FLIGHT Hangars

Order Number: TBD

June 1991

This document describes the FLIGHT notion of hangars, how users may construct their own hangars, including information on how to invoke the FLIGHT hangar compiler and use of the FLIGHT hangar editor.

Revision/Update Information: This document supersedes *FLIGHT Hangars*, V2.4

Operating System and Version: VMS Version 5.4

Software Version: FLIGHT, Version 2.5

digital equipment corporation

Table of Contents

List of Figures

List of Tables

1 Hangars

This chapter describes FLIGHT hangars and their use in the simulator.

1.1 Overview

A FLIGHT hangar is a collection of object and instrument descriptions in binary form; that is, FLIGHT has compressed and converted them into a form which is accepted by the simulator program. In this chapter, you will learn how to use FLIGHT's hangar compiler to convert a source to the final hangar form. See Section 2.1 for a description of how to invoke the hangar compiler.

A FLIGHT hangar source file is a text file which a designer creates to describe the contents of a hangar.

Each object is described by a series of commands in the hangar source file. An example of such a command is the FUEL command.

```
FUEL/ALLOWANCE=7000.0
FUEL/CONSUMPTION=3.5
```

The above example shows two lines of a sample source file, both using the FUEL command.

1.2 Organization of Hangar Source Files

Some FLIGHT commands reference hangar objects. For example, the command to add an instrument to an instrument panel is PLACE. This command's first parameter is the name of an instrument. This means that the instrument must have been defined before the PLACE command was encountered. The same is true for weapons, which are referenced by the WEAPONS command.

Because of this referencing mechanism, hangar source files should be organized such that all objects which are going to be referenced should be at the beginning of the file, and objects which make the references should be at the end of the file.

Below is an example of the organization in a typical hangar source file.

```
OPEN INSTRUMENT=DIAL="Altimeter"

.
.
.

CLOSE INSTRUMENT

OPEN INSTRUMENT=RADAR="10-mile Radar"

.
.
.

CLOSE INSTRUMENT

OPEN OBJECT="Sidewinder"

.
.
.

CLOSE OBJECT

OPEN OBJECT="F4 Phantom"

PLACE "Altimeter" 0.45 3.5
PLACE "10-mile Radar" 13.1 13

WEAPONS "Sidewinder" 0.0 0.0 10.0
WEAPONS "Sidewinder" 0.0 0.0 -10.0

CLOSE OBJECT

OPEN OBJECT="F16 Fighting Falcon"

.
.
.

CLOSE OBJECT

.
.
.
```

1.2.1 Structuring Hangar Source Files

Because object descriptions can be quite lengthy (500 source lines is not uncommon for the example objects), FLIGHT allows designers to include multiple source files into a single hangar's compilation using the INCLUDE command. The INCLUDE command compiles the hangar commands in an included file just as if they were in the original file. A convenient way to mix and match objects in any desired order or combination is to place one object in each source file and then create another file with only INCLUDE commands in it.

For example, if you reduced a large hangar file to a smaller source, the file might look like the following:

```
INCLUDE INSTRUMENTS.DAT
INCLUDE SIDEWINDER.DAT
INCLUDE F4_PHANTOM.DAT
INCLUDE F16_FIGHTING_FALCON.DAT
```

1.2.2 Commenting Designs

Although the hangar commands which FLIGHT uses are usually self-explanatory, it is often useful for the designer to add comments to the source files to make the designs more understandable. This is especially useful when the designer needs to change some aspect of a design.

FLIGHT allows you to place comments in source files by using the exclamation character. When you begin a line in a source file with an exclamation character at the beginning of the line, you can add arbitrary text to the source which the hangar compiler will ignore. This is true, even if the lines which begin with the exclamation character have valid hangar commands on them.

For example, if you wanted to add information to a source file using comments, the source file might look like the following:

```
OPEN OBJECT="F4 Phantom"

OPEN DISPLAY="Pilot"
!
! Instrument panel stuff
!
OPEN PANEL
PLACE "Altimeter" 0.45 3.5
PLACE "10-mile Radar" 13.1 13.0
CLOSE PANEL
!
CLOSE DISPLAY
!
! Two sidewinder missiles. One is placed at each wingtip.
!
WEAPONS "AIM-9L Sidewinder" 0.0 0.0 10.0
WEAPONS "AIM-9L Sidewinder" 0.0 0.0 -10.0
!
CLOSE OBJECT
```

1.3 Basic Design

In order to design objects for FLIGHT, you need to have a basic understanding of what FLIGHT is expecting in terms of units, etc.

FLIGHT represents hangar objects in terms of four unrelated components; its geometric description, volumetric description, performance description and cockpit description.

Note that you must have an object open in order for any of the commands which specify these components to be considered valid by the hangar compiler. Other than that condition, you may place the individual components in any order in your object description.

Objects are opened by use of the OPEN OBJECT command. Only one object may be open at any one time. The CLOSE OBJECT command causes the currently open object to be closed.

1.3.1 Geometric Description

A geometric description describes the appearance of the hangar object while it is visible during a simulation run.

Object geometry is always specified in units of feet and is three dimensional. That is, you always specify (X, Y, Z) coordinates. Geometry may be specified in either wireframe form (a collection of lines) or in a solid, faceted form (a collection of triangular plates). At present, only the wireframe form is support for display.

When designing object geometry, it is important to understand that the geometry may take any appearance you like. The geometry of the object has nothing to do with the performance of the object. If you like, you can create a Mach 5 aircraft which looks like a loaf of bread.

1.3.1.1 The GEOMETRY Commands

The procedure for describing object geometry is through the use of the GEOMETRY commands. The bottom-most command is the one which specifies where all the points of the object are. This is the GEOMETRY POINT command. Each such command defines the location of one point. So a series of points are defined by multiple GEOMETRY POINT commands.

Once the GEOMETRY POINT commands have been specified, you supply a series of GEOMETRY LINE commands. These commands specify the wireframe form of the object by telling FLIGHT to draw lines between pairs of points. The points that the GEOMETRY LINE command references are the ones you supplied via the GEOMETRY POINT commands. The points were numbered as you specified them. The first GEOMETRY POINT command created point number one, the second was point number two, etc.

In addition to describing the geometry in wireframe, you may also describe it in a solid form by use of the GEOMETRY TRIANGLE command. This command specifies three points, again taken from the list of points defined by the GEOMETRY POINT commands. Remember that FLIGHT does not currently support display of solid objects.

The following example shows construction of a cube. Note that it is 2 feet by 2 feet by 2 feet. Units are in feet.

```
GEOMETRY POINT  1.0  1.0  1.0
GEOMETRY POINT -1.0  1.0  1.0
GEOMETRY POINT -1.0  1.0 -1.0
GEOMETRY POINT  1.0  1.0 -1.0
GEOMETRY POINT  1.0 -1.0  1.0
GEOMETRY POINT -1.0 -1.0  1.0
GEOMETRY POINT -1.0 -1.0 -1.0
GEOMETRY POINT  1.0 -1.0 -1.0

GEOMETRY LINE 1 2
GEOMETRY LINE 2 3
GEOMETRY LINE 3 4
GEOMETRY LINE 4 1
GEOMETRY LINE 5 6
GEOMETRY LINE 6 7
GEOMETRY LINE 7 8
GEOMETRY LINE 8 5
GEOMETRY LINE 1 5
GEOMETRY LINE 2 6
GEOMETRY LINE 3 7
GEOMETRY LINE 4 8
```

1.3.1.2 Orientation

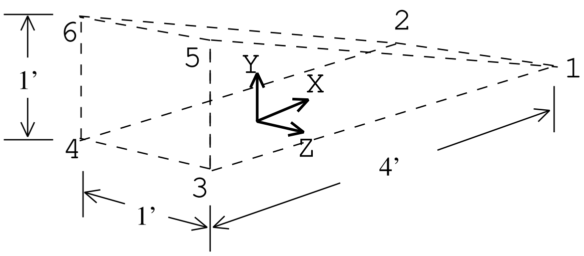
So the units are feet, and GEOMETRY commands are used to describe the object appearance. What about orientation? An aircraft has to be designed such that it flies forwards. It’s entirely possible to design an aircraft such that it appears to fly backwards because you have the geometry backwards.

If you are sitting in the cockpit of your aircraft, forward is positive X, up is positive Y, and right is positive Z. This means that wings will extend predominantly in the Z direction, their vertical thickness is in the Y direction, and the front-to-back width of the wing (the chord) is predominantly in the X direction. The tip of the nose of the aircraft will have the largest X value of any part of the aircraft, the landing gear will have the smallest Y value, and the top of the tail will have the largest Y value.

The cube example above is hardly a good illustration of orientation because it is symmetrical in all directions. The following example of a wedge illustrates orientation such that the point of the wedge is forwards and the top surface is sloping.

```
GEOMETRY POINT 3.0 0.0 1.0
GEOMETRY POINT 3.0 0.0 -1.0
GEOMETRY POINT -1.0 0.0 1.0
GEOMETRY POINT -1.0 0.0 -1.0
GEOMETRY POINT -1.0 1.0 1.0
GEOMETRY POINT -1.0 1.0 -1.0

GEOMETRY LINE 1 2
GEOMETRY LINE 1 3
GEOMETRY LINE 1 5
GEOMETRY LINE 2 4
GEOMETRY LINE 2 6
GEOMETRY LINE 3 4
GEOMETRY LINE 4 5
GEOMETRY LINE 5 6
GEOMETRY LINE 6 4
```



Note that the object has been designed without regard to where the object will appear in the world. The location of the point where all three values is zero is important. It is the point about which the object will rotate. In the wedge example, this location, known as the object origin, is on the underside of the object, 3 feet back from the front and centered left/right.

1.3.1.3 Color

The color of an object is set by use of the GEOMETRY COLOR command. Color may only be specified on a per object basis. That is, individual lines cannot be separate colors.

When specifying colors for objects, you may either specify the name of a color which has been defined earlier in the hangar, or rely on the display machine to know the color by name. For example, using a color name of "Red" is probably safe. However, expecting a display machine to understand what you mean by "Slightly off-white" is rather doubtful unless you have used the DEFINE COLOR command prior to starting this object definition.

The DEFINE COLOR command is used for two purposes:

- To define what odd names of colors should look like. For example, "Slightly off-white" could be defined to have any given color, and FLIGHT would attempt to display objects with that color as faithfully as possible.
- To establish the priority of colors. The order in which colors are defined determines the likelihood of their being exactly that color on the screen. Note that FLIGHT will perform a DEFINE COLOR automatically for you if you have not defined a color name used in a GEOMETRY COLOR command. When it comes time to try to display geometry in the specified color, FLIGHT will ask the workstation what the meaning of the color (e.g. "Red") is.

Remember that the DEFINE COLOR command is hangar-wide. It is *not* a definition of color priorities per hangar object.

If you wanted to set the color of the wedge to blue, it would be a simple matter of issuing the appropriate command.

```
GEOMETRY COLOR "blue"
```

Color names are case-sensitive and should be enclosed in quotes. If you do not, the color name will be converted to all upper case.

1.3.1.4 Levels of Detail

As an object moves away from you, it appears to become smaller and smaller. It is entirely non-optimal for the simulator to display all of the lines that you specified as the geometry of your object when the object is just a few pixels wide on the workstation screen. It is also cumbersome to the user for objects which are distant to always appear as blobs instead of appearing in some meaningful manner.

It would seem desirable to have something like a simpler stick figure of your object be used as the object grows visibly smaller. FLIGHT supports such a concept via what are known as *levels of detail*. A level of detail is a complete geometric description of your object, but keeping in mind that the geometric description will be used in a certain range of visual sizes of the object.

As an example, imagine an aircraft taking off on a runway. You are standing at the end of the runway that the aircraft is starting from. While you are standing next to the aircraft, it may fill your field of view. If you were to point at the extreme right edge of the aircraft with your right arm and then point at the extreme left edge of the aircraft with your left arm, you would create some angle between the two outstretched arms. Call this angle the *view angle*. While you are close, this angle will be very large, perhaps larger than 90 degrees.

The aircraft starts its engines and starts to roll down the runway. If you keep your arms pointing at the sides of the aircraft, you'll find that the angle between your arms, the view angle, is rapidly getting smaller. By the time the aircraft is to the other end of the runway, your arms are probably nearly touching and you're tempted to touch the sides of the aircraft with thumb and forefinger.

FLIGHT allows you to specify up to 3 full geometric descriptions of your object. You are also able to specify at what range of view angles these geometric descriptions should be used by FLIGHT to display your object. The GEOMETRY LOD command is used to introduce a new level of detail description and to specify the smallest angle at which the geometric description following the LOD command should be used.

If you specify three levels of detail for your object and the third level of detail has a minimum angle greater than zero degrees, then FLIGHT will cause the object to vanish when the view angle to the object becomes smaller than that third level of detail's minimum angle.

The following example shows the general flow of GEOMETRY LOD, POINT and LINE commands.

```
GEOMETRY LOD 1 10.0

GEOMETRY POINT 0.0 0.0 0.0
GEOMETRY POINT ...

GEOMETRY LINE 1 2
GEOMETRY LINE 4 12
GEOMETRY LINE ...

GEOMETRY LOD 2 1.0

GEOMETRY POINT 0.0 0.0 1.0
GEOMETRY POINT ...

GEOMETRY LINE 1 4
GEOMETRY LINE 3 9
GEOMETRY LINE ...

GEOMETRY LOD 3 0.1

GEOMETRY POINT 0.0 0.0 1.0
GEOMETRY POINT ...
```

```
GEOMETRY LINE 1 2
```

In the example, all three levels of detail are being specified. Note that you do not need to specify all levels of detail. If you specify only one detail level, FLIGHT will use that geometry description until the minimum view angle of that detail level is reached, at which point the object will vanish. When the view angle to the object again goes above the minimum for that level of detail, the object will appear again.

According to the example, the first level of detail (LOD 1) will be used to display the object while the view angle to the object is 10 degrees or more. The second level of detail will be used when the view angle is between 10 and 1 degrees. The last detail level will be used below 1 degree, but above 0.1 degrees. When it becomes smaller than 0.1 degrees, it vanishes.

You should not attempt to design the object smaller and smaller for the smaller levels of detail. Merely provide less and less detail to the object. FLIGHT will ensure that the object is displayed at the correct size.

You should also realize that the geometric descriptions of detail levels are completely self-contained. They may have different numbers of points and lines. You may not reference a point in one level of detail to create a line in another level of detail.

Each level of detail may have up to 300 points, 300 lines and 200 triangles.

1.3.2 Volumetric Description

A volumetric description describes the solidity of an object. This solid description is not used for display, but for determining collisions between objects and for determining what result those collisions have.

The units involved with volumes are, like geometry, specified in units of feet and are three dimensional. Unlike geometric descriptions, volumetric descriptions are specified only once. That is, there are no detail levels to volumetric descriptions.

1.3.2.1 Volumes

A volume is a sphere of a given radius at a location with respect to the object origin, with a damage rating, and a destruct effect. The location of the sphere is specified just as a point is in a geometric description. Forward is positive X, up is positive Y, and right is positive Z.

The damage rating of a volume is arbitrarily assigned and only has meaning in relation to the volumes of other objects. For example, if the object for a machine gun tracer has a damage rating of 10, then an aircraft wanting to be able to survive 10 hits of tracer rounds should have a volume with a damage rating of 100.

However, up to 20 volumes can be specified per hangar object, allowing the designer to distribute the volumes throughout the object, assigning whatever damage rating seems reasonable for the associated part of the aircraft. For example, if a volume were placed in the wing to represent damage to the wing, it might receive a damage rating of 30 so that it could survive 2 hits by tracer rounds and be considered destroyed by a third.

Realize that many volumes on an object can slow the performance of all components of FLIGHT, and may impair multi-user simulations significantly. In general, the fewer the number of volumes, the better.

The simple notion behind damaging a volume is that whenever two volumes from different object come in contact with either other, one volume will be destroyed. This is because one volume is always rated higher than the other. In the case where two volumes have equal damage ratings, they will both be destroyed.

1.3.2.2 Damage Effects

When a volume is destroyed, its associated damage effect is applied to the object that the volume is bound to. Table 2-12 lists the possible effects on an aircraft when a volume is destroyed. Note that the LOCALDESTRUCT keyword refers to subobjects. They are described in Section The Hangar Editor, *Subobjects*.

If the SELFDESTRUCT damage effect is bound to a volume, and that volume is destroyed, the object that the volume is in will be destroyed. If that object is the user's aircraft, that ends the user's simulation.

That is the only case (apart from the LOCALDESTRUCT effect) where a volume affects the geometry of an object. At all other times, the geometry will be unaffected. However, the LOCALDESTRUCT exception grants the designer a wide range of possibilities to have volume damage affect the geometry of an object (see the section describing subobjects).

1.3.2.3 Surface Contact

In a FLIGHT world, terrain and water can exist which the object you are defining may be able to drive or float over. Although you have defined the volume that your object occupies, FLIGHT does not check that volumetric description against the curves and contours of the terrain or water. For example, if your aircraft were driving along next to a steep cliff, the wing of the aircraft may protrude through the wall of the cliff, even though there might be a volume created to indicate where the wing is.

1.3.2.4 Standardization

All of the aircraft and weapons which FLIGHT provides are standardized on a certain level of damage points. Those standards are listed here to allow object designers to create combat aircraft and weapons systems which will work well with the standard objects.

- *Kinetic energy weapons* are those weapons such as machine guns, cannon, etc. FLIGHT defines each object fired by such a weapon to contain anumber of projectiles (bullets) equal to a 0.05 second burst from the weapon. So if the weapon in question is the M61A1 gatling gun, which fires at 6000 rounds per second, a total of 6000/60*0.05 = 5 rounds per shot. The Calibre of the M61A1 rounds is 20mm, and muzzle velocity of the rounds is approximately Mach 3.1. Taking this information and placing it into the following equation yields the damage rating of a single burst from the weapon:

$$\frac{\sqrt{calibre^3 * number-rounds * speed}}{10}$$

For the M61A1, this results in a value of 34 points per burst of 5 tracers. The F-16C Fighting Falcon carries an M61A1 and 500 rounds of ammunition, giving it 100 shots, or a total destructive ability of 3400 points by its gun.

Note that calibre is measured in millimeters, and speed is measured as a Mach number. To convert from feet per second (a common measurement for gun muzzle velocities) to a Mach number, divide the feet per second value by 1103.52.

Although the calculations used for the FLIGHT kinetic weapons treat WWI era weapons as Mach 1, WWII as Mach 2 and modern era as Mach 3, the actual speeds of the FLIGHT rounds are Mach 1.9, 2.4 and 3.1, respectively.

As a rough rule of thumb, armor piercing and high explosive rounds are awarded twice the normal damage rating. Note that the complex calculation above is fabricated. It has no significant basis in physical principles.

- *Warhead weapons* are those weapons which, quite simply, have an explosive warhead. The allotment of damage to such weapons is 5 points per pound of warhead. The M82 500 lb. bomb, therefore, is awarded 2500 points. Guided missiles are awarded damage only on the basis of their warhead, not from the kinetic energy of the missile itself.

Now that there are guidelines for the weapons, aircraft should undoubtedly be standardized tomatch these guidelines. The following is a suggested table for allotting damage to aircraft components.

Table 1-1 Aircraft Component Damage

Component	WWI	WW2	Modern
Wing	8	30	90
Control Surface	2	10	30/60
Pilot	2	2	2
Engine	4	10	30
Single Volume	8	39	90

The *Single Volume* row is for aircraft which do not want to create individual volumes for each of the named components. Designers are expected to use their discretion in following this table. For example, an aircraft may have very good survivability features (e.g. the A-10 Thunderbolt II) which would tend to increase these numbers.

FLIGHT recommends that for combat aircraft, only the named areas above are significant and that excessive numbers of volumes are not needed. Non-combat aircraft may very well use the single volume technique, or give the aircraft the UNDETECTABLE characteristic, meaning that they cannot be hit or successfully hit anything else.

1.3.3 Cockpit Description

The cockpit description of an object determines the appearance of the cockpit to the possibly multiple users involved in the simulation of the object you are designing. This description of laying out a cockpit will restrict itself to single user aircraft.

1.3.3.1 The DISPLAY

A DISPLAY is a collection of instrument panels and world viewports which are displayed on a single workstation. A DISPLAY is given a name. This name appears on the aircraft configuration dialog as the name of a crew station. See Section Subobjects, *Subobjects*, for a description of building an aircraft which displays on multiple workstations for multiple users.

The procedure for defining the cockpit of an object is to open a display (OPEN DISPLAY), create the viewports and instrument panels that appear on the display, then close the display (CLOSE DISPLAY).

It should be noted that although the designer may place multiple viewports or instrument panels into a display, the user may iconify any portion of those windows. When a window is iconified during a simulation run, the simulator stops expending CPU resources devoted to updating that window.

1.3.3.2 The VIEWPORT

A VIEWPORT is a rectangular region of the workstation screen which is used to display a view of the world. In general, this means a view from the cockpit of the aircraft. Up to 8 viewports may be created per display. Each viewport may be any size and location on the display. It is suggested that you limit your displays to having a single viewport because additional viewports can slow the simulator dramatically.

Each viewport may also be set up to face in a different direction relative to the current direction that the user is looking in. In a sense, you can create a cluster of cameras all bolted together by having multiple viewports which use the FACING qualifier to face in different directions. For example, the following example creates three viewports. One faces forwards (the default), and the others face left and right, respectively.

```
OPEN DISPLAY="Pilot"

VIEWPORT 10.0 5.0 3.0 15.0
VIEWPORT 10.0 5.0 13.0 15.0/FACING=LEFT
VIEWPORT 10.0 5.0 23.0 15.0/FACING=RIGHT

<instrument panel commands>

CLOSE DISPLAY
```

This will display three simultaneous views for the user. If the user is looking forward out of the cockpit, then the second viewport will be looking over the left wing and the third viewport will be looking over the right wing. If the user changes the view such that he is looking to the right (the pilot turns his head to the right), then the second viewport is looking over the nose of the aircraft and the third viewport is looking at the rear of the aircraft. This is why the camera cluster analogy is used.

By default, a viewport displays a 90 degree field of view. If a viewport is facing forward, it is subject to the current PILOTVIEW selected by the user. Because a PILOTVIEW may provide magnification, the 90 degree field of view may be reduced. However, magnifications only apply to a viewport which is bound to the forward direction (i.e. /FACING=FORWARD).

1.3.3.3 The PANEL

A PANEL is an actual instrument panel, completely analogous to a physical panel found in an aircraft. It, like a viewport, occupies a rectangular region of the display. Up to 8 instrument panels may be created per display. In general, there is no significant performance impact in having multiple instrument panels. However, it is again suggested that only one instrument panel be created if at all convenient.

An instrument panel may be placed and sized arbitrarily on a display. Also, each panel may have up to 100 instruments on it. Instruments are defined on a per hangar basis and must be defined prior to being added to an instrument panel. Instruments are referenced by their name in PLACE commands. Each such command occurs between an OPEN PANEL and CLOSE PANEL command pair and also specifies the positioning of the instrument on the instrument panel.

```
OPEN DISPLAY="Commander"

<viewport commands>

OPEN PANEL
```

```
PANEL 33.80 17.70 0.40 0.40

PLACE "F18_WING_FOLD_SWITCH" 32.00 5.00
PLACE "F18_HOOK_SWITCH" 30.40 5.00
PLACE "F18_FUEL_PROBE_SWITCH" 32.00 1.80
PLACE "F18_LADDER_SWITCH" 30.40 1.80

CLOSE PANEL

CLOSE DISPLAY
```

All measurements are in centimeters. The position of an instrument panel is for the lower left corner of the panel relative to the lower left corner of the display. The position of an instrument is for the center of the instrument relative to the lower left corner of the instrument panel. The FLIGHT Hangar Editor provides a tool to allow interactive placement of instruments.

1.3.3.4 Defining Instruments

The following list shows the available types of instruments:

- Dial
- Radio
- Radar
- Control
- Digital
- Artificial Horizon
- Turn & Slip Indicator
- Slider
- ADF
- VOR
- Lamp
- Toggle

Each instrument is introduced by a command of the form

```
OPEN INSTRUMENT=type="name"
```

where *type* is ADF, VOR, DIAL, etc, and *name* is a name that you give the instrument. Once opened, a single command verb may be used to describe the opened instrument. That verb is identical to the type of the instrument. For example, dial instruments use the DIAL command to define them:

```
DIAL/RADIUS=3.0
DIAL/HAND=(0,82,2,1,0)
```

Once the instrument has been described, it is closed by using the CLOSE INSTRUMENT command. See the description of the individual instrument type commands for details on what can and cannot be specified in an instrument design.

1.3.4 Performance Description

The performance description of an object determines how it behaves in the simulated world. An object may exist without geometry, a cockpit or volumes, but the performance information for the object is required.

1.3.4.1 FUEL Storage and Consumption

The information required for fuel is the total amount that the aircraft can carry, in pounds, and the amount of fuel that it consumes per second at full power, also in pounds. See the FUEL command for details.

When an engine is afterburning, it consumes fuel at three times the full power rate.

Note that FLIGHT does not support multiple fuel sources, nor does it support modifying the fuel consumption rate due to damage.

The user has control over what percentage of the total fuel load should be carried at the beginning of a simulation. When an object is used as a weapon and released from the parent object, it always carries full fuel.

1.3.4.2 THRUST Generation

You must specify the amount of power that the object's engine (if any) produces. This includes the maximum power of the engines, the afterburning power of the engines, and the throttle increment available to the user. All values are specified in pounds of thrust.

