DISCLAIMER

This book was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

USER MANUAL FOR IOSYM:
AN INPUT-ORIENTED SIMULATION LANGUAGE FOR CONTINUOUS SYSTEMS

Joseph Polito*
Geo-Energy Systems Analysis Division 4756
Sandia National Laboratories
Albuquerque, New Mexico 87185

ABSTRACT

IOYSM is an extension of the GASP IV simulation language. It permits systems which are sequences of continuous processes to be modeled graphically. Normally the system can be described by data input only. The language permits stochastic sequencing and termination criteria for processes and allows crossing conditions for ending operations that are more general than GASP IV. Extensive capability exists for conditional branching and logical modification of the network. IOSYM has been used to model the cost of geothermal drilling where the various costly processes of drilling are represented by IOSYM operations. The language is much more general, however, since it retains most of GASP IV's discrete event capabilities and permits easy modeling of continuous processes.

*Presently employed by Pritsker and Associates, Albuquerque, NM

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

ACKNOWLEDGMENT

The author is indebted to Charles C. Carson and Y. T. Lin of Division 4744, Sandia National Laboratories, and David M. Smith of Scientific Simulation, Inc. Their careful editing and aid in organizing and preparing figures for this document have substantially improved its final form.

In addition, Mr. Smith (at that time with the BDM Corporation) shared in the programming and program design tasks. Without his aid, this work would not have been completed.

CONTENTS

<u>Section</u>		<u>Page</u>
1	INTRODUCTION	9
	1.1 A Description of IOSYM	9
	1.2 Relationship of IOSYM to GASP IV	10
2	NETWORK MODELS AND SYMBOLS	13
	2.1 Two Examples	13
	2.2 Network Models of Examples 1 and 2	24
3	THE IOSYM NETWORK LANGUAGE	27
	3.1 How IOSYM Works	27
	3.2 IOSYM Input Conventions	28
	3.3 Networks, Variables, and Tables	29
	3.4 Nodes	33
	3.5 Executable Commands	34
	3.6 Operations	46
	3.7 Arcs	49
	3.8 Global	56
	3.9 Library, Menu, and Probability Functions	58
	3.10 User-Written FORTRAN Programs	62
	3.11 Utility Subprograms for Use in User Programs	67
	3.12 Reserved Names and Conventions in IOSYM	76
4	IOSYM STATISTICS	77
	4.1 Tabulations	77
	4.2 Report Statistics	80
	4.3 Collect Statistics	80
	4.4 IOSYM Plots	81
	4.5 GASP IV Statistics	83
5	DATA INPUT AND RUNNING IOSYM	87
	5.1 Creating a Network	87
	5.2 Simulating the Network	89
	5.3 Making Changes to a Network	94
	5.4 The SAVE and RESTORE Commands	96
	5.5 GASP Input	98
6	IOSYM OUTPUT	101
	6.1 Network Summary	101

CONTENTS (Continued)

<u>Section</u>	<u>Page</u>
6.2 Tabulations	106
6.3 Plots	106
6.4 REPORT Statistics	113
6.5 COLLECT Statistics	113
6.6 Multiple Runs and Sensitivity Analysis	113
REFERENCES	118
APPENDICES	
A IOSYM Commands, Syntax, and Network Symbolism	A-1
B IOSYM Networks and Data Input for Examples 1 and 2	B-1
C Advanced GASP IV Applications	C-1
D Library, Probability, and Menu Functions	D-1
E Glossary of IOSYM Terms	E-1
F IOSYM Error Processing and Error Numbers	F-1
G IOSYM COMMON With Selected Explanations	G-1

ILLUSTRATIONS

<u>Figure</u>		
1	Coordinate System for Example 1	14
2	Geometry for EJECT	15
3	Geometry for FREEFLY	17
4	Geometry for Pilot Striking Tail of Airplane	18
5	Geometry for SNAP	19
6	Coordinate System and Variables for Example 2	22
7	Schematic Network for Example 1	24
8	Schematic Network for Example 2	25
9	Example of Abbreviated PICK Commands	51
10	The Difference Between TIMST and TIMSA Integration	85
11	Concise Summary for Example 1	103
12	Node Summary for Example 1	103
13	Arc Summary for Example 2	103
14	Operation Summary for Example 1	104
15	Variable Summary for Example 1	104

ILLUSTRATIONS (Continued)

<u>Figure</u>		<u>Page</u>
16	Table Summary for Example 1	105
17	Statistics Summary for Example 1	105
18	Plot Summary for Example 1	107
19	Tabulation for Example 1	107
20	First Page of Tabulation for Example 2	107
21	Plot 1 from Example 1 -- Line Printer Plot	108
22	Plot 3 from Example 2 -- Line Printer Plot	108
23	Plot 1 -- Example 1	109
24	Plot 2 -- Example 1	109
25	Plot 3 -- Example 1	110
26	Plot 4 -- Example 1	110
27	Plot 5 -- Example 1	111
28	Plot 1 -- Example 2	111
29	Plot 3 -- Example 2	112
30	Plot 4 -- Example 2	112
31	First Page of GASP IV Plot for Example 2	114
32	REPORT Statistics for Example 2	114
33	COLLECT Statistics -- Part 1, Example 2	115
34	COLLECT Statistics -- Part 2, Example 2	115
35	IOSYM Message at Start of a Simulation	117
36	Plot of Cost Per Foot versus ROPMULT	117
A-1	IOSYM Network Symbolism	A-3
B-1	IOSYM Network for Example 1	B-1
B-2	Data Cards for Example 1	B-4
B-3	FUNCTION U for Example 1	B-7
B-4	SUBROUTINE UEVENT for Example 1	B-7
B-5	IOSYM Network for Example 2	B-7
B-6	Data Cards for Example 2	B-8
B-7	FUNCTION U for Example 2	B-12
B-8	SUBROUTINE UEVENT for Example 2	B-12
B-9	SUBROUTINE USSAVE for Example 2	B-12
G-1	IOSYM COMMON Blocks	G-2

