SAE Aerospace
An SAE International Group

AEROSPACE RECOMMENDED PRACTICE

SAE ARP5571A

Issued 2005-01
Revised 2008-12

Superseding ARP5571

(R) Gas Turbine Engine Performance Presentation and Nomenclature For Object-Oriented Computer Programs

RATIONALE

Gas turbine engine manufacturers (suppliers) have long provided their customers with computer programs which simulate engine performance. Application manufacturers and others (customers) use these programs, often called models or simulations, in design studies, mission analysis, life cycle analysis, and performance prediction of their products. These models are used throughout the life of a product, from conceptual design through production, deployment, field use, maintenance, and overhaul. Communication between suppliers and customers is more productive and less error prone if all engine models adhere to common guidelines with respect to presentation of data and interface with other computer programs. No guidelines or recommended practices previously existed for Object-Oriented models.

Revision A has been created to correct minor typographical errors as well as address integer switch values that have been added in Appendix A, also some revisions were made in the Program Status Indicaton section.

TO PLACE A DOCUMENT ORDER: **Tel: 877-606-7323 (inside USA and Canada)**
Tel: 724-776-4970 (outside USA)
Fax: 724-776-0790
Email: CustomerService@sae.org
SAE WEB ADDRESS: **http://www.sae.org**

SAE values your input. To provide feedback on this Technical Report, please visit
http://www.sae.org/technical/standards/ARP5571A

FOREWORD

Historically, the typical engine model is delivered as a FORTRAN program to be used in either stand-alone or subprogram mode by the customer. AS4191 governs delivery of such models.

Advancements in computing since the FORTRAN processes were developed provide an invaluable opportunity to create new processes and standards governing engine performance computer programs. Three specific developments contribute to this opportunity:

1. Programming languages other than FORTRAN are now widely used. These languages do not share the limitations of early versions of FORTRAN which shaped earlier standards (limitations on variable name length, for example).

2. The advent of object-oriented programming, using languages such as C++, has introduced new features such as hierarchical object naming and object reuse which provides improved capabilities not addressed in the existing standards.

3. A trend toward sharing entire modeling environments rather than individual programs makes it desirable to standardize features of model architecture that have not been addressed before.

TABLE OF CONTENTS

1. SCOPE

This document provides recommendations for several aspects of air-breathing gas turbine engine performance modeling using object-oriented programming systems. Nomenclature, application program interface, and user interface are addressed with the emphasis on nomenclature. The Numerical Propulsion System Simulation (NPSS) modeling environment is frequently used in this document as an archetype. Many of the recommendations for standards are derived from NPSS standards. NPSS was chosen because it is an available product. The practices recommended herein may be applied to other object-oriented systems.

While this document applies broadly to any gas turbine engine, the great majority of engine performance computer programs have historically been written for aircraft propulsion systems. Aircraft and propulsion terminology and examples appear throughout.

1.1 Purpose

The purpose of this document is to define an object-oriented paradigm for the simulation of gas turbine engine performance, recommend nomenclature for objects and variables within such a simulation, and recommend practices for user interaction with such a simulation.

1.2 Field of Application

This document applies to air-breathing gas turbine engine performance simulations using an object-oriented programming approach.

2. REFERENCES

2.1 Applicable Documents

The following publications form a part of this document to the extent specified herein. The latest issue of SAE publications shall apply. The applicable issue of other publications shall be the issue in effect on the date of the purchase order. In the event of conflict between the text of this document and references cited herein, the text of this document takes precedence. Nothing in this document, however, supersedes applicable laws and regulations unless a specific exemption has been obtained.

2.1.1 SAE Publications

Available from SAE International, 400 Commonwealth Drive, Warrendale, PA 15096-0001, Tel: 877-606-7323 (inside USA and Canada) or 724-776-4970 (outside USA), www.sae.org.

AS681 Gas Turbine Engine Performance Presentation for Computer Programs

AS755 Aircraft Propulsion System Performance Station Designation and Nomenclature

ARP5571 incorporates the recommendations of AS755 with respect to station designations, in particular the “Alternate Numbering System for Reduced Ambiguity” introduced in Revision D. The nomenclature of AS755 applies broadly to all forms of communication regarding propulsion system performance. AS681 allows “use of fully descriptive labels for variable identification” and provides that “In cases where symbols are to be substituted for fully descriptive labels, the list of symbols contained in AS755 will be used …”. The nomenclature in this document is more narrowly focused on the specific area of “fully descriptive symbols” to be used in the interface between users and computer performance simulations.

2.2 Related Publications

The following documents contain material relevant to the subject area of this document, but do not form a part of this document.

2.2.1 SAE Publications

Available from SAE International, 400 Commonwealth Drive, Warrendale, PA 15096-0001, Tel: 877-606-7323 (inside USA and Canada) or 724-776-4970 (outside USA), www.sae.org.

ARP210 Definition of Commonly Used Day Types (Atmospheric Ambient Temperature Characteristics Versus Pressure Altitude)

AS4191 Gas Turbine Engine Performance Presentation for Computer Programs Using FORTRAN

ARP4868 Application Programming Interface Requirements for the Presentation of Gas Turbine Engine Performance on Digital Computers

2.2.2 Other Publications

Available from Wolverine Ventures, Inc, 16593 121 Terrace N, Jupiter, FL 33478-6005, www.virtualtestcell.com.

NPSS User Guide

NPSS Customer Deck User Guide

2.3 Definitions

API: Application Program Interface, a protocol agreed to between the supplier and the user for interaction between the customer's computer program and the supplier's engine performance simulation, when the supplier's program is operating as a subprogram.

ATTRIBUTE: Characteristic of an object. Similar in concept to a variable in FORTRAN – efficiency is an attribute of a compressor, for example. But variables themselves have attributes in object-oriented programming; value, description, units are all typical attributes of a variable.

CLASS: The code which forms the pattern or blueprint for an object type.

CUSTOMER: The person or organization which receives and executes the engine simulation program. Same as USER.

COMPONENT: See ELEMENT.

ELEMENT: The primary building block of a model. Somewhat analogous to a FORTRAN subroutine. An element may represent either a physical engine component or a process. Same as COMPONENT.

ESI: Engineering Status Indicator. An 8 or 9 digit integer conveying error, warning, and message information. This indicator is an extension of the 4-digit NSI (Numerical Status Indicator) Concept defined in AS4191.

GUI: Graphical User Interface.

HOOK SOCKET INTERFACE: A specific object type, provided by the supplier, which allows the user to plug into (or hook up to) certain sockets in the model. Hook sockets allow the user to provide his own custom code for installation effects.

INTERPRETED CODE: Uncompiled code which is interpreted and executed at run time.

INSTANCE: A particular occurrence of an object type. For example, CmpH might be one instance of an object type called Compressor. There can be multiple instances of the same object type.

INSTANTIATE: To create an instance of an object type.

MEMBER FUNCTION: A function (an action) which belongs to an object.

MODEL: (1) A computer program together with the appropriate input which simulates the performance (and possibly other characteristics) of a particular gas turbine engine design. (2) A specific member of a gas turbine product line, e.g., PW120 or CFM56-7.

NODE: A connection between components or elements which passes fluid, fuel, or other properties.

NPSS: Numerical Propulsion System Simulation. An object-orientated simulation development environment developed jointly by NASA and industry.

OBJECT-ORIENTED: "Object orientation means that the program's structure revolves around objects of various types, each of which has a set of attributes accessible by other objects, and certain functions – actions particular to the object – that can be performed at the request of other objects. An object may be a simple entity such as a variable, or a complex entity such as a solver [an object which performs the necessary numerical iteration]" (NPSS User Guide). Java, Ada, and C++ are object-oriented languages.

OBJECT: An instance of a class. Objects may be instantiated within other objects (subobjects). Subobjects (child objects) are said to belong to the object containing them (parent object).

PATH: Analogous to directory path in a file system. Path is the Object/subobject/… chain of ownership of a variable, function, or other object.

PARAMETER: An entity that is used to describe the condition or state of a particular object in the engine or engine program. Example: Amb.alt_in, CbldECS.W.

CALCULATED PARAMETER: Any engine program parameter with a value calculated by the engine program. Often, these are referred to as output parameters. The term calculated is used to avoid ambiguity with input parameters that may also be available for and displayed as output.

INPUT PARAMETER: A parameter that can be set by the customer and which the engine program should not overwrite.

INTERMEDIATE PARAMETER: A parameter which provides the target or demand value for the model solver. Its value may come from the corresponding input parameter, from a user's hook socket instantiation, or from a calculation in the supplier's code.

SOCKET: A particular type of object created within an element or subelement that acts as a placeholder within the code. Subelements and other objects plug into sockets; each socket can accept only one object, of a specific, predetermined type.