In addition to the raw engine output, you may also specify the rate at which the engine responds to changes in throttle settings, the engine type, and the altitude at which the engine cannot maintain function. The last is called the flameout altitude. When the object reaches that altitude, the engine will shut down.

FLIGHT differentiates between in-air, on-terrain (land), and on-water engine performance. Be sure to set the appropriate characteristic(s) on the CHARACTERISTICS command corresponding to the regime(s) that your object is capable of operating in.

~~In order for the object to have an afterburning capability, not only must the afterburning thrust value be supplied, but the characteristic of AFTERBURNER must be specified. See the CHARACTERISTICS command for a description of this attribute.~~

Note that FLIGHT does not support multiple engines. If an aircraft has 4 engines producing 3,000 pounds of thrust each, the FLIGHT model must treat that as a single engine with 12,000 pounds of thrust. However, there are damage effects which may be assigned to object volumes causing partial loss of power. See Section 1.3.2 for a description of object volumes.

1.3.4.3 Brakes

There are two values specified for the braking ability of an aircraft: the maximum braking power and the braking increment available to the user. The braking power is specified as the effective thrust output which the brakes appear to provide. In general, specifying a maximum brake output equal to the maximum power of the aircraft's engine is a good start.

The braking capabilities of an object are specified via the BRAKE command.

1.3.4.4 Flaps

Flaps are control surfaces that augment both the lift and drag of the object. You are able to specify the maximum amount of flaps that may be lowered and the increment available to the user when lowering flaps.

```
FLAPS/MAX=45
FLAPS/INCREMENT=15
```

In the above example, a maximum of 45 degrees of flaps may be lowered by the user, and in 15 degree steps.

~~In order for an object to have flaps, not only must the flaps values be supplied, but the characteristic of FLAPS must be specified. See the CHARACTERISTICS command for a description of this attribute.~~

1.3.4.5 Control Rates

Control rates are explicitly specified by the designer in approximate degrees per second of roll, pitch and yaw. Roll is rotation along the object X axis, pitch is rotation along the object Z axis and yaw is rotation along the object Y axis.

The rates are attenuated for low speeds. The *control speed* of an aircraft is the speed at and above which the full roll, yaw and pitch rates of the aircraft are in effect. Below that speed, the rates are linearly scaled down. Control speeds are specified in knots.

For example, if an aircraft's control speed is 150 knots, then at 75 knots, the control rates would be at approximately half of their maximums for the aircraft. At an airspeed of zero, the object will not roll, pitch or yaw in response to pilot controls.

Some aircraft have small thrusters positioned at selected points on the aircraft to allow them to maneuver when not normally possible due to low airspeeds or being in a vacuum.

~~To provide for this capability, FLIGHT allows for specification of the characteristic of REACTION_CONTROL. See the CHARACTERISTICS command for a description of this attribute.~~

To provide for this capability, FLIGHT allows you to specify the capabilities of the aircraft via the REACTION_CONTROL command indepenantly of the AERODYNAMICS command. The latter controls the aircraft's performance as a function of normal flight. The REACTION_CONTROL command allows you to specify the parameters of the thruster-based control system.

Aircraft with a reaction control system (RCS) are able to maintain constant roll, pitch and yaw rates regardless of air speed so long as the aircraft's air speed remains below the control speed specified by the REACTION_CONTROL command.

1.3.4.6 Weight

The only value which the designer needs to specify is the empty weight of the object. FLIGHT computes gross weights from the empty weight, the fuel carried and any objects loaded into the weapon bays. Therefore, you should include all other weights into the aircraft weight value, including pilot weight, etc.

1.3.4.7 Aerodynamics

A significant component of how an aircraft performs is embodied in four components:

- The *wing areas* of the object.
- The *lift curve* of the object.
- The *side lift curve* of the object.
- The *drag curve* of the object.

The wing areas of concern are those of the actual aircraft wing, and that of the fuselage. The fuselage area may be thought of as the effective lifting area of the body of an aircraft as it flies such that the wings are vertical. Both values are specified in square feet.

The lift and drag curves are a bit more complicated to describe and for you to specify to FLIGHT.

Lift is the force which cause the aircraft to resist gravity and is generated by airflow across the wing. The lift curve specifies coefficients (unitless values) which are fed into the simulator equations to determine how much lift is being generated by the wing. Therefore, one axis of the curve is the lift coefficient. The other axis is the angle at which the air is striking the wing. This is known as the *angle of attack* or AOA.

Drag is the force which is caused by the aircraft's forward movement through the airflow. It resists that forward movement, attempting to slow the aircraft. The drag curve specifies coefficients as the lift curve does, except that the drag coefficients apply to the computations surrounding the effects of drag.

Each of these curves is specified for angles of attack of -90 through 90 degrees. This gives FLIGHT 181 values for each curve. These curves are 'real' curves in that these curves are actually computed for all airfoils. The airfoils are assigned an NACA[1] number which may then be referenced by aircraft designers and manufacturers. For example, the F-14A Tomcat has an airfoil section number of NACA 64 A2. If you are able to locate a source for NACA airfoil information, you will be able to supply those curves to FLIGHT

[1] *NACA stands for National Advisory Committee for Aeronautics and was one of the forerunners of NASA, the National Aeronautics and Space Administration.*

and immediately start tuning your aircraft with that information as a ’known’.

The side lift curve is just like a lift curve except that it specifies lift generated by the side of the aircraft fuselage as a function of the angle of attack on the side of the fuselage. This lift curve is optional, with FLIGHT providing a set of reasonable default values.

If you do not have accurate numbers for an airfoil, you are faced with either using the closest airfoil from the aircraft supplied with FLIGHT, or with manufacturing your own through trial and error. Unless you are familiar the physics of aerodynamics, we suggest the first approach.

1.3.5 Structural Definition

Some information about the mechanical nature of the object must be supplied to FLIGHT, in order for the proper aerodynamic and mechanical behavior to be simulated. This is specified via the STRUCTURE command.

1.3.5.1 Wing Structure

At a minimum, the span (total left-to-right extent) and chord (total front-to-back extent) of the wing must be specified.

The span is specified via the STRUCTURE WING/SPAN=n command, where “n” is the wing span in feet.

The chord is broken into two metrics: root chord (chord as measured where the wing joins the fuselage) and tip chord (chord as measured at the outter tip of the wing, not counting any final rounding or pinching off). Root chord is specified via the STRUCTURE WING/ROOT_CHORD=n1 statement, and tip chord is specified via the STRUCTURE WING/TIP_CHORD=n2 statement. “n1” and “n2” are in feet.

Optionally, the strength of the wing may be specified via the STRUCTURE WING/GLOAD=g command, where “g” is measured in Earth standard G’s.

1.3.5.2 Fuselage Structure

At a minimum, the fuselage length (total nose-to-tail extent), radius, and ground pivot point must be specified.

The length is specified via the STRUCTURE FUSELAGE/LENGTH=n command, where “n” is in feet.

The radius, which represents the average radius as measured along the whole extent of the fuselage, is specified via the STRUCTURE FUSELAGE/RADIUS=n command, where “n” is in feet.

The ground pivot point is specified via the STRUCTURE FUSELAGE/PIVOT_POINT=n command. “n" represents the location (in feet, in the X axis relative to the object’s origin) of the point at which the object pivots in pitch while on a surface. Normally this is the location of the aircraft’s main gear. A typical tricycle-gear aircraft would have a negative value (because the main gear are behind the center of gravity) such as -3.0 feet. A typical tail-dragger aircraft would have a positive value such as 2.0 feet.

Optionally, the rest angle of the object may be specified via the STRUCTURE FUSELAGE/REST_ANGLE=d command. “d” represents the pitch angle (in degrees) at which the object should sit when stopped on a surface, relative to the surface. A value of 0 causes the object to be level (parallel to the terrain or water surface) when stopped. Negative values result in a nose-up/tail-down stance.

1.3.6 Miscellaneous

There are a variety of other aspects of design which do not directly relate to any specific area.

1.3.6.1 Restricting Object Lifetimes

This is intended as a performance measure. Through the use of the LIFETIME command, you can specify how long the object may survive after it is created. For example, a tracer round should not be allowed to race out, miss its target at ten thousand feet and then spend the next 5 or 10 minutes falling to the ground. Setting its LIFETIME attribute to 3 or 4 seconds will greatly aid in keeping the simulator responding well even during furious combat.

Related to this optimization is that of the /TERMINATION qualifier on the LIFETIME command. It allows you to specify whether the object should exit the simulation by exploding or merely vanishing when the object’s LIFETIME expires.

1.3.6.2 Radar Images

Objects may carry instrumentation to display radar information to the pilot. Whether or not a particular radar set detects a particular object is determined by the object's radar signature. The radar signature is defined through the SIGNATURE RADAR command, which specified the radar cross section of the object in three orthogonal axes: X, Y and Z.

Note that radar signatures are also significant to radar guided weapons.

1.3.6.3 Signing Your Name

Aircraft designs typically take upwards of several hundred hours to faithfully reproduce the object so that an enthusiast can keep interest in flying that model for many hours. That's a lot of work, and FLIGHT likes to give credit to its designers. The DESIGNER command allows you to do this. It allows you to specify your name and when you did the work. When FLIGHT presents a list of objects to the user to begin a simulation with, the designers of each of those aircraft appear alongside the object name.

1.3.6.4 Loading Weapons

A weapon can be any object in a hangar. In truth, FLIGHT does not have knowledge of weapons *per se*, but allows for a general notion of detachable objects. The weapon approach is used because that is the most common scenario for these detachable objects.

Weapons are loaded into one of four banks, with each bank holding from zero to 20 weapon loads. Each load of a weapon may specify a repeat count, causing the individual load to explicitly add many copies of the specified weapon. Once a simulation starts, the weapons are made available, but are not visible.

The following example places 200 tracers and 4 bombs into bank 2.

```
WEAPON/BANK=2 "Tracers" 5.0 1.0 1.0/REPEAT=200
WEAPON/BANK=2 "Bomb" 0.0 -1.0 5.0
WEAPON/BANK=2 "Bomb" 0.0 -1.0 -5.0
WEAPON/BANK=2 "Bomb" 0.0 -1.0 5.5
WEAPON/BANK=2 "Bomb" 0.0 -1.0 -5.5
```

The bombs are individually placed at particular release points on the aircraft, while all 200 tracers are to be fired from the same location. This set of commands consumes 5 of the possible 20 weapon loads in bank 2.

When a weapon is released (fired), the weapon's full engine power is applied, and its control surfaces are set in a neutral position to provide for straight flight. If the object has displays defined and the user has requested that they be displayed, FLIGHT attempts to draw them on the appropriate workstations. Such displays might be present for a reconnaissance drone or for flying an experimental aircraft such as the X-15, which was dropped from the wing of a B-52 bomber.

Because objects may be guided (see the GUIDANCE commands), they may be able to guide themselves to a target, providing an exception to the neutral controls statement above.

The orientation of a weapon release is forward by default. However, the placement of a weapon release is set individually for each weapon placed in a weapon bank. This allows for bombs to fall from the fuselage, or missiles to fire from under wings, etc. The placement of weapons is in the object's coordinate system and works just like the object geometry: forward is positive X, up is positive Y, and right is positive Z. All units are in feet.

1.4 Advanced Design

The previous section discussed the basics of object design. However, there is a broad range of capabilities which have not been discussed.

1.4.1 Subobjects

Any discussion of advanced geometry modeling in FLIGHT must begin with a discussion of subobjects, for they are the key to the most dynamic aspects of object modeling.

A subobject is much the same as the geometric and volumetric description of an object. Note that a subobject does not have performance information, nor does it have cockpit information. The geometric and volumetric information is described exactly as in a normal object, except that they are bounded not by OBJECT open and close operations, but by SUBOBJECT open and close operations.

A subobject is defined in its own coordinate system, just as a root object is. The root object responds to user control inputs that affect roll, pitch and yaw. Subobjects are far more flexible, allowing for rotations and translations in response to not only user actions such as moving the control stick, but also in response to simulation parameters such as air speed and altitude.

A subobject may not exist by itself. It must have a parent object.

1.4.1.1 POSITIONING and Data Channel Values

A *data channel* is a parameter of a simulation at a given time. For example, airspeed is a data channel. The airspeed data channel is constantly updated to reflect the current simulated object's forward air speed in knots through the airflow. Table 2.4 lists all the data channels supported by FLIGHT.

A data channel is very significant to discussions of subobjects because all subobjects are bound to some data channel. It is the data channel that drives all movement of subobjects during a simulation.

Specification of a subobject's data channel is performed by use of the SUBOBJECT command. Once the data channel is supplied, between one and five POSITIONING commands are specified. Each positioning command establishes a relationship between a data channel value and a subobject position. When the subobject's data channel reaches the specified value, FLIGHT will place the subobject in the location and at the orientation specified by the POSITIONING command.

If one POSITIONING command is supplied, then the subobject will always occupy the single position specified by that command.

If multiple POSITIONING commands are supplied, then FLIGHT will order them according to the their data channel values. As the data channel changes its value, FLIGHT will determine how the subobject should be positioned. If the data channel value is between positions, FLIGHT will linearly interpolate to determine the intermediate position. That is, a data channel value which is half way between two design positions will cause the subobject to be positioned half way between the rotations and translations specified for the two design positions that the data channel value is between.

If the data channel value is greater than the largest value to which the subobject is bound, the position of that greatest value is used. Similarly, if the data channel value is less than the smallest value to which the subobject is bound, the position of that least value is used.

FLIGHT performs an optimization such that between any two design positions, FLIGHT will only change the subobject's position if the data channel value changes by a certain amount. By default, this change is 5 percent of the difference in the data channel values of the two design positions. You can change this through use of the SUBOBJECT/POSITIONS command.

1.4.1.2 Specifying Positioning Information

As stated above, the positions of a subobject are supplied through the use of a POSITIONING command. These commands allow specification of an X-Y-Z position and rotations in each axis. The position being specified is the placement of the subobject origin.

1.4.2 Vectorable Thrust

Normally, an aircraft's engines are mounted on the aircraft such that the engine thrust is sent straight back. Therefore, that is the default in FLIGHT. You merely specify the attributes of the engines via the THRUST command and FLIGHT assume that the engines are mounted in a conventional manner.

More exotic aircraft sometimes have the ability to change the direction that the engine thrust goes in. Examples of this are the AV-8B Harrier II, the V-22 Osprey and any number of helicopters, including the Bell 206B JetRanger III. The process of directing the engine thrust in different directions is called *thrust vectoring*.

Like most advanced modeling capabilities, thrust vectoring is accomplished through subobjects. The task of vectoring thrust becomes trivial once subobjects have been mastered. All that needs to be done is to include the command

```
THRUST/BINDING="subobject-name"
```

The named subobject will control where the engine thrust is sent. The thrust is always trying to move the subobject in its +X direction, meaning that the engine thrust is being pushed out the -X end of the subobject.

In the case of the Harrier, the engine thrust is bound to one of the nozzles which rotate forwards and backwards in response to control inputs by the pilot. Because subobjects can be moved in response to any number of different simulation parameters (including user input), this means that engine thrust can be delivered in response to just about anything.

When vectoring thrust, don't worry about having the engine output flipping over the aircraft because the subobject you specify isn't near the center of gravity of the aircraft. FLIGHT always assumes that the engine always thrusts through the center of mass of the aircraft.

Note that if the subobject that the aircraft's thrust is bound to is destroyed, the engine is assumed to be non-functional.

1.4.3 Directed Weapon Release

When weapons are placed onto an aircraft using the WEAPONS command, the only required information is the name of the weapon and where on the aircraft the weapon launches from. Optionally, you can specify a bank. Also optional is the name of a subobject which the weapon should launch from. By default, the object that the weapon is launched from is the aircraft itself. By the use of the WEAPONS/BINDING qualifier, you can tell FLIGHT to launch the weapon as if it were mounted to, or fired through the named subobject.

For example,

```
WEAPON "Tracers" 1.0 0.0 0.0/BINDING="Turret Gun"
```

might be the command used to say that this particular tracer burst should be fired according to where the subobject called "Turret Gun" is pointed. The position information of 1.0, 0.0, 0.0 indicates that the tracer burst should actually originate 1.0 feet in front of the subobject's origin in the X axis.

Note that if the subobject that the weapon is bound to is destroyed, the weapon is considered destroyed and it will not fire when it is next to be fired from its weapon bank.

1.4.4 Crew Aircraft

FLIGHT provides the designer with the ability to create multiple *displays* (a maximum of 4). Each of these displays is a complete crew station with instruments and/or viewports. See Section 1.3.3 for a description of the construction of displays.

Because multiple displays can be created, FLIGHT allows multiple workstations to be used to fly a single aircraft (or man a single ground vehicle, etc). The designer's task is merely to describe the displays. This is done by opening and then closing multiple displays within an object, just as is done for a single cockpit description.

```
OPEN DISPLAY="Pilot"
<display-commands>
CLOSE DISPLAY
OPEN DISPLAY="Copilot"
<display-commands>
CLOSE DISPLAY
```

In this example, two displays are being created. The first is named "Pilot" and the second is called "Copilot". The names are significant because they are presented to the user in the Configure Aircraft dialog box. Therefore, you should use descriptive names.

Once created, it is the user's responsibility to request multiple crew stations. The first display is always created on the workstation that the application is started from. But the user may request that additional displays that you have defined should be presented on a workstation.

You should keep this in mind when you design your displays. If you put the start switch on the third display's instrument panel, then the user won't have any way of starting the aircraft's engines unless the third display is presented somewhere.

Note that you can create empty displays as placeholders so that you can define, for example, only the third display of an object. This would be useful for a weapon for which you wanted to provide the option of remote piloting.

```
OPEN DISPLAY="Unused"
CLOSE DISPLAY
OPEN DISPLAY="Remote Pilot Console"
<display-commands>
CLOSE DISPLAY
```

In this example, the first display is empty and the second has the real cockpit for flying the weapon. When the weapon is launched, the first display will be skipped, and the second display will check to see if the user asked for the second displays of objects to go anywhere. If the user specified a workstation, then the remote piloting console would appear on that workstation. If not, then the weapon would be left to fly however it would normally. This would be a useful technique for a weapon such as the Cruise Missile or other similar weapons. Conceptually, the user sitting at the second workstation is the aircraft Weapons Officer.

1.4.5 Head-Up Displays

Head-up displays (HUDs) can be accomplished by using lots of subobjects. Each of these subobjects can respond to some data channel (a simulation parameter) and move or rotate as appropriate. All that needs to be done to make them seem to work like a HUD is to make sure that you put the subobjects somewhere in front of the PILOTVIEW used for the aircraft pilot so that the pilot can see it. The F/A-18 Hornet uses this capability very well.

One drawback to this approach of using geometry to implement a HUD is that you must have the entire aircraft geometry displayed in order to see just the HUD. This is accomplished by pressing PF3-KP7 during a simulation. But it is a drawback because drawing all the geometry of an object may be distracting and it will definitely slow the simulation by some amount.

In order to alleviate this, you can use FLIGHT's notion of subobject *groups*. A group is a set of subobjects which can affect each other's visibility. Normally, the root object (the aircraft) and all subobjects are in group 1. So when the aircraft is visible, the subobjects that want to be visible can be and those that don't want to be can be invisible. But when the aircraft is invisible (which is what happens locally on your workstation when you use KP7 to view the simulation from a PILOTVIEW), all the subobjects will unconditionally become invisible because they are in an invisible object.

Groups can change all that. By placing the HUD subobjects in a separate group, you can control their visibility independant of the aircraft. If the subobjects are in a separate group and are visible, then when the aircraft goes invisible, the subobjects in that separate group will ignore the aircraft visibility and continue to be visible. This is exactly what we want for the HUD. To place a subobject into a particular group, use the SUBOBJECT/GROUP command inside each subobject of your HUD.

```
SUBOBJECT/GROUP=2
```

So design your HUD using subobjects which respond to the appropriate data channel values (such as airspeed, altitude, angle of attack, etc), position them so that they are visually correct from the pilot's PILOTVIEW, and place them in a separate *group*.

Now you've got a piece of geometry, your HUD, which is visible regardless of whether or not the aircraft is visible. This may not be all that desireable, either. We'd like the option of getting rid of the HUD when we don't want it cluttering the display. One simple applicationof subobjects will solve our problem. Merely place all of your HUD subobjects within another subobject (which is also in your HUD group). Then bind that enclosing subobject to the WENABLED data channel such that when the data channel is 0.0, the subobject is invisible, and when the data channel is 1.0, the subobject is visible. This will result in having the HUD objects visible when the weapons system is enabled, and invisible otherwise.

If you were to leave the enclosing object out of the HUD group, then it would become visible and invisible as desired, but it would have no effect on the subobjects within it because they are in a different object group - the HUD group.

2 The Hangar Compiler

The hangar compiler is the processor which converts hangar source files into the binary form which the FLIGHT simulator can read. The hangar compiler may only be invoked as a batch processor. That is, there is a single command which invokes the compiler and then it runs to completion without any user interaction.

2.1 DCL Command

The following describes the invocation of the hangar compiler.

FLIGHT/COMPILE=HANGAR

Invokes FLIGHT's hangar compiler.

FORMAT

FLIGHT/COMPILE=HANGARinput-filespec

Qualifiers	Defaults
/(NO)LISTING(=filespec)	*/NOLISTING*
/(NO)OUTPUT(=filespec)	*/OUTPUT=*.hangar*
/(NO)LOG	*/NOLOG*
/(NO)COMPRESS	*/COMPRESS*
/(NO)PILOTABLE	*/PILOTABLE*
/(NO)PRELOAD=(filespec, ...)	*/NOPRELOAD*
/DEFINE=(name, (name,...))	*None*

PARAMETERS

input-filespec

Specifies the input file to be compiled. Note that wildcards are not allowed in the input file specification. The default file type for this parameter is DAT.

DESCRIPTION

The FLIGHT/COMPILE=HANGAR command starts FLIGHT's hangar compiler. See Section 2.2 for a description of the commands that are used in hangar source files to describe the contents of a hangar.

As the hangar source file is compiled, errors may be encountered. FLIGHT will list the source line containing the error along with an informative message about the error to the terminal. If a listing file was requested via the LISTING qualifier, then the informative message is placed there as well.

QUALIFIERS

/(NO)LISTING(=filespec)

Specifies whether or not the compiler creates a listing file. If you omit the file specification, the compiler defaults to the name of the source file and a file type of LIS. The default is NOLISTING.

/(NO)OUTPUT(=filespec)

Specifies whether or not the compiler creates an output file containing the compiled hangar. If you omit the file specification, the compiler defaults to the name of the source file and a file type of hangar. The default is OUTPUT=<inputfile>.hangar.

Specifying NOOUTPUT can be useful to ensure that a hangar compiles correctly, but you do not want to use the disk space required to store the compiled hangar file.

/(NO)LOG

Specifies whether or not the compiler outputs an informational message whenever it encounters the beginning of a hangar object or instrument definition. This may be used to watch the progress of the compiler as it compiles each object in your hangar sources.

/(NO)COMPRESS

Specifies whether or not the compiler compresses the hangar before writing it to disk. Compression consists of elimination of unreferenced instruments, colors and weapons in the hangar. For example, a set of instruments may be included in the compilation, but only a few of them are actually used. The compression capability eliminates the unused instruments in order to make the hangar file as small as possible. The default is COMPRESS.

/(NO)PILOTABLE

Specifies whether or not the compiler checks to see if a pilotable object would be inthe hangar written to disk. If PILOTABLE is specified, the compilation will fail and no output file will be produced if no pilotable object can be found. Pilotable objects are those objects which have the PILOTABLE characteristic. The default is PILOTABLE.

/(NO)PRELOAD=(filespec, ...)

Specifies whether or not the compiler loads the specified hangars before beginning the compilation of the hangar source file. This feature can be useful for preloading of collections of instruments or weapons to avoid having to repeatedly compile the utilitarian objects. The default is NOPRELOAD.

/DEFINE=(name, (name, ...))

Specifies the set of symbol names which should be considered as defined for the duration of this compilation. This has the same effect as if "_DEFINE name" directives were placed into the beginning of the source file.

2.2 Compiler Directives

The following section provides complete descriptions of a set of directives which may be specified in a source file to control the behavior of the compiler.

All compiler directives begin with an underscore (_).

_DEFINE

Directs the compiler to define a symbol

FORMAT

_DEFINE *symbol*

Qualifiers	**Defaults**
None.	*None.*

PARAMETERS

symbol

Specifies the name of the symbol to define. Symbol names are case sensitive, but if the symbol name is not in quotes, it is converted to uppercase.

DESCRIPTION

Use the _DEFINE directive to tell the compiler that a particular symbol name should be defined as valid. Once defined, the _IFDEFINED and _IFNOTDEFINED directives may be used to conditionally compile sections of source code.

Up to 16 symbols may be defined during a single compilation. Symbols definitions may not be removed or negated in any way.

QUALIFIERS

None

_IFDEFINED

Directs the compiler to conditionally compile source lines based on the existence of a symbol definition.

FORMAT

_IFDEFINED *symbol*

Qualifiers	**Defaults**
None.	*None.*

PARAMETERS

symbol

Specifies the name of the symbol to check. If the symbol is defined, then the lines following the _IFDEFINED directive and up to the matching _ENDIF directive will be compiled. If not defined, then those lines will be ignored.

DESCRIPTION

Use the _IFDEFINED directive to tell the compiler that lines between this directive and its matching _ENDIF directive should be compiled if the specified symbol name is defined.