TABLES

<u>Table</u>		<u>Page</u>
1	Variables for Example 1	16
2	Variables for Example 2	24
D-1	Parameter Set Values	D-3
F-1	IOSYM Error Numbers	F-2

USER MANUAL FOR IOSYM:
AN INPUT-ORIENTED SIMULATION LANGUAGE FOR CONTINUOUS SYSTEMS

1. INTRODUCTION

Complete documentation for the IOSYM language is contained in this manual, which is intended primarily for IOSYM users, and in Reference 1, which contains a detailed description of the computer program.

1.1 A DESCRIPTION OF IOSYM

IOSYM is a simulation language for modeling complex systems of different continuous processes characterized by their being performed in sequence. Systems that can be represented in this fashion include

1. Drilling of deep boreholes into the earth -- in this system, the operations of drilling, tripping the bit, surveying the borehole, etc., are performed in a sequence,
2. Numerically controlled manufacturing -- in this system, the sequential operations consist of positioning the work piece, tool change, tool selection (such as on a turret lathe), rapid traverse, and cutting, and
3. Preparing a missile for launch -- this system is characterized by many operations that must be performed in a sequence prescribed by a strict precedence relationship in which delays and resequencing occur when contingencies arise.

All of the above systems contain several operations which can be described by differential equations. Also, the criteria for ending one operation and beginning the next depend on which operation is being

performed and, perhaps, on what operations have already been accomplished. Indeed, there may be more than one criterion for termination. For example, a machine tool cutting operation may terminate because the cut is complete, the tool has worn out, or the tool has failed.

IOSYM permits the description of the operations in a system, their selection and termination criteria, and the precedence relations among them to be specified through data input. In most cases, the user is not required to write a computer program. A graph is constructed by the user with arcs and nodes that represent the operations and decision points in the system. This network is translated into data cards with additional information to specify selection and termination rules. IOSYM reads the data cards and simulates the system. The following sections describe the network modeling philosophy and the various data input commands.

1.2 RELATIONSHIP OF IOSYM TO GASP IV

IOSYM is an extension of the GASP IV simulation language.² GASP IV will model combined continuous/discrete systems. It is a FORTRAN program and requires that the user write FORTRAN subprograms to describe the operations and events for the system under study. The programming can become very complex when many continuous operations must be represented. IOSYM relieves the user of the need to write FORTRAN and provides a convenient framework to specify the characteristics of the many continuous operations. The user must still write subprograms to describe events that occur at discrete intervals.

IOSYM includes most of the capabilities of GASP IV but is not a replacement for it. If the system being represented has no continuous aspect or if only a few continuous operations are performed, it may be more convenient to use GASP. When complex systems like those described above are encountered, however, IOSYM will provide a considerable savings of analyst effort.

The user need not have intimate knowledge of GASP IV in order to use IOSYM's capabilities for modeling continuous systems. A working

knowledge of some of GASP's input variables is sufficient, and this is provided in this document where necessary. Combined continuous/discrete simulations with extensive discrete event aspects will, of course, require a working familiarity with GASP IV.

.

.

.

.

2. NETWORK MODELS AND SYMBOLS

The philosophy of modeling continuous systems as networks is presented in this section. To make the discussion of this and subsequent sections concrete, two examples are introduced in subsection 2.1. These examples will be used to demonstrate IOSYM modeling, data preparation, and output. Subsection 2.2 will introduce the network models of Examples 1 and 2.

2.1 TWO EXAMPLES

2.1.1 Example 1--Pilot Ejection

The first example considers a pilot ejecting from an airplane. The action takes place in six phases which will be modeled as IOSYM operations. The phases are listed below.

EJECT	During this phase, the pilot and seat move up the rails inside the aircraft; the pilot is being ejected from the aircraft.
FREEFLY	After leaving the airplane, the pilot and seat are in free flight together.
SNAP	After a short period of free flight, a strap under the pilot is automatically reeled to a taut position, and the pilot's seat belt is opened. The result is a rapid separation of the pilot and seat.
FREEFAL	The pilot free falls to 14,000 feet.
OPEN	When the pilot reaches 14,000 feet (above sea level), his parachute automatically opens.
CHUTE	The pilot descends to the ground with the parachute open.

The coordinate system for this problem is shown in Figure 1. The pilot's initial position is at $(x,y) = (0,H)$ where H is the initial altitude. The functions which describe the motion of the pilot are $x(t)$, $y(t)$, $V(t)$, and $\theta(t)$. Equations of motion for each phase are presented below. The variable t is time, and ejection begins at time zero. The notation x_i , y_i , etc. denotes the value of the variable at the start of an operation. A dot over a variable indicates the time derivative, e.g., $\dot{x} = dx(t)/dt$.