SUBELEMENT: An object which is the child of an element or of another subelement.

SUBOBJECT: A more generic term than subelement to describe an object that is the child of another object.

SUPPLIER: The organization which creates and delivers the engine simulation program.

SWITCH: An input parameter with a predefined list of allowable options which selects among alternate paths through the program logic.

USER: See CUSTOMER.

3. MODEL INTERFACE

This section addresses user interface and Application Program Interface (API) issues which do not concern the internal architecture of the model itself. These issues need to be understood by both the supplier and the customer so that an interface protocol beneficial to both may be agreed upon. This document provides a basis for identifying the issues to be discussed. Elements of this recommended practice should only be waived by mutual consent of the program supplier and customer.

3.1 Model Usage Modes

Models are used in two fundamentally different ways or modes. Which mode is used is determined by whether the model supplier or the model user is responsible for (1) executive control of the simulation and (2) the formatting of input and output.

3.1.1 Stand-Alone Mode

For this mode, the program provided by the supplier includes a main program that is used as delivered. There is no Application Program Interface (API), and the supplier is responsible for executive control of the simulation, and for its input and output. The user controls execution through a user interface.

3.1.2 Application Program Interface (API) Mode

For this mode, the program provided by the supplier is used as a subprogram. The model user provides the main program that governs executive control of the simulation, and provides for input and final output. The engine program is thus an application which interacts with the user's software through a mutually agreed API. The supplier generally provides a sample main program (in source code form) for check-out purposes, but the user is free to modify or replace this program.

3.2 Model Types

Model suppliers can provide their models in several different forms. Three common types of models are described in this section.

3.2.1 Engine-Specific Model

The supplier provides a complete program to the user, together with any input files required by the program. The program cannot be modified for other uses; it can only produce results for the specific engine configuration for which it was written. To provide an updated model, the supplier must generally provide a new program as well as new input files. In principle, the supplier could provide program source to the user. Historically, however, it has been more common for the supplier to compile and link an executable for the user's platform.

3.2.2 Shared-Environment Model

The supplier and customer use the same environment kernel to develop and run models. There will likely be differences in the full computing environment of the two companies due to, for example, different configuration management systems. However, the supplier and customer share the essential software (the simulation program itself and any supporting software required by that program) used to produce simulation results. The essential characteristic of shared-environment models is that the supplier need only provide a model definition to the customer, not the complete simulation software. The model definition quite often will consist of one or more files read as input by the simulation program, and may consist of special software modules (either in source or compiled form) called by the simulation program. Ideally, the supplier sends the very same model definition files as used by the supplier internally. This situation is analogous to one user providing a Microsoft Excel workbook model to another Excel user. The supplier needs only send an Excel input file and any macros used by that file. Both supplier and customer share the computing environment kernel (Excel) required to use those files.

3.2.3 Restricted-Environment Model

Programs like NPSS make a third model type possible that stands between the two previously described. As stated above, with shared-environment models the supplier and customer need only share the environment kernel required to run models. However, each company may have, for example, different configuration management systems. Suppose that the supplier's use of the simulation program is deeply integrated with the supplier's configuration management system. In effect, the configuration management system has become part of the environment kernel. For the supplier to send the same model definition files as used internally would require that the customer duplicate the supplier's configuration management system. This may be impractical.

In such a case, the supplier can deliver a model in which the modeling environment footprint is reduced. In this example, a model would be delivered which has been extracted from the supplier's configuration management system. It can still have all the capabilities of a shared-environment model if the supplier chooses to provide them. However, another feature of a restricted-environment model is the ability of the supplier to selectively restrict the user's access to certain aspects of the model such as government security regulations, export restrictions, proprietary agreements (or lack thereof), or any other reasons mutually agreeable to user and supplier.

Historically, often the only way a supplier could reduce the footprint of a delivered model, or restrict access to it, was to make the delivered program engine-specific (as discussed above). Simulation programs like NPSS, however, provide far greater flexibility, allowing the supplier to produce a restricted-environment model that is not limited to a single engine configuration. A restricted-environment model can be created to model many different engine configurations by loading the appropriate input files and/or files containing supplementary code. Some such files may be encrypted for privacy, but others may be open for user modification or replacement. In this way, the user may be able to override agreed-to parts of the simulation as provided by the supplier, or add his own calculations.

For example, a vehicle manufacturer could receive a restricted-environment program from an engine supplier together with the files defining a specific engine, and use the program input capabilities to add installation effects, and even construct a complete vehicle simulation. When the engine model needs to be updated, the engine supplier provides a new set of input files tailored to the program sent earlier. The engine supplier could also provide input files defining a completely different engine as long as they are designed to work with the original program. From the user's point of view, this arrangement is much like a shared-environment model. The difference is that the supplier must initially send a simulation program somewhat modified from the one the supplier uses internally, and may need to modify some model definition files before sending them in order to make them compatible with the program used by the customer.

3.2.4 Shared-Environment and Restricted-Environment Model Architecture

Shared- and restricted-environment models have a great deal in common. Both types of model should be delivered to support the structure shown in Figure 1.

This architecture provides great flexibility in integrating the engine model with the application if desired. The intent is to modularize the different aspects of an integrated engine/vehicle installed performance analysis such that each aspect can be coded, maintained, and even operated independently of all the others. Models need not be hard-coded specifically for one installation.

The "Program Kernel" is defined in this context as the shared environment (e.g., the NPSS binary) which interprets all of the objects and solves the system at the desired conditions.

The "Supplier Engine Model Definition Files" are the files delivered by the supplier. Collectively, these files will include any source code that is not part of the kernel, and any data or input files required to define the engine configuration. The code will generally include provisions for hook sockets which allow users to provide their own code for installation effects such as inlet loss.

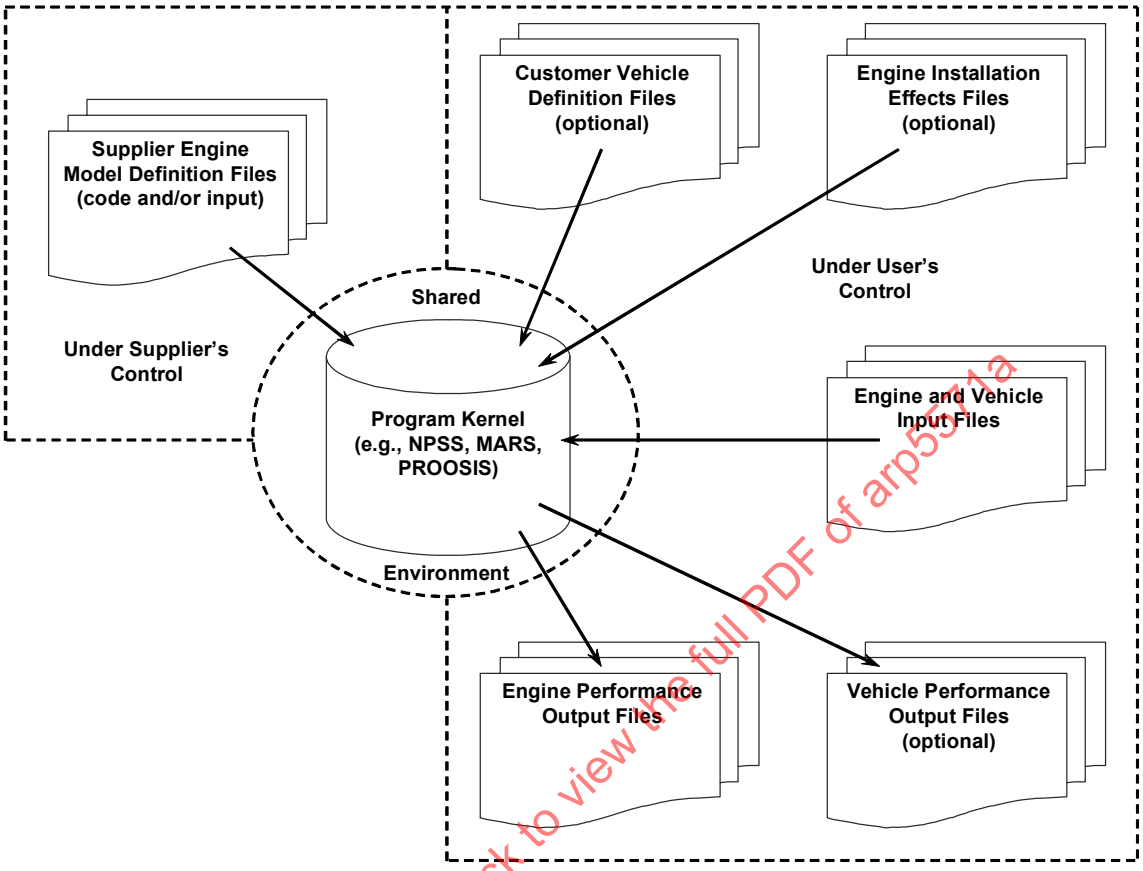

FIGURE 1 - SHARED/RESTRICTED-ENVIRONMENT ARCHITECTURE

"Customer Vehicle Definition Files" are optional files, compatible with the program kernel, which can be used to model the customer's application, or more likely, subsystems of the application. These files could contain transmission or propeller models, for example.