QUALIFIERS

None

EXAMPLES

```
_DEFINE BOMB

.
.
.
OPEN OBJECT="Mk IX Spitfire"
.
.
.
_IFDEFINED BOMB
WEAPON ... ! Load bombs
```

```
WEAPON ... ! Load bombs
WEAPON ... ! Load bombs
_ENDIF

_IFNOTDEFINED BOMB
WEAPON ... !  Load tracers
WEAPON ... !  Load tracers
_ENDIF
.
.
.
CLOSE OBJECT
```

This example shows a _DEFINE directive causing the compiler to consider the SOLID symbol defined. Later in the source, an object conditionally compiles a VOLUME command if the SOLID symbol is defined. Due to the presence of the _DEFINE directive, that symbol is defined.

A more common scenario is to use the /DEFINE qualifier on the FLIGHT/COMPILE command. This allows for selection of the form of the object that is desired at the command line; no editing of the source file is needed in order to change the outcome of a particular compilation.

_IFNOTDEFINED

Directs the compiler to conditionally compile source lines based on the absence of a symbol definition.

FORMAT

_IFNOTDEFINED *symbol*

Qualifiers	**Defaults**
None.	*None.*

PARAMETERS

symbol

Specifies the name of the symbol to check. If the symbol is not defined, then the lines following the _IFNOTDEFINED directive and up to the matching _ENDIF directive will be compiled. If defined, then those lines will be ignored.

DESCRIPTION

Use the _IFNOTDEFINED directive to tell the compiler that lines between this directive and its matching _ENDIF directive should not be compiled if the specified symbol name is defined.

QUALIFIERS

None

_ENDIF

Directs the compiler to end the processing effect produced by a matching _IFDEFINED or _IFNOTDEFINED directive.

FORMAT

_ENDIF

Qualifiers	Defaults
None.	*None.*

PARAMETERS

none

DESCRIPTION

Use the _ENDIF directive to end the processing effect produced by a matching _IFDEFINED or _IFNOTDEFINED directive. Note that these directives may be nested and that an _ENDIF directive is required for each _IFDEFINED or _IFNOTDEFINED directive specified.

QUALIFIERS

None

2.3 Hangar Commands

The following section provides complete descriptions of all the source commands used to describe a hangar.

ADF

Specifies a component of an ADF-type instrument definition within the confines of OPEN INSTRUMENT and CLOSE INSTRUMENT. The ADF command is not valid for non-ADF instruments.

FORMAT

ADF

Qualifiers	Defaults
/BIGTICKS=number	*None.*
/BUTTON=(xpos, ypos, width, height, fill, label, xscale, yscale)	*None.*
/C_LABEL=(title, xpos, ypos, xscale, yscale)	*None.*
/CONSOLE=(xpos, ypos, width, height)	*None.*
/DECIMALS=decimals	*None.*
/DIGITS=digit-count	*None.*
/DIVISOR=value	*None.*
/FREQUENCY_READOUT=(xpos, ypos, xscale, yscale)	*None.*
/I_LABEL=(title, position, radius, xscale, yscale)	*None.*
/IN_TITLE=(title, yposition, xscale, yscale)	*None.*
/INITIAL_FREQUENCY=freq-in-Hz	*None.*
/KNOB=(xpos, ypos, radius, fill, label, xscale, yscale)	*None.*
/MAXFREQUENCY=freq-in-Hz	*None.*
/MINFREQUENCY=freq-in-Hz	*None.*
/RADIUS=size-in-cm	*None.*
/SID_READOUT=(xpos, ypos, xscale, yscale)	*None.*
/SMLTICKS=number	*None.*
/TICKS=(number, start, end, minradius, maxradius, width)	*None.*

PARAMETERS

None

QUALIFIERS

/BIGTICKS=number

Specifies the number of large tick marks drawn around the inside of the ADF indicator. The number of tick marks are spread evenly through the full 360 degree sweep of the dial.

See the /TICKS qualifier for a description of how to explicitly control the appearance of ticks.

/BUTTON=(xpos, ypos, width, height, fill, label, xscale, yscale)

Adds a button graphic to the ADF console. The button is purely for aesthetics; it is non-functional. You may specify up to 10 buttons per ADF.

'xpos' and 'ypos' specify the location of the center of the button. 'width' and 'height' specify the size of the button image. 'fill' controls how the interior of the button is drawn; the value is interpreted as a gray-level percentage, where 0 is empty and 100 is solid filled. 'label' specifies a text string which is drawn centered above the button (up to six characters). The text character size is specified by the two scale values. The magnitude of all dimension values is based on 100 units being equal to the console's height. (If the scale values are omitted, the label is drawn at a standard, fixed size.)

/C_LABEL=(title, xpos, ypos, xscale, yscale)

Specifies the position, size, and title of a label positioned on the interior of the ADF console. 'xpos' and 'ypos' specify the center of the label. The text character size is specified by the two scale values. The magnitude of all dimension values is based on 100 units being equal to the console's height. (If the scale values are omitted, the label is drawn at a standard, fixed size.)

Up to 10 labels may be placed on a single ADF console, but each must be placed by a separate ADF/C_LABEL command.

/CONSOLE=(xpos, ypos, width, height)

Specifies where and how large the console is. The position values specify where the center of the console is relative to the center of the indicator. The width/height values specify the console's size. All four values are specified in centimeters.

/DECIMALS=decimal-digit-count

Specifies how many decimal digits to the right of the decimal point should be displayed in the console frequency readout. A value of 0 omits the decimal point altogether.

/DIGITS=digit-count

Specifies how many total digits are displayed on the console frequency readout. This count should include digits to the left of the decimal point, the decimal point (if present), and digits to the right of the decimal point.

/DIVISOR=value

The ADF console provides a digital readout of the frequency that the unit is currently tuned to. The actual number that is displayed is computed by dividing the frequency in Hertz by the DIVISOR value. For example, DIVISOR=1000 would make the display be in kHz.

/FREQUENCY_READOUT=(xpos, ypos, xscale, yscale)

Specifies where the frequency readout is located within the ADF console, and how large it is. 'xpos' and 'ypos' determine the center of the readout. The readout text character size is specified by the two scale values. The magnitude of all four values is based on 100 units being equal to the console's height. (If the scale values are omitted, the text is drawn at a standard, fixed size.)

/I_LABEL=(title, position, radius, xscale, yscale)

Specifies the position, size, and title of a label positioned around the circumference of the ADF indicator. The angular position is expressed in degrees. The 'radius' value controls where the text is written, in terms of distance out from the dial center. A magnitude of 100 equals the radius of the dial face; values larger than 100 may be specified. The title string is centered at that point. The text character size is specified by the two scale values. The magnitude of the size values is based on 100 units being equal to the dial's radius. (If the scale values are omitted, the label is drawn at a standard, fixed size.)

Up to 40 labels may be placed on a single ADF indicator, but each must be placed by a separate ADF/I_LABEL command.

/IN_TITLE=(title, yposition, xscale, yscale)

Specifies the position and text of a title located in the inside of the ADF indicator. The text is drawn with characters sized as specified by the two scale values, and centered horizontally. The vertical position is specified by the second parameter (which may be negative). The magnitude of the numeric values is based on 100 units being equal to the dial's radius. (If the scale values are omitted, the title is drawn at a standard fixed size.)

For example, IN_TITLE=(ADF, 30, 50, 50) would put the string "ADF" above the dial center, 30% up toward the dial edge, with square proportioned letters that were half the radius tall and wide. The string would be noticibly larger.

Up to 20 interior titles may be placed on a single ADF indicator, but each must be placed by a separate ADF/IN_TITLE command.

/INITIAL_FREQUENCY=freq-in-Hz

Controls what frequency the ADF console is tuned to when you first enter the object. The value in specified in hertz.

/KNOB=(xpos, ypos, radius, fill, label, xscale, yscale)

Adds a knob graphic to the ADF console. The knob is purely for aesthetics; it is non-functional. You may specify up to 10 knobs per ADF.

'xpos' and 'ypos' specify the location of the center of the knob. 'radius' specifies the radial dimension of the knob image. 'fill' controls how the interior of the knob is drawn; the value is interpreted as a gray-level percentage, where 0 is empty and 100 is solid filled. 'label' specifies a text string which is drawn centered below the knob (up to six characters). The text character size is specified by the two scale values. The magnitude of all dimension values is based on 100 units being equal to the console's height. (If the scale values are omitted, the label is drawn at a standard, fixed size.)

/MAXFREQUENCY=freq-in-Hz

Specifies the highest frequency to which the ADF console may be tuned. Attempts to tune to higher frequencies will be ignored. The value is specified in Hertz.

/MINFREQUENCY=freq-in-Hz

Specifies the lowest frequency to which the ADF console may be tuned. Attempts to tune to lower frequencies will be ignored. The value is specified in Hertz.

/RADIUS=size-in-cm

Specifies the radial dimension of an ADF indicator instrument in centimeters.

/SID_READOUT=(xpos, ypos, xscale, yscale)

Specifies where the station ID readout is located within the ADF console, and how large it is. 'xpos' and 'ypos' determine the center of the readout. The readout text character size is specified by the two scale values. The magnitude of all four values is based on 100 units being equal to the console's height. (If the scale values are omitted, the text is drawn at a standard, fixed size.)

When the ADF is picking up the signal from some NDB, the station ID readout displays that NDB's three-letter callsign. When the ADF is not picking up any signal, "OFF" is displayed.

/SMLTICKS=number

Specifies the number of small tick marks drawn around the inside of the ADF indicator. The number of tick marks are spread evenly through the full 360 degree sweep of the dial.

See the /TICKS qualifier for a description of how to explicitly control the appearance of ticks.

/TICKS=(number, start, end, minradius, maxradius, width)

Places a generalized set of ticks around the ADF indicator's face. 'Number' controls how many tick marks there are in the set. 'Start' and 'end' control the angular region where the marks are drawn (specified in degrees). The tick marks are evenly distributed in this region, inclusive of ends (that is, the first tick is located at 'start' and the last at 'end'). The two radial parameters control how long the tick is (100 tick radius units equals the radius of the indicator face). 'Width' controls how wide the tick marks are (same units as the radius parameters).

Up to 9 tick sets may be specified per ADF. The BIGTICKS and SMLTICKS qualifiers may be used to construct simple cases of ticks.

AERODYNAMICS

Specifies aerodynamics information about an object.

FORMAT

AERODYNAMICS

Qualifiers	Defaults
/ROLL_RATE=degrees-per-second	*None.*
/PITCH_RATE=degrees-per-second	*None.*
/YAW_RATE=degrees-per-second	*None.*
/AREA=(square-feet,square-feet)	*None.*
/CONTROL=knots	*None.*
/LIFT=(lift-coefficients)	*None.*
/S_LIFT=(lift-coefficients)	*A general-purpose curve.*
/DRAG=(drag-coefficients)	*None.*

PARAMETERS

None

DESCRIPTION

The AERODYNAMICS command specifies aerodynamics information about the currently open object. All of the information specified by the AERODYNAMICS command is required.

QUALIFIERS

/ROLL_RATE=degrees-per-second

Specifies approximately how fast the object will roll at the object's control speed.

/PITCH_RATE=degrees-per-second

Specifies approximately how fast the object will pitch at the object's control speed.

/YAW_RATE=degrees-per-second

Specifies approximately how fast the object will yaw at the object's control speed.

/AREA=(wing-square-feet, side-square-feet)

The first value specifies how much (top) surface area of the object should be treated as a lifting/dragging surface. You should not include surface area of the object fuselage. Typically, this means the wing area.

Also, area specified by the /AREA qualifier on the position command of any subobjects should not be included.

The second value specifies the net area of the object, when viewed from the side. This should include any vertical surface(s) as well as fuselage, engine nacelles, etc. In the case of the parts of the airframe which have rounded cross sections (such as typical fuselages), the area should be the effective "barn-door" side area, not the actual area of the projected side view. The side area defaults to 0.

/CONTROL=knots

Specifies the object's control speed. The control speed is used to graduate the object's responsiveness to roll, pitch and yaw controls on the object. During flight, the control speed is taken from the velocity of the forward airflow across the object's surfaces. When this velocity is zero, all object controls do not respond. As this velocity increases, object controls begin to respond. At and above the control speed, object controls are responding at the full rates specified by the /ROLL_RATE, /PITCH_RATE and /YAW_RATE qualifiers to the AERODYNAMICS command. A typical value for the control speed is half the top speed of the object.

/LIFT=(lift-coefficients)

Specifies the lift curve of the object. The lift coefficients are specified in order for angles of attack from -90 degrees to +90 degrees. This means that 181 values (including a value for 0 degrees) must be specified. These values are unitless and are based on what are known as NACA curves for specific airfoil types. You can either obtain the real numbers for the object airfoil that you are working with, or make them up and tune them until you see the flying behavior you expect. Be aware, the correct definition of this curve is crucial to accurate simulation of an object. Specifying other than exactly 181 values will produce an error when the object is compiled.

For more information about editing curves using the FLIGHT Hangar Editor, see Chapter 3.

/S_LIFT=(lift-coefficients)

Specifies the side lift curve of the object. The lift coefficients are specified in order for angles of sideslip from -90 degrees to +90 degrees. This means that 181 values (including a value for 0 degrees) must be specified. These values are unitless and are based on the same coefficient scheme used by the LIFT curve. The side lift curve is used to generate lateral forces as the simulated object experiences varying angles of slip, in conjunction with the side area of the object. For example, an aircraft like the F-15 Eagle has a large net side area (twin vertical tails plus large flat sides to the air intakes) which allows reasonably sustained knife-edge flight.

If no side lift curve is specified, a general-purpose curve will be used by default.

For more information about editing curves using the FLIGHT Hangar Editor, see Chapter 3.

/DRAG=(drag-coefficients)

Specifies the drag curve of the object. See the /LIFT qualifier for a description of the value of this qualifier.

EXAMPLES

```
AERODYNAMICS/LIFT=( 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000)
AERODYNAMICS/LIFT=( 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000)
AERODYNAMICS/LIFT=( 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000)
AERODYNAMICS/LIFT=( 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000)
AERODYNAMICS/LIFT=( 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000)
AERODYNAMICS/LIFT=( 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000)
AERODYNAMICS/LIFT=( -0.0339, -0.0339, -0.0508, -0.0508, -0.0678, -0.0847)
AERODYNAMICS/LIFT=( -0.1186, -0.1695, -0.2712, -0.3729, -0.4915, -0.6441)
AERODYNAMICS/LIFT=( -0.7797, -0.8983, -1.0169, -1.1017, -1.2373, -1.3390)
AERODYNAMICS/LIFT=( -1.4407, -1.5763, -1.6610, -1.7403, -1.7403, -1.7403)
AERODYNAMICS/LIFT=( -1.7050, -1.6701, -1.6348, -1.5999, -1.5297, -1.4576)
AERODYNAMICS/LIFT=( -1.4068, -1.3220, -1.2719, -1.2373, -1.1607, -1.1069)
AERODYNAMICS/LIFT=( -1.0348, -0.9630, -0.9153, -0.8644, -0.8012, -0.7293)
AERODYNAMICS/LIFT=( -0.6753, -0.6034, -0.5496, -0.5135, -0.4068, -0.3898)
AERODYNAMICS/LIFT=( -0.3220, -0.2542, -0.1864, -0.1356, -0.0678, 0.0000)
AERODYNAMICS/LIFT=( 0.0508, 0.1356, 0.1525, 0.2203, 0.2712, 0.3220)
AERODYNAMICS/LIFT=( 0.3559, 0.4068, 0.4576, 0.5085, 0.5932, 0.6441)
AERODYNAMICS/LIFT=( 0.6949, 0.7627, 0.8012, 0.8644, 0.9153, 0.9630)
AERODYNAMICS/LIFT=( 1.0348, 1.1069, 1.1607, 1.2373, 1.2719, 1.3220)
AERODYNAMICS/LIFT=( 1.4068, 1.4576, 1.5297, 1.5999, 1.6348, 1.6701)
AERODYNAMICS/LIFT=( 1.7050, 1.7403, 1.7403, 1.7403, 1.6610, 1.5763)
AERODYNAMICS/LIFT=( 1.4407, 1.3390, 1.2373, 1.1017, 1.0169, 0.8983)
AERODYNAMICS/LIFT=( 0.7797, 0.6441, 0.4915, 0.3729, 0.2712, 0.1695)
```

```
AERODYNAMICS/LIFT=( 0.1186, 0.0847, 0.0678, 0.0508, 0.0508, 0.0339)
AERODYNAMICS/LIFT=( 0.0339, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000)
AERODYNAMICS/LIFT=( 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000)
AERODYNAMICS/LIFT=( 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000)
AERODYNAMICS/LIFT=( 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000)
AERODYNAMICS/LIFT=( 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000)
AERODYNAMICS/LIFT=( 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000)
AERODYNAMICS/LIFT=( 0.0000)
```

This is the lift curve for the F-16C which is provided as a part of the standard hangars.

BRAKE

Specifies information about the object's braking abilities.

FORMAT

BRAKE

Qualifiers	**Defaults**
/MAX=max-brakes	*/MAX=0.0*
/INCREMENT=brake-increment	*/INCREMENT=0.0*

PARAMETERS

None

DESCRIPTION

The BRAKE command allows for specification of the currently opened object's braking abilities.

QUALIFIERS

/MAX=max-brakes

Specifies the maximum braking thrust of the object. This value is specified in pounds. When full brakes are in effect, this many pounds of force are simulated to oppose movement of the object.

These are not the same as thrust reversers, which must be simulated using subobjects and binding of the aircraft's thrust. Thrust reversers may be used to back up an aircraft, while brakes may not. Brakes will oppose backing up of the aircraft, while thrust reversers will not.

/INCREMENT=brake-increment

Specifies the increment by which the brakes increase or decrease their effectiveness relative to the value specified by BRAKE/MAX.

CHARACTERISTICS

Specifies information about general characteristics of an object.

FORMAT

CHARACTERISTICS characteristic (, characteristic...)

Qualifiers	Defaults
None	*None*

PARAMETERS

characteristic (, characteristic...)

Specifies one or more characteristic keywords which are to be applied to the object. By default, no characteristics are applied to the open object.

Table 2-1 shows the list of possible object characteristics.

Table 2-1 Characteristics Keywords

Keyword	Description
STALL_WARNING	Specifies that the object has the ability to detect a stall and to inform the pilot of that stall
~~AFTERBURNER~~	~~Specifies that the object has afterburner capability~~
~~FLAPS~~	~~Specifies that the object has flaps~~
INVERTABLE	Specifies that the object's engine can operate when the object is inverted.
~~REACTION_CONTROL~~	~~Specifies that there are thrusters of one sort or another to power the object through yaw, pitch, and roll maneuvers. This is typically employed by VTOL aircraft or spacecraft.~~
PILOTABLE	Specifies that this object may be selected by a user to begin a simulation
UNDETECTABLE	Specifies that collision detection for the object (and any released objects) should be disabled.
TERRAIN_CAPABLE	Specifies that the object is capable of sustained operation while in contact with terrain (dry land). If an object is not terrain capable, its engine will not produce propulsive force while on terrain (such as a boat's screw).

WATER_CAPABLE	Specifies that the object is capable of sustained operation while in contact with water. If an object is not water capable, its engine will not produce propulsive force while on water (such as an automobile's wheel drive train).
AIR_CAPABLE	Specifies that the object is capable of sustained operation while free flying (above any surface). If an object is not air capable, its engine will not produce propulsive force while free-flying (such as a boat's screw).

DESCRIPTION

The CHARACTERISTICS command specifies information about general characteristics of the currently open object.

QUALIFIERS

None

EXAMPLES

```
CHARACTERISTICS STALL_WARNING, INVERTABLE, PILOTABLE
```

The object has a stall warning, and the engine will function while the object is flown inverted. The PILOTABLE attribute indicates that the object will appear in listings of pilotable aircraft (i.e. to be used in a simulation as the starting aircraft).

CLOSE OBJECT

Completes the description of the open object.

FORMAT

CLOSE OBJECT

Qualifiers	Defaults
None	*None*

PARAMETERS

None

DESCRIPTION

The CLOSE OBJECT command closes the currently open object. FLIGHT checks to make sure that 181 data points were specified for the aerodynamics lift and drag curves, that all open subobjects have been closed, and that if the object is pilotable, that it has at least one display defined.

QUALIFIERS

None

CLOSE DISPLAY

Completes the description of the open display.

FORMAT

CLOSE DISPLAY

Qualifiers	Defaults
None	*None*

PARAMETERS

None

DESCRIPTION

The CLOSE DISPLAY command closes the currently open display.

QUALIFIERS

None

CLOSE PANEL

Completes the description of the open instrument panel.

FORMAT

CLOSE PANEL

Qualifiers	**Defaults**
None	*None*

PARAMETERS

None

DESCRIPTION

The CLOSE PANEL command closes the currently open instrument panel.

QUALIFIERS

None

CLOSE SUBOBJECT

Completes the description of the open subobject.

FORMAT

CLOSE SUBOBJECT

Qualifiers	Defaults
None	*None*

PARAMETERS

None

DESCRIPTION

The CLOSE SUBOBJECT command closes the currently open subobject. Note that this command only closes the most recently open subobject. This is significant where you are describing subobjects of multiple levels, requiring multiple, nested, subobjects to be open simultaneously.

QUALIFIERS

None

CONTROL

Specifies a component of a CONTROL-type instrument definition within the confines of OPEN INSTRUMENT and CLOSE INSTRUMENT. The CONTROL command is not valid for non-CONTROL instruments.

FORMAT

CONTROL

Qualifiers	**Defaults**
/BORDER	*None.*
/TITLE="title"	*None.*
/SURFACE_SIZE=size-in-centimeters	*None.*
/CONCAVITY=concavity-value	*None.*
/HORIZONTAL_CHANNEL=data-channel	*None.*
/VERTICAL_CHANNEL=data-channel	*None.*
/HORIZONTAL_BOUNDS=(left, right, home (,min, max))	*None*
/VERTICAL_BOUNDS=(bottom, top, home (,min, max))	*None.*
/HORIZONTAL_TYPE=control-type	*/HORIZONTAL_TYPE=VALUE*
/VERTICAL_TYPE=control-type	*/VERTICAL_TYPE=VALUE*

PARAMETERS

None

QUALIFIERS

/BORDER

Specifies that a thick border should be drawn around the CONTROL instrument.

/TITLE="title"

Specifies the string which will be displayed under the CONTROL instrument.

/SURFACE_SIZE=size-in-centimeters

Specifies the width and height of the control surface in centimeters.

/CONCAVITY=concavity-value

Specifies the response profile of the CONTROL instrument. A concavity-value of between 0 and 1 makes the control instrument very sensitive, specifically near the center. A value greater than 1 makes the control instrument less sensitive, especially near the center.

/VERTICAL_CHANNEL=data-channel

Specifies the data channel which is modified by this control in the vertical axis. Only a subset of the total number of data channel types may be specified here. They are listed in Table 2-2. If this qualifier is omitted, movement of the mouse in the vertical axis of the control instrument will have no effect.

Table 2-2 Data Channel Keywords for CONTROL Instruments

Keyword	Qualifier(s)	Value
ELEVATOR	=PERCENTAGE	
	=ANGLE	This value is expressed as a percentage of the elevator angle of the object's maximum elevator settings.
AILERONS	=PERCENTAGE	
	=ANGLE	This value is expressed as a percentage of the ailerons angle of the object's maximum ailerons settings.
RUDDER	=PERCENTAGE	
	=ANGLE	This value is expressed as a percentage of the maximum or the actual amount of rudder deflection.
THRUST	None	This value is expressed as a percentage of the idle-to-full setting of the object's throttle, where 100 represents full (possibly afterburning power.)
BRAKES	=PERCENTAGE	
	=ANGLE	This value is expressed as a percentage of the maximum or the actual amount of brake force being applied.
USER1-USER16	None	These values have no limit and are controlled through execution of the SET USER_VALUE simulator function command.

/HORIZONTAL_CHANNEL=data-channel

Specifies the data channel which is modified by this control in the horizontal axis. Table 2-2 lists valid data channel types. If this qualifier is omitted, movement of the mouse in the horizontal axis of the control instrument will have no effect.

/VERTICAL_BOUNDS=(bottom, top, home (, min, max))

Specifies the values of the data channel which correspond to the bottom and top of the control. In a RATE-based control instrument, these values are the rate at which the control's value changes per second. In a VALUE-based control instrument, these values are absolutes which the control's value reaches instantaneously. Positions between top and bottom are interpolated according to the concavity of the control instrument.

The *home* position is the position considered 'neutral' or 'natural' for the instrument's data channel.

The *min* and *max* values specify the limits on the value that the control instrument can reach. Usually, they are only significant for a RATE-based control instrument. If omitted, they are given the values $-2^{32}-1$ and $2^{32}-1$ respectively.

/HORIZONTAL_BOUNDS=(left, right, home (,min, max))

Specifies the same information as the /VERTICAL_BOUNDS qualifier, except that they specify the left and right limits of the control instrument, the horizontal home position and the horizontal minimum and maximum.

/VERTICAL_TYPE=control-type

Specifies whether the instrument is a RATE-based instrument or a VALUE-based instrument. A RATE-based instrument is one in which the control instrument controls the rate at which a value is changing. A VALUE-based instrument is one in which the control instrument controls the exact value.

/HORIZONTAL_TYPE=control-type

Specifies the same information as the VERTICAL_TYPE qualifier, except that it specifies the control type for the horizontal axis of the control instrument.