Figure 1. Coordinate System for Example 1

The variables in Figure 1 are defined as follows:

(x,y) = horizontal and vertical distance

V = pilot velocity

θ = inclination of velocity measured from the horizon

H = initial altitude of the airplane

The dynamics of EJECT are shown in Figure 2. The aircraft has initial velocity V_a at flight angle θ_a . The pilot and seat move up the

Figure 2. Geometry for EJECT

rail at constant velocity V_r , and the rail is aligned at an angle θ_r with the vertical axis of the aircraft. EJECT ends when the seat reaches the end of the rail and leaves the airplane. The rail has length h_r , so EJECT ends at time h_r/V_r . Table 1 lists all of the variables in this problem. The variables are defined after the figures which describe the example and in Appendix B.

The time derivatives for the variables can be obtained as follows:

$$\begin{aligned}\dot{x}(t) &= V_a \cos \theta_a - V_r \sin(\theta_a + \theta_r) \\ \dot{y}(t) &= V_a \sin \theta_a + V_r \cos(\theta_a + \theta_r) \\ \dot{V}(t) &= 0 \\ \dot{\theta}(t) &= 0\end{aligned}$$

Also, $x_i = 0$, $y_i = H$, and

$$\begin{aligned}V_i &= \sqrt{(\dot{x}_i)^2 + (\dot{y}_i)^2} \\ \theta_i &\begin{cases} = \tan^{-1}(\dot{y}_i/\dot{x}_i) & \text{if } \dot{x}_i > 0, \dot{y}_i \geq 0 \\ = \pi/2 & \text{if } \dot{x}_i = 0, \dot{y}_i \geq 0 \end{cases}\end{aligned}$$

$$\dot{x}_i = 0, \dot{y}_i \geq 0$$

Table 1
Variables for Example 1

<u>Variable</u>	<u>IOSYM Name</u>	<u>Initial Value</u>
θ_a	THETAA	-5
V_a	VA	400 ft/s
θ_r	THETAR	15
V_r	VR	45 ft/s
h_r	HR	4 ft
m_p	MP	5 slugs
m_s	MS	2.5 slugs
H	H	22,000
LT	LT	35 ft
HT	HT	9 ft
S_{sp}	SSP	12 ft ²
S_p	SP	8 ft
S_c	SC	315 ft
C_{Dsp}	CDSP	1
C_{Dp}	CDP	1
C_{Dc}	CDC	1
K_{snap}	KSNAP	200
Δ_{max}	DELMAX	1 ft
Δ	DELTA	0
TSEAT	TSEAT	1 s
TCHUTE	TCHUTE	0.5 s
ρ	RHO	Table function
x	X	0
y	Y	H
v	V	-
θ	THETA	-
x_t	XT	-
y_t	YT	-
x_p	XP	-
y_p	YP	-
F_{snap}	FSNAP	-

The FREEFLY phase is depicted in Figure 3. The forces on the pilot and seat are gravity and aerodynamic drag. One of two events will end the FREEFLY phase. The pilot and seat will automatically separate (the SNAP) after a time interval, here designated TSEAT. FREEFLY will also end if the pilot and seat strike the tail of the aircraft.

Figure 3. Geometry for FREEFLY

The variables and motion equations for the FREEFLY phase (as illustrated in Figure 3) are defined as follows:

subscript sp = seat and pilot

D_{sp} = aerodynamic drag

m_p = pilot mass

m_s = seat mass

g = acceleration of gravity

ρ = air density

C_{Dsp} = drag coefficient

S_{sp} = surface area

$\dot{x}(t) = V \cos \theta$

$\dot{y}(t) = V \sin \theta$

$\dot{V}(t) = -D_{sp}/(m_p + m_s) - g \sin \theta$

$\dot{\theta}(t) = -g \cos \theta / V$

$D_{sp} = 1/2 C_{Dsp} \rho V^2 S_{sp}$

If the airplane continues its original flight path after EJECT, then the geometry of FREEFLY is as shown in Figure 4. The tail is a distance LT behind the cockpit and is at height HT. The tip of the tail has coordinates (x_t, y_t) , and the base has coordinates (x_p, y_p) . For this example, the pilot will hit the tail if he passes through the line joining the tip and base of the tail before seat separation occurs.

Figure 4. Geometry for Pilot Striking Tail of Airplane

At time zero,

$$x_t(0) = -LT \cos \theta_a - HT \sin \theta_a$$

$$y_t(0) = y(0) - LT \sin \theta_a + HT \cos \theta_a$$

and during EJECT and FREEFLY,

$$\dot{x}_t = V_a \cos \theta_a$$

$$\dot{y}_t = V_a \sin \theta_a$$

The position of the base of the tail can be computed from

$$x_p(t) = x_t(t) + HT \sin \theta_a$$

$$y_p(t) = y_t(t) - HT \cos \theta_a$$

The pilot passes the line connecting the tip and base of the tail when

$$y - \left(\frac{y_t - y_p}{x_t - x_p} \right) x \leq y_t - \left(\frac{y_t - y_p}{x_t - x_p} \right) x_t \quad \text{for } \theta_a \neq 0$$

or

$$x \leq x_t$$

$$\text{for } \theta_a = 0$$

The pilot strikes the tail if, when the above condition is satisfied, the following conditions also occur:

$$[(x - x_t)^2 + (y - y_t)^2]^{1/2} \leq HT$$

and

$$[(x - x_p)^2 + (y - y_p)^2]^{1/2} \leq HT$$

The SNAP phase is modeled as though a spring separates the pilot and seat. Several simplifying assumptions have been made (see Figure 5) as follows:

1. The pilot and seat do not rotate during FREEFLY.
2. The separation force, F_{snap} , is perpendicular to the pilot-seat orientation.
3. Aerodynamic forces do not affect the separation velocity.
4. Drag is the same as during FREEFLY, i.e., the drag coefficient and area are the same.