"Engine Installation Effects Files" provide the user owned hook socket instantiations which link up to predetermined sockets in the engine model.

"Engine and Vehicle Input Files" contain the typical inputs to control vehicle and engine operating condition – altitude, ambient conditions, bleed demand settings, etc. This category may also include files to control model output content and format.

Although a model may be delivered with a default output format, output content and format should be under the user's control.

With this architecture it is possible to replace any of the various components of a configuration without modification to any other. This is demonstrated by the following use case:

An engine manufacturer has provided a simulation for a family of engine models. The simulation is in regular use for prediction of engine performance on a specific vehicle configuration.

A preliminary design task has begun, using the existing engine simulation to evaluate changes in the aircraft inlet, bleed, horsepower extraction, exhaust nozzle, and force accounting systems.

All of the aircraft systems being modified interact with the engine model through pre-determined hook sockets in the simulation code.

The customer can quickly evaluate the impact of the proposed system changes on engine and vehicle performance without any changes to the supplier's engine simulation code. He can also evaluate the performance of other members of the same engine family in the modified installation.

3.3 Relationship between Model Usage Mode and Model Type

Engine-specific programs can be delivered either in stand-alone or API mode. With shared-environment models, supplier and customer are generally using the same environment kernel in stand-alone mode. If the customer wants to call the simulation from another main program, it is the customer's responsibility to create an application version of the simulation program. Restricted-environment models may be supplied in either stand-alone or API mode, as agreed between supplier and customer.

3.4 Application Program Interface (API)

The purpose of the API is to provide communication between the customer's main program and supplier's application program. This communication is used to control the execution of the application program, and to exchange data with the main program. It is important for supplier and customer to agree on the API.

3.5 User Interface

A user interface is an interface by which a user communicates with a program, usually being run in stand-alone mode. Command-line interfaces and Graphical User Interfaces (GUIs) are two broad classes of user interface types.

Related to the concept of user interface is program input format. Input files prepared by a user are also a means by which a user controls the execution of a simulation program. Input format is applicable to stand-alone programs and programs run through an API. Output format is also important to all model types and modes. This document will address some issues of user interface, input format, and output format, especially since they take on increased importance in shared-environment models and restricted-environment models.

4. NOMENCLATURE

This section makes detailed recommendations for descriptive names to be used in object-oriented engine models. The lists of parameter and object names in this section are not to be construed as requirements to provide customer access to their values. As has always been the case, suppliers and customers should agree in advance to a mutually acceptable degree of openness in their information exchange, subject to the strictures of government classification, export control, proprietary agreement, and need to know.

4.1 General

Object-oriented nomenclature is quite different from FORTRAN-based nomenclature. Name length is not restricted; fully descriptive labels are encouraged, as are mnemonic abbreviations for names which are frequently typed. Mixed case names are common, because they improve readability of code and input.

4.1.1 Hierarchical Structure

The structure of object-oriented code is hierarchical – objects are the parent or child of other objects. Specifying the full name of an object or variable requires specifying the chain of parent-child relationships leading to the object. This chain of ownership is known as the path of an object. This path is very much analogous to the directory path required to fully describe a file in a directory structure. See Figure 2. Whereas directory names are delineated by a slash (/), object names are delineated by a period, or “dot” (.).

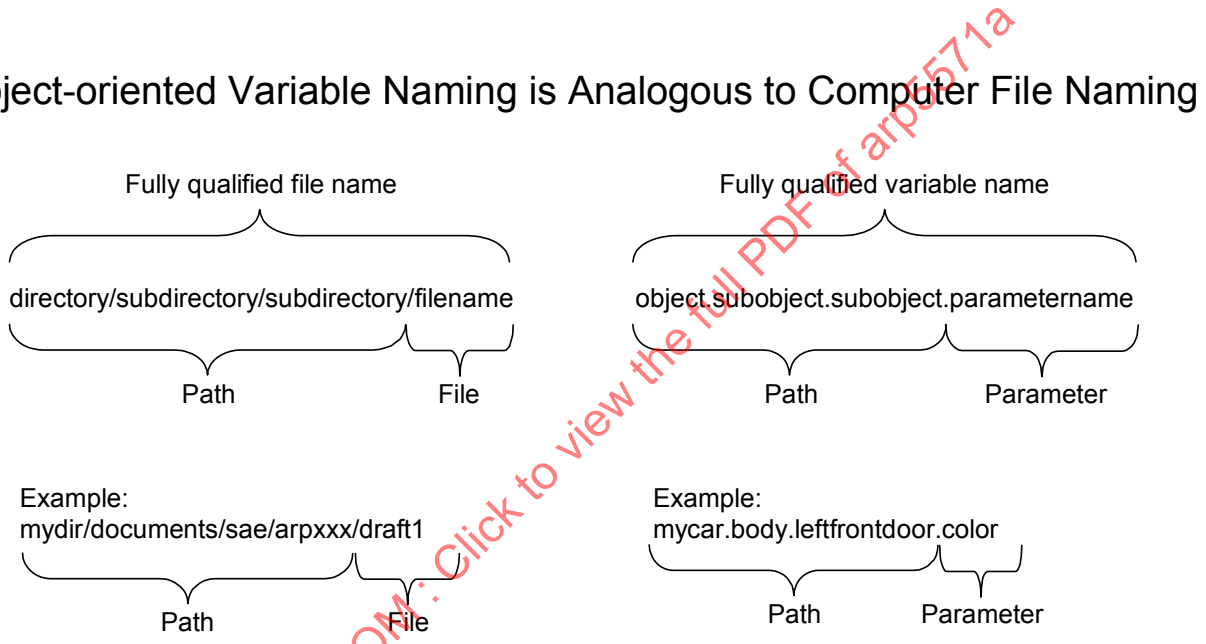

FIGURE 2 - VARIABLE PATH TO DIRECTORY PATH ANALOGY

There are several important details about path which are important to understand.

- Path is relative, just as it is for directories. Any path information which appears in the code for an object is relative to that object.
- A name may be used without any path information in the object where it is created and in all children of that object. The part of the model where an object name can be used without specifying a path is called the object scope. Object names must be unique within their scope.
- Some objects belong to the top level of the model, and require no path in their names. The leading dot is inferred.
- A common name such as T3 might become something like F030.Tt, which is short for Flow Station 030 Total Temperature.

Parameter is not necessarily the lowest level of the hierarchy – parameters in turn own attributes such as their value, description, units, and input/output status.

4.1.2 Object Naming

It is desirable to minimize the amount of path information needed when dealing with common interface variables between the engine and the application. It is also extremely desirable for both the path and parameter names to be common across a wide range of models from different suppliers. To that end, the nomenclature which follows establishes naming conventions for several categories of objects:

- Top level objects representing components or processes that customers frequently interact with, such as inlets, compressors, or bleeds.
- Attributes of these objects, such as input, output, and switch parameter names and allowable options. (Switches are akin to SIM and SERAM in FORTRAN models.)
- Nodes, which are the connection points between components. Node types are fluid (the equivalent of flow stations), fuel, and mechanical.

Suggestions for naming new objects not anticipated in this document are offered in 4.2 for components and in 4.6 for parameters.

4.1.3 Case Sensitivity

Fully descriptive parameter names tend to be quite long. Readability is enhanced if upper and lower case letters are used in predictable ways. To take advantage of mixed-case parameter names, the program kernel must obviously be case-sensitive. The case conventions of the sections which follow should be observed when writing code for any case-sensitive system.

4.2 Component Instance Naming

Using standard instance names improves communication about models between suppliers and customers. Standard naming allows a single application program to be used with different models. Table 1 provides examples of instance names for common components and processes. Each instance name is constructed from a name base and an appended identifier. The name base indicates generically what type the component is, e.g., Cmp for Compressor. The identifier makes the name specific and unique within a given engine model. For example, a model might have CmpFan and CmpH modules. Both the name base and the identifier should have only their initial letters in upper case. Names of components not anticipated in this table should be constructed using these guidelines.

