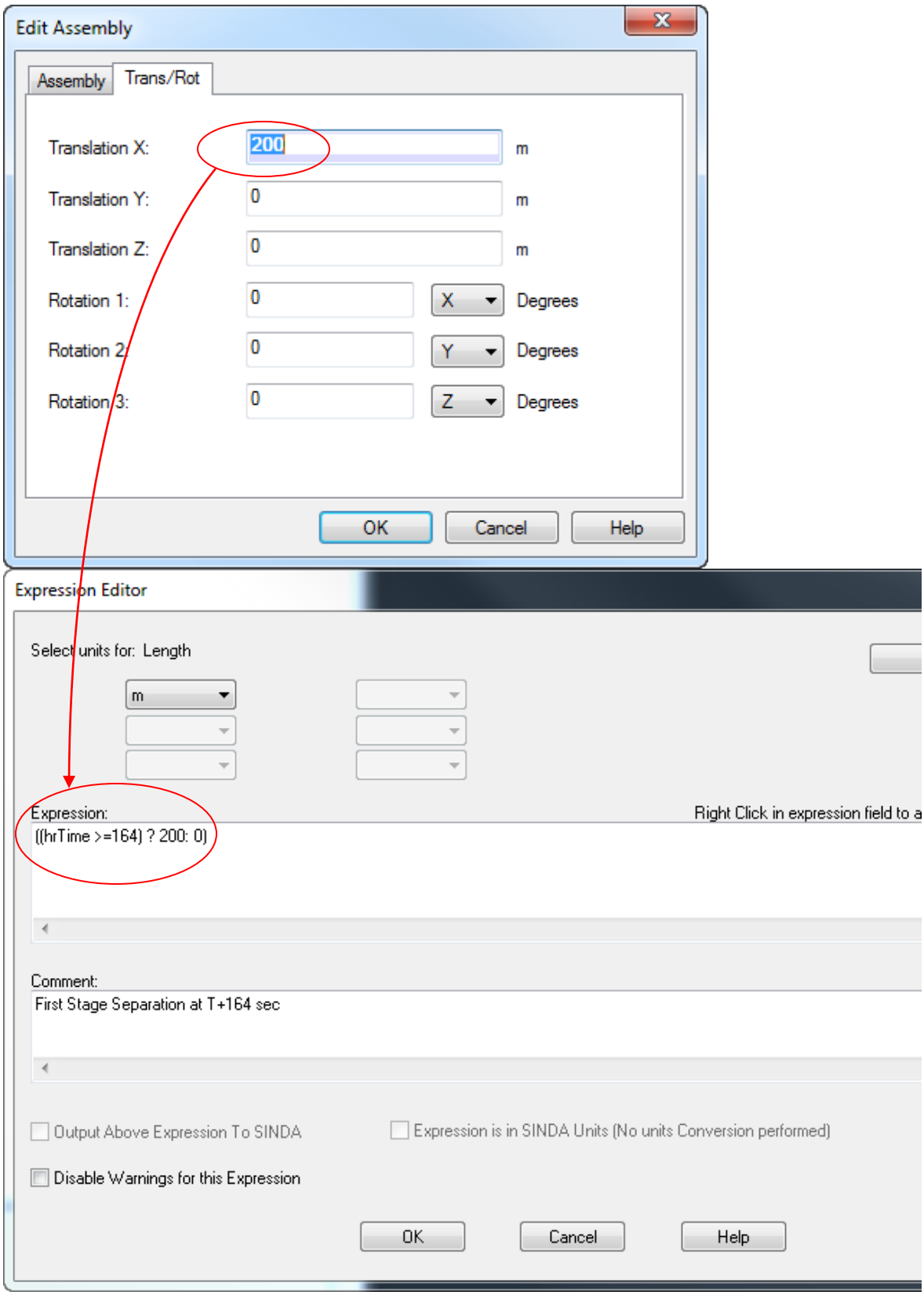


Figure 3 Assembly Expression Editor (access by double clicking on Translation X field)