Figure 5. Geometry for SNAP

Let Δ be the distance between the pilot and seat during SNAP, and let the separation force be given by

$$F_{\text{snap}}(\Delta) = K_{\text{snap}}(\Delta_{\text{max}} - \Delta) \quad 0 \leq \Delta \leq \Delta_{\text{max}}$$

where K_{snap} is a "spring constant" and Δ_{max} is the maximum extension of the "spring." In other words, the pilot and seat are separated when the distance between them reaches Δ_{max} . The relative velocity between the pilot and seat is denoted by V_{Δ} . The SNAP phase is over when the pilot and seat have separated.

The equations of motion for SNAP are as follows:

$$\begin{aligned} \dot{x}(t) &= V \cos \theta \\ \dot{y}(t) &= V \sin \theta \\ \dot{V}(t) &= -D_{\text{sp}}/(m_p + m_s) - g \sin \theta + F_{\text{snap}}(\Delta) \cos(\theta_r + \theta_a - \theta) \\ \dot{\theta}(t) &= -g \cos \theta / V + F_{\text{snap}}(\Delta) \sin(\theta_r + \theta_a - \theta) / V \\ \dot{\Delta}(t) &= V_{\Delta} \\ \dot{V}_{\Delta}(t) &= F_{\text{snap}}(\Delta) (1/m_s + 1/m_p) \\ \Delta_i &= 0 \quad V_{\Delta_i} = 0 \end{aligned}$$

The FREEFAL phase is similar to FREEFLY except that the mass and area of the seat are absent. The equations of motion are

$$\begin{aligned} \dot{x}(t) &= V \cos \theta \\ \dot{y}(t) &= V \sin \theta \\ \dot{V}(t) &= -D_p/m_p - g \sin \theta \\ \dot{\theta}(t) &= -g \cos \theta / V \end{aligned}$$

FREEFAL terminates when $y \leq 14,000$. If the initial altitude is less than 14,000 feet, the FREEFAL phase does not occur.

The opening of the parachute is represented by the operation OPEN. A simple representation is used in which the equations of motion are the

same as for FREEFAL except that the drag term, D_p , is replaced by $1.2 D_p$. OPEN terminates after the time interval TCHUTE.

The flight to the ground with the parachute open, CHUTE, is similarly simplified. The same equations of motion are used as for FREEFAL except the drag coefficient and area are for the parachute. It is assumed that the centers of pressure and mass are the same so the pilot does not oscillate pendulum fashion beneath the chute.

2.1.2 Example 2--Deep Drilling

The economics of deep drilling for oil, natural gas, and other mineral resources are often examined with one-equation cost model that takes into account rate of penetration, bit life, depth interval, bit cost, drilling rig cost, and tripping rate. Tripping rate is the rate at which the drill bit is withdrawn from the well when a bit change is needed. The rate of penetration and bit life are inputs to this model. A simple model that computes penetration rate and bit life as functions of control variables and wellbore parameters is constructed here. Five operations are modeled:

DRILL	The process of deepening the well.
TRIPOUT	Withdrawing the drill pipe and bit from the well.
CHGBIT	The time during which a new bit is installed on the end of the drill pipe.
TRIPIN	Lowering the new bit to the bottom of the borehole.
SURVEY	A period during which drilling is stopped and a directional instrument lowered down the drill pipe to determine the location of the bottom of the hole. No trip takes place.

The coordinate system for this example is shown in Figure 6. Of interest is the cost to drill the interval from Z_1 to Z_2 . Empirical equations for penetration rate, bit tooth wear, and bit bearing wear³ are presented in the following pages.

Figure 6. Coordinate System and Variables for Example 2

The rate of penetration (ROP) equation is

$$ROP = f_1 f_2 f_3 f_4 f_5 f_6 f_7 f_8$$

where

$$f_1 = a_1$$

$$f_2 = \exp[2.303 a_2 (10,000 - D)]$$

$$f_3 = \exp[2.303 a_3 D^{.69} (a_9 - 9)]$$

$$f_4 = \exp[2.303 a_4 D (a_9 - a_{10})]$$

$$f_5 = \left[\frac{w/d - a_{12}}{4 - a_{12}} \right]^{a_5}$$

$$f_6 = (N/100)^{a_6}$$

$$f_7 = \exp[-a_7 h]$$

$$f_8 = (a_{11}/1000)^{a_8}$$

and where $D = |\text{DEPTH}|$, w is the weight on the bit divided by 1000, d is the bit diameter, N is the rpm, h is the fraction of bit tooth wear, and a_1 through a_{12} are constants and other borehole parameters.

The equation for rate of tooth wear is

$$dh/dt = (H_3/H_5) (N/100) H_1 [(H_4 - 4)/(H_4 - w/d)] (1 + H_2/2)/(1 + H_2 h)$$

where H_1 through H_5 are constant borehole parameters. The rate of bit bearing wear is given by

$$dB/dt = (N/100) [(w/4d)^{b_1}]/b_2$$

where B is the fraction of bearing life that has been used, and b_1 and b_2 are constants relating to the bearings. Bit life is the minimum of tooth life and bearing life.