It should be noted that, in models conforming to this document which are of different engine configurations, or come from different suppliers, identically named components may not have identical roles in the cycle. As an example, in one model, CmpH might represent the entire gas generator compressor, while in another, CmpH might represent only the centrifugal stage of a gas generator compressor comprising several axial stages and a centrifugal stage on the same shaft. Also, a component in the context of this table may be a single object or a collection of objects. CmpH might represent solely the compression process between two fluid nodes, or it might also include additional processes such as bleed and exit diffusion. All these examples conform to this recommended practice. Communication and agreement between supplier and customer are critical.

TABLE 1 - RECOMMENDED COMPONENT INSTANCE NAMES

Component or Process	Name Base	Recommended Instances[1]	Instance Description
Ambient	Amb	Amb	Freestream conditions
Bleed	B	---[2,3]	
Burner	Brn	BrnPri	Primary burner
		BrnAug	Augmentor (afterburner)
Compressor	Cmp	CmpFSec	Fan OD or secondary (split fan)
		CmpFPri	Fan ID or primary (split fan)
		CmpFan	Fan (not split)
		CmpL	Low pressure compressor
		CmpI	Intermediate pressure compressor
		CmpH	High pressure compressor
		CmpLd	Load compressor
Control	Ctrl	Ctrl	Engine control
Customer Bleed	Cbld	CbldECS	Customer bleed – environmental control syst.
		CbldAI	Anti-ice customer bleed
		CbldWAI	Wing anti-ice customer bleed
		CbldCAI	Cowl anti-ice customer bleed
		CbldPreC	Precooler bleed
Customer Drag	Cdrag	CdragSt	Stream Tube Drag
		CdragIn	Inlet Drag
		CdragNoz	Nozzle boat tail Drag
		CdragPod	Engine Pod Drag
Customer Power Extraction	Cpwr	CpwrL	Power extraction from low spool
		CpwrI	Power extraction from intermediate spool
		CpwrH	Power extraction from high spool
Cycle Summary	Cycle	Cycle	Overall cycle parameters, e.g., bypass ratio
Duct	D	---[2]	
Duct Instrumented	Di	---[2]	
Engine Interface	Eint	Eint	Engine Interface
Flow-duplicator	Fdup	FdupPri	Primary stream flow duplicator
		FdupSec	Secondary stream flow duplicator
Flow-end	Fe	FePri	Primary stream flow end
		FeMix	Mixed stream flow end
		FeSec	Secondary stream flow end
Flow-start	Fs	FsEng	Total engine stream flow start
Fuel-splitter	Fusplt	FuspltEng	Splits fuel output of FusEng
Fuel-start	Fus	FusEng	Single fuel start for engine[4]
		FusPri	Multiple fuel starts - primary[4]
		FusAug	Multiple fuel starts - augmentor[4]

TABLE 1 - RECOMMENDED COMPONENT INSTANCE NAMES (CONTINUED)

Component or Process	Name Base	Recommended Instances[1]	Instance Description
Gear	Gear	GearFan	Gearbox driving fan
		GearProp	Gearbox driving Propeller
Heat exchanger	Hx	---[2]	
Inlet	In	InEng	Engine inlet (non-split flow)
		InPri	Primary inlet (split flow)
		InSec	Secondary inlet (split flow)
Mixer	Mix	MixExh	Engine Exhaust stream mixer
Nozzle	Noz	NozPri	Primary stream nozzle
		NozSec	Secondary stream nozzle
		NozMix	Mixed stream nozzle
Propeller	Prop	Prop	Propeller
Performance	Perf	Perf	Engine performance (base-UnInstalled)
		PerfSt	Stream-tube-thrust engine performance
		PerfPod	Podded-thrust engine performance
		PerfMin	Minimum-engine performance
Power Setting	Pset	Pset	Power Setting
Shaft	Sh	ShL	Low Pressure shaft
		ShLG	Low Pressure shaft – geared
		ShI	Intermediate Pressure shaft
		ShH	High Pressure shaft
		ShP	Power turbine shaft
Splitter	Splt	SpltFan	Fan Splitter (Pre or Post)
Turbine	Trb	TrbL	Low Pressure turbine
		TrbI	Intermediate Pressure turbine
		TrbH	High Pressure turbine
		TrbP	Free Power Turbine

Notes to Table 1

(1) For components not described in the table, or for components that do not have recommended instances, the identifying suffix may be a string or a station label. If the identifier is a string, its first letter is capitalized with following letters in lower case unless they start a new word. If the instance identifier is a station number, it should conform to the “Alternate System for Reduced Ambiguity” of AS755.

(2) The identifying suffix for the indicated components may be either a station designation or a commonly used abbreviation. Examples: B030, D125, DEGV, and Di030.

(3) The Bleed object does the bookkeeping of bleed flow with respect to the main flow. The bleed object may be identified by either its entrance or its exit station. In Typical usage, bleed extraction processes should be named after their entrance stations, and the bleed return processes are named after their exit stations. For example: B030 (entrance station 030, exit station 031) might model bleed extracted after the compressor, and B041 (entrance 040, exit station 041) bleed returned to the main flow ahead the turbine rotor.

(4) Use FusEng when a single fuel source is used. This may provide fuel to a single burner or, when used with a fuel splitter, provide fuel for two burners. Use FusPri and FusAug if two separate fuel sources are modeled, one for the main burner and one for the augmentor (afterburner). If two or more fuel sources are used for other reasons, select meaningful names – FusJP4, FusJP5, FusTank1, FusTank2.

4.3 Node Naming

Nodes provide access to variables that link components. Like all variable names, the node variable names consist of two parts; the path and the parameter name. The path is simply the node name. The node name begins with a prefix that defines its type:

a. Fluid flow node names begin with F

b. Fuel node names begin with Fu

c. Mechanical node names begin with Me

4.3.1 Fluid Flow Nodes

A fluid flow node provides the connection for flow passing from one component to another. A component may have multiple fluid input ports and fluid output ports. By definition, a fluid node connects one output port to one input port of another component. The fluid flow node name is the F prefix followed by a station number. The conventions for selecting station numbers are as follows (see also AS755):

a. The only one-digit station allowed is 0; all other stations require three digits.

b. The first digit of a station number indicates flow stream

- The primary stream is indicated by a leading 0
- Additional streams are numbered consecutively, beginning with 1
- Streams may be numbered from innermost to outermost, or in their order of branching

c. The second digit of a station number indicates process, e.g., compression from x2x to x3x, expansion from x4x to x5x, etc.

d. The third digit (required even if 0) and subsequent digits (optional) indicate intermediate stations.

e. Flow at a station may be characterized or subdivided into multiple, co-planar stations by the addition of one or more characters, with first character being a letter to avoid confusion with the station number. Examples:

- A for average
- a, b, c, … for flow stratification

f. Stations intermediate between two stations with consecutive numbers may be created by appending digits to the upstream station number. e.g., 0252 is intermediate between 025 and 026, 02524 is intermediate between 0252 and 0253.

g. If two streams mix, succeeding station numbers will be consistent with the lowest leading digit stream.

h. If multiple, coplanar streams are totaled or averaged, the station number will be consistent with the innermost (lowest leading digit) stream.

i. Within a stream, station identifiers should always be in ascending numerical or alphabetic order in the direction of flow.

Examples of fluid node naming are shown in Table 2.

TABLE 2 - FLUID NODE NAMING EXAMPLES

Traditional Station No.	Recommended Station No.	Fluid Node Name
2	020	F020
2.5	025	F025
2A	020A	F020A
4	040	F040
4.1	041	F041
4.18	0418	F0418
14	140	F140
4.90	049	F049
49.00	490	F490

4.3.2 Fuel Nodes

Fuel node names start with the letters Fu followed by an identifier for the fuel stream they apply to. For examples see Table 3.

TABLE 3 - FUEL NODE NAMING EXAMPLES

Fuel Stream	Node Name
Engine total fuel supply	FuEng
Main burner fuel supply	FuPri
Afterburner fuel supply	FuAug

4.3.3 Mechanical Nodes

Mechanical nodes link components to shafts. Mechanical node names begin with letters Me followed by the description of the component that it is being linked to the shaft. The examples of Table 4 illustrate some possibilities. (Note that in case of the gear component, there are 2 nodes connecting to the appropriate shafts).

TABLE 4 - MECHANICAL NODE NAMING EXAMPLES

Description	Component Name	Node Name
Fan OD	CmpFSec	MeCmpFSec
Fan ID	CmpFPri	MeCmpFPri
Low pressure compressor	CmpL	MeCmpL
High pressure turbine	TrbH	MeTrbH
Low pressure turbine	TrbL	MeTrbL
Prop gearbox	GearProp	MeGearPropIn
		MeGearPropOut

4.4 Parameter Naming

Table 5 provides recommended names for attributes (parameters) of the components and nodes described in 4.2 and 4.3. This table is not intended to be an exhaustive list, nor is it intended to be a minimum list of parameters that must always be available; rather it is a list of common parameters that at least some users will need to interact with. If customer needs and supplier-customer agreements dictate that access be given to a particular parameter, then it should have the name shown in the table. It should also be noted that suppliers differ in their methodology; some parameters defined in this table may not exist in a given simulation.