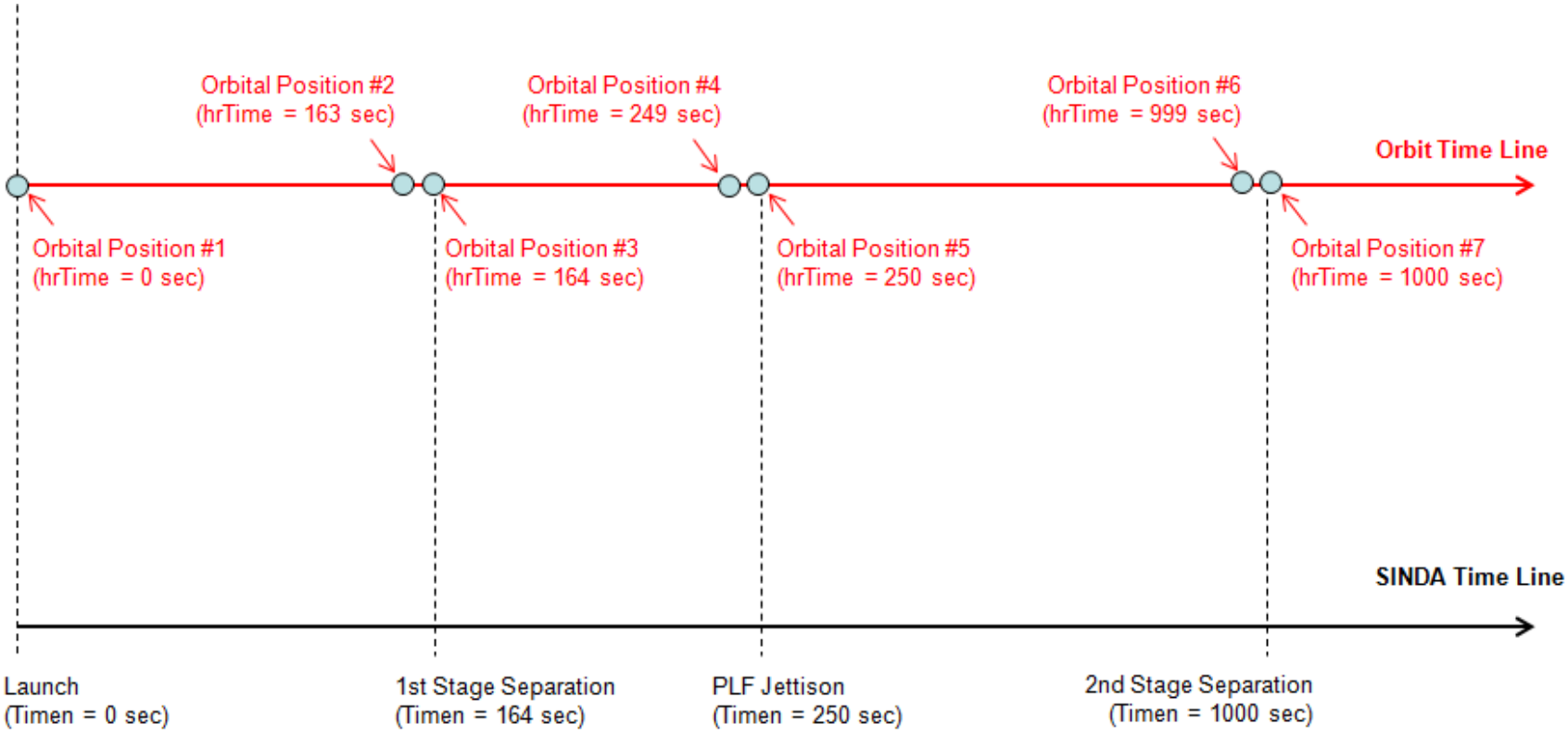


Figure 4 Synchronization of SINDA and Orbit Time Lines

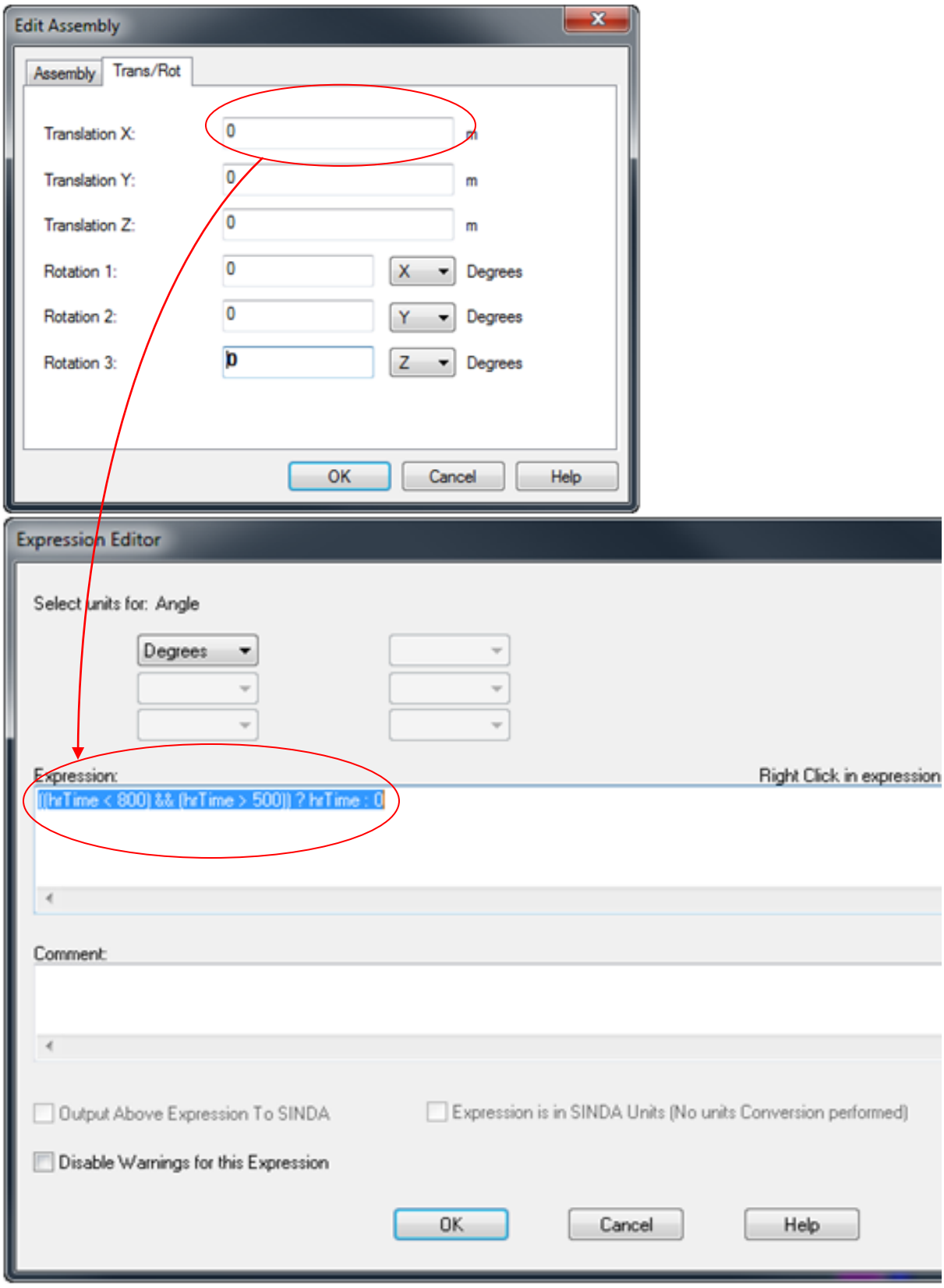
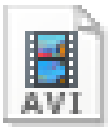


Figure 5 Assembly Expression Editor (access by double clicking on Rotation 3 field)

APPENDIX A – ANIMATIONS FOR ILV ASCENT TRAJECTORY

Staging Events.avi



PTC Roll.avi