Variables for Example 2 are shown in Table 2. The differential equations describing the dynamics are given below. A dot above a variable indicates the time derivative.

$$\dot{COST}(t) = \text{rig rental rate}$$

Let b_p be the position of the bit. Then for operation DRILL

$$\dot{D}(t) = -ROP$$

$$\dot{b}_p(t) = \dot{D}$$

$$\dot{h}(t) = dh/dt$$

$$\dot{B}(t) = dB/dt$$

DRILL ends when the teeth or bearings are worn, a survey depth is reached, or the end depth of the interval is reached.

For TRIPOUT, let TRIPRT be the tripping rate in hours per thousand feet. Then

$$\dot{D}(t) = \dot{h}(t) = \dot{B}(t) = 0$$

$$\dot{b}_p(t) = 1000/TRIPRT$$

TRIPOUT ends when the bit reaches the surface.

Table 2
Variables for Example 2

<u>Variables</u>	<u>IOSYM Name</u>	<u>Initial Value</u>
COST	COST	0
D	DEPTH	Z1
b_p	BIT	Z1
h	TOOTH	0
B	BEARING	0
Z_1	Z1	3000
Z_2	Z2	-8000
N	RPM	60
w	WOB	60
d	BITDIA	12.25
Rig rental	RIGRATE	250
TRIPRT	TRIPRT	0.5
-	SURVINT	500
-	BITCOST	5200

For TRIPIN,

$$\dot{D}(t) = \dot{h}(t) = \dot{B}(t) = 0$$

$$\dot{b}_p(t) = -1000/\text{TRIPRT}$$

TRIPIN ends when the bit reaches the bottom of the hole.

For CHGBIT and SURVEY, all derivatives except cost equal zero. These operations end when a specified time interval is completed.

2.2 NETWORK MODELS OF EXAMPLES ONE AND TWO

This section shows how the example problems can be visualized as networks. IOSYM symbolism will be introduced, but a complete description must wait until additional concepts are discussed in Section 3.

The pilot ejection model is a simple sequence of events that can be represented by the network in Figure 7. The lines joining the numbered boxes are "arcs" and represent the phases (or operations) of the ejection. The numbered boxes are "nodes" and represent the epochs at which one operation terminates and another begins. The arrowheads from node 3 to node 7 represent an alternate routing from node 3. If FREEFLY ends with the pilot striking the airplane, then the simulation terminates at node 7. Similarly, the dotted line from node 4 to node 5 represents a transfer directly from SNAP to OPEN. No FREEFAL portion takes place. Such a circumstance would occur if the ejection took place below 14,000 feet. The arrowheads and dashed lines represent alternate ways to cause transfers to take place and will be discussed further in Section 3. For now it is sufficient to realize that the network represents the sequence in which operations occur and that, by appropriate input, the initial conditions, equations of motion, and termination criteria for each operation can be specified as shown in Figure 7.

Figure 7. Schematic Network for Example 1

The computer simulation begins at node 1. The arc leaving node 1 is selected, and the equations of motion are integrated until the termination criteria are satisfied. Then at node 2 the next arc is selected, in this case FREEFLY. The process continues until node 7 or 8 is reached, where the simulation ends.

The network for Example 2 is shown in Figure 8. This network is substantially different than the network for Example 1 in that the arcs loop or form "cycles" in the network. This is because, in drilling the interval from Z_1 to Z_2 , the same operations may be performed more than once. For example, when a bit is worn, DRILL stops and TRIPOUT begins. Then a new bit is installed during CHGBIT. Next the bit is inserted into the well during TRIPIN and DRILL resumes. Similarly, at survey depths DRILL stops while the survey is performed and resumes afterward. When Z_2 is reached, a transfer is made from node 3 to node 4 and the simulation ends.

Figure 8. Schematic Network for Example 2

The operation DRILL may end in one of four ways: (1) the bit teeth are worn, (2) the bit bearings are worn, (3) a survey depth is reached, or (4) the final depth, Z_2 , is reached. This is represented by the four branches of the arc DRILL. At node 2, IOSYM determines which of the arcs leaving the node should be selected.

Implementation of these examples with IOSYM requires a knowledge of the language's procedures and syntax. The following sections discuss the IOSYM language in detail and use the two examples above to demonstrate its features.

3. THE IOSYM NETWORK LANGUAGE

All of the elements and most of the commands for IOSYM networks are described in this section. A complete list of the data input for the examples is contained in Appendix B. Frequent reference will be made to this information.

3.1 HOW IOSYM WORKS

The user prepares data that describe the network. These data are in several parts. First the variables and arrays are defined. Entities can be assigned names much as in FORTRAN. When appropriately defined, IOSYM will recognize these variable names whenever they occur in the data input. One-dimensional arrays are also permitted. They are called "Tables" in IOSYM terminology. Variables and tables may be assigned initial values during the initial phase of input. Tables 1 and 2 are partial lists of the IOSYM variables for the examples. Their IOSYM name and initial values are also given.

Next, a section of input specifies what statistics are to be maintained and printed. Thirdly, the operations are described. Each operation is assigned a name and the equations of motion typed on the data cards. IOSYM interprets these equations and executes them properly.

The nodes and arcs are described next. Arcs are the most complex structures in IOSYM. Three pieces of information are required on each arc: (1) the criteria for selecting the arc, (2) the operation to be performed, and (3) the criteria for ending the operation.

The IOSYM program will simulate the system described in the input and print the appropriate statistics. The following sections explain the various commands and syntax for preparing network input.