EXAMPLES

```
CONTROL/VERTICAL_CHANNEL=USER2
CONTROL/VERTICAL_BOUNDS=(30.0, -30.0, 0.0, -20.0, 75.0)
CONTROL/VERTICAL_TYPE=RATE
```

This is an extract from the Anti-Aircraft gun's control instrument. It is a rate-based control instrument which affects the USER2 data channel. The rates at which it can control this user value are ±30 units per second. The home position is 0.0, signifying that centering the stick causes a zero rate of change to the USER2 data channel. The limits of values that the control instrument can reach are -20.0 units to +75.0 units.

In the Anti-Aircraft gun, the USER2 data channel is used to determine the elevation angle of the gun. So by moving the mouse in this control instrument, the elevation angle of the gun will change.

```
OPEN INSTRUMENT=CONTROL=CONTROL_STICK

CONTROL/SURFACE_SIZE=(11.0,11.0)
CONTROL/CONCAVITY=(1.6,1.6)
CONTROL/VERTICAL_CHANNEL=ELEVATOR=PERCENTAGE
CONTROL/HORIZONTAL_CHANNEL=AILERON=PERCENTAGE
CONTROL/VERTICAL_BOUNDS=(100.0, -100.0, 0.0)
CONTROL/HORIZONTAL_BOUNDS=(-100.0, 100.0, 0.0)

CLOSE INSTRUMENT
```

This is one of the typical control instruments used for normal aircraft control. It is a full specification of a two-axis VALUE-based control instrument. This means that it specifies exact values instead of rates at which a value is supposed to change. The values that it changes are the elevator and aileron deflections in the vertical and horizontal axes, respectively.

The instrument will be 11.0 by 11.0 centimeters and the concavity for both axes is 1.6. That means that the edges of the instrument are more sensitive than the center. Or, in other words, that it requires less control movement near the edges of the control instrument than towards the center for the same change in the data channel values.

CRASH_IMPACT

Specifies the impact weight which will cause the object to be destroyed when coming into contact with the ground.

FORMAT

CRASH_IMPACT impact-load

Qualifiers	**Defaults**
None	*None*

PARAMETERS

impact-load

Specifies the impact weight which will cause the object to be destroyed. Values are expressed in pounds and a good value for this parameter is three times the object's dry weight. The object's dry weight is specified by the WEIGHT/EMPTY command.

DESCRIPTION

The CRASH_IMPACT command specifies the impact weight which will cause the currently open object to be destroyed.

For object intended as weapons, this parameter should be set to a value which is less than the root object's gross weight. This will cause the object to destroy itself upon contact with the ground.

QUALIFIERS

None

DEFINE COLOR

Specifies a color value for use in the current hangar.

FORMAT

DEFINE COLOR color-name

Qualifiers	Defaults
/RGB_VALUE=(red,green,blue)	*None*

PARAMETERS

color-name

Specifies the color to be defined. If the name specified is known by the X server on which this color will be displayed, then the RGB_VALUE qualifier is not necessary. If you define your own color name, then you must specify the RGB_VALUE value.

Color names are case-sensitive.

DESCRIPTION

The DEFINE COLOR command specifies a color name which can be used by objects in their GEOMETRY COLOR commands. If a GEOMETRY COLOR command references an undefined color name, a color definition is automatically performed.

Colors are assigned during a simulation on a first-come first-served basis. So the order by which colors are defined determines the likelihood of them being used as requested. For example, 30 colors might be defined in a hangar. However, on a system with few color resources, perhaps only new 5 colors may be allocated. The 5 colors that will be allocated will be the first 5 defined in the hangar. Therefore, the remaining 25 colors must be mapped to the closest color that the display can handle.

QUALIFIERS

/RGB_VALUE=(red,green,blue)

Specifies the exact red, green and blue components of the named color. Each value is on a scale from 0.0 to 1.0. If this qualifier is not specified, then FLIGHT will rely on the X server's knowledge of colors to determine what the appearance of objects using this color will be.

EXAMPLES

```
DEFINE COLOR "Brown"
```

Defines a color called Brown. The command relies on the display device knowing what the color Brown looks like. If it does not, the results are undefined.

```
DEFINE COLOR "Off-White"/RGB_VALUE=(0.83,0.90,1.0)
```

Defines a color called Off-White. The command specifies the exact appearance of the color.

```
DEFINE COLOR "Red"/RGB_VALUE=(1.0, 0.0, 0.0)
```

Defines a color called Red. The command specifies the exact appearance of the color.

DESIGNER

Specifies ownership information about a particular object.

FORMAT

DESIGNER "designer-name" "design-date"

Qualifiers	Defaults
None	*None*

PARAMETERS

designer-name

Specifies the name of the designer of the currently opened object.

design-date

Specifies the date on which the object was designed or made publicly available. The format of the date is DD-MMM-YYYY HH:MM:SS.SS, which is a standard VMS date/time specification.

DESCRIPTION

The DESIGNER command allows the object designer to specify ownership of the object design. The designer's name and time of creation are both kept with the compiled form of the object. If the hangar editor is used to generate the source which originated the compiled hangar, the designer's name will be output into the source, and the time on the design will be updated to the time that the reverse-compiler wrote the source file out.

QUALIFIERS

None

EXAMPLES

```
DESIGNER "FLIGHT development" "15-DEC-1986 00:00:00.00"
```

The designer's name is "FLIGHT development" and the source file was created the 15th of December, 1986.

DIAL

Specifies a component of a dial instrument definition within the confines of OPEN INSTRUMENT and CLOSE INSTRUMENT. The DIAL command is not valid for non-DIAL instruments.

FORMAT

DIAL

Qualifiers	**Defaults**
/BASE=number	*None*
/BIGTICKS=number	*None*
/SMLTICKS=number	*None*
/LABEL=(label, position, radius, xscale, yscale)	*None*
/DATA=data-type	*None*
/MAXVALUE=value	2^{32}*-1*
/MINVALUE=value	2^{32}*-1*
/RADIUS=centimeters	*1.7*
/ROTATION	*0.0*
/PRECISION=number	*0*
/HAND=(inner,outer,width,scale,offset)	/HAND=(0.0, 82.0, 2.0, 1.0, 0.0)
/TICKS=(number,start,end,in,out,thickness)	*None*
/IN_TITLE=(title,position,xscale,yscale)	*None*

PARAMETERS

None

QUALIFIERS

/BASE

Specifies a value which corresponds to one full revolution of the dial hand on the instrument in the units of the DIAL's data channel.

/BIGTICKS=number

Specifies the number of large tick marks drawn around the inside edge of the DIAL. The number of tick marks are spread evenly through 360°. For more exact control over tick marks on the DIAL, see the TICKS qualifier.

/SMLTICKS=number

Specifies the number of the small tick marks drawn along the inside edge of the DIAL. The number of tick marks are spread evenly through 360°. For more exact control over tick marks on the DIAL, see the TICKS qualifier.

/LABEL=(title, position, radius, xscale, yscale)

Specifies the position, size, and title of a label positioned around the circumference of the DIAL. The angular position is expressed in term's of the dial's base value. The 'radius' value controls where the text is written, in terms of distance out from the dial center. A magnitude of 100 equals the radius of the dial face; values larger than 100 may be specified. The label is centered at that point. The text character size is specified by the two scale values. The magnitude of the size values is based on 100 units being equal to the dial's radius. (If the scale values are omitted, the label is drawn at a standard, fixed size.)

Up to 40 labels may be placed on a single DIAL, but each must be placed by a separate DIAL/LABEL command.

/DATA=data-type

Specifies the parameter of the simulation that should be displayed in the DIAL. For more information about the available data types which can be used to drive the instrument, see Section 2.4.

/ROTATION=degrees

Specifies the amount of rotation for the zero point of the DIAL. The zero value of the DIAL's data channel aligns on this point. A rotation of 0° is at the 12 o'clock position, 90° is at the 3 o'clock position.

/HAND=(inner, outer, width, scale, offset)

Specifies a needle hand for the dial. The inner and outer values specify the ends of the needle. The values are expressed as percentages of the dial radius. The width value is similarly expressed, except that it specifies the width of the needle. The scale and offset are applied to the dial's data channel value before indicating a value.

The value actually indicated by the dial hand is

```
data-channel-value * scale + offset
```

A maximum of 5 hands may be specified per dial.

/MAXVALUE=value

Specifies the largest value which can be displayed by the DIAL.

/MINVALUE=value

Specifies the smallest value which can be displayed by the DIAL.

/RADIUS=radius-in-centimeters

Specifies the radius of the DIAL.

/PRECISION=number

Specifies the amount of change required in the data channel feeding the DIAL instrument before a screen update takes place to reflect the change. For example, a value of 1 indicates that a change of 10.0 or greater will cause an update to the instrument. A value of -3 indicates that a change of 0.001 or greater will cause an update to the instrument.

/TICKS=(number,start,end,in,out,thickness)

Places a generalized set of ticks in the DIAL. *Number* controls how many tick marks there are in the set. *Start* and *end* specify the vertical location where the marks are drawn (specified in the dial's BASE value units). The tick marks are evenly distributed in this region, inclusive of ends (that is, the first tick is located at *start* and the last at *end*). The *in* and *out* parameters control the positions of the endpoints of the ticks (100 tick units equals the radius of the DIAL face and would cause a tick end to coincide with the edge of the dial face). *Thickness* controls how thick the tick marks are (same units as the radius parameters).

Up to 9 tick sets may be specified per DIAL.

/IN_TITLE=(title,position,xscale,yscale)

Places a text title within the face of the DIAL. This title is horizontally centered on the dial and the *position* value indicates the vertical placement of the title as a percentage of the dial radius. Negative values indicate positions below the center of the dial and positive values indicate positions above the center.

Up to 10 titles may be specified per DIAL.

DIGITAL

Specifies a component of a digital-type instrument definition within the confines of OPEN INSTRUMENT and CLOSE INSTRUMENT. The DIGITAL command is not valid for non-DIGITAL instruments.

FORMAT

DIGITAL

Qualifiers	**Defaults**
/(NO)BORDER	*None.*
/TITLE="title"	*None.*
/LABEL="label"	*None.*
/DATA=data-type	*None.*
/MAXVALUE=value	*None.*
/MINVALUE=value	*None.*
/DIGITS=number	*None.*
/PRECISION=number	*None.*
/SCALE_FACTOR=number	*1.0*
/OFFSET=number	*0.0*

PARAMETERS

None

QUALIFIERS

/(NO)BORDER

Specifies that a border should be drawn around the DIGITAL instrument.

/TITLE="title"

Specifies the string which will be displayed under the DIGITAL instrument. This string may be up to 32 characters long.

/LABEL="label"

Specifies a label to be displayed immediately to the right of the DIGITAL instrument. This label usually indicates units displayed by the instrument. This string may be up to 6 characters long.

/DATA=data-channel

Specifies the parameter of the simulation that should be displayed in the DIGITAL instrument. For information about the available data types which can be used to drive the instrument, see Section 2.4.

/MAXVALUE=value

Specifies the largest value which can be displayed on the DIGITAL instrument.

/MINVALUE=value

Specifies the smallest value which can be displayed on the DIGITAL instrument.

/DIGITS=number

Specifies the total number of digits to display, including the decimal point.

/PRECISION=number

Specifies the amount of change required in the DATA type feeding the DIGITAL instrument instrument before a screen update takes place to reflect the change. For example, a PRECISION=1 value indicates that a change of 10.0 or greater will cause an update to the instrument. A PRECISION=-3 value indicates that a change of 0.001 or greater will cause an update to the instrument.

Note that PRECISION also controls the number of digits after the decimal point in the DIGITAL instrument. For example, suppose a DIGITAL instrument had the following attributes:

```
DIGITAL/DIGITS=10/PRECISION=-3
```

In the preceeding case, the instrument would display its value in the form "nnnnnn.nnn", where the overall length of the display is 10 digits, but one is used for the decimal point, and 3 are used for fractional precision.

/SCALE_FACTOR=number

The data channel value fed to the instrument will be multiplied by this number, in the course of computing the final value displayed by the instrument. There is another qualifier, /OFFSET, which also plays a part. The formula used to compute the final displayed number is:

```
displayed value = (data value * scale factor) + offset
```

For example, if the offset is set to 10 and the scale factor is set to 2, and the data value being fed to the instrument is 30.0, then the displayed value will be 70.0.

/OFFSET=number

The scaled data channel value fed to the instrument will be added to this number, in the course of computing the final value displayed by the instrument. There is another qualifier, /SCALE_FACTOR, which also plays a part.

FUEL

Specifies information about the object's use of fuel.

FORMAT

FUEL

Qualifiers	**Defaults**
/ALLOWANCE=pounds	*None.*
/CONSUMPTION=pounds-per-second	*None.*

PARAMETERS

None

DESCRIPTION

The FUEL command allows for specification of the currently opened object's use of fuel.

QUALIFIERS

/ALLOWANCE=pounds

Specifies the maximum amount of fuel (in pounds) which the object may carry. No special provision is made for external fuel tanks, so if you want that amount of fuel to be included in the total fuel allowance, merely add it into this value.

/CONSUMPTION=pounds-per-second

Specifies the amount of fuel (in pounds) which the object consumes per second while the engines are at maximum thrust. Maximum thrust is specified using the THRUST/DRY command.

EXAMPLES

```
FUEL/ALLOWANCE=7000.0/CONSUMPTION=3.5
```

This example command sets the object to be able to carry a maximum of 7,000 lbs of fuel and with the engines at maximum power they will consume 3.5 pounds of fuel per second. Assuming a full load of fuel at full power, the object will run out of fuel in

$$\frac{(7000.0\ \mathit{pounds})}{(3.5\ \mathit{pounds\ per\ second})} = 2000.0\ \mathit{seconds} = 33\ \mathit{minutes}$$

FIRING_SEQUENCE

Specifies information about the order in which weapon banks are accessed when releasing weapons.

FORMAT

FIRING_SEQUENCE bank(,bank,...) (...)

Qualifiers	Defaults
None	*None.*

PARAMETERS

bank(,bank,...) (...)

Specifies the banks to fire from in one step of the firing sequence.

DESCRIPTION

The FIRING_SEQUENCE command controls which banks are fired each time the interactive FIRE WEAPON command is invoked. A firing sequence consists of 4 sequential steps, each of which may cause between 1 and 4 banks to release their current weapon.

As firing sequences are stepped through by virtue of users firing weapons, FLIGHT remembers where each firing sequence is in terms of the last step invoked. So if the user switches between firing sequences interactively, FLIGHT ensures that a firing sequence is never executed out of order. That is, the second step always follows the first , etc. Note that the first step in a sequence logically follows the last step in a firing sequence.

Only the first step of a firing sequence must be specified. Additional steps are optional.

QUALIFIERS *None*

EXAMPLES

```
FIRING_SEQUENCE 1
FIRING_SEQUENCE 2
FIRING_SEQUENCE 3
FIRING_SEQUENCE 4
```

These commands show the default firing sequences which FLIGHT defines for all aircraft. Each firing sequence merely releases a single weapon from a single bank. Firing sequence 1 releases from bank 1, sequence 2 releases from bank 2, etc.

```
FIRING_SEQUENCE 1,2
```

This command shows a single step sequence in which the lead weapons from banks 1 and 2 are both released when the user performs a single FIRE WEAPON command using this sequence.

```
FIRING_SEQUENCE 1 2
```

This command shows a two-step sequence. The first step is invoked when the user performs a FIRE WEAPON command using this sequence. The second step is invoked on the next FIRE WEAPON directive from the user. The next FIRE WEAPON directive would cause the first step to again be used.

The first step of the sequence releases the lead weapon from bank 1 and the second step releases the lead weapon from bank 2. This sequence might be used to fire tracers from alternating sides of the aircraft for visual effect. One bank would fire from one side of the aircraft and the other bank would fire from the other side. Note the small difference in the syntax between the second and third examples.

```
FIRING_SEQUENCE 1,2 3 4
```

This command shows a three-step sequence. The first step releases the lead weapon from banks 1 and 2. The second step releases the lead weapon from bank 3, and the third step releases the lead weapon from bank 4.

GEOMETRY COLOR

Specifies the color of an object's geometry.

FORMAT

GEOMETRY COLOR color

Qualifiers	Defaults
None	*None*

PARAMETERS

color

Specifies the color of the object's geometry. The set of color names is determined by the colors known by the X server being used, or those colors which have been defined prior to this GEOMETRY COLOR command by DEFINE COLOR commands.

DESCRIPTION

The GEOMETRY COLOR command specifies the color of the currently open object's geometry. Each subobject and root object may have different colors. If the command is not specified during the description of the object, a default color will be used.

The specified color is not always possible on the display device. FLIGHT automatically figures out the display color for you if the color you have specified is not available. For example, on a black and white display device, RED cannot be displayed when specified within a subobject; this command sets the color of that subobject.

QUALIFIERS

None

EXAMPLES

```
GEOMETRY COLOR GREEN
```

The object's geometry will be drawn using green lines where possible.

GEOMETRY LINE

Specifies the endpoints of a single line in an object.

FORMAT

GEOMETRY LINE point-index1 point-index2

Qualifiers	**Defaults**
None	*None*

PARAMETERS

point-index1 point-index2

Specifies the indices of the two points which make up the line.

DESCRIPTION

The GEOMETRY LINE command specifies the location of a single line in the currently open object's geometry. GEOMETRY LINE commands may only be used after a GEOMETRY LOD command has been specified. A maximum of 300 lines may be loaded into each level of detail for each object.

The color of the line is determined by the color of the object or subobject that contains the line. The color is specified by the GEOMETRY COLOR command. The order in which the point indices are specified in a GEOMETRY LINE command is not significant, but the points must have been loaded before being referenced. Other than that restriction, GEOMETRY POINT and GEOMETRY LINE commands may be mixed freely.

QUALIFIERS

None

EXAMPLES

```
GEOMETRY LINE 8 12
```

A single line is loaded into the most recently specified level of detail for the currently open object. The line is defined to be drawn from the eighth point loaded into the current level of detail to the twelfth point loaded into the current level of detail.

GEOMETRY LOD

Specifies a change in an object's geometry level of detail specification.

FORMAT

GEOMETRY LOD level-index min-angle

Qualifiers	**Defaults**
None	*None*

PARAMETERS

level-index

Specifies the new geometry level of detail to be loaded with points and lines. Valid level of detail indices are 1 through 3.

min-angle

Specifies an angle in degrees. This angle is used to determine when the level of detail specification should be used to display the object. The angle is computed by figuring out how wide the object appears to be to someone viewing it. The further away the object is, the smaller the angle. Careful tuning of this number will produce imperceptible changes between levels of detail as the object recedes into the distance or approaches the viewer.

DESCRIPTION

The GEOMETRY LOD command is used to specify the conditions under which a level of detail specification should be used to display the object. A level of detail specification is a geometric description of the object with some given level of complexity. LOD 1 is used for the greatest level of geometric complexity (most lines and points). LOD 3 is used for the least complicated geometry. FLIGHT also provides a fourth LOD, which merely results in a single point at the center of the object. The rationale for levels of detail at all is so that as the object moves further and further away from the viewer, less and less complicated geometries can be used to display the object. This is an important performance consideration.

Each level of detail specification is prefixed by a GEOMETRY LOD command, followed by GEOMETRY LINE and GEOMETRY POINT commands. Further, each specification is a complete description of the geometry of the object which is totally unrelated from the descriptions of other levels of detail, except, of course, in that they all describe the same object.

You do not have to specify anything more than 1 level of detail, but you must specify the "GEOMETRY LOD 1 min-angle-applies" command for that level of detail. When the visual size of the object goes below min-angle-applies, the FLIGHT-supplied fourth level of detail will be used.

QUALIFIERS *None*

EXAMPLES

```
GEOMETRY LOD 1 20.0

 GEOMETRY POINT and GEOMETRY LINE commands for level of detail 1

GEOMETRY LOD 2 8.0

 GEOMETRY POINT and GEOMETRY LINE commands for level of detail 2

GEOMETRY LOD 3 0.5

 GEOMETRY POINT and GEOMETRY LINE commands for level of detail 3
```

In this example, FLIGHT would use the description outlined in the first level of detail while the object was visually greater than or equal to 5 degrees of angular width. The second level of detail would automatically be used when the object was visually reduced to between 5.0 and 1.0 degrees of width. The third would be used once the 1.0 degree threshhold had been crossed.

GEOMETRY ORIGIN

Specifies the location of the geometry origin relative to the ground.

FORMAT

GEOMETRY ORIGIN offset-in-feet

Qualifiers	**Defaults**
None	*None*

PARAMETERS

offset-in-feet

Specifies the distance from the ground to the object's geometry origin.

DESCRIPTION

The GEOMETRY ORIGIN command specifies the distance from the ground to the currently open object's geometry origin. The origin of an object's geometry is always defined to be (0, 0, 0). The GEOMETRY ORIGIN command is provided to allow a vertical displacement adjustment, generally to move the object up.

If the GEOMETRY ORIGIN value were 0.0, parts of the object would appear to pass through the ground when the object was in contact with the ground. Using the GEOMETRY ORIGIN command allows the aircraft designer to raise the object to a height which is appropriate for the object.

QUALIFIERS

None

EXAMPLES

```
GEOMETRY ORIGIN 12.0
```

The entire object is moved 12 feet off the ground to accomodate for the normal height of the geometry center. This object is probably a large airliner, owing to the height off the ground. Twelve feet is rather high up. A private airplane might only be a few feet up.

GEOMETRY POINT

Specifies the location of a single point in an object.

FORMAT

GEOMETRY POINT x y z

Qualifiers	**Defaults**
None	*None*

PARAMETERS

x y z

Specifies the location of the point relative to the origin of the object.

DESCRIPTION

The GEOMETRY POINT command specifies the location of a single point in the currently open object's geometry. GEOMETRY POINT commands may only be used after a GEOMETRY LOD command has been specified. Points are numbered in the order that they are loaded, starting at 1 and going up to a maximum of 300 points per each level of detail. So each level of detail may have up to 300 points, each numbered from 1 to 300, not 1 to 1200 or any other variant.

Points may be referenced in geometry line commands multiple times in a geometric description, but only within the same level of detail description.

Units are in feet. Forward is +X, up is +Y and right is +Z. This is a required orientation for the geometry if you want the object to fly forward. Further, you should describe the geometry such that the origin of the geometry is at the center of gravity for the object, since all rotation of the object in flight is around the geometry origin.

QUALIFIERS *None*

EXAMPLES

```
GEOMETRY POINT 12.0 -6.66 0.11
```

A single point is loaded into the most recently specified level of detail for the currently open object. The point is defined to be 12.0 feet in front of, 6.66 feet below, and 0.11 feet to the right of the geometry center.

GEOMETRY TRIANGLE

Creates a triangle polygon.

FORMAT

GEOMETRY TRIANGLE point-index1 point-index2 point-index3

Qualifiers	Defaults
None	*None*

PARAMETERS

point-index1 point-index2 point-index3

Specifies the indices of the three points which make up the triangle.

DESCRIPTION

Use the GEOMETRY TRIANGLE command to create a triangle for the current level of detail in the currently open object. A maximum of 200 triangles may be created in each level of detail for each object.

The color of the triangle is determined by the color of the object or subobject that contains the triangle. The color is specified by the GEOMETRY COLOR command. The order in which the point indices are specified is not significant, but the points must have been loaded before being referenced by a GEOMETRY TRIANGLE command.

FLIGHT does not currently support display of solid objects. Instead, the wireframe description of the object is used. See the GEOMETRY LINE command for a description of how to describe that wireframe form.

QUALIFIERS *None*

EXAMPLES

```
GEOMETRY TRIANGLE 4 1 2
```

A single triangle is created in the current level of detail for the currently open object. The triangle is defined to fill the triangular area encompassed by the fourth, first, and second points in the current level of detail's point list.

GUIDANCE NONE

Specifies that the weapon is unguided

FORMAT

GUIDANCE NONE

Qualifiers	**Defaults**
None	*None*

PARAMETERS

None

DESCRIPTION

Use the GUIDANCE NONE command to indicate that the currently open weapon has no guidance and will follow a ballistic trajectory subject to the effect of engines and the other flight characteristics of the weapon. This is the default guidance provided for weapons if no GUIDANCE command is specified.

QUALIFIERS

None

GUIDANCE RADAR

Specifies that the guidance mode of the current weapon is radar-based.

FORMAT

GUIDANCE RADAR

Qualifiers	**Defaults**
~~*/RATING=max target rating*~~	~~*500.0*~~
/SIZE=feet	*8.0*
/SEPARATION=degrees	*60.0*
~~*/ANGULAR_SIZE=degrees*~~	~~*0.5*~~
/RESOLUTION=resolution	*1E-8*

PARAMETERS

None

DESCRIPTION

Use the GUIDANCE RADAR command to indicate that FLIGHT's radar guidance software should be used to allow the weapon to track targets at its own discretion. The GUIDANCE RADAR command should be specified somewhere between an OPEN OBJECT and CLOSE OBJECT command pair.

While in flight, the weapon will constantly consider targets and pursue the one most appealing, according to the guidance parameters you specify with this command. If the weapon loses guidance on all targets, it becomes ballistic in its flight (no guidance). It may reacquire a target at any time.

~~Note that the radar guidance software requires a moving target for guidance. If the object being tracked stops moving, the weapon will lose lock-on.~~

To lock a radar-guided weapon onto a target requires waiting for it to identify a target. Only the weapons which are next to be launched in the selected firing sequence will actively search for a target. See the data channel documentation for a description of the information which a weapon may return as a result of a target lock-on.