The parameter names shown in the table are part of full variable names in the path.parameter format, where path = the component instance name. For example, the parameter eff is listed for compressors, burners, and turbines. Full variable names might be CmpH.eff, BrnPri.eff, and TrbL.eff, respectively.

Note that there are up to three parameter names listed for some quantities. All three names should be available for display or printout. The values of parameters listed in the Input column will never be changed by the code. Parameters in the Input column with the suffix _in are likely to be superseded either by the user's hook sockets as described in Section 5 or by constraints in the engine code. Thus the values used by the program may be different than those requested via input. Parameters with a _in suffix preserve the input value and are analogous to those with a Z prefix as specified in AS681.

A few parameters with a suffix _dmd appear in the Intermediate column. Their values are those which are demanded of the engine model. Such a parameter will frequently have the same value as the corresponding _in parameter. In other cases, its value may be set by the user's hook functionality, or it may be set by the engine code. In any case, _dmd values are the target values for the engine solver balance, but they may be overridden by constraints.

The values of parameters in the Output column are the final values calculated by the simulation. In the most general situation, parameter, parameter_in, and parameter_dmd may all have different values.

4.5 Units

The units listed in Table 5 are in compliance with AS681 for U.S. Customary Units. The equivalent SI or Imperial Units are also acceptable.

For certain parameters, the units in customary use throughout the aircraft industry are exceptions to U.S Customary Units and/or to SI units. Knot for flight speed is one example; cockpit temperature gauges in °C are another; rotational speeds are generally rendered in rpm, regardless of the system of units.

Suppliers should consult with customers and make provisions for allowing input and output in the most useful units.

TABLE 5 - RECOMMENDED ATTRIBUTE (PARAMETER) NAMES

Ambient Object (Amb)[1]				
Input	Intermediate	Output	Description	Units
alt_in		alt	Altitude	ft
dTs_in		dTs	Delta static temperature from selected atmosphere	°R
		dTsStd	Delta static temperature from standard atmosphere	°R
humRel_in		humRel	Relative humidity = Pvapor / Psaturation	
humSp_in		humSp	Specific humidity = mass of water vapor / mass of dry air	
MN_in		MN	Mach number	
Ps_in		Ps	Ambient (static) pressure	lbf/in^2
Pt_in		Pt	Free stream total pressure	lbf/in^2
Ts_in		Ts	Ambient (static) temperature	°R
Tt_in		Tt	Free stream total temperature	°R
		TsDay	Ambient (static) temperature based on selected day	°R
VEAS_in		VEAS	Equivalent air speed	knot
VCAS_in		VCAS	Calibrated air speed	knot
VTAS_in		VTAS	True free stream air speed	knot
switchDay			Ambient temperature day selector	
switchHum			Humidity calculation method selector	
switchMode			Flight conditions input mode selector	
Burner Object (Brn*)				
Input	Intermediate	Output	Description	Units
		eff	Adiabatic burner efficiency	
		dPqP	Pressure loss (delta P / P)	
		Wfuel	Burner fuel flow	lbm/sec
Compressor Object (Cmp*)				
Input	Intermediate	Output	Description	Units
		eff	Adiabatic efficiency	
		effPoly	Polytropic efficiency	
		Nc	Corrected speed, N / sqrt(T/Tref)	rpm
		NcPct	Percent corrected speed	
		Np	N / sqrt(Tt)	rpm/sqrt(°R)
		PR	Pressure ratio	
		pwr	Aerodynamic power or work	hp
		SMN	Stall margin at constant speed	
		SMW	Stall margin at constant flow	
		TR	Temperature ratio	

TABLE 5 - RECOMMENDED ATTRIBUTE (PARAMETER) NAMES (CONTINUED)

Customer Bleed Object (Cbld*) – (other than Customer Bleed PreCooler Object)				
Input	Intermediate	Output	Description	Units
		ht	Total specific enthalpy of bleed flow	Btu/lbm
		Pt	Total pressure of bleed flow	lbf/in^2
		Tt	Total temperature of bleed flow	°R
W_in	W_dmd	W	Bleed mass flow	lbm/sec
WqWref_in		WqWref	Bleed mass flow fraction	
		Wref	Reference mass flow	lbm/sec
		WrefName	Name of bleed reference object	
switchW			Customer bleed flow calculation method selector	
switchSource			Customer bleed source selector(use only if multiple sources)	
Customer Bleed PreCooler Object (CbldPreC)				
Input	Intermediate	Output	Description	Units
		effect	Effectiveness	
		ht	Total specific enthalpy of exiting cooling flow	Btu/lbm
		htServ	Total specific enthalpy of exiting service bleed flow	Btu/lbm
		Pt	Total pressure of exiting cooling flow	lbf/in^2
		PtServ	Total pressure of exiting service bleed flow	lbf/in^2
		Tt	Total temperature of exiting cooling flow	°R
TtServ_in	TtServ_dmd	TtServ	Total temperature of exiting service bleed flow	°R
W_in	W_dmd	W	Bleed mass flow	lbm/sec
WqWref_in		WqWref	Bleed mass flow fraction	
		Wref	Reference mass flow	lbm/sec
		WrefName	Name of bleed reference object	
switchW			Precooler bleed flow calculation method selector	
Customer Drag Object (Cdrag*)				
Input	Intermediate	Output	Description	Units
Fd_in	Fd_dmd	Fd	Drag	lbf
switchFd			Drag calculation method selector	
Customer Power Extraction Object (Cpwr*)				
Input	Intermediate	Output	Description	Units
pwr_in	pwr_dmd	pwr	Customer power extraction	hp
pwrqpwrRef_in		pwrqpwrRef	Power extraction as a fraction of pwrRef	
		pwrRef	Reference power	hp
		pwrRefName	Variable name that provides pwrRef value	
switchPwr			Power extraction calculation method selector	

TABLE 5 - RECOMMENDED ATTRIBUTE (PARAMETER) NAMES (CONTINUED)

Cycle Summary Object (Cycle)				
Input	Intermediate	Output	Description	Units
		BPR	Engine bypass ratio, typically W120/W020	
		EPR	Engine pressure ratio	
		FPR	Fan pressure ratio	
		OPR	Overall compressor pressure ratio	
Duct Object (D*)				
Input	Intermediate	Output	Description	Units
dPqP_in	dPqP_dmd	dPqP	Pressure loss (delta P / P)	
Q_in	Q_dmd	Q	Heat flow into fluid	Btu/sec
switchDP			Pressure loss calculation method selector	
switchQ			Heat flow calculation method selector	
Engine Interface Object (Eint)				
Input	Intermediate	Output	Description	Units
AOA_in		AOA	Angle of attack (Alpha)	
AOS_in		AOS	Angle of sideslip (Beta)	
PLA_in		PLA	Power lever angle	
switchWOW			Weight on wheels indicator	
switchLG			Landing gear indicator	
switchAI			Engine anti-ice on/off (controls internal engine anti-ice)	
switchPA			Powered approach mode indicator	
Fuel Start Object (Fus*)				
Input	Intermediate	Output	Description	Units
		CHRatio	Carbon to hydrogen ratio	
LHV_in		LHV	Lower heating value	Btu/lbm
		Tfuel	Fuel temperature	°R
		Wfuel	Fuel mass flow	lbm/sec
spGrav_in		spGrav	Specific gravity	
Gear Object (Gear*)				
Input	Intermediate	Output	Description	Units
		gearRatio	Gear ratio, speed out / speed in	
		eff	Power out / power in	

TABLE 5 - RECOMMENDED ATTRIBUTE (PARAMETER) NAMES (CONTINUED)

Heat Exchanger Object (Hx*)				
Input	Intermediate	Output	Description	Units
		cap1	Capacity of flow stream1 (W*Cp)	Btu/(sec*°R)
		cap2	Capacity of flow stream2 (W*Cp)	Btu/(sec*°R)
		dPqP1	Pressure loss in stream 1	
		dPqP2	Pressure loss in stream 2	
		effect	Heat transfer effectiveness	
Q_in	Q_dmd	Q	Heat flow between sides	Btu/sec
switchQ			Heat flow calculation method selector	
Inlet Object (In*)				
Input	Intermediate	Output	Description	Units
		Afs	Freestream area	In^2
Aref_in		Aref	Reference area for inlet recovery	in^2
		Fram	Ram drag	lbf
PqP_in	PqP_dmd	PqP	Inlet recovery (average)	
PqP1_in	PqP1_dmd	PqP1	Inlet recovery (stream 1)	
PqP2_in	PqP2_dmd	PqP2	Inlet recovery (stream 2)	
PtRef_in		$PtRef^2$	Target downstream pressure (typically fan face), for use with option switchRec="PTREF"	lbf/in^2
		$WcRef^2$	Target corrected mass flow at reference location (typically fan face); useful in situations where PqP_dmd = f(WcRef)	lbm/sec
switchRec			Inlet recovery option	
Mixer Object (Mix*)				
Input	Intermediate	Output	Description	Units
		impMixed	Total impulse of two incoming streams (pressure force + momentum)	lbf
		impOut	Total impulse of exiting stream	lbf
			(pressure force + momentum)	
		partialMix	Partial mixing correction term	