3.2 IOSYM INPUT CONVENTIONS

All data are input to IOSYM without regard to column spacing. Data fields are separated by blanks, commas, slashes, or parentheses depending upon the syntax of the particular command. In general, commas and blanks are interchangeable as separators. So,

```
VAR, X, 1, 10
```

```
VAR X 1 10
```

are interpreted exactly the same.

Integer variables in the FORTRAN sense do not exist in IOSYM. All values and variables are "real." Numeric information can be input in any of the following forms:

```
100      100.      1.E2      1E2
```

Data cards are complete unless the last interpreted character is a comma or operation (+, -, /, *, **) or the first character on the next card is "*". In this case, the next card is considered a continuation. If the first character is "*", that character is ignored. A dollar sign (\$) encountered on a card causes interpretation to stop on that card, so anything that follows the "\$" will be treated as a comment. The VAR card previously shown could be written like this:

```
VAR X 1,    $ EXAMPLE OF COMMENTS
```

```
10          $ AND CONTINUATION CARDS
```

Only the first 80 columns of a data card are scanned by IOSYM. However, 90 columns are read, and 90 columns will be echoed.

Names are assigned by the user to variables, nodes, arcs, etc. These names are alphanumeric strings of seven characters or less. Certain names should be avoided because they conflict with IOSYM commands. Reserved names are discussed in subsection 3.12.

3.3 NETWORKS, VARIABLES, AND TABLES

The first command in every network is NETWORK. The syntax is shown below.

```
NETWORK, network name
```

When IOSYM encounters NETWORK, it clears its internal array storage and begins to read data cards that describe the new system. The network name is used to identify the information which, taken together, is the network.

IOSYM reads data cards as part of the network until it encounters an END card as follows:

```
NETWORK, EJECTN
```

```
    other network data cards
```

```
END
```

The END statement is used frequently to signify the end of a block of related data cards. IOSYM automatically keeps track of the "level" or depth of nested information that requires an END command. Any characters that follow END (so long as a separator follows the D) are treated as a comment, so it is often helpful to include a few characters to identify the entity that the END refers to. For example, a network can be schematically represented by

```
NETWORK EJECTN
```

```
    .
```

```
    .
```

```
    .
```

```
    NODE
```

```
        END NODE
```

```
    ARC
```

```
        END ARC
```

```
    .
```

```
    .
```

```
    .
```

```
END NETWORK
```

The comments on the END commands improve the readability of the data.

A command CLEAR is available which clears the internal storage arrays but does not cause a new network to be read. The syntax is

CLEAR.

The NETWORK command first performs a CLEAR and then reads a network.

GASP IV has a format-free input scheme of its own that permits direct setting of GASP variables. It is sometimes convenient to be able to effect GASP variables through data input. To facilitate this, IOSYM will accept the command GASP immediately after NETWORK. This will cause control to be passed to the GASP input subprograms and GASP-formatted data cards will be read until a FIN* card is encountered. Then reading of IOSYM data will resume. Schematically, the input stream is

```

NETWORK EJECTN
GASP
•      }
•      }   GASP-formatted data cards
•      }
FIN*   }
•      }
        }
•      }   additional IOSYM data
•      }

```

The GASP command and input are optional but, if present, must immediately follow NETWORK. GASP input is discussed in subsection 5.5 and Appendix C.

Variables and tables may be defined immediately after NETWORK (or GASP input, if present). Simple variables are defined with the VARIABLE command. The syntax is

VARIABLE, name, type, initial value

"VARIABLE" may be abbreviated by "VAR". The "name" is an IOSYM name which will thereafter identify the variable.

There are three types of variables. Type "0" is the independent variable for integration. Type "1" variables are "primary" variables

and are defined by differential equations. Type "2" variables are "secondary" variables which include all other simple variables.

The independent variable exists by default and has the name TIME and initial value zero. Hence, it is not necessary to have a VAR card to define it if the name and initial value are satisfactory. If not, then a VAR card can be included to change the characteristic. Suppose the independent variable is to be named DIST, with an initial value of 100. Then the following command could be used:

```
VAR DIST    0    100
```

Primary variables are treated in a special way by IOSYM, so every primary variable must be defined on a VAR card. IOSYM, through GASP, numerically integrates the differential equations which describe the primary variables. Therefore, the principal means of defining primary variables is by assigning values to their derivatives. Once a primary variable has been defined on a VAR card (e.g., VAR X 1 5), IOSYM will recognize the variable name preceded by a "D" (e.g., DX) as its derivative. No VAR card is needed for "DX" and none is permitted. Derivative variables are the only names (other than command names such as VARIABLE) in IOSYM that may have eight characters (seven for the primary variable name plus "D").

Eight secondary variables are automatically provided. The first six, I, J, K, L, M, and N, are primarily for use with tables and will be discussed subsequently. Their initial values are zero, but this can be changed by using a VAR card (e.g., VAR I 2 3. will set I to 3 initially). The variables EEE and PII are provided and have initial values e and π , respectively. The variable OPTIME has initial value zero, but it is unique in that its value is automatically changed by IOSYM during the simulation to give the TIME interval spent on each operation. OPTIME will be discussed more fully in subsection 3.6.

Additional secondary variables may be defined as needed. However, only variables that require nonzero initial values need be defined. If IOSYM encounters undefined variable names in the subsequent part of the

network, it automatically makes them secondary variables and assigns an initial value of zero. Finally, if the initial value is omitted from any VAR card, IOSYM will assign zero, but a warning message will be printed.