QUALIFIERS

~~/RATING=max-target-rating~~

~~Specifies the highest (worst) target rating which the weapon will consider as a viable target. This value is computed as follows;~~

~~Add one point for every 50 feet of distance separating the potential target from the weapon.~~

~~Add one point for every 1 degree of turn needed to aim the nose of the weapon towards the potential target.~~

~~By default, this value is 500.0, meaning that a target no farther than 50, 000 feet (about 9½ miles) will be considered for pursuit. And that, only if the target is directly ahead of the weapon. The maximum distance will go down if the object is off-center from the weapon's flight path.~~

/SIZE=feet

Specifies the smallest physical size of an object that would be considered for pursuit.

/SEPARATION=degrees

Specifies the farthest angular sweep that the weapon's radar will look to the left, right, up and down for locating targets. By default, this value is 60 degrees, meaning that the weapon will look a maximum of 60 degrees away from its path in attempts to locate a target.

~~/ANGULAR_SIZE=degrees~~

~~Specifies the smallest angular width which a target object may appear to have on radar and still be considered for pursuit. By default, an object must appear to be 0.5 degrees wide (the approximate resolution power of the human eye). The farther an object is from the weapon, the smaller the angular width of the object.~~

/RESOLUTION=resolution

Specifies the resolving ability of the radar unit which the weapon carries. For a given object cross section **C** (defined by the object's SIGNATURE RADAR commands) at a distance **D**, the radar unit will detect the object if the resolution value is greater than $\mathbf{C} / (2 * \mathbf{D}^2)$. Note that an object's cross section is dependant upon the orientation of the object relative to the radar unit. That is, a head-on

view of an aircraft presents a smaller cross section than an over-
head view.

GUIDANCE SURFACE

Specifies that the guidance mode of the current weapon is based on a surface target tracking ability.

FORMAT

GUIDANCE SURFACE

Qualifiers	**Defaults**
/APPROACH=value	*2.0*

PARAMETERS

None

DESCRIPTION

Use the GUIDANCE SURFACE command to indicate that FLIGHT's surface guidance software should be used to allow the weapon to guide to a single location in the world. The GUID-ANCE RADAR command should be specified somewhere between an OPEN OBJECT and CLOSE OBJECT command pair.

While in flight, the weapon will track to strike the target location, following the approach trajectory established for the weapon by the guidance qualifiers you specify.

To lock a surface-guided weapon onto a target, merely sight the aircraft's crosshairs on the ground next to the target of interest. When the user fires the weapon, it locks onto that location and guides to it.

QUALIFIERS

/APROACH=value

Specifies the approximate glide slope that the weapon will take to the target. A large value (e.g. 100) will cause the weapon to attempt to take a shallower approach to the target. A smaller value (e.g. 10) will cause the weapon to attempt to take a steeper approach to the target.

Note that the weapon must be tuned for the approach value that you specify. A steeper approach is best for low-lift devices such as missiles which are ballistic, while a shallower approach is possible for lifting devices such as cruise missiles.

The exact approach that the weapon takes is to keep itself (distance-from-target / approach-value) feet off the ground. If the weapon were 1 mile (5280 feet) away from the target and it had an approach value of 100 defined for it, the weapon would go no lower than 52.8 feet from the ground.

HORIZON

Specifies a component of a HORIZON-type instrument definition within the confines of OPEN INSTRUMENT and CLOSE INSTRUMENT. The HORIZON command is not valid for non-HORIZON instruments.

FORMAT

HORIZON

Qualifiers	**Defaults**
/RADIUS=size-in-cm	*None.*
/RATION=radius-ration	*None.*
/PITCH_SCALE=pitch-angle-degrees	*None.*
/(NO)FILLED	*None.*
/B_ANNOTATION	*None.*
/X_ANNOTATION=(xvalue, xvalue2, ...)	*None.*
/Y_ANNOTATION=(yvalue, yvalue2, ...)	*None.*
/RULE=(x1, y1, x2, y2)	*None.*

PARAMETERS

None

QUALIFIERS

/RADIUS=size-in-cm

Specifies the radial dimension of a HORIZON instrument in centimeters. This radius applies to the outer-most circle of the instrument.

/RATIO=radius-ratio

HORIZON instruments are composed of (among other things) two concentric circles. The size of the outer circle is specified using the RADIUS qualifier; the size of the inner circle is indirectly specified using RATIO. For example, /RADIUS=2.0/RATIO=0.5 would mean that the inner circle had a 1 cm. radius (half of 2.0).

/PITCH_SCALE=pitch-angle-degrees

This value binds the amount of horizon gimbal deflection that a given pitch attitude will cause. Specifically, this value defines what pitch (up or down) will cause the horizontal simulated horizon line inside the instrument face to be deflected up or down all the way to the top or bottom of the inner circle.

For example, PITCH_SCALE=30 says that when the object is pitched up 30 degrees, the horizon line will be up at the top of the instrument face. Other pitch amounts will cause proportional deflections.

/(NO)FILLED

Specifies whether the earth-half of the horizon gimbal is drawn as a solid filled region, or is drawn as a hollow shape with a thick outline.

/B_ANNOTATION

Specifies the beginning of a new annotation (also automatically completes any current annotation). For a complete description of annotations, see the help for XANNOTATION.

/X_ANNOTATION=(xvalue1, xvalue2, ...)

Specifies a list of X-coordinate values for the current annotation.

A horizon instrument may have a number of annotations drawn on it. Annotations may be created to represent various parts of the instrument, for example the toy airplane that indicates the zero pitch attitude, simulated adjustment knobs, etc. All annotations are drawn with a 50% halftone fill pattern.

There are three HORIZON command qualifiers which together construct annotations: B_ANNOTATION, which specifies the beginning of a new annotation, and the X_ANNOTATION, Y_ANNOTATION pair which supply X and Y-coordinate values that define the shape and position of the annotation. The coordinate system which the X and Y values are based on is as follows: the origin is at the center of the instrument face, and the outer face circle radius represents a size of 100 units.

There can be any number of separate annotations, but the total number of X coordinate values specified for all annotations for a single horizon instrument can be at most 99 (similarly you can specify up to 99 total Y values).

/YANNOTATION=(yvalue1, yvalue2, ...)

Specifies a list of Y-coordinate values for the current annotation (see XANNOTATION).

/RULE=(x1, y1, x2, y2)

Specifies a rule (line segment) that should be placed on the face of the gimballed horizon ball. Since they are "inscribed" on the face of the moving ball, the ruled lines will move as the object pitches and rolls. For example, a horizontal rule drawn below center could represent a 10 degree down pitch mark (assuming it were placed appropriately, based on the PITCH_SCALE of the instrument).

A single HORIZON instrument may have up to 40 rules. Each rule is defined by two coordinate pairs (one for each end-point of the line). See the help for XANNOTATION for a description of the coordinate space which the end-point values are specified in.

EXAMPLES

The following set of sample commands creates a square annotation, centered on the face and with total side length equal to the outer circle radius:

```
HORIZON/B_ANNOTATION
HORIZON/XANNOTATION=(-50, 50, 50, -50, -50)
HORIZON/YANNOTATION=(50, 50, -50, -50, 50)
```

INCLUDE

Specifies a hangar source file to include to the current compiler stream. The default location for an included file is the default directory and the default file extension is DAT.

FORMAT

INCLUDE input-file-spec

Qualifiers	Defaults
None	*None*

PARAMETERS

input-file-spec

Specifies the name of the source file to include into the current compiler stream.

You must specify a complete file specification. No wildcards are allowed.

DESCRIPTION

The INCLUDE command takes the text contents of the specified file and includes it exactly as the compiler finds it. There is no restriction on the number of include levels permitted. That is, an included file may include other files using the INCLUDE command, and those files may, in turn, include additional files. The nesting of INCLUDE commands is indicated in the listing file produced by the compiler (available using the LISTING qualifier on the FLIGHT/COMPILE=HANGAR command).

QUALIFIERS *None*

EXAMPLES

```
INCLUDE MYDISK:[MYDIRECTORY]SPITFIRE.TXT
```

The INCLUDE command takes the contents of the specified file and begins to compile it. The line immediately following the INCLUDE line is not compiled until MYDISK:[MYDIRECTORY]SPITFIRE.TXT has completed its compilation. Note that the default .DAT extension is overrided by explicitly specifying .TXT as the file extension.

```
INCLUDE ULTRALIGHT
  0 0003  INCLUDE ULTRALIGHT
%FLT-E-UNAOPEDES, unable to open description file $DISK1:[FLIGHT]ULTRALIGHT.DAT;
-RMS-E-FNF, file not found
```

The INCLUDE command take the contents of ULTRALIGHT.DAT, assumed to be in the default directory, and begins to compile it. In this case, the terminal output shows that the specified file could not be found. Compilation will continue at the next line in the file.

LAMP

Specifies a component of a LAMP-type instrument definition within the confines of OPEN INSTRUMENT and CLOSE INSTRUMENT. The LAMP command is not valid for non-LAMP instruments.

FORMAT

LAMP

Qualifiers	**Defaults**
/HEIGHT=cm	*1.5*
/WIDTH=cm	*2.5*
/DATA=data-type	*None.*
/MINVALUE=minimum-value	*0.5*
/MAXVALUE=maximum-value	*See text*
/LABEL=(title, xposition, yposition, xscale, yscale)	*None.*

PARAMETERS

None

QUALIFIERS

/HEIGHT=cm

Specifies the height of the LAMP in centimeters.

/WIDTH=cm

Specifies the width of the LAMP in centimeters.

/DATA=data-channel

Specifies the parameter of the simulation that should feed the LAMP. For more information about the available data types which can be used to drive the lamp instrument, see Section 2.4.

/MINVALUE=minimum-value

Specifies the minimum value of the LAMP's data channel which causes the lamp to light up.

/MAXVALUE=maximum-value

Specifies the maximum value of the LAMP's data channel which causes the lamp to light up. By default, this value is the maximum possible value for the LAMP's data channel.

/LABEL=(title, xposition, yposition, xscale, yscale)

Specifies the position, size, and title of a label positioned on the LAMP. The position values control where the text is written, in terms of distance out from the lamp center. A magnitude of 100 equals the height of the LAMP; values larger than 100 may be specified. The title string is centered at that point. The text character size is specified by the two scale values. The magnitude of the size values is based on 100 units being equal to the LAMP's height. (If the scale values are omitted, the label is drawn at a standard, fixed size. If the position values are omitted, the label is centered in the LAMP.)

Up to 40 labels may be placed on a single lamp, but each must be placed by a separate LAMP/LABEL command.

LANDING

Specifies the limits of orientations which can be considered a valid landing for an object.

FORMAT

LANDING

Qualifiers	Defaults
/ROLL=(roll-left-limit, roll-right-limit)	*/ROLL=(-20.0, 20.0)*
/PITCH=(pitch-up-limit, pitch-down-limit)	*/PITCH=(-50.0, 10.0)*
/YAW=(yaw-left-limit, yaw-right-limit)	*/YAW=(-20.0, 20.0)*

PARAMETERS

None

DESCRIPTION

The LANDING command allows for specification of the limits of what is permitted in pitch, roll and yaw while the object is on the ground. Exceeding the roll limits is used to indicate that a wing tip is being dragged, exceeding the yaw limit is used to indicate that the landing was too much out of line, etc.

Note the necessity of using a negative value for the first parameter of each pair.

QUALIFIERS

/ROLL=(roll-left-limit, roll-right-limit)

Specifies the limits on rolling during landing. If either of the limits specified is exceeded, the object is destroyed.

/PITCH=(pitch-up-limit, pitch-down-limit)

Specifies the limits on pitching during landing. If either of the limits specified is exceeded, the object is destroyed.

/YAW=(yaw-left-limit, yaw-right-limit)

Specifies the limits on yawing during landing. If either of the limits specified is exceeded, the object is destroyed.

LIFETIME

Specifies the lifetime of an object.

FORMAT

LIFETIME duration

Qualifiers	Defaults
/TERMINATION=type	*/TERMINATION=EXPLODE*

PARAMETERS

duration

Specifies how long, in seconds, the object will be permitted to exist. By default, an object's lifetime is unlimited.

DESCRIPTION

The LIFETIME command specifies how long the currently open object may be simulated before being automatically destroyed. This is primarily an optimization feature for weapons. Once launched and their fuel is consumed, they become little more than an annoyance as each weapon launched consumes important simulator resources.

QUALIFIERS

/TERMINATION

Specifies how the object should be taken out of the simulation when its lifetime limit is reached.

Table 2-3 lists the termination types available.

Table 2-3 Termination Types

Keyword	Description
EXPLODE	Causes the object to visually explode when the lifetime limit is reached This is the default.
VANISH	Causes the object to vanish when the lifetime limit is reached

OPEN OBJECT

Begins an object definition

FORMAT

OPEN OBJECT="object-name"

Qualifiers	**Defaults**
None	*None*

PARAMETERS

None

DESCRIPTION

The OPEN OBJECT command begins a new object definition. The CLOSE OBJECT command indicates the completion of that object definition. Commands describing the attributes of the object should be placed between the OPEN OBJECT and CLOSE OBJECT commands. There may be no more than 64 objects in a single hangar.

Object are given names so that when FLIGHT or the user needs to refer to the object, there is a meaningful way of doing so. An important instance of this is the simulator allowing selection of object in a hangar. It is the object name which appears in the list of object menu. Note that only those objects with the PILOTABLE characteristic will appear in the object selection menu.

Table 2-4 shows the commands used to describe an object.

Table 2-4 Commands Applicable to Aircraft

Commands		
AERODYNAMICS	BRAKE	CHARACTERISTICS
CRASH_IMPACT	DESIGNER	FLAPS
FUEL	GEOMETRY	GUIDANCE
LANDING	LIFETIME	REACTION_CONTROL
SIGNATURE	STRUCTURE	THRUST
VOLUME	WEAPONS	WEIGHT

In addition to the commands listed in the table, there are also commands to describe the workstation cockpit and what are known as *subobjects*. For more information about subobjects, see the description of the SUBOBJECT command in this chapter. Subobjects can add considerably to the reality of an object, allowing for animated parts of the object and other, more esoteric, capabilities. The section on subobjects is definitely a must for a serious FLIGHT object designer.

OPEN DISPLAY

Begins a display definition

FORMAT

OPEN DISPLAY="display-name"

PARAMETERS

None

DESCRIPTION

The OPEN DISPLAY command begins a new display definition. The CLOSE DISPLAY command indicates the completion of that display definition. Commands describing the attributes of the display should be placed between the OPEN DISPLAY and CLOSE DISPLAY commands.

A display is a collection of windows which are shown on the user's workstation. There may be up to 4 displays defined in a single object, and in each of those displays, there may be up to 8 viewports (see the VIEWPORT command and 4 instrument panels (see the OPEN PANEL command). An object must be opened before a display can be defined.

During an interactive simulation, the user specifies where each of the 4 displays is to be shown. This is done through the Configure Aircraft dialog box.

Table 2-5 shows the commands used to describe a display.

Table 2-5 Commands Applicable to Displays

Commands		
OPEN PANEL	CLOSE PANEL	VIEWPORT

QUALIFIERS *None*

EXAMPLES

```
OPEN DISPLAY="Pilot"
VIEWPORT ... VIEWPORT ...
OPEN PANEL PLACE "Altimeter" 3.88 9.41 PLACE ...
.
.
.
CLOSE PANEL
CLOSE DISPLAY
```

The OPEN DISPLAY command begins the description of a display called "Pilot". The display is completed with a CLOSE DISPLAY command. The presence of the two VIEWPORT commands indicate that two windows, updated in realtime are to be placed on the display to show the user what is visible. At least two instruments are being placed onto the display's instrument panel by use of the PLACE commands. One of them is called "Altimeter".

OPEN INSTRUMENT

Begins an instrument definition

FORMAT

OPEN INSTRUMENT=instrument-type="instrument-name"

Qualifiers	**Defaults**
None	*None*

PARAMETERS

None

DESCRIPTION

The OPEN INSTRUMENT command begins a new instrument definition. The CLOSE INSTRUMENT command indicates the completion of that instrument definition. Commands describing the attributes of the instrument should be placed between the OPEN INSTRUMENT and CLOSE INSTRUMENT commands. There may be no more than 256 instrument definitions in a single hangar.

Instruments are given names so that when FLIGHT or the user needs to refer to the instrument, there is a meaningfull way of doing so. The real use of this is when a designer is laying out an instrument panel using the PLACE command. Designers refer to the instruments by name when using that command.

Only a single command type is permitted in an instrument definition. That command is identical to the type of the instrument. All descriptive information is available as qualifiers to that verb. For example, the DIAL instrument type only accepts the DIAL command (which has numerous qualifiers).

Table 2-6 shows the instrument types available.

Table 2-6 Instrument Types

Name	Description
Dial	Indicates a value using a rotating needle
Radio	Allows typed transmissions to be broadcast
Radar	Shows relative locations of nearby objects
Digital	Indicates a value using a numeric display
Control	Allows control of the object using the mouse
Horizon	Indicates object orientation
Slider	Indicates a value using a sliding needle
ADF	Indicates navigation information
VOR	Indicates navigation/landing information
Lamp	Indicates an object status
Toggle	A switch which controls an object function
Turn_Slip	Indicates changes of object orientation

QUALIFIERS *None*

EXAMPLES

```
OPEN INSTRUMENT=DIAL="Altimeter"
DIAL/...
DIAL/...
DIAL/...
CLOSE INSTRUMENT
```

The OPEN INSTRUMENT command begins the description of a Dial instrument called "Altimeter". The instrument is completed with a CLOSE INSTRUMENT command. Note that only the DIAL command is valid within a DIAL instrument.

OPEN PANEL

Begins an instrument panel definition

FORMAT

OPEN PANEL

PARAMETERS

None

DESCRIPTION

The OPEN PANEL command begins a new instrument panel definition. The CLOSE PANEL command indicates the completion of that panel definition. Commands describing the attributes of the panel should be placed between the OPEN PANEL and CLOSE PANEL commands. A display must be opened before an instrument panel can be defined. There may be no more than 8 instrument panel definitions in a single display.

Table 2-7 shows the commands used to describe an instrument panel.

Table 2-7 Commands Applicable to Instrument Panels

Commands	
PANEL	PLACE

QUALIFIERS

None

OPEN SUBOBJECT

Begins a subobject definition

FORMAT

OPEN SUBOBJECT="subobject-name"

PARAMETERS

None

DESCRIPTION

The OPEN SUBOBJECT command begins a new subobject definition. The CLOSE SUBOBJECT command indicates the completion of that subobject definition. Commands describing the attributes of the subobject should be placed between the OPEN SUBOBJECT and CLOSE SUBOBJECT commands.

For more information about subobjects, see the description of the SUBOBJECT command in this chapter.

Subobjects are given names so that when FLIGHT or the user needs to refer to the subobject, there is a meaningful way of doing so. One example of a need for this capability is for vectorable thrust, as is found in the AV-8B Harrier II. The thrust output of that object is bound to a subobject. The subobject that the thrust is bound to is specified by the subobject's name.

Up to 255 subobjects may be specified per parent object, and subobjects may be nested up to 16 levels deep. That is, an object may have may have 255 subobjects, and each of those may have 255 subobjects and so on, for 16 levels. Many thousands of subobjects may be created for a single object. However, that would surely slow the simulator to a crawl.

Table 2-8 lists the commands used to describe a subobject.

Table 2-8 Commands Applicable to Subobjects

Commands		
GEOMETRY	POSITIONING	SUBOBJECT

QUALIFIERS *None*

EXAMPLES

```
OPEN OBJECT="Spitfire"
.
.
.
   OPEN SUBOBJECT = "Right Wing"
   .
   .
   .
      OPEN SUBOBJECT = "Right Gear"
      .
      .
      .
      CLOSE SUBOBJECT
      OPEN SUBOBJECT = "Right Aileron"
      .
      .
      .
      CLOSE SUBOBJECT
   .
   .
   .
   CLOSE SUBOBJECT
.
.
.
CLOSE OBJECT
```

The OPEN OBJECT command begins the description of an object called "Spitfire". Within the description, a subobject is opened by use of the OPEN SUBOBJECT command to begin the description of the object’s right wing.

While that subobject is still open, two more subobjects are created. Because they are created within the confines of the right wing subobject, they are considered subobjects of the right wing. If the right wing moves for any reason, its subobjects are also affected. If the right wing is destroyed, so are all its subobjects.

Each subobject is closed as needed by the CLOSE SUBOBJECT command, and finally the object itself is closed by the CLOSE OBJECT command.

The indentation on the OPEN SUBOBJECT and CLOSE SUBOBJECT command are permitted, but not required.

PANEL

Specifies attributes of an instrument panel

FORMAT

PANEL width height xpos ypos

PARAMETERS

width height

Specifies the size of the instrument panel in centimeters.

xpos ypos

Specifies the position of the lower left corner of the instrument panel in centimeters from the lower left corner of the display.

Note that the top of an instrument panel is always visible. Therefore, if you create a large instrument panel and attempt to specify a lower left corner position which would cause the top of the window to be off the top of the display, the *ypos* argument will be ignored and the window will be placed such that its top is touching the top of the display.

DESCRIPTION

The PANEL command specifies attributes about the currently open instrument panel. An instrument panel definition is begun by use of the OPEN PANEL command.

QUALIFIERS

None

PILOTVIEW

Specifies the location of the pilot's eye in the cockpit relative to the geometry origin of the object.

FORMAT

PILOTVIEW x y z description

Qualifiers	**Defaults**
/BINDING="subobject-name"	*See Text*
/MAGNIFICATION=(value (,value...)	*/MAGNIFICATION=(1)*

PARAMETERS

x y z

Specifies the location of the viewpoint in the coordinate system of the root object, or if the binding qualifier is specified, the subobject.

description

The 'description' is used to label the world view window when the user is viewing from this viewpoint. Because the simulator tacks on a string to the end of the world view window label automatically, you should always place an extra blank on the end of your description string.

QUALIFIERS

/BINDING="subobject-name"

Specifies which subobject the viewpoint should be bound to. If the subobject moves or rotates, the pilotview position or orientation changes with it. The default is to use the coordinate system of the root object.

/MAGNIFICATION=(value (,value...))

Specifies up to 10 magnification levels which may be requested by a user during a simulation while viewing from this pilot view. The magnification levels may be in the range 1 through 250 inclusive. In the case of an object display with multiple viewports, only the viewport which is facing forward will be affected by a magnification change. All other viewports will use a magnification factor of 1.

Note that magnification levels do not need to be in increasing order, nor does the first magnification level need to be 1.

EXAMPLES

```
PILOTVIEW 10.0 3.0 0.1 "Maverick "
```

This would result in a label of "Maverick (forward)" while looking forward.

```
PILOTVIEW 10.0 3.0 0.1 "Maverick"
```

This would result in a label of "Maverick(forward)" which you may not find visually pleasing.

By default, the pilot view position is at the geometry origin of the object, a display magnification level of 1X is used, and the description is "Pilot View ".

```
PILOTVIEW 10.0 2.0 0.1 "Maverick "/MAGNIFICATION=(1, 4, 12)
```

This would result in the definition of a pilotview which provides three levels of magnification to the user looking out the viewport. They are 1X, 4X, and 12X. When the viewport is first displayed, the first magnification level is used by default (1X).

PLACE

Positions an instrument on the instrument panel of the object that is currently opened.

FORMAT

PLACE instrument-name x-position y-position

Qualifiers	**Defaults**
None	*None*

PARAMETERS

instrument-name

Specifies the name of the instrument to position onto the object's instrument panel. An instrument's name is specified on the OPEN INSTRUMENT command and is case-sensitive. Because of this case-sensitivity, the instrument name should be specified in double-quotes.

x-position

Specifies the horizontal placement of the instrument in centimeters from the left edge of the instrument panel. FLIGHT only retains two decimal places of precision in the placement value.

y-position

Specifies the vertical placement of the instrument in centimeters from the bottom edge of the instrument panel.

DESCRIPTION

The PLACE command allows the object's instrument panel to be laid out. Only those instruments which have been defined prior to the PLACE command may be reference by it. Instruments may be placed on top of each other or off of the instrument panel altogether and in any combination of instrument types, sizes, etc.

See also FLIGHT's hangar editor, which allows interactive placement of instruments. Note that the editor does not allow for addition/deletion of instruments, only changes in placement. Therefore, you must place a complete set of instruments somewhere on the instrument panel before running the editor.

QUALIFIERS *None*

EXAMPLES

```
PLACE "Altimeter" 12.04 3.55
```

This example command will cause the Altimeter instrument to be placed 3.55 centimeters from the bottom of the object's instrument panel and 12.04 centimeters from the left edge of the instrument panel.

POSITIONING

Specifies information about the movement of the subobject geometry.

FORMAT

POSITIONING x-position y-position z-position x-rotation y-rotation z-rotation

Qualifiers	**Defaults**
/VALUE_EQUIVALENT=value	*None.*
/INVISIBLE	*/NOINVISIBLE*
/LIFT=(lift-coeff-scale, coeff-offset)	*1.0, 0.0*
/DRAG=(drag-coeff-scale, coeff-offset)	*1.0, 0.0*
/AREA=square-feet	*0.0*

PARAMETERS

x-position y-position z-position

Specifies the position of a subobject at a particular subobject setting. This position is relative to the subobject's parent object's geometry origin. Because subobjects may be nested 4 levels, the parent of the subobject may be another subobject.

x-rotation y-rotation z-rotation

Specifies the rotation of a subobject at a particular subobject setting. As in the position, rotation is relative to the parent object's rotation. This is useful for having flaps on variable-sweep wing aircraft where the flaps must rotate up and down relative to the sweep position of the wing. This happens automatically for you (assuming that you have defined the subobject correctly).