TABLE 5 - RECOMMENDED ATTRIBUTE (PARAMETER) NAMES (CONTINUED)

Nozzle Object (Noz*)				
Input	Intermediate	Output	Description	Units
		AeTh	Nozzle throat effective area determined from the cold throat area, and the thermal expansion and flow coefficients	in^2
		Aexit	Physical exit area not fully expanded	in^2
		Ath	Throat area determined from the cold throat area and the thermal expansion coefficient	in^2
CdTh_in	CdTh_dmd	CdTh	Nozzle throat discharge coefficient	
Cfg_in	Cfg_dmd	Cfg	Nozzle exit gross thrust coefficient	
Cv_in	Cv_dmd	Cv	Velocity coefficient	
		Fg	Nozzle gross thrust	lbf
		FgIdeal	Nozzle ideal gross thrust	lbf
		PsExh	Nozzle exhaust static pressure	lbf/in^2
		PR	Nozzle pressure ratio	
		Vactual	Nozzle exit actual velocity	ft/sec
switchCoef			Select how losses are determined, via velocity or thrust coefficient	
switchCdTh			Discharge coefficient calculation method	
switchCfg			Gross thrust coefficient calculation method	
switchCv			Velocity coefficient calculation method	
Performance Object (Perf*)				
Input	Intermediate	Output	Description	Units
		Fd	Installation Drag (Excluding Fram)	lbf
		Fg	Gross thrust	lbf
		Fn	Net thrust	lbf
		Fnc	Corrected net thrust	lbf
		Fram	Ram drag	lbf
		SFC	Specific fuel consumption (power or thrust)	
		Wfuel	Fuel flow used in SFC calculation	lbm/hr
		pwrSD	Delivered shaft power	hp

TABLE 5 - RECOMMENDED ATTRIBUTE (PARAMETER) NAMES (CONTINUED)

Power Setting Object (Pset)				
Input	Intermediate	Output	Description	Units
RCfrac_in	RCfrac_dmd	RCfrac	Rating fraction – fraction of rated thrust or power given by switchRC .	
parm_in	parm_dmd	parm	Demanded value of variable set by switchParm when switchRC=”PARM”	
switchLimit			Constraint parameter group selector	
switchLimitSet			Constraint value group selector	
switchRating			Rating group selector	
switchRC			Rating code selector	
switchParm			Power setting option selector when switchRC=”PARM”	
Propeller Object (Prop)				
Input	Intermediate	Output	Description	Units
		advRatio	Propeller advance ratio	
		Cpwr	Power coefficient	
		CT	Thrust coefficient	
		dia	Diameter	in
		Fg	Thrust	lbf
		gearRatio	Gear ratio	
		inertia	Rotational inertia	$slug*in^2$
		Utip	Tip speed	ft/sec
		pwr	Power from shaft	hp
Shaft Object (Sh*)				
Input	Intermediate	Output	Description	Units
		dNqdt	Derivative of speed with respect to time	rpm/sec
		inertia	Inertia of shaft only	$slug*in^2$
		inertiaSum	Total inertia on shaft (including attached components)	$slug*in^2$
		Nmech	Mechanical speed of shaft	rpm
		pwrNet	Total of all power on the shaft	hp
		trqNet	Total of all torque on the shaft	ft*lbf
Splitter Object (Splt*)				
Input	Intermediate	Output	Description	Units
		W2qW1	Flow ratio, W2/W1	
		dPqPavg	Mass average delta P/P	
		dPqP1	Normalized pressure loss in first output stream	
		dPqP2	Normalized pressure loss in second output stream	
		W1qW	Flow fraction = W1/ (W1+W2)	
		W2qW	Flow fraction = W2/(W1+W2)	

TABLE 5 - RECOMMENDED ATTRIBUTE (PARAMETER) NAMES (CONTINUED)

Turbine Object (Trb*)				
Input	Intermediate	Output	Description	Units
		dhqT	Total specific enthalpy change over inlet temperature	Btu/(lbm*°R)
		dht	Total specific enthalpy change	Btu/lbm
		eff	Adiabatic efficiency	
		Nc	Corrected speed, N / sqrt(theta)	rpm
		Np	Corrected speed, N / sqrt(Tt)	rpm/sqrt*°R
		PR	Expansion pressure ratio	
		pwr	Aerodynamic power or work	hp
Fluid node (F*)				
Input	Intermediate	Output	Description	Units
		A	Physical cross sectional area	in^2
		s	Entropy	Btu/(lbm*°R)
		FAR	Fuel to air ratio	
		gams	Gamma based on static conditions	
		gamt	Gamma based on total conditions	
		hs	Static specific enthalpy	Btu/lbm
		ht	Total specific enthalpy	Btu/lbm
		WAR	Specific humidity = mass of water vapor / mass of dry air	
		MN	Mach number	
		mu	Viscosity	lbm/(in*sec)
		Ps	Static pressure	lbf/in^2
		Pt	Total pressure	lbf/in^2
		rhos	Density at static conditions	lbm/ft^3
		rhot	Density at total conditions	lbm/ft^3
		Rs	Gas constant based on static conditions	Btu/(lbm*°R)
		Rt	Gas constant based on total conditions	Btu/(lbm*°R)
		V	Velocity	ft/sec
		Ts	static temperature	°R
		Tt	total temperature	°R
		W	Mass flow	lbm/sec
		Wc	Corrected mass flow, W sqrt(theta) / delta	lbm/sec
		Wp	Mass flow parameter, W sqrt(Tt) / Pt	$\frac{lbm*sqrt(°R)}{sec*lbf/in^2}$

TABLE 5 - RECOMMENDED ATTRIBUTE (PARAMETER) NAMES (CONTINUED)

Fuel node (Fu*)				
Input	Intermediate	Output	Description	Units
		CHRatio	Carbon to hydrogen ratio	
		hFuel	Specific enthalpy	Btu/lbm
		LHV	Lower heating value	Btu/lbm
		Pfuel	Pressure	lbf/in^2
		Tfuel	Temperature	°R
		Wfuel	Mass flow	lbm/sec
Mechanical node (Me*)				
Input	Intermediate	Output	Description	Units
		inertia	Inertia	$slug*in^2$
		Nmech	Rotational speed	rpm
		pwr	Power	hp
		trq	Torque	ft*lbf

Notes to Table 5:

(1) The * in some object names in this table denotes one or more characters to complete the instance name per Table 1.

(2) PtRef and WcRef are normally target values for parameters which exist at flow stations downstream of the Inlet object. Therefore to avoid a used-before-calculated situation, one typically guesses their values and allows the solver to vary them to match the specific downstream condition.

4.6 Parameter Naming Guidelines

When a new name must be created that is not already defined in the Table 5, use the following general guidelines to create a base or root name. The name may be further qualified by using the prefixes and suffixes of Table 6.

a. Names need be unique only within scope.

b. Use only alphabetic characters, digits, and the underscore character, "_". Names cannot include spaces, dots (periods), hyphens, or any other punctuation marks or special characters.

c. There is no limit on name length but keep name short and make it just long enough to be meaningful.

d. Remember that the name is case sensitive.

e. Do not begin the name with a digit.

f. Do not use a program keyword (check program manual).

g. Names are lower case except for common usage names such as A, P, T, C, Re, FAR, MN, N, PR, TR, W, WAR.

h. Second and all subsequent words have their initial letter alternate case from the last letter of the previous word. i.e. WcRef, pwrRefName.

i. Suffixes for total and static are always lower case: t, s.

j. Suffix for percent is always lower case: pct.

k. Suffix for corrected is always lower case: c.

l. All suffixes in i), j), and k) are lower case, even when used in combination. The order of the suffixes when combined is arbitrary.

m. The abbreviation for quotient is always lower case: q.

n. Do not include station label as part of any parameter name. This information is included with the variable scope.