Tables are created with the TABLE command whose syntax is as follows:

```
TABLE name(length) initial values
```

Tables are one-dimensional arrays, and the initial values can be assigned in the same way that information is assigned by FORTRAN DATA statements. For example,

```
TABLE A(10) 3*6.3 15.2 18 26 4 * 0
```

If initial values are omitted on the TABLE card, zeros will be assigned and a warning printed. Every user-specified table must be defined by a TABLE command. Any locations not defined with initial values will be set to zero. Hence, the "4 * 0" at the end of the example above is not needed.

For convenience, four tables are provided automatically. Their names and default lengths are

```
GV(50) NV(25) AV(50) MV(10)
```

The default initial values are all zero. The lengths and initial values can be changed by a TABLE command; e.g.

```
TABLE GV(25) 25*-1.E10
```

The command will change the length of GV to 25 and initialize all locations to -1.E10.

The GV table is a global table in that it is treated by IOSYM just like user-created tables. Values stored in GV remain until changed by the user. The NV and AV tables are special, however. IOSYM will zero out the NV table as the first action in processing a node. Therefore, any nonzero values stored in NV are destroyed when a node is reached. The AV table is set to zero as the first action in processing a new arc.

Therefore, values stored in AV are available only on the arc during which they were set.

The MV table is for use with the MINIMIZE command which has not been implemented in this version of IOSYM. MV can be used in the same way as GV in this version.

The subscript or "index" used to reference a table element can be a number or a simple variable. Only the variables I, J, K, L, M, and N may be used. Legal table references are

GV(3) AV(6) GV(I) NV(N)

Table indexes may not be expressions or variables other than the six listed above.

A special provision is made to determine the length of a table. If a variable index whose value is zero is used, the value returned is the length of the table. In a sense, the "zeroth" element is equal to the length of the table. As an example, consider

I = 0

J = GV(I)

J will have the value 50 after the above statements. The statement J = GV(0) is not permitted and will produce a fatal error. Also, index values less than zero or greater than the table length will produce errors.

3.4 NODES

Nodes are represented by the symbol

On input, the node command has syntax

NODE, name

It is important to remember that "name" is an alphanumeric identifier.

The command

NODE 3

does not refer to the "third" node, but to the node named "3".

Nodes may have executable commands associated with them, so the complete block of data describing a node terminates with an END command as follows:

```

NODE ONE
    } other commands
END

NODE TWO
    } other commands
END
```

If other commands are not present at the node, then the NODE and END commands are not needed. IOSYM will recognize that a node is needed and create it automatically. (If fatal input errors occur and some nodes are not explicitly defined with NODE commands, the network summary that is printed may not correctly reflect those nodes and the arcs emanating from them.)

3.5 EXECUTABLE COMMANDS

IOSYM includes several commands that are executable in the FORTRAN sense. In other words, they are processed sequentially as they are encountered. For example, the "other commands" at nodes must be executable commands. They are processed when the node is encountered during a simulation.

SET is the IOSYM analog of a FORTRAN assignment statement. In its simplest form it involves only a variable, an equal sign, and an expression. Examples of simple set commands are

```
GV(J) = SIN(THETA)
```

```
DX = VA * COS(THETA) - VR * SIN(THETAA + THETAR)
```

AV(3) = HR/VA

V = SQRT(AV(1)**2 + AV(2)**2)

Simple SET commands are identical in most respects to FORTRAN assignments statements. A number of specialized functions which include a subset of FORTRAN library functions (SIN, COS, ABS, etc.) are available. These will be discussed in subsection 3.9 and Appendix D. SET commands have another feature, however, that is revealed by the general syntax:

SET, variable name(truth condition) = expression

The user may, optionally, include a logical or "truth condition" on the SET command. Allowable truth conditions are

LT -- Less than zero

LE -- Less than or equal to zero

EQ -- Equal to zero

GE -- Greater than or equal to zero

GT -- Greater than zero.

If no truth condition is specified, then the value of the variable named on the left of the equal sign is set equal to the value of the expression. If a truth condition is specified, then the value of the expression is acted upon by the condition. If the resulting logical expression is true, then 1.0 is stored in the variable. If the logical expression is false, then zero is stored in the variable. For example, consider

SET, X(LE) = 1

SET, GV(J)(EQ) = SIN(X)

The above sequence would result in $X = 0$ (because 1.LE.0 is false) and $GV(J) = 1$ (because $SIN(0).EQ.0$ is true). The characters "SET" may appear on any SET command, but they are optional. The equal sign signifies a SET command. Finally, the truth condition need not be enclosed in parentheses; any separator (comma or blank) will do. In particular, the following are equivalent:

```
SET, X(LE) = 1
SET, X, LE, = 1
SET, X LE = 1
```

The GOTO command permits transfers to take place that do not follow normal arc-node sequences. The GOTO command has two syntaxes:

```
GOTO, N, node name
GOTO, A, arc name
```

As can be inferred from the above syntaxes, transfer can be made to an arc or a node. GOTO is not like the FORTRAN GO TO statement in that it does not cause an immediate transfer. The meaning of GOTO is as follows: when the next transfer to an arc or node is to be made, go to the entity specified on the GOTO command.