DESCRIPTION

The POSITIONING command is used to specify between 1 and 5 settings that the subobject may occupy according to changes in the subobject's data channel. If the data channel specifies a value between two subobject settings, FLIGHT will automatically interpolate to produce intermediate positions, orientations and aerodynamic effects.

See the example section below to see how multiple settings work. Specifying a single subobject settings causes the subobject to remained fixed at that one location and orientation.

Subobjects can be used for many different types of animation on object, as evidenced by the object in the default hangars provided by FLIGHT.

QUALIFIERS

/VALUE_EQUIVALENT=value

Specifies what data channel value of this subobject should cause the subobject to assume this position and orientation. A subobject's data channel is specified by the SUBOBJECT command.

/INVISIBLE

Specifies that the subobject should become invisible when the data channel value feeding this subobject reaches or exceeds the VALUE_EQUIVALENT of this subobject. This can be useful for things such as animating landing gear which, when retracted, should disappear since they are inside the body of the object.

/LIFT=(lift-coeff-scale, coeff-offset)

Specifies that when the subobject reaches this limit of positioning that the object's lift aerodynamics should be modified according to the specified values. By default, the scale value is 1.0 and the offset is 0.0, meaning no effect.

This can be useful for subobjects which modify the object's lift characteristics, such as swing-wings.

/DRAG=(drag-coeff-scale, coeff-offset)

Specifies that when the subobject reaches this limit of positioning that the object's drag aerodynamics should be modified according to the specified values. By default, the scale value is 1.0 and the offset is 0.0, meaning no effect.

This can be useful for subobjects which modify the object's drag characteristics, such as landing gear and air brakes.

/AREA=square-feet

Specifies that when the subobject reaches this limit of positioning that the object's wing area should be added to by this many square feet. Negative values are valid. The value 0.0 is the default.

EXAMPLES

```
SUBOBJECT/DATA_CHANNEL=AIR_SPEED

POSITIONING 3.0 0.0 0.0 0.0 0.0 0.0/VALUE_EQUIVALENT=0.0     1

POSITIONING 6.0 0.0 0.0 0.0 0.0 0.0/VALUE_EQUIVALENT=10.0    2

POSITIONING 6.0 0.0 0.0 0.0 0.0 45.0/VALUE_EQUIVALENT=15.0   3
```

When the object is stopped (the air speed is zero), the subobject geometry will be 3 feet ahead of the parent object's origin with no rotation. As the air speed increases, the subobject will begin to move in the positive X direction. This is because the second setting changes from the first setting only in its X position. So the subobject will move forward from 3 feet ahead of the parent's origin to 6 feet ahead of that origin.

Once a speed of 10 knots is built up, the subobject will stop moving forward, but will start to rotate. This is because the X position is no longer changing between the second two subobject settings, but the Z rotation is changing. Therefore, as the air speed moves between 10 and 15 knots, the subobject will rotate through 45 degrees in the Z axis. Figure 2-1 shows how these positions work.

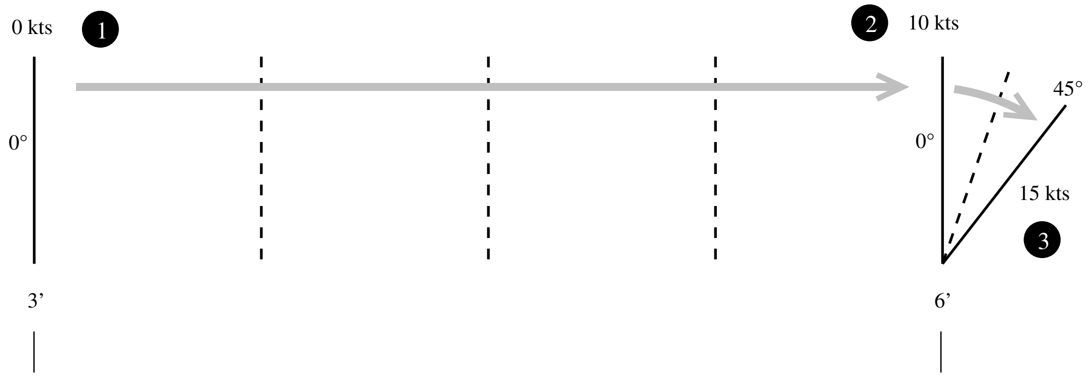


Figure 2-1 Position Interpolation

Note that when the air speed is greater than 15 knots, the position specified for 15 knots will be used.

```
POSITIONING 4.3 20.0 10.0  0.0 20.0 0.0/VALUE=0.0/LIFT=(0.8,0.0)
POSITIONING 4.3 20.0 10.0  0.0 20.0 0.0/VALUE=10.0/LIFT=(1.0,0.0)
```

In this example, there are no geometric changes because the POSITIONING commands both specify the exact same position and rotation information. However, a lift modifier has been specified.

In the same way that geometric changes are interpolated, so are the lift, drag and wing area modifications. At a value of 0.0, this subobject will cause a 20% reduction in the object's lifting capability. At a value of 10.0, this subobject will have no effect on the object's lifting capability. At intermediate positions, it will have intermediate effects.

Note that the subobject has a 20% reducing effect when the value is 0.0. This is because the scale modifier is 0.80. If two subobjects of this sort were created, then at a value of 0.0, the scale modifier for both would be 0.80. This would result in 1.0 - 0.20 - 0.20 = 0.60, or 60% of the normal lift capability; a 40% reduction.

If the POSITIONING commands had specified that an increase in lifting capability were created by this subobject instead of a reduction, the lift scale value would be greater than 1.0. If the value were 1.30, the lift coefficient would be scaled up by 30%. If there were four subobjects with a lift scale value of 1.30, then the resulting increase in lift would be 1.0 + 0.30 + 0.30 + 0.30 + 0.30 = 2.20, or 220% of the normal lift capabilty; a 120% increase.

RADAR

Specifies a component of a RADAR-type instrument definition within the confines of OPEN INSTRUMENT and CLOSE INSTRUMENT. The RADAR command is not valid for non-RADAR instruments.

FORMAT

RADAR

Qualifiers	Defaults
/RADIUS=coverage-in-feet	*4.0*
/SCOPE_RADIUS=display-size-in-cm	*26400.0*
~~*/TRACKS=(class-a, class-b, ...class-n)*~~	~~*None.*~~
/RESOLUTION=ratio	*0.001*
/COVERAGE=(v-min,v-max,h-min-hmax)	*(-180,180,-180,180)*
/I_LABEL=(label,location,radius,xscale,yscale)	*None.*

PARAMETERS

None

QUALIFIERS

/RADIUS=coverage-in-feet

Specifies how large an area is represented on the radar display. A value of 5280 would mean that the radius monitors traffic within one mile of the vehicle.

/SCOPE RADIUS=display-size-in-cm

Specifies how large the radar display is on the instrument panel, measured in centimeters.

~~*/TRACKS=(class-a, class-b, ...class-n)*~~

~~Specifies which radar classes this radar instrument can detect. For example TRACKS=(1, 2) means that objects of radar classes 1 and 2 can be seen on this radar, but objects of class 3 cannot be seen. The radar class of an object is determined by the rclass command. Valid radar classes are 1 through 10.~~

~~Note that an object appears as a blip, regardless of its radar class.~~

/RESOLUTION=ratio

Specifies the resolving ability of the radar unit. For a given object cross section **C** (defined by the object's SIGNATURE RADAR commands) at a distance **D**, the radar unit will detect the object if the resolution value is greater than $\mathbf{C} / (2 * \mathbf{D}^2)$. Note that an object's cross section is dependant upon the orientation of the object relative to the radar unit. That is, a head-on view of an aircraft presents a smaller cross section than an overhead view.

/I_LABEL=(title, position, radius, xscale, yscale)

Specifies the position, size, and title of a label positioned around the circumference of the RADAR instrument. The angular position is expressed in degrees. The 'radius' value controls where the text is written, in terms of distance out from the dial center. A magnitude of 100 equals the radius of the dial face; values larger than 100 may be specified. The title string is centered at that point. The text character size is specified by the two scale values. The magnitude of the size values is based on 100 units being equal to the dial's radius. (If the scale values are omitted, the label is drawn at a standard, fixed size.)

Up to 40 labels may be placed on a single RADAR instrument, but each must be placed by a separate RADAR/I_LABEL command.

/COVERAGE=(v-min, v-max, h-min, h-max)

Specifies the coverage of the radar instrument relative to the forward heading of the aircraft. The list of values specifies the vertical and horizontal coverage of the radar instrument in degrees. Negative vertical values indicate angles below the aircraft, while positive vertical values indicate angles above the aircraft. Negative horizontal values indicate angles to the left of the aircraft, while positive horizontal values indicate angles to the right of the aircraft. The *min* values must be less than the *max* values.

The shape of the radar instrument is taken from the horizontal coverage values, indicating visually what area the radar instrument is covering. Note that radar instruments are fixed to the orientation of the aircraft.

RADIO

Specifies a component of a RADIO-type instrument definition within the confines of OPEN INSTRUMENT and CLOSE INSTRUMENT. The RADIO command is not valid for non-RADIO instruments.

FORMAT

RADIO

Qualifiers	**Defaults**
/(NO)BORDER	*None.*
/DECIMALS=decimal-digit-count	*None.*
/DIGITS=digit-count	*None.*
/DIVISOR=value	*None.*
/INITIAL_FREQUENCY=freq-in-Hz	*None.*
/MAXFREQUENCY=freq-in-Hz	*None.*
/MESSAGE_LENGTH=number-characters	*None.*
/MINFREQUENCY=freq-in-Hz	*None.*
/BUTTON=(xpos, ypos, width, height, fill, label, xscale, yscale)	*None.*
/C_LABEL=(title, xpos, ypos, xscale, yscale)	*None.*
/KNOB=(xpos, ypos, radius, fill, label, xscale, yscale)	*None.*
/FILL=fill-percentage	*/FILL=50*
/LINE_SCALE=height-percentage	*None.*
/FREQUENCY_SCALE=height-percentage	*None.*

PARAMETERS

None

QUALIFIERS

/(NO)BORDER

Specifies that a fat border should be drawn around the RADIO window.

/DECIMALS=decimal-digit-count

Specifies how many decimal digits to the right of the decimal point should be displayed in the frequency readout. A value of 0 omits the decimal point altogether.

/DIGITS=digit-count

Specifies how many total digits are displayed on the frequency readout. This count should include digits to the left of the decimal point, the decimal point (if present), and digits to the right of the decimal point.

/DIVISOR=value

The radio provides a digital readout of the frequency that the unit is currently tuned to. The actual number that is displayed is computed by dividing the frequency in Hertz by the DIVISOR value. For example, DIVISOR=1000000 would make the display be in mHz.

/INITIAL_FREQUENCY=freq-in-Hz

Controls what frequency the radio is tuned to when the object is first entered.

/MAXFREQUENCY=freq-in-Hz

Specifies the highest frequency to which the radio may be tuned. Attempts to tune to higher frequencies will be ignored. The value is specified in Hertz.

/MESSAGE_LENGTH=number-characters

Specifies the width of a RADIO window in number of characters.

/MINFREQUENCY=freq-in-Hz

Specifies the lowest frequency to which the radio may be tuned. Attempts to tune to lower frequencies will be ignored. The value is specified in Hertz.

/BUTTON=(xpos, ypos, width, height, fill, label, xscale, yscale)

Adds a button graphic to the RADIO console. The button is purely for aesthetics; it is non-functional. You may specify up to 10 buttons per RADIO console.

'xpos' and 'ypos' specify the location of the center of the button. 'width' and 'height' specify the size of the button image. 'fill' controls how the interior of the button is drawn; the value is interpreted as a gray-level percentage, where 0 is empty and 100 is solid filled. 'label' specifies a text string which is drawn centered above the button (up to six characters). The text character size is specified by the two scale values. The magnitude of all dimension values is based on 100 units being equal to the console's height. (If the scale values are omitted, the label is drawn at a standard, fixed size.)

/C_LABEL=(title, xpos, ypos, xscale, yscale)

Specifies the position, size, and title of a label positioned on the interior of the RADIO console. 'xpos' and 'ypos' specify the center of the label. The text character size is specified by the two scale values. The magnitude of all dimension values is based on 100 units being equal to the console's height. (If the scale values are omitted, the label is drawn at a standard, fixed size.)

Up to 10 labels may be placed on a single RADIO console, but each must be placed by a separate RADIO/C_LABEL command.

/KNOB=(xpos, ypos, radius, fill, label, xscale, yscale)

Adds a knob graphic to the RADIO console. The knob is purely for aesthetics; it is non-functional. You may specify up to 10 knobs per RADIO console.

'xpos' and 'ypos' specify the location of the center of the knob. 'radius' specifies the radial dimension of the knob image. 'fill' controls how the interior of the knob is drawn; the value is interpreted as a gray-level percentage, where 0 is empty and 100 is solid filled. 'label' specifies a text string which is drawn centered below the knob (up to six characters). The text character size is specified by the two scale values. The magnitude of all dimension values is based on 100 units being equal to the console's height. (If the scale values are omitted, the label is drawn at a standard, fixed size.)

/FILL=fill-percentage

Specifies a fill value for background to the entire radio console as a gray-level percentage, where 0 is empty and 100 is solid filled. The default is a 50% gray fill.

/LINE_SCALE=height-percentage

Specifies a scale value for the two broadcast lines of the radio. The magnitude of this scale value is based on 100 units being equal to the radio console's height. If omitted, a standard, fixed size is chosen.

/FREQUENCY_SCALE=height-percentage

Specifies a scale value for the frequency indicator of the radio. The magnitude of this scale value is based on 100 units being equal to the radio console's height. If omitted, a standard, fixed size is chosen.

~~RCLASS~~

~~Specifies information about the radar detectability of an object.~~

~~FORMAT~~

~~RCLASS~~ ~~radar-class blip-type~~

~~**Qualifiers**~~	~~**Defaults**~~
~~*None*~~	~~*None*~~

~~PARAMETERS~~

~~*radar-class*~~

~~Specifies the radar class of the object. A radar class is a unitless number which is merely an arbitrary classification of the radar detectability of the object. The higher the radar class, the more difficult it is to detect the object. Valid radar classes are 1 through 10.~~

~~The RADAR/TRACKS command (specified when a radar instrument is being defined) is the counterpart to this command. If the radar class of an object is not listed in the RADAR/TRACKS value of the radar instrument, then the object will not be detected on the radar instrument.~~

~~*blip-type*~~

~~Specifies how the object will appear on the radar scope where it is detected.~~

~~*Note that in the present version of the software, all objects will appear as blips.*~~

~~Table 2-9 shows the list of possible blip types.~~

Table 2-9 ~~**Blip Type Keywords**~~

~~**Keyword**~~	~~**Description**~~
~~CHARACTER=character~~	~~Specifies that a single character should be used to indicate the position of the tracked object~~
~~BLIP~~	~~Specifies that a single point should be used to indicate the position of the tracked object~~

~~DESCRIPTION~~

~~The RCLASS command specifies information about the radar detectability of the currently open object.~~

~~QUALIFIERS~~

~~*None*~~

~~EXAMPLES~~

~~`RCLASS 1 CHARACTER="+"`~~

~~The radar class of the object described by the example RCLASS command is 1, by convention indicating that it is easily detectable on most radar scopes.~~

~~`RCLASS 10 BLIP`~~

~~The radar class of the object described by the example RCLASS command is 10, by convention indicating that it is nearly impossible to detect by radar.~~

REACTION_CONTROL

Specifies information about the reaction control system of the aircraft.

FORMAT

REACTION_CONTROL

Qualifiers	**Defaults**
/CONTROL=knots	*None*
/ROLL=degrees-per-second	*/ROLL=0.0*
/YAW=degrees-per-second	*/YAW=0.0*
/PITCH=degrees-per-second	*/PITCH=0.0*

PARAMETERS

None

QUALIFIERS

/CONTROL=knots

Specifies the control speed observed by the aircraft's reaction control system (RCS). Below this speed, the reaction control system automatically becomes active, providing the specified roll, pitch and yaw rates. In contrast to aerodynamic control rates, these rates do not change as a function of speed or altitude.

/ROLL=degrees-per-second

Specifies the maximum roll rate that the RCS system provides while the RCS system is active.

/YAW=degrees-per-second

Specifies the maximum yaw rate that the RCS system provides while the RCS system is active.

/PITCH=degrees-per-second

Specifies the maximum pitch rate that the RCS system provides while the RCS system is active.

SIGNATURE RADAR

Specifies the radar signature of an object or subobject

FORMAT

SIGNATURE RADAR *forward,side,top*

Qualifiers	**Defaults**
None	*None*

PARAMETERS

forward,side,top

Specifies the cross section of the object in square feet in each of three orthogonal views of the object or subobject. Therefore, the *forward* value indicates the cross section of the object as viewed from its front, the *side* value indicates the cross section of the object as viewd from the side, etc. Note that the cross section of the rear of the object is assumed to be the same as the front, the two sides are symmetrical, and the bottom the same as the top.

When the object is viewed from a location which is not perpendicular to one of the above 'faces', FLIGHT interpolates according to the view angle.

QUALIFIERS

None

SLIDER

Specifies a component of a SLIDER-type instrument definition within the confines of OPEN INSTRUMENT and CLOSE INSTRUMENT. The SLIDER command is not valid for non-SLIDER instruments.

FORMAT

SLIDER

Qualifiers	Defaults
/(NO)BORDER	*None*
/TITLE="title"	*None*
/BIGTICKS=number	*None*
/SMLTICKS=number	*None*
/LABEL=(title, xpos, ypos, scale)	*None*
/DATA=data-type	*None*
/MAXVALUE=value	*None*
/MINVALUE=value	*None*
/HEIGHT=slider-height-in-centimeters	*None*
/WIDTH=slider-width-in-centimeters	*None*
/PRECISION=number	*None*
/(NO)FILL	*/NOFILL*
/TICKS=(number, start, end, minx, maxx, height)	*None*

PARAMETERS

None

QUALIFIERS

/(NO)BORDER

Specifies that a fat border should be drawn around the SLIDER.

/TITLE="title"

Specifies the string which will be displayed under the SLIDER.

/BIGTICKS=number

Specifies the number of large tick marks drawn along the inside right edge of the SLIDER. The number of tick marks are spread evenly through the full height of the SLIDER.

/SMLTICKS=number

Specifies the number of the small tick marks drawn along the inside right edge of the SLIDER. The number of tick marks are spread evenly through the full height of the SLIDER.

/LABEL=(title, xpos, ypos, scale)

Specifies the position and title of a label. The xpos specifies where on the vertical axis of the slider that the label appears. It is expressed in terms of the slider's MINVALUE and MAXVALUE. Ypos specifies where the label appears horizontally. It is expressed as a percentage of the width of the slider. Negative values place the label to the left of the slider. The label is drawn such that the left edge of the slider is positioned at the stated position.

Up to 40 labels may be placed on a single slider, but each must be placed by a separate SLIDER/LABEL command.

/DATA=data-type

Specifies the parameter of the simulation that should be displayed in the SLIDER. For more information about the available data types which can be used to drive the instrument, see Section 2.4.

/MAXVALUE=value

Specifies the largest value which can be displayed on the SLIDER.

/MINVALUE=value

Specifies the smallest value which can be displayed on the SLIDER.

/HEIGHT=slider-height-in-centimeters

Specifies the height in centimeters of the SLIDER.

/WIDTH=slider-width-in-centimeters

Specifies the width in centimeters of the SLIDER.

/PRECISION=number

Specifies the amount of change required in the DATA type feeding the SLIDER instrument before a screen update takes place to reflect the change. For example, a PRECISION=1 value indicates that a change of 10.0 or greater will cause an update to the instrument. A PRECISION=-3 value indicates that a change of 0.001 or greater will cause an update to the instrument.

/(NO)FILL

Specifies whether or not the slider should fill the area below the slider needle. The default is /NOFILL.

/TICKS=(number, start, end, minx, maxx, height)

Places a generalized set of ticks vertically in the SLIDER. 'Number' controls how many tick marks there are in the set. 'Start' and 'end' specify the vertical location where the marks are drawn (specified in the dial's BASE value units). The tick marks are evenly distributed in this region, inclusive of ends (that is, the first tick is located at 'start' and the last at 'end'). The minx/maxx parameters control how wide the tick is (100 tick units equals the width of the slider face). 'Height' controls how thick the tick marks are (same units as the radius parameters).

Up to 9 tick sets may be specified per SLIDER.

STRUCTURE

Specifies mechanical structural details of an object. The first parameter to the STRUCTURE command is a keyword specifying which element of the structure is being described.

FORMAT

STRUCTURE FUSELAGE, WING

Qualifiers	Defaults
/ROOT_CHORD=feet	*None.*
/RADIUS=feet	*None.*
/TIP_CHORD=feet	*None.*
/GLOAD=gravities	*None.*
/LENGTH=feet	*None.*
/SPAN=feet	*None.*
/REST_ANGLE=degrees	*0.*
/PIVOT_POINT=feet	*None.*

PARAMETERS

STRUCTURE FUSELAGE

Specifies details of the structure of the object's fuselage.

QUALIFIERS

/RADIUS=feet

Specifies the mean radius of the fuselage.

/LENGTH=feet

Specifies the length of the fuselage.

/REST_ANGLE=degrees

Specifies the pitch angle (in degrees) at which the object should sit when stopped on a surface, relative to the surface. A value of 0 causes the object to be level (parallel to the terrain or water surface) when stopped. Negative values result in a nose-up/tail-down stance. Positive values result in a nose-down/tail-up stance.

/PIVOT_POINT=feet

Specifies the location (in feet, in the X axis relative to the object's origin) of the point at which the object pivots in pitch while on a surface. Normally this is the location of the aircraft's main gear. A typical tricycle-gear aircraft would have a negative value (because the main gear are behind the center of gravity) such as -3.0 feet. A typical tail-dragger aircraft would have a positive value such as 2.0 feet.

PARAMETERS

STRUCTURE WING

Specifies details of the structure of the object's wings.

QUALIFIERS

/ROOT_CHORD=feet

Specifies the chord of the wing, measured at the wing root.

/TIP_CHORD=feet

Specifies the chord of the wing, measured at the wing tip.

/GLOAD=gravities

Specifies the maximum number of vertical g's which the wing can withstand before wing failure. During a simulation, a warning tone will sound when the pilot induces more than half of this number of g's.

When the object experiences twice the GLOAD value (in any direction), the object will be destroyed.

/SPAN=feet

Specifies the wingspan of the aircraft, measured from wingtip to wingtip.

EXAMPLES

```
STRUCTURE FUSELAGE/PIVOT_POINT=-2.5
```

This example describes a tricycle-gear aircraft whose main gear wheels contact the ground 2.5 feet behind the aircraft's center of gravity (origin of geometry).

```
STRUCTURE FUSELAGE/REST_ANGLE=-8
STRUCTURE FUSELAGE/PIVOT_POINT=1.6
```

This example describes a conventional (tail-dragger) aircraft whose main wheels contact the ground 1.6 feet ahead of the aircraft's center of gravity, and whose fuselage ground stance is 8 degrees nose-up.

SUBOBJECT

Specifies information about a subobject of an object.

FORMAT

SUBOBJECT

Qualifiers	**Defaults**
/DATA_CHANNEL=data-channel	*None.*
/POSITIONS=number	*/POSITIONS=20*
/GROUP=number	*/GROUP=1*
/CHARACTERISTICS=(list)	*None*

PARAMETERS

None

DESCRIPTION

The SUBOBJECT command specifies information about the most recently opened subobject of the currently open object.

QUALIFIERS

/DATA_CHANNEL=data-channel

Specifies the data value which should be used to control the positioning and rotation of the subobject at simulation time. The data channel is computed once every display frame and the position and rotation of the subobject is computed based on that value. This is how animation of subobjects (as well as other effects) is achieved. For more information about positioning, see the description for the POSITIONING command.

Table 2-2 shows the list of possible data channel feeds for subobjects.

/POSITIONS=number

Specifies the number of positions to which the current subobject will interpolate to. The limits of interpolation are established by the POSITIONING command. This value should be set as low as is convenient due to performance considerations.

/GROUP=number

Specifies the subobject group that this subobject belongs to. When determining whether or not a subobject is visible, FLIGHT will look up through the subobject's parents and try to find a parent which is both invisible *and* in the same group as the subobject.

The root object is always in group 1.

/CHARACTERISTICS=(list)

Specifies the list of subobject characteristics which apply to the currently open subobject.

Table 2-10 shows the list of subobject characteristics

Table 2-10 Subobject Characteristics

Keyword	Description
PRIVATE	Specifies that this subobject is only to be managed locally. No other simulation client or the FLIGHT server is to be aware of this subobject. This is an optimization which should be taken advantage of for components of a model which are not significant to other clients. For example, the HUD geomtry of aircraft is a prime candidate for having the PRIVATE attribute. Other candidates would be similar, small or barely noticeable components of the aircraft. This attribute is inherited by a subobject's subobjects. In other words, subobjects of a subobject with the PRIVATE subobject are also considered to be private.

THRUST

Specifies information about the object's engine thrust output.

FORMAT

THRUST

Qualifiers	**Defaults**
/DRY=pounds	*None.*
/AFTERBURNER=pounds	*None.*
/INCREMENT=pounds	*None.*
/DELTA_TIME=seconds	*/DELTA_TIME=3.0*
/FLAMEOUT_ALTITUDE=feet	*/FLAMEOUT_ALTITUDE=45000*
/BINDING=subobject-name	*see text*
/ENGINE_TYPE=engine-type	*/ENGINE_TYPE=PISTON*

PARAMETERS

None

DESCRIPTION

The THRUST command allows for specification of the currently opened object's engine thrust output.