TABLE 6 - QUALIFYING PREFIXES AND SUFFIXES

Prefix	Description
_	Denotes private or local variable
a_	Denotes adder variable (always lower case)
s_	Denotes scaler variable (always lower case)
switch	Option switch variable
C_	Constant value (always upper case)
Suffix	Description
base	Unadjusted base variable
bld	Bleed
c	Corrected variable (always lower case)
des	Design
e	Effective (always lower case)
ideal	Ideal
map	Map variable
mech	Mechanical
pct	Percent (always lower case)
poly	Polytropic
ref	Reference
s	Static (always lower case)
t	Total (always lower case)
th	Throat
tip	Tip
_dmd	Demand variable (see 4.4) (always lower case)
_in	Input variable (see 4.4) (always lower case)

4.7 Option Switches

Table 5 lists many variables with the prefix “switch”. These switches control how the various objects perform their calculations. Each switch has a limited number of allowed values.

Table 7 provides a recommended list of allowed values (with brief descriptions) for every switch parameter listed in Table 5. Many of the switches have a setting of “CUSTOM” recommended. The “CUSTOM” setting allows the user to provide his own code to link to predetermined hook sockets in the simulation code. The table also lists the socket type required for the customer’s hook classes. The first option listed for each switch in the table is the switch default option.

TABLE 7 - RECOMMENDED SWITCH VARIABLE OPTION SETTINGS

Ambient Object (Amb)					
Switch Description	Switch Name	Option Description	Option Setting	Socket Name	Variable Set by Socket
Ambient temperature day types as defined in ARP210	switchDay	Minimum Recorded	MINREC		
		1% Cold Day	COLD1PCT		
		5% Cold Day	COLD5PCT		
		10% Cold Day	COLD10PCT		
		20% Cold Day	COLD20PCT		
		Cold Day	COLD		
		Polar Day	POLAR		
		Standard Day	STD		
		Tropical Day	TROP		
		Hot Day	HOT		
		20 % Hot Day	HOT20PCT		
		10 % Hot Day	HOT10PCT		
		5 % Hot Day	HOT5PCT		
		1 % Hot Day	HOT1PCT		
		Maximum Recorded	MAXREC		
		Customer hook socket	CUSTOM	S_customDay	TsDay
Humidity type	switchHum	Specific humidity, humSp_in is input	SPECIFIC		
		Relative humidity, humRel_in is input	RELATIVE		
		Humidity per Federal Aviation Regulation Part 25 and Part 23 (which is the same as Part 25)	FAR25		
Ambient conditions calculation mode	switchMode	Select demand variables to define flight conditions	ALDTMN		
			ALDTVC		
			ALDTVT		
			ALTSMN		
			PSTSMN		
			PSTSTT		
Note:	Options are constructed from three, 2-character strings indicating demand variables. Additional options may be provided.				
	First string indicates a pressure or altitude demand				
	AL = altitude				
	PS = static pressure				
	PT = total pressure				
	Second string indicates a temperature demand				
	DT = difference from day temperature				
	TS = static temperature				
	TT = total temperature				
	Third string indicates a vehicle speed demand				
	MN = Mach number				
	VC = calibrated air speed				
	VT = true air speed				
	VE = equivalent air speed				
Example:	Amb.switchMode="ALDTVT"				
	allows input of demand variables				
	Amb.alt_in, Amb.dTs_in, and Amb.VTAS_in				

TABLE 7 - RECOMMENDED SWITCH VARIABLE OPTION SETTINGS (CONTINUED)

Customer Bleed Object (Cbld*) – (other than Customer Bleed PreCooler Object)					
Switch Description	Switch Name	Option Description	Option Setting	Socket Name	Variable Set by Socket
Customer bleed calculation method	switchW	Bleed is not flowing	OFF		
		Bleed flow rate is input	INPUT		
		Bleed flow fraction is input	FRAC		
		Bleed flow rate and fraction are input	INPUT+FRAC		
		Supplier-provided calculation	CALCULATE		
		Customer hook socket	CUSTOM	S_customW	W_dmd
Customer bleed source selector (use only for multiple sources)	switchSource	Lowest pressure source	LOW		
		Medium pressure source (use only if 3 sources)	MED		
		Highest pressure source	HIGH		
		Bleed source selected automatically	AUTO		
Customer Bleed Precooler Object (CbldPreC)					
Switch Description	Switch Name	Option Description	Option Setting	Socket Name	Variable Set by Socket
Precooler bleed calculation method	switchW	Bleed is not flowing	OFF		
		Bleed flow rate is input	INPUT		
		Bleed flow fraction is input	FRAC		
		Bleed flow rate and fraction are input	INPUT+FRAC		
		Bleed flow is calculated to satisfy input service bleed temperature	TSERV		
		Supplier-provided calculation	CALCULATE		
		Customer hook socket	CUSTOM	S_customW	W_dmd
Customer Drag Object (Cdrag*)					
Switch Description	Switch Name	Option Description	Option Setting	Socket Name	Variable Set by Socket
Drag calculation method	switchFd	Drag is not calculated	OFF		
		Drag is input	INPUT		
		Supplier-provided calculation	CALCULATE		
		Customer hook socket	CUSTOM	S_customFd	Fd_dmd

TABLE 7 - RECOMMENDED SWITCH VARIABLE OPTION SETTINGS (CONTINUED)

Customer Power Extraction Object (Cpwr*)					
Switch Description	Switch Name	Option Description	Option Setting	Socket Name	Variable Set by Socket
Customer power extraction calculation method	switchPwr	No power extraction	OFF		
		Power extraction is input	INPUT		
		Power extraction fraction is input	FRAC		
		Power extraction and fraction are input	INPUT+FRAC		
		Supplier-provided calculation	CALCULATE		
		Customer hook socket	CUSTOM	S_customPwr	pwr_dmd
Duct Object (D*)					
Switch Description	Switch Name	Option Description	Option Setting	Socket Name	Variable Set by Socket
Pressure loss calculation method	switchDP	Supplier-provided calculation	CALCULATE		
		No Pressure loss	OFF		
		Pressure loss, dPqP_in, is input	INPUT		
		Customer hook socket	CUSTOM	S_customDP	dPqP_dmd
External heat load calculation method	switchQ	No heat load	OFF		
		Heat load is input	INPUT		
		Supplier-provided calculation	CALCULATE		
		Customer hook socket	CUSTOM	S_customQ	Q_dmd
Engine Interface Object (Eint)					
Switch Description	Switch Name	Option Description	Option Setting	Socket Name	Variable Set by Socket
Weight-on-wheels indicator	switchWOW	Off	OFF		
		On	ON		
Landing gear indicator	switchLG	Up	UP		
		Down	DOWN		
Engine anti-ice selector	switchAI	Off	OFF		
		On	ON		
Heat Exchanger Object (Hx*)					
Switch Description	Switch Name	Option Description	Option Setting	Socket Name	Variable Set by Socket
External heat load calculation method	switchQ	No heat load	OFF		
		Heat load is input	INPUT		
		Supplier-provided calculation	CALCULATE		
		Customer hook socket	CUSTOM	S_customQ	Q_dmd

TABLE 7 - RECOMMENDED SWITCH VARIABLE OPTION SETTINGS (CONTINUED)

Inlet Object (In*)					
Switch Description	Switch Name	Option Description	Option Setting	Socket Name	Variable Set by Socket
Inlet recovery calculation method	switchRec	Ideal average recovery	IDEAL		
		MIL-E-5007D average recovery	MILE5007D		
		Input average recovery	INPUT		
		Average recovery defined by setting PtRef_in	PTREF		
		Supplier-provided calculation	CALCULATE		
		Customer hook socket	CUSTOM	S_customRec	PqP_dmd PqP1_dmd PqP2_dmd
Nozzle Object (Noz*)					
Switch Description	Switch Name	Option Description	Option Setting	Socket Name	Variable Set by Socket
Loss calculation mode	switchCoef	Velocity coefficient, Cv, mode	CV		
		Gross thrust coefficient, Cfg, mode	CFG		
Discharge coefficient calculation method	switchCdTh	Ideal (Discharge coefficient = 1.0)	IDEAL		
		Supplier-provided calculation	CALCULATE		
		Discharge coefficient input	INPUT		
		Customer hook socket	CUSTOM	S_customCdTh	CdTh_dmd
Gross thrust coefficient calculation method	switchCfg	Ideal (Thrust coefficient = 1.0)	IDEAL		
		Supplier-provided calculation	CALCULATE		
		Gross thrust coefficient input	INPUT		
		Customer hook socket	CUSTOM	S_customCfg	Cfg_dmd
Velocity coefficient calculation method	switchCv	Ideal (Velocity coefficient = 1.0)	IDEAL		
		Supplier-provided calculation	CALCULATE		
		Velocity coefficient input	INPUT		
		Customer hook socket	CUSTOM	S_customCv	Cv_dmd

TABLE 7 - RECOMMENDED SWITCH VARIABLE OPTION SETTINGS (CONTINUED)

Power Setting Object (Pset)					
Switch Description	Switch Name	Option Description	Option Setting	Socket Name	Variable Set by Socket
Rating code selector	switchRC	Augmented maximum	MAXAB		
		Augmented minimum	MINAB		
		Wet Takeoff	WETTO		
		Emergency or Contingency	EMER		
		30 second OEI (One Engine Inoperative)	OEI30S		
		2 minute OEI	OEI2MIN		
		2 1/2 minute OEI	OEI2P5MIN		
		30 minute OEI	OEI30MIN		
		Continuous OEI	OEICT		
		Automatic power reserve	APR		
		Non-augmented max	MRP		
		Non-augmented intermediate	IRP		
		Non-augmented max takeoff	MTO		
		Non-augmented takeoff	TO		
		Non-augmented takeoff / go around	TOGA		
		Non-augmented max continuous	MCT		
		Non-augmented max climb	MCL		
		Non-augmented max cruise	MCR		
		Approach Idle	APRIDLE		
		Flight Idle	FLTIDLE		
		Ground Idle	GNDIDLE		
		Reverse Idle	REVIDLE		
		Reverse maximum	MAXREV		
		Rating defined by switchParm and parm_in	PARM		
Power option selector	switchParm	Net thrust	FN		
		Fuel flow	WF		
		Low rotor speed	NL		
		Intermediate rotor speed	NI		
		High rotor speed	NH		
		Fan Corrected speed	NCFAN		
		Low pressure compressor corrected speed	NCLPC		
		Intermediate pressure corrected speed	NCIPC		
		High Pressure compressor corrected speed	NCHPC		
		Engine pressure ratio	EPR		
		Actual Flow at station xxx	Wxxx		
		Corrected Flow at station xxx	WCxxx		
		Total pressure at station xxx	Pxxx		
		Static pressure at station xxx	PSxxx		
		Total temperature at station xxx	Txxx		
		Static temperature at station xxx	TSxxx		

TABLE 7 - RECOMMENDED SWITCH VARIABLE OPTION SETTINGS (CONTINUED)

Rating group selector	switchRating	Example options: BASE, GROWTH, 50KFN	As required		
Limit parameter group selector	switchLimit	Default group	NORMAL		
		No group defined	OFF		
		Additional example options: STUDY, REDLINE	As required		
Limit values group selector	switchLimitSet	Example options: MTO, MCL, FLTIDLE	As required		

5. PROGRAM CAPABILITIES

The architecture of 3.2 combined with the hook sockets mentioned throughout provide enhancements to the program capabilities described in AS681. Hook sockets in particular convey to the user extensive flexibility in evaluating what are commonly known as installation effects. Installation effects include customer bleed, customer power extraction, and aircraft inlet and exhaust losses, among others. The user has full control and ownership of the installation code.

This section recommends implementation methods for hook sockets. The recommendations are demonstrated by way of an example taken from NPSS; user-defined inlet pressure recovery.

5.1 Supplier Provision for Custom Hook Socket Installation Interface

The annotated example shown is in NPSS interpreted code syntax. This document does not require any particular approach to the coding; it requires only that the delivered software conform to the following:

a. Component instance names per Table 1 - Recommended Component Instance Names

b. Switch parameter names per Table 5 - Recommended Attribute (Parameter) Names

c. Demand parameter names per Table 5 - Recommended Attribute (Parameter) Names

d. Customer Hook Socket names per Table 7 – Recommended Switch Variable Option Settings

e. Hook socket is activated by setting the value of the switch parameter to “CUSTOM” per Table 7 - Recommended Switch Variable Option Settings

f. Hook socket interface type (if applicable to simulation system) should be “CUSTOM_HOOK”

The example presented is a flight inlet. It has been simplified for clarity and has only two of the option settings called out in Figure 3; “INPUT” and “CUSTOM”.

In the code fragments shown in Figure 3, any text following a double slash (//) is a comment. Object names and values explicitly required for compliance with this document are in **boldface**.

```
// Supplier's class code for object type Inlet
class Inlet extends Element {
  // Instantiate class variables and provide suitable default values
  real PqP=1;
  real PqP_in=1;
  real PqP_dmd=1;
  real WcRef=1000;
  // Instantiate option switch and provide allowed option values
  Option switchRec {
    allowedValues = { "INPUT", "CUSTOM" }
  }
  // Instantiate custom hook socket and define the variable(s)
  // the socket may set
  Socket S_customRec {
    allowedValues = { "PqP_dmd" }
    socketType = "CUSTOM_HOOK";
  }
  // Code for required calculations
  void calculate() {
    if ( switchRec == "INPUT" ) {
      PqP_dmd = PqP_in;
    }
    else { // switchRec == "CUSTOM"
      if( ! S_customRec.isEmpty() ) {
        S_customRec.execute(); // returns PqP_dmd
      } else {
        PqP_dmd = 1;
      }
    }
    PqP = PqP_dmd;
  }
}

// Part of supplier's model with instantiation of the Inlet element
Element Inlet InEng {}
```


FIGURE 3 - EXAMPLE OF SUPPLIER INLET CLASS CODE WITH HOOK SOCKET PROVISION

5.2 Customer Implementation of Custom Hook Socket Installation Interface

To utilize the custom hook socket capability shown in 5.1, the customer need only create a custom socket class, instantiate it into the indicated socket and supply any required data for the new functionality. Figure 4 provides an example of the user supplied code to calculate inlet recovery via a simple table lookup as a function of WcRef. Again, object names and values explicitly required for compliance with this document are in **boldface**. The user can instantiate new variables, create tables, or create much more complicated code than shown here.

It is common practice for the customer to provide the supplier with their installation calculations and data, or some subset of same, as this facilitates optimization of the installed vehicle. The supplied model should contain the latest customer provided installation code and data in a cleanly packaged form, this makes it easier for the customer to make updates as needed. As part of a delivered model, the supplier should include check cases demonstrating proper functionality of custom hook sockets available in the simulation.

```
// Customer's class code for inlet recovery
class CustomerRecovery extends Subelement {
  addInterface("CUSTOM_HOOK");
  void calculate() {
    // Evaluate inlet recovery table
    PqP_dmd = TB_PqP(WcRef)
  }
}

// Customer's instantiation of inlet recovery subelement into the
// socket of existing inlet element instance provided by supplier
InEng {
  Subelement CustomerRecovery S_customRec;
}

// Customer's inlet recovery table required by their code
InEng.S_customRec {
  Table TB_PqP (real Wc) {
    Wc    = { 1000, 2000 }
    PqP   = { 0.95, 0.95 }
  }
}
```

FIGURE 4 - EXAMPLE OF CUSTOMER INLET RECOVERY CLASS CODE USED WITH HOOK SOCKET

To select the custom hook logic the customer should simply set the appropriate effect switch to “CUSTOM” and run the simulation.

```
InEng.switchRec = "CUSTOM";
run();
```

6. PROGRAM STATUS INDICATION

AS681 establishes a four-digit NSI (Numerical Status Indicator) to convey the severity, category and purpose of an error or warning. The function of an NSI is to concisely display and store status information about each point run. This function is especially important when large numbers of cases are run.

ESI (Engineering Status Indicator) is an extension of the NSI concept, and replaces it. An ESI is an eight- or nine-digit integer: the first one or two digits indicate severity, the next three digits indicate purpose, the next two digits indicate category, and the final two digits indicate organizational ownership. The concept of ownership was added to avoid duplication of ESIs in code written by different organizations in collaborative models.

The ESI integer then has the form:

(S)SPPPCCOO

where:

(S)S = Severity
PPP = Purpose
CC = Category
OO = Owner

An example of an ESI is: 9 123 22 02. (White spaces added for clarity) This example would be a GE defined, kill point, invalid input error. The meaning of purpose 123 would be found in the model documentation.

ESI values shall be presented in an array, similar to the way NSI values are presented in AS4191 models. The items in the array should be sorted to place severity 99 or severity 9 errors in the first position in the array. The array should be named ESI, and should be instantiated at the top level of the model.

Specifically the following top-level scope variables should be provided by the supplier:

ESI[] – An integer array of ten or more elements to hold ESI values

ESI01, ESI02, ….ESI10 – Equivalent scalar integer variables holding at least the first 10 ESIs. Desired for databases and viewers.

ESIDescription[] – String array that holds the ESI values along with their descriptions; may or may not align with ESI integer array.

ESIseverityMax – Scalar integer variable that holds the most severe severity for a case.

warnStreamDisplay 1=on (default) 0= Off

errorStreamDisplay 1=on (default) 0= Off