Note that FLIGHT differentiates between in-air, on-terrain (land), and on-water engine performance. Be sure to set the appropriate characteristic(s) on the CHARACTERISTICS command corresponding to the regime(s) that your object is capable of operating in.

QUALIFIERS

/DRY=pounds

Specifies the maximum amount of thrust normally available from the engine. At the dry thrust level, the engine consumes fuel at the rate specified by the FUEL/CONSUMPTION command.

/AFTERBURNER=pounds

Specifies the amount of thrust available from the engine when its afterburners are turned on. Jet engines often have afterburners, but piston engines never do. Piston engines may have turbocharging, but FLIGHT makes no provision for turbocharging. At the afterburner thrust level, the engine consumes fuel at triple the rate specified by the FUEL/CONSUMPTION command.

~~In order to make afterburners available, you must also specify the AFTERBURNER characteristic using the CHARACTERISTICS command.~~

/INCREMENT=pounds

Specifies the amount of thrust by which the engine increases for each engine throttle setting.

/DELTA_TIME=seconds

Specifies the minimum time it takes for the engine to accelerate from zero to dry thrust.

/FLAMEOUT_ALTITUDE=feet

Specifies the maximum service ceiling of the engine. When operated at altitude above this, the engine does not produce thrust.

/BINDING=subobject-name

Specifies the subobject to which the thrust direction should be bound. Engine exhaust is expelled along the negative X axis of the subobject.

The default binding is to the root object..

/ENGINE_TYPE=engine-type

Specifies the type of engine which generates the thrust. FLIGHT offers six types of engines.

- `REACTION`
- `PISTON`
- `ROTARY_PISTON`
- `TURBO_PISTON`
- `TURBOFAN`
- `TURBOJET`

Engine effectiveness, except for the REACTION engine, are attenuated according to altitude.

TOGGLE

Specifies a component of a TOGGLE-type instrument definition within the confines of OPEN INSTRUMENT and CLOSE INSTRUMENT. The TOGGLE command is not valid for non-TOGGLE instruments.

FORMAT

TOGGLE

Qualifiers	**Defaults**
/T_LABEL=text-string	*None.*
/B_LABEL=text-string	*None.*
/L_LABLE=text-string	*None.*
/R_LABEL=text-string	*None.*
/ON_POSITION=UP | DOWN	*Up*
/ON_FUNCTION="command-string"	*None.*
/OFF_FUNCTION="command-string"	*None.*

PARAMETERS

None

QUALIFIERS

/T_LABEL=text-string

Specifies the label that should be centered over the top of the TOGGLE instrument.

/B_LABEL=text-string

Specifies the label that should be centered underneath the TOGGLE instrument.

/L_LABEL=text-string

Specifies the label that should be placed to the immediate left of the TOGGLE instrument.

/R_LABEL=text-string

Specifies the label that should be placed to the immediate right of the TOGGLE instrument.

/ON_POSITION= UP | DOWN

Specifies whether the TOGGLE is in the on position while the physical toggle switch is in the UP or DOWN position. The default is UP.

/ON_FUNCTION="command-string"

Specifies a simulator function to invoke as a result of the pilot flipping this toggle to the ON position. Up to 32 ON_FUNCTION commands may be specified per toggle. For more information about the available commands, see *The FLIGHT Flight Simulator*.

/OFF FUNCTION="command-string"

Specifies a simulator function to invoke as a result of the pilot flipping this toggle to the OFF position. Up to 32 OFF_FUNCTION commands may be specified per toggle. For more information about the available commands, see *The FLIGHT Flight Simulator*.

TURN_SLIP

Specifies a component of a TURN_SLIP-type instrument definition within the confines of OPEN INSTRUMENT and CLOSE INSTRUMENT. The TURN_SLIP command is not valid for non-TURN_SLIP instruments.

FORMAT

TURN_SLIP

Qualifiers	**Defaults**
/RADIUS=size-in-cm	*2.0 cm*
/GYRO_ANGLE=angle-in-degrees	*0 degrees*
/(NO)FILLED	*FILLED.*
/B_ANNOTATION	*None.*
/X_ANNOTATION=(xvalue, xvalue2, ...)	*None.*
/Y_ANNOTATION=(yvalue, yvalue2, ...)	*None.*
/ANNOT_PATTERN=gray-level	*100%*
/LABEL=(text, x, y,width,height)	*None.*

PARAMETERS

None

QUALIFIERS

/RADIUS=size-in-cm

Specifies the radial dimension of a TURN_SLIP instrument in centimeters.

/GYRO_ANGLE=angle-in-degrees

Specifies the orientation of the simulated gyroscope inside the simulated TURN_SLIP instrument. The angle represents the angle of the gimballing frame in which the gyro is mounted. 0 means horizontal (the instrument would only respond to yaw rate); 90 means vertical (the instrument would only respond to roll rate). Intermediate angles are allowed; e.g. 45 would cause the instrument to respond in equal amounts to both yaw rate and roll rate.

/(NO)FILLED

Specifies whether the needle and the ball of the instrument are drawn as solid filled regions, or are drawn as hollow shapes with a thin outline.

/B_ANNOTATION

Specifies the beginning of a new annotation (also automatically completes any current annotation). For a complete description of annotations, see the help for XANNOTATION.

/X_ANNOTATION=(xvalue1, xvalue2, ...)

Specifies a list of X-coordinate values for the current annotation.

A TURN_SLIP instrument may have a number of annotations drawn on it. Annotations may be created to represent various parts of the instrument, for example the "dog-house" needle deflection markers on some turn & slip instruments, simulated adjustment knobs, etc. All annotations are drawn with the halftone fill pattern specified by the /ANNOT_PATTERN qualifier.

There are three TURN_SLIP command qualifiers which together construct annotations: B_ANNOTATION, which specifies the beginning of a new annotation, and the X_ANNOTATION, Y_ANNOTATION pair which supply X and Y-coordinate values that define the shape and position of the annotation. The coordinate system which the X and Y values are based on is as follows: the origin is at the needle origin location (see TURN_SLIP NEEDLE), and the radius of the overall instrument represents a size of 100 units.

There can be any number of separate annotations, but the total number of X coordinate values specified for all annotations for a single TURN_SLIP instrument can be at most 99 (similarly you can specify up to 99 total Y values).

/YANNOTATION=(yvalue1, yvalue2, ...)

Specifies a list of Y-coordinate values for the current annotation (see XANNOTATION).

/ANNOT_PATTERN=gray-level

Specifies the gray-level halftone pattern which is used to draw all the annotations in the instrument. 0 means all background, 100 means all foreground.

/LABEL=(text, xposition, yposition, xscale, yscale)

Specifies the position, size, and title of a label positioned in the instrument. The position values control where the text is written, in terms of distance from the needle origin location. A magnitude of 100 equals the radius of the instrument; values larger than 100 may be specified. The text string is centered at that point. The text character size is specified by the two scale values. The magnitude of the size values is based on 100 units being equal to the instrument's radius.

Up to 10 labels may be placed in a single TURN_SLIP instrument, but each must be placed by a separate TURN_SLIP/LABEL command.

EXAMPLES

The following set of sample commands creates a typical turn & slip instrument:

```
OPEN INSTRUMENT=TURN_SLIP=TURN_COORDINATOR
TURN_SLIP/RADIUS=2.0
TURN_SLIP/B_ANNOTATION
TURN_SLIP/XANNOTATION=(61,81,81,61)
TURN_SLIP/YANNOTATION=(3,3,-3,-3)
TURN_SLIP/B_ANNOTATION
TURN_SLIP/XANNOTATION=(58,77,75,56)
TURN_SLIP/YANNOTATION=(-19,-25,-31,-25)
TURN_SLIP/B_ANNOTATION
TURN_SLIP/XANNOTATION=(-61,-81,-81,-61)
TURN_SLIP/YANNOTATION=(3,3,-3,-3)
TURN_SLIP/B_ANNOTATION
TURN_SLIP/XANNOTATION=(-58,-77,-75,-56)
TURN_SLIP/YANNOTATION=(-19,-25,-31,-25)
TURN_SLIP/LABEL=("TURN", 0, -16, 4, 8)
TURN_SLIP/LABEL=("COORDINATOR", 0, -30, 4, 8)
TURN_SLIP/LABEL=("2 MIN", 0, -44, 8, 14)
TURN_SLIP/LABEL=("L", -64, -41, 8, 14)
TURN_SLIP/LABEL=("R", 64, -41, 8, 14)
TURN_SLIP/GYRO_ANGLE=30
TURN_SLIP NEEDLE/XNEEDLE=(3,3,9,25,25,56,56,9,7,0,-7,-9,-56,-56,-25,-25,-9,-3,-3)
TURN_SLIP NEEDLE/YNEEDLE=(23,11,6,6,3,3,-3,-4,-7,-11,-7,-4,-3,3,3,6,6,11,23)
TURN_SLIP NEEDLE/RATE=60
TURN_SLIP NEEDLE/MAXROT=42
TURN_SLIP NEEDLE/MAXYAW=6
TURN_SLIP BALL/RADIUS=13
TURN_SLIP BALL/TUBE_RADIUS=280
```

```
TURN_SLIP BALL/YORIGIN=210
TURN_SLIP BALL/ARC=6
CLOSE INSTRUMENT
```

TURN_SLIP BALL

Specifies a ball-related component of a TURN_SLIP-type instrument definition within the confines of OPEN INSTRUMENT and CLOSE INSTRUMENT. The TURN_SLIP BALL command is not valid for non-TURN_SLIP instruments.

FORMAT

TURN_SLIP BALL

Qualifiers	**Defaults**
/RADIUS=size	*10%*
/TUBE_RADIUS=size	*150%*
/PATTERN=gray-level	*100%*
/RATE=degrees-per-second	*10 degrees per second*
/XORIGIN=position	*0%*
/YORIGIN=position	*0%*
/ARC=angle-in-degrees	*20 degrees*

PARAMETERS

None

QUALIFIERS

/RADIUS=size

Specifies the radial dimension of the ball in a TURN_SLIP instrument as a percentage of the overall instrument's radius. For example, BALL/RADIUS of 100 would make the ball the same size as the total instrument.

/TUBE_RADIUS=size

Specifies the bending radius of the glass tube in which the ball floats. (The radius of the glass tube itself is derived from the radius of the ball.) See figure 2-2 for an illustration of the dimensions.

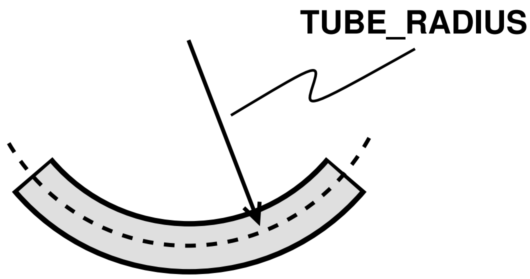


Figure 2-2 TURN_SLIP Tube Dimensions

/PATTERN=gray-level

If the ball is drawn filled, this qualifier determines the gray level of the fill pattern used to draw the ball. 0 is all background, 100 is all foreground.

/RATE=degrees-per-second

Specifies an upper limit on the rate of angular travel of the ball, in degrees per second (considering the ball motion as sweeping out an arc, constrained by the glass tube). A value of 10 means that the ball cannot move in the tube faster than 10 degrees per second (in either direction), regardless of the forces acting to move the ball.

/XORIGIN=position

Specifies the X coordinate of where the center of the ball's bent glass tube is. A value of zero means that the glass tube's origin is horizontally aligned with the center of the overall instrument. Values greater than zero displace the tube to the right. The position value is specified as a percentage of the overall instrument radius.

/YORIGIN=position

Specifies the Y coordinate of where the center of the ball's bent glass tube is. A value of zero means that the glass tube's origin is vertically aligned with the center of the overall instrument. Values greater than zero displace the tube up. The position value is specified as a percentage of the overall instrument radius.

/ARC=angle-in-degrees

Specifies the extent of the ball's bent glass tube, as an arc of a circle. The tube will extend to the left and right by the specified value (e.g. /ARC=10 results in a tube which extends through 20 degrees overall).

TURN_SLIP NEEDLE

Specifies a needle-related component of a TURN_SLIP-type instrument definition within the confines of OPEN INSTRUMENT and CLOSE INSTRUMENT. The TURN_SLIP NEEDLE command is not valid for non-TURN_SLIP instruments.

FORMAT

TURN_SLIP NEEDLE

Qualifiers	**Defaults**
/XNEEDLE=(xvalue1, xvalue2,...)	*None.*
/YNEEDLE=(yvalue1, yvalue2,...)	*None.*
/PATTERN=gray-level	*100%*
/RATE=degrees-per-second	*10 degrees per second*
/XORIGIN=position	*0%*
/YORIGIN=position	*0%*
/MAXROT=angle-in-degrees	*20 degrees*
/MAXYAW=degrees-per-second	*6 degrees per second*

PARAMETERS

None

QUALIFIERS

/XNEEDLE=(xvalue1, xvalue2,...)

Specifies an ordered sequence of X coordinate positions which define the shape of the needle. Up to 99 coordinates may be specified. This information, combined with the /YNEEDLE data, defines a polygon which represends the instrument's turn rate indicator needle. A single TURN_SLIP instrument must have exactly one needle.

/YNEEDLE=(yvalue1, yvalue2,...)

Specifies an ordered sequence of Y coordinate positions which define the shape of the needle. Up to 99 coordinates may be specified.

/PATTERN=gray-level

If the needle is drawn filled, this qualifier determines the gray level of the fill pattern used to draw the needle. 0 is all background, 100 is all foreground.

/RATE=degrees-per-second

Specifies an upper limit on the rate of rotation of the needle, in degrees per second. A value of 10 means that the needle cannot rotate faster than 10 degrees per second (clockwise or counter-clockwise), regardless of the turn rates acting to rotate the needle.

/XORIGIN=position

Specifies the X coordinate of where the center of the needle's geometry (and rotation point) is. A value of zero means that the needle's origin is horizontally aligned with the center of the overall instrument. Values greater than zero displace the needle to the right. The position value is specified as a percentage of the overall instrument radius.

The instrument's /LABEL labels are also mapped in the needle's coordinate system (i.e., they move as the needle's origin moves).

/YORIGIN=position

Specifies the Y coordinate of where the center of the needle's geometry (and rotation point) is. A value of zero means that the needle's origin is vertically aligned with the center of the overall instrument. Values greater than zero displace the needle up. The position value is specified as a percentage of the overall instrument radius.

/MAXROT=angle-in-degrees

Specifies the limits of rotation of the needle, to either side of center (i.e., /MAXROT=20 would allow the needle to rotate within the range of +20 to -20 degrees).

/MAXYAW=degrees-per-second

Specifies the turn rate corresponding to the needle's MAXROT indication. For example if the needle should deflect 10 degrees when a 3 degree per second turn is experienced, and the needle is limited to a maximum of 20 degrees deflection, then the MAXYAW value should be set to 6 (i.e. a turn rate of 6 degrees per second corresponds to a 20 degree needle deflection).

USERVALUE

Specifies information about one of the user data channel values maintained for the current object.

FORMAT

USERVALUE user-value-number

Qualifiers	**Defaults**
/INITIAL_VALUE=value	*/INITIAL_VALUE=0.0*
/CYCLE=(min, max)	*/CYCLE=*$(-2^{32}-1, 2^{32}-1)$

PARAMETERS

user-value-number

Specifies which of the 16 user values should have its attributes modified. Specify only a number. Do not specify a data channel keyword, such as USER1, USER2, etc.

DESCRIPTION

The USERVALUE command allows control over the attributes of a user-controlled data channel. Any one USERVALUE command can modify only a single data channel. However, any number of data channel attributes may be modified.

QUALIFIERS

/INITIAL_VALUE=value

Specifies the value that the user data channel will have when the currently opened object is created during a simulation. There is no limit on this value.

/CYCLE=(min, max)

Specifies a pair of values which, when one is reached or exceeded, the user value immediately switches to the other value. The *min* value must be less than the *max* value.

EXAMPLES

```
USERVALUE 4/INITIAL_VALUE=10.0/CYCLE=(-100.0,100.0)
```

This command would modify user value 4 (known as USER4). It would set its initial value to be 10.0 when the object this command is modifying is created during a simulation. It would also set the cycle limits of the user data channel to be -100.0 and 100.0. If any interactive command attempted to change the value of the USER4 data channel to be 101.0, FLIGHT would begin changing the value until it exceeded 100.0 and then would 'wrap' the advancement towards 101.0 back through -100.0. This would leave the USER4 data channel at somewhere around -99.0.

Because the command has been issued that is supposed to set the user value to a value of 101.0, FLIGHT begins to advance the value until it crosses the 100.0 boundary and again pushes the value back down to the *min* limit. This will continue forever until FLIGHT is told to set the value of USER4 to some value between the cycle *min* and *max* values. This is why it is called cycling.

It can be useful for causing a particular pattern to be repeated over and over again. Effects such as flashing lights can be achieved using the CYCLE qualifier in conjunction with subobjects.

VIEWPORT

Specifies location and size of a world viewport on the display that is currently opened.

FORMAT

VIEWPORT width height xplace yplace

Qualifiers	**Defaults**
/FACING=direction	*/FACING=FORWARD*

PARAMETERS

None

width

Specifies the width of the viewport in centimeters.

height

Specifies the height of the viewport in centimeters.

xplace

Specifies the horizontal placement of the lower left corner of the world viewport as measured from the left size of the display in centimeters.

yplace

Specifies the vertial placement of the lower left corner of the world viewport as measured from the bottom of the display in centimeters.

DESCRIPTION

The VIEWPORT command allows for specification of the currently opened display's positioning and sizing of a world viewport. A world viewport is a window on the display at simulation time which shows the pilot's view of the world. Regardless of the viewport size, the larger dimension of the viewport allows display of a 90 degree spread. The smaller dimension is appropriately scaled.

Positioning of the viewport on the display does not impact what the simulator displays in the viewport.

The viewport is added to the display that is currently open.

QUALIFIERS

/FACING=direction

Specifies which direction, relative to the direction the pilot is looking, that should be displayed in the viewport. Table 2-11 lists the valid directions. If the ROTATIONS keyword is used, the values are expressed in degrees and the order of rotation specification is X, Y, Z.

Table 2-11 Direction Keywords

Keyword	Description
FORWARD	Equivalent to ROTATIONS=(0, 0, 0)
BACKWARD	Equivalent to ROTATIONS=(0, 180, 0)
UP	Equivalent to ROTATIONS=(0, 0, 90)
LEFT	Equivalent to ROTATIONS=(0, -90, 0)
RIGHT	Equivalent to ROTATIONS=(0, 90, 0)
DOWN	Equivalent to ROTATIONS=(0, 0, -90)
ROTATIONS=(x, y, z)	Allows specification of arbitrary orientations

EXAMPLES

```
VIEWPORT 8.75 18.66 7.05 17.47
```

This command would create a viewport 8.75 cm wide, 18.66 cm wide, positioned 7.05 cm from the left edge of the display and 17.47 cm from the bottom edge.

VOLUME

Specifies information about the spherical volumes that an object occupies.

FORMAT

VOLUME volume-radius center-x center-y center-z volume-damage

Qualifiers	Defaults
/DESTROY=destroy-keyword	*/DESTROY=SELFDESTRUCT*

PARAMETERS

volume-radius

Specifies the radius of the spherical volume being added to the object's volume list.

center-x center-y center-z

Specifies the location of the center of the spherical volume being added to the object's volume list.

volume-damage

Specifies the amount of damage that this volume can accrue before being considered destroyed. When a volume is destroyed, it's destruction effect is performed. This effect is specified by using the /DESTROY qualifier.

When two objects collide, it is actually their volumes which collide. When two volumes collide, the one with lesser remaining damage will be destroyed. Therefore, weapons such as bullets should have low volume damage levels, whereas the tail section of an aircraft might have a relatively high damage level. All damage values are unitless and arbitrarily assigned. What is important is how the damage values assigned relate to each other.

DESCRIPTION

The VOLUME command is used to specify what space (relative to the object's geometry center) is actually occupied by the object currently being defined. This is used for collision detection checks. Up to 20 volumes may be specified per object, and volumes are unrelated to levels of detail, which are used in geometry specification. The order in which volumes are defined is of no importance. You may choose not to specify any volumes, in which case a single volume is defined for you which encompasses all the points of the geometry, centered at the object's geometry origin with the SELF-DESTRUCT destruction effect.

The VOLUME command is also valid within the scope of a subobject description. Again, up to 20 volumes may be specified per subobject. If no volumes are specified, a volume which completely encompasses the subobject's geometry is created with a collision effect of LOCALDESTRUCT.

QUALIFIERS

/DESTROY=destroy-keyword

Specifies what should happen if this volume is destroyed due to collision damage. This allows you to bind destruction of the volume to changes in the object's performance and capabilities.

Table 2-12 shows the list of possible destruction effects.

Table 2-12 Destruction Keywords

Keyword	Description
THRUST=percentage	Reduces thrust by a multiple-of-10 percentage
ROLL=percentage	Reduces the object's ability to roll by a multiple-of-25 percentage
PITCH=percentage	Reduces the object's ability to pitch by a multiple-of-25 percentage
YAW=percentage	Reduces the object's ability to yaw by a multiple-of-25 percentage
LEFTWING=percentage	Reduces the surface area on the object's left wing by a multiple-of-25 percentage
RIGHTWING=percentage	Reduces the surface area on the object's right wing by a multiple-of-25 percentage
CREW=name	Destroys the named display. If this is the last display of the object, the object is destroyed as well.
SELFDESTRUCT	Destroys the root object
LOCALDESTRUCT	Destroys the object or subobject in which the volume was defined

EXAMPLES

```
VOLUME 3.0 -10.0 0.5 2.0 20.0/DESTROY=PITCH=50
```

In this example, the volume has a radius of 3.0 feet and is positioned on the right side elevator. A total of 20 damage points have been allotted to the volume before it is considered destroyed. When it is destroyed, it will reduce the object's ability to pitch by 50% (which is why it is assumed to be the volume of an elevator).

VOR

Specifies a component of a VOR-type instrument definition within the confines of OPEN INSTRUMENT and CLOSE INSTRUMENT. The VOR command is not valid for non-VOR instruments.

FORMAT

VOR

Qualifiers	Defaults
/BIGSTICKS=number	*None.*
/BUTTON=(xpos, ypos, width, height, fill, lable, xscale, yscale)	*None.*
/C_LABEL=(title, xpos, ypost, xscale, yscale)	*None.*
/CONSOLE=(xpos, ypos, width, height)	*None.*
/DIGITS=digit-count	*None.*
/DIVISOR=value	*None.*
/FLAGS=(off-flag, to-flag, from-flag)	*None.*
/FREQUENCY_READOUT=(xpos, ypos, xscale, yscale)	*None.*
/(NO)GLIDE_SLOPE	*None.*
/GS_DOTS=(count, spread)	*None.*
/I_LABEL=(title, position, radius, xscale, yscale)	*None.*
/IN_TITLE=(title, yposition, xscale, yscale)	*None.*
/INITIAL_FREQUENCY=freq-in-Hz	*None.*
/KNOB=(xpos, ypos, radius, fill, label, xscale, yscale)	*None.*
/MAXFREQUENCY=freq-in-Hz	*None.*
/MINFREQUENCY=freq-in-Hz	*None.*
/NAV_DOTS=(count, spread)	*None.*
/NEEDLE_LENGTH=length	*None.*
/RADIUS=size-in-cm	*None.*
/RATIO=radius-ratio	*None.*
/SMLTICKS=number	*None.*
/TICKS=(number, start, end, minradius, maxradius, width)	*None.*

QUALIFIERS

/BIGTICKS=number

Specifies the number of large tick marks drawn around the inside of the VOR indicator (the Omni-Bearing Selector). The number of tick marks are spread evenly through the full 360 degree sweep of the dial.

/BUTTON=(xpos, ypos, width, height, fill, label, xscale, yscale)

Adds a button graphic to the VOR console. The button is purely for aesthetics; it is non-functional. You may specify up to 10 buttons per VOR.

'xpos' and 'ypos' specify the location of the center of the button. 'width' and 'height' specify the size of the button image. 'fill' controls how the interior of the button is drawn; the value is interpreted as a gray-level percentage, where 0 is empty and 100 is solid filled. 'label' specifies a text string which is drawn centered above the button (up to six characters). The text character size is specified by the two scale values. The magnitude of all dimension values is based on 100 units being equal to the console's height. (If the scale values are omitted, the label is drawn at a standard, fixed size.)

/C_LABEL=(title, xpos, ypos, xscale, yscale)

Specifies the position, size, and title of a label positioned on the interior of the VOR console. 'xpos' and 'ypos' specify the center of the label. The text character size is specified by the two scale values. The magnitude of all dimension values is based on 100 units being equal to the console's height. (If the scale values are omitted, the label is drawn at a standard, fixed size.)

Up to 10 labels may be placed on a single VOR console, but each must be placed by a separate VOR/C_LABEL command.

/CONSOLE=(xpos, ypos, width, height)

Specifies where and how large the console is. The position values specify where the center of the console is relative to the center of the indicator. The width/height values specify the console's size. All four values are specified in centimeters.

/DECIMALS=decimal-digit-count

Specifies how many decimal digits to the right of the decimal point should be displayed in the console frequency readout. A value of 0 omits the decimal point altogether.

/DIGITS=digit-count

Specifies how many total digits are displayed on the console frequency readout. This count should include digits to the left of the decimal point, the decimal point (if present), and digits to the right of the decimal point.

/DIVISOR=value

The VOR console provides a digital readout of the frequency that the unit is currently tuned to. The actual number that is displayed is computed by dividing the frequency in Hertz by the DIVISOR value. For example, DIVISOR=1000000 would make the display be in mHz.

/FLAGS=(off-flag, to-flag, from-flag)

Defines the text which should be used in the TO-FROM flag. The first string is displayed in the TO-FROM flag window when the azimuthal portion (ILS localizer or VOR radial) of the VOR receiver is not receiving a signal. It is also used to flag the glide slope indicator (if one is present) when no GS signal is being received. The second and third strings are displayed in the TO-FROM window during VOR navigation when the OBS setting indicates a heading approximately to or from the station, respectively.

/FREQUENCY_READOUT=(xpos, ypos, xscale, yscale)

Specifies where the frequency readout is located within the VOR console, and how large it is. 'xpos' and 'ypos' determine the center of the readout. The readout text character size is specified by the two scale values. The magnitude of all four values is based on 100 units being equal to the console's height. (If the scale values are omitted, the text is drawn at a standard, fixed size.)

/(NO)GLIDE_SLOPE

If this qualifier is negated, then no glide slope components will be configured in this VOR receiver instrument. If not negated, glide slope capability will be present.

/GS_DOTS=(count, spread)

Specifies the configuration of the scale of dots that is drawn vertically across the middle of the indicator to show how far from center the glide slope needle is. 'count' is how many total dots there should be; 'spread' is how tall an area they should be drawn over, measured out from the center. The units are percentage of VOR radius, so 100 would mean spread the dots from full bottom to full top across the entire face of the dial.

/I_LABEL=(title, position, radius, xscale, yscale)

Specifies the position, size, and title of a label positioned around the circumference of the VOR indicator (OBS). The angular position is expressed in degrees. The 'radius' value controls where the text is written, in terms of distance out from the dial center. A magnitude of 100 equals the radius of the dial face. The title string is centered at that point. 'xscale' and 'yscale' should be set to 0.

Up to 40 labels may be placed on a single VOR indicator, but each must be placed by a separate VOR/I_LABEL command.

/IN_TITLE=(title, yposition, xscale, yscale)

Specifies the position and text of a title located in the inside of the VOR indicator. The text is drawn with characters sized as specified by the two scale values, and centered horizontally. The vertical position is specified by the second parameter (which may be negative). The magnitude of the numeric values is based on 100 units being equal to the dial's radius. (If the scale values are omitted, the title is drawn at a standard fixed size.)

Up to 20 interior titles may be placed on a single VOR indicator, but each must be placed by a separate VOR/IN_TITLE command.

/INITIAL_FREQUENCY=freq-in-Hz

Controls what frequency the VOR console is tuned to when the object is first entered.

/KNOB=(xpos, ypos, radius, fill, label, xscale, yscale)

Adds a knob graphic to the VOR console. The knob is purely for aesthetics; it is non-functional. You may specify up to 10 knobs per VOR.

'xpos' and 'ypos' specify the location of the center of the knob. 'radius' specifies the radial dimension of the knob image. 'fill' controls how the interior of the knob is drawn; the value is interpreted as a gray-level percentage, where 0 is empty and 100 is solid filled. 'label' specifies a text string which is drawn centered below the knob (up to six characters). The text character size is specified by the two scale values. The magnitude of all dimension values is based on 100 units being equal to the console's height. (If the scale values are omitted, the label is drawn at a standard, fixed size.)

/MAXFREQUENCY=freq-in-Hz

Specifies the highest frequency to which the VOR console may be tuned. Attempts to tune to higher frequencies will be ignored. The value is specified in Hertz.

/MINFREQUENCY=freq-in-Hz

Specifies the lowest frequency to which the VOR console may be tuned. Attempts to tune to lower frequencies will be ignored. The value is specified in Hertz.

/NAV_DOTS=(count, spread)

Specifies the configuration of the scale of dots that is drawn horizontally across the middle of the indicator to show how far from center the CDI needle is. 'count' is how many total dots there should be; 'spread' is how wide an area they should be drawn over, measured out from the center. The units are percentage of VOR radius, so 100 would mean spread the dots from full left to full right across the entire face of the dial.

/NEEDLE_LENGTH=length

Specifies how long the Course Deviation Indicator (and Glide Slope Indicator) needles should be. 100 length units equals the radius of the VOR indicator.

/RADIUS=size-in-cm

Specifies the radial dimension of a VOR indicator instrument in centimeters.

/RATIO=radius-ratio

VOR indicators are composed of (among other things) two concentric circles; the outter circle contains the OBS dial and the inner circle contains the CDI and GS needles and the TO-FROM flag. The size of the outer circle is specified using the RADIUS qualifier; the size of the inner circle is indirectly specified using RATIO. For example, RADIUS=2.0/RATIO=0.5 would mean that the inner circle had a 1 cm. radius (half of 2.0).

/SMLTICKS=number

Specifies the number of small tick marks drawn around the inside of the VOR indicator (OBS). The number of tick marks are spread evenly through the full 360 degree sweep of the dial.

/TICKS=(number, start, end, minradius, max-radius, width)

Places a generalized set of ticks around the VOR indicator's OBS face. 'Number' controls how many tick marks there are in the set. 'Start' and 'end' control the angular region where the marks are drawn (specified in degrees). The tick marks are evenly distributed in this region, inclusive of ends (that is, the first tick is located at 'start' and the last at 'end'). The two radial parameters control how long the tick is (100 tick radius units equals the radius of the indicator face). Width controls the thickness of the tick marks.

Up to 9 tick sets may be specified per VOR. The BIGTICKS and SMLTICKS qualifiers may be used to construct simple cases of ticks.

WEAPONS

Specifies information about the weapons loaded onto an object.

FORMAT

WEAPONS object-name x-position y-position z-position

Qualifiers	Defaults
/BANK=bank-number	*/BANK=1*
/BINDING=subobject-name	*See text*
/REPEAT=repeat-count	*/REPEAT=1*

PARAMETERS

weapon-name

Specifies the name of the weapon object which should be loaded in the currently open object. Object names are case-sensitive and, therefore, should be enclosed in double-quotes.

x-position y-position z-position

Specifies the firing location of the weapon when it is fired. The position is relative to the origin of the object's geometry and is specified in feet. Forward is +X, up is +Y and right is +Z.

DESCRIPTION

The WEAPONS command allows the object's weapons stores to be specified. Only those objects which have been defined prior to the WEAPONS command may be referenced by it. A maximum of 20 weapons may be loaded into each of 4 weapons banks, and only one weapon may be loaded per WEAPONS command. There are no constraints on combinations of weapons which may be stored in each bank.The order of WEAPONS commands determines the firing order of the weapons in each bank. That is, the first weapon loaded into bank 1 will be the first weapon fired from bank 1, etc.

When a weapon is fired during an interactive simulation, the displays of the weapon being fired (if there are any) will be created by FLIGHT. This allows aircraft to launch remotely piloted drone aircraft, camera-guided cruise missiles and other exotic, piloted weapons. The ability to have displays on a weapon can also be used to research the behavior of the weapon in flight, even if it has some sort of on-board guidance, specified through the use of the GUIDANCE command.

An example of a weapon which has displays is the Cruise Missile found on the B-1B Bomber and the B-52 Stratofortress in the default hangars. Note that any aircraft could be treated as a weapon by another aircraft. That is, the B52 Stratofortress could carry the X15 as a weapon and 'fire' it, allowing testing of such an experimental aircraft. The X15 could be fully instrumented and even carry weapons of its own.

QUALIFIERS

/BANK=bank-number

Specifies the number of the weapon bank into which the weapon should be stored. Each object has up to 4 banks of weapons which may be loaded with up to 20 unique weapon definitions. Note that each weapon definition may have a repeat count associated with it (see the REPEAT qualifier below).

If this qualifier is not specified, weapon bank number 1 is loaded by default.

/BINDING=subobject-name

Specifies the subobject whose orientation and position are used to determine the direction of the flight of the weapon. Weapons are aimed along the positive X axis of the subobject.

If this qualifier is not specified, the weapon is bound to the root object; that is, the weapon is bound to the object.

/REPEAT=repeat-count

Specifies a repeat on this weapon placement command. The repeat count indicates the number of identical weapons to place into the specified weapon bank. A repeat count does not subtract that amount from the total weapons which may be placed into a bank. See the examples for more information.

EXAMPLES

```
WEAPONS "AIM-9L Sidewinder" 3.0 -1.0 -10.0
```

A Sidewinder missile is being loaded into the default weapon bank (number 1). The Sidewinder missile object must have been defined earlier in this same hangar, or an error will be reported. The missile, when launched, will appear 3.0 feet forward, 1.0 feet below and 10.0 feet to the left of the object's geometry origin. That probably puts it under the left wing.

```
WEAPONS/BANK=2 "AIM-9L Sidewinder" 3.0 -1.0 10.0
```

A Sidewinder missile is again being loaded, but this time into bank number 2. That fact is explicity specified by the presence of the /BANK qualifier. The missile appears to want to be launched from under to right wing, according to the X-position, Y-position and Z-position parameters.

```
WEAPONS "SA-9 SAM" 0.0 0.0 0.0/BINDING="Launcher"
```

This time an SA-9 Soviet surface-to-air missile (SAM) is being loaded into the default weapon bank. However, the BINDING qualifier has been specified to cause the weapon to be launched according to the position and orienation of the Launcher subobject in the current object. The X/Y/Z position values indicate that the missile should be released from the origin of the Launcher subobject. Should the subobject be destroyed, this weapon will not launch.

```
WEAPONS/BANK=3 "Tracers" 10.0 0.0 2.0/REPEAT=100
```

This command adds 100 tracer objects into bank 3. The objects will be released/fired in succession until all 100 objects have been released. Then the next object in bank 3 will be available for release.

WEIGHT

Specifies information about the weight of an object.

FORMAT

WEIGHT

Qualifiers	**Defaults**
/EMPTY=pounds	*None.*

PARAMETERS

None

DESCRIPTION

The WEIGHT command specifies information about the weight of the currently open object.

QUALIFIERS

/EMPTY

Specifies the empty weight of the currently opened object. This should not include fuel or weapon load considerations, but should include the crew's weight and any hypothetical optional hardware, such as radar, radio gear, and so on.

2.4 Data Channel Keywords

Table 2-13 lists the keywords that the DATA qualifier accepts for the DIAL, SLIDER, DIGITAL, and LAMP commands.

Table 2-13 Data Channel Keywords

Keyword		Value
HEADING		This value is expressed in degrees.
FUEL	=PERCENTAGE	
	=POUNDAGE	This value is expressed as a percentage or poundage of the object's total fuel available.
ALTITUDE	=ABSOLUTE	
	=RELATIVE	This value is expressed in feet; ABSOLUTE is height above zero (MSL, mean sea level, whereas RELATIVE is height above the GROUND_LEVEL plain (AGL, above ground level).)
THRUST	=PERCENTAGE	
	=POUNDAGE	This value is expressed as a percentage or poundage of the object's total thrust being produced by it's engine(s.)
VERTICAL_SPEED		This value is expressed in feet per minute.
AIR_SPEED		This value is expressed in knots.
FLAPS	=ANGLE	
	=PERCENTAGE	This value is expressed as a percentage or angle of the object's flaps (if available.)
GFORCE		This value is expressed as G's (standard Earth gravities which the object's structure is currently experiencing.)
AOA		This value is expressed in degrees, and represents the object's current angle-of-attack.
AOP		This value is expressed in degrees, and represents the object's current angle-of-pitch.
AOR		This value is expressed in degrees, and represents the object's current angle-of-roll.

AOS		This value is expressed in degrees, and represents the object's current angle-of-slip (sideslip.)
ROLL_RATE		This value is expressed in degrees per second, and represents the current roll rate of the object (rate of change of roll attitude).
PITCH_RATE		This value is expressed in degrees per second, and represents the current pitch rate of the object (rate of change of pitch attitude).
YAW_RATE		This value is expressed in degrees per second, and represents the current yaw rate of the object (rate of change of yaw attitude).
THROTTLE		This value is expressed as a percentage of the idle-to-full setting of the object's throttle, where 100 represents full (possibly afterburning power.)
AFTERBURNERS		Value is 1.0 if afterburners are lit, 0.0 if not.
STALLED		Value is 1.0 if the object stall warning is enabled, 0.0 if disabled.
ON_TERRAIN		Value is 1.0 if the object is in contact with terrain (dry land), 0.0 otherwise.
ON_WATER		Value is 1.0 if the object is in contact with water, 0.0 otherwise.
IN_AIR		Value is 1.0 if the object is free-flying (not in contact with any surfaces), 0.0 otherwise.
BRAKES	=PERCENTAGE	
	=POUNDAGE	This value is expressed as a percentage of the maximum or the actual amount of brake force being applied.
WENABLED		Value is 1.0 if the object's weapons system is armed, 0.0 if disarmed.
WSELECTED		This values is the number of the currently-selected weapon firing sequence. The possible range of values is 1 through 4.
ENGINE_ON		Value is 1.0 if the object's engines are on, 0.0 if off.
IGNITION_ON		Value is 1.0 if the object's ignition switch is on, 0.0 if off.

USER1-USER16		These values have no limit and are controlled through execution of the SET USER_VALUE simulator function command.
RUDDER	=PERCENTAGE	
	=ANGLE	This value is expressed as a percentage of the maximum or the actual amount of rudder deflection.
AILERONS	=PERCENTAGE	
	=ANGLE	This value is expressed as a percentage of the maximum or the actual amount of aileron deflection.
ELEVATOR	=PERCENTAGE	
	=ANGLE	This value is expressed as a percentage of the maximum or the actual amount of elevator deflection.
LOCKn		Value is 1.0 if the lead weapon in weapon bank n is locked on target and weapons are armed. Value is 0.0 otherwise.
COUNTn		Value is the number of individual weapons remaining in bank n.
L*n*X, L*n*Y		These values are expressed in degrees and they represent horizontal and vertical angles from the aircraft's forward direction. They are angles to the object that the lead weapon in bank *n* of the aircraft is locked onto. If the lead weapon is not radar guided or has not achieved lock-on, then these values are 0.0. The keywords valid are, therefore, L1X through L4X and L1Y through L4Y.
LnR		These values are expressed in feet and they represent the linear distance to the object that the lead weapon in bank *n* of the aircraft is locked onto. If the lead weapon is not radar guided or has not achieved lock-on, then these values are 0.0. The keywords valid are, therefore, L1R through L4R.
LIFETIME	=REMAINING	

	=ELAPSED	This value is expressed in seconds, indicating either the elapsed lifetime of the object, or the remaining lifetime of the object. The lifetime of an object is specified by the LIFETIME command.
FOREIGN_LOCK		This value is 1.0 if another aircraft in the simulation environment has a weapon lock on your aircraft. The value is 0.0 otherwisze.
TIME		This value is expressed in seconds, indicating the time elapsed since 00:00 o'clock on the day that the simulation started.

3 The Hangar Editor

The FLIGHT hangar editor is an interactive tool which is intended to aid in the development of hangars. It provides only the most rudimentary capabilities.

3.1 DCL Command

The following describes the invocation of the hangar editor.

FLIGHT/EDIT=HANGAR

Invokes FLIGHT's hangar editor.

FORMAT

FLIGHT/EDIT=HANGAR input-filespec

Qualifiers	**Defaults**
None	*None*

PARAMETERS

input-filespec

Specifies the input file to be edited. Note that wildcards are not allowed in the input file specification. The default file type for this parameter is HANGAR. The default location for a hangar is your default directory. If the hangar you are attempting to edit is not in that directory, you must specify a full path to the hangar file.

DESCRIPTION

The FLIGHT/EDIT=HANGAR command starts FLIGHT's hangar editor. After the hangar file is read in, messages stating how many objects and instruments which were read are output. Then a dialog is presented to the user.

QUALIFIERS

None

3.2 Interactive Use

The following figure shows the dialog displayed when the hangar editor is invoked. It allows selection from among the objects in the hangar, and selection from the various options available through the *File*, *Edit* and *View* menu options.

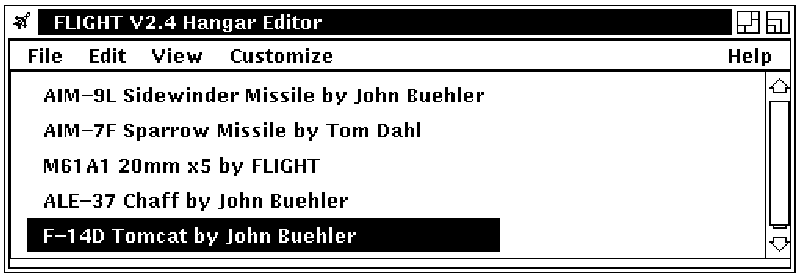


Figure 3-1 Hangar Editor Dialog

3.3 The File Menu

The *File* menu allows you to

- Open a new hangar
- Open a new hangar, merging its contents into the currently loaded hangar
- Compress the currently loaded hangar, removing all unreferenced objects, instruments and colors.
- Save the source description of the currently selected object to a file
- Save a pictorial representation of the components of the currently selected object in *PostScript*™ format.
- Save the hangar being edited to a file that you name
- Save the hangar being edited to the file it came from
- Exit the hangar editor, saving any changes you have made
- Quit the hangar editor. This does not save the current hangar.

3.3.1 Open Hangar

Select this option to open a new hangar for editing. This does not save the hangar which you are currently editing, nor does it warn you if you have not saved any changes made to the current hangar.

When selected, a file selection box will be presented to you, allowing ou to specify the name of the hangar what you want to open.

3.3.2 Merge Hangar

Select this option to open a new hangar. This hangar does not replace the contents of the current hangar. Rather, the two hangars are merged, with the objects from the new hangar being added to the end of the object list shown in the main hangar editor dialog.

When selected, a file selection box will be presented to you, allowing ou to specify the name of the hangar what you want to merge.

3.3.3 Compress Hangar

Select this option to compress the current hangar. Compression eliminates all unreferenced objects, instruments and colors in the hangar. This has the benefit of making the hangar smaller because it is not carrying unneeded information. Objects with the PILOTABLE characteristic are always retained.

3.3.4 Save Object as Source

Select this option to create a source file describing the currently selected object. When selected, a file selection box will be presented to you, allowing you to specify the name of the file that FLIGHT should write the source description into.

The source description will be without any comments, and will only describe the actual object selected, not including any placement commands relating to it.

3.3.5 Save Object as PostScript™

Select this option to create a file containing a *PostScript*™ description of the components of the object. When selected, a file selection box will be presented to you, allowing you to specify the name of the file that FLIGHT should write the *PostScript*™ desription into.

Each page of the output file will show three views of a component of the selected object. The views displayed are top, front and rights side views. Each point of the object is numbered, and the name of the object is shown in the upper right corner of the page.

3.3.6 Save Hangar to New File

Select this option to create a new hangar file containing any changes made so far. When selected, a file selection box will be presented to you, allowing you to specify the name of the file that FLIGHT should write the hangar file into.

3.3.7 Save Hangar

Select this option to create a new version of a hangar file containing any changes made so far. When selected, FLIGHT writes the hangar contents into the file that was originally edited.

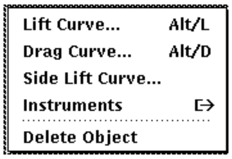
3.3.8 Exit

Select this option to exit the hangar editor and save the current hangar contents to a new version of the file being edited. When selected, FLIGHT writes out the hangar, then exits.

3.3.9 Quit

Select ths option to quit the hangar editor. When selected, FLIGHT immediately exits without saving any changes you have made.

3.4 The Edit Menu



The *Edit* menu allows you to

- Edit the lift curve of the currently selected object.
- Edit the side lift curve of the currently selected object.
- Edit the drag curve of the currently selected object.
- Edit the layout of the currently selected object's displays.
- Delete the currently selected object.

3.4.1 Edit Lift Curve

Select this option to edit the lift curve of the currently selected object. When selected, a window is created with the lift curve of the current object displayed in it. Use MB1 to change values in the lift curve. MB2 acts a s a toggle. When you press it once, the current angle and lift coefficient are displayed in the upper left corner of the window. As you move the mouse, the value will change to reflect the portion of the curve you are on. When you press MB2 a second time, the display is turned off.

In addition, the following keys are defined to allow you to modify the curve:

Table 3-1 Curve Editor Keyboard

Key	Description
KP1	Mirrow the left side of the curve vertically.
KP2	Scales curve up by 5%. You may scale the curve such that it extends off the top and bottom of the window without losing any information.
KP3	Mirrors right side of the curve vertically.
KP4	Mirrors right side of the curve to the left side. The existing left side of the curve is, therefore, lost.
KP6	Mirrors left side of the curve to the right side. The existing right side of the curve is, therefore, lost.
KP7	Shjfts curve to the left by 1 degree. This causes the coefficient for -90° to be lost and the coefficient for +90^0 to be duplicated.
KP8	Scales the curve down by 5%
KP9	Shifts the curve to the right by 1 degree. This causes the coefficient for +90° to be lost and the coefficient for -90° to be duplicated.
KP,	Shifts the curve down by a 0.05 coefficient.
KP−	Shifts the curve up by a 0.05 coefficient.
F8	Quit the curve editor. This causes the editor to be dismissed without modifying the original lift curve of the object.
F10	Exit the geometry viewer. This causes the editor to be dismissed, with the changes to the lift curve being applied to the object that it came from.

3.4.2 Edit Side Lift Curve

Select this option to edit the side lift curve of the currently selected object. This functions exactly as the lift curve editor except that it operates on the side lift information instead of the lift information of the object. See Section 3.4.1 for a description of the curve editor.

3.4.3 Edit Drag Curve

Select this option to edit the drag curve of the currently selected object. This functions exactly as the lift curve editor except that it operates on the drag information instead of the lift information of the object. See Section 3.4.1 for a description of the curve editor.

3.4.4 Edit Instruments

This option has a pullright listing four items. They are labelled according to the names of the displays of the currently selected object. Those labelled 'Unused' will produce no result if selected.

When any other display is selected, the instrument panels and viewports of that display are created, complete with all instrumentation. As you move the mouse across instruments, they will highlight, indicating the current instrument.

You may move and resize windows to control the layout of the display. Windows may not be added or deleted.

The following keys are defined to allow you to modify the instrument layout:

Table 3-2 Display Editor Keyboard

Key	Description
MB1	Press and drag allows you to move instruments. As the instrument is dragged on the instrument panel, its outline is shown. When you release the button, the instrument is moved and redrawn at its new location. There is no way to cancel a move operation once it has started.
KP1	'Shrinkwrap' the instrument panel to the instruments it contains. If there are multiple instrument panels in the display, be sure to have input focus directed to the correct instrument panel before pressing KP1.
F8	Quit the display editor. This causes the editor to be dismissed without modifying the original display definition.
F10	Exit the display editor. This causes the editor to be dismissed, with the changes to the display being applied to the object that it came from.

3.4.5 Delete Object

This option deletes the selected object. If other objects in the hangar rely on the selected object's existence (for example, the selected object is a weapon used by an aircraft in the hangar), then the delete operation will be disallowed. Otherwise, the object is removed from the hangar.

Referenced instruments, colors and weapons are not affected by the delete operation. To remove unused instruments, colors and weapons, use the Compress Hangar function in the File menu.

3.5 The View Menu

Geometry... Alt/G

The *View* menu allows you to

- View the geometry of the currently selected object.

3.5.1 View Object Geometry

Select this option to view the geometry of the currently selected object. When selected, a window is created with the geometry of the current object displayed in it. You may use the mouse to view the geometry from a variety of angles. In general, the front of the aircraft will attempt to keep pointing at the mouse pointer.

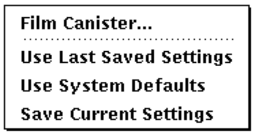
In addition, the following keys are defined to allow you to control other aspects of viewing.

Table 3-3 Viewer Keyboard

Key	Description
Next Screen	Zooms in by 3X
Prev Screen	Zooms out by 3X
Up Arrow	Zooms in by 0.3X
Down Arrow	Zooms out by 0.3X
MB1	(Mouse Button 1) Locks the view at the current orientation. Another press of MB1 causes the view to unlock and follow the mouse pointer again.
F8	Quit the geomtry viewer. This is identival to the exit function.
F10	Exit the geometry viewer.

Note that in the lower left corner of the viewer window is a numeric display. This display indicates the current view angle to the object in the window. It can be useful for tuning the level of detail transitions of the object.

3.6 The Customize Menu



The *Customize* menu allows you to

- Specify the name of the file which should be used as your film anister for storing snapshots taken during the editing session.
- Reload the last set of preferences which you saved to disk.
- Set your preferences to the default FLIGHT preferences.
- Save your current preferences to disk.

3.6.1 Film Canister

Select this option to specify a new film canister filename. The film canister file is written to whenever a snapshot is taken in FLIGHT. In the world editor, this may be done in the geometry viewer.

3.6.2 Use Last Saved Settings

Select this option to reestablish the preferences which you last saved to disk.

3.6.3 Use System Defaults

Select this option to set the preferences for the current setting to the defaults defined by FLIGHT. If you had no preferences saved to disk prior to starting the world editor, the defaults would be in effect.

3.6.4 Save Current Settings

Select this option to save your preferences set through the *Customize* menu to disk. The file created is DECW$USER_DEFAULTS:FLT$DEFAULTS.DAT