Transfers can be made only at the end of processing an arc and at the end of processing a node. For example, recall that node processing consists of executing the commands, if any, at the node and then selecting the proper outgoing arc. Suppose node QY consists of the following:

```
NODE QY
  .
  .   } executable commands
  .
GOTO A ONE
  .
  .   } executable commands
  .
END
```

Control would not transfer to arc ONE immediately when the GOTO command was encountered. The arc selection process after all of the commands had been executed would be overridden, however, and arc ONE would be selected. Note that arc ONE need not emanate from node QY. GOTO is general; any node or arc in the network may be specified. Indeed, a user may transfer from an arc directly to another arc and from a node to another node. For example, if the GOTO command above were GOTO N PP,

all of the commands at node QY would be executed, control would then go directly to node PP. All of the commands at PP would be executed and then an arc would be selected at node PP (assuming another GOTO command was not executed at node PP).

Clearly, when a transfer time occurs and a GOTO is in effect, only one transfer can be made. It is possible to execute multiple GOTO's, and therefore, a hierarchy exists to select which GOTO to use. First, the last executed GOTO within type (N or A) has precedence. So the sequence

```
GOTO N 6  
GOTO N 3  
GOTO N 4
```

is equivalent to GOTO N 4. The GOTO A command is similar.

The procedure between types N and A GOTO commands depends on whether they are on a node or an arc. At a node, if both a GOTO N and a GOTO A command have been executed, the GOTO N command has precedence. Arcs have not been discussed in detail yet, but it is possible to specify executable commands on arcs as well. If both types of GOTOs have been specified on an arc, then the GOTO A type has precedence. In all cases, if only one GOTO has been executed, it will override normal sequencing and cause a transfer to the specified arc or node.

Also, GOTO commands have no memory through a transfer. So, for example, if there is a type A and a type N GOTO at a node, the type N will take precedence. When the transfer is made to the specified node, the type A command will be forgotten, and normal logic will be followed in sequencing from the new node.

The GOTO command affects a single transfer and then is forgotten. It is a "temporary" network modification. An executable command is available that causes a permanent modification of the network.

The REPLACE command replaces one node with another node. The syntax is

REPLACE, node out, node in

where "node out" and "node in" are node names. For example,

REPLACE OLD NEW

will cause all transfers to node OLD to go to node NEW instead. No processing is done at node OLD. It is simply bypassed in favor of node NEW. The replacement will remain in effect until revised by the command

REPLACE OLD OLD

This will put node OLD "back" into the network. REPLACE's can be changed also, e.g.,

REPLACE OLD NEW

REPLACE NEW NEWER

All transfers to OLD and to NEW will be routed to NEWER. A subsequent command

REPLACE OLD NEWEST

would route transfers from OLD to NEWEST. NEW would still be replaced by NEWER. The command

REPLACE OLD OLD

has no effect (and is not an error) if OLD had not been previously REPLACED. A REPLACE command may appear anywhere an executable command is permissible and any nodes in the network may be specified. Replacement occurs at the time the command is executed but has no effect until the next transfer to the replaced node. In particular, in the following situation,

NODE ONE

REPLACE ONE TWO

.
.
.
.

END

all of the commands at node ONE will be executed, and normal arc selection will occur. The replacement will have effect only at the next transfer to node ONE.

GOTO and REPLACE commands have symbolisms associated with them for use in drawing networks. A dashed line indicates a GOTO command. Figure 7 (page 25) shows the effect at a GOTO N 5 command at node 4. A test is made at node 4 to determine if the ejection took place below 14,000 feet. If so, there is no FREEFAL phase. The GOTO N 5 bypasses the arc from node 4 to node 5. Appendix A has a more complete description of GOTO symbolism.

REPLACE commands are depicted on the network by lines of arrowheads. REPLACE commands are shown from nodes 3 to 7 in Figure 7 and nodes 3 to 4 in Figure 8 (page 26). In each case, the REPLACE commands are used to end the simulation. In Figure 7, if the pilot hits the tail of the airplane, a REPLACE command routes transfer from the FREEFLY arc directly to node 7. Similarly, in Example 2, if DRILL ends because the end depth, Z2, is reached, a "REPLACE 3 4" command causes transfer to node 4 after TRIPOUT.

Figures 7 and 8 are only schematics; REPLACE symbolism is explained in more detail in Appendix I. In particular an indication can be placed on the network of where the REPLACE command is located. For example,

indicates that a "REPLACE A B" command is located at node C and a "REPLACE A A" is located on arc ONE.

To make full use of network modification features, it must be possible to conditionally execute groups of commands. IOSYM has this capability.

The IF command provides the capability to make logical tests. The syntax for an IF group is

IF, label, variable, truth condition, expression

TRUE

•
• } commands if true
•

FALSE

•
• } commands if false
•

LABEL label

The "label" is an alphanumeric string indicating the "range" of the IF. The IF group is terminated when the label is again encountered on the LABEL command. The "variable" is a variable name or a constant. "Truth conditions" are LT, LE, EQ, GE, and GT. The "expression" is an arithmetic expression of arbitrary complexity. The value of the "variable" is compared with the "truth condition" to the value of the expression. If the resulting logical expression is true, the executable commands following the TRUE command are executed. If not, the commands following FALSE are executed. If the TRUE or FALSE side has no commands, then TRUE or FALSE may be omitted. If both are present, TRUE need not precede FALSE.

As an example, at node 4 in Figure 7 a test must be made to determine if the altitude, Y, of the pilot is below 14,000 feet. If so, a GOTO command is needed to transfer to node 5. The following node 4 commands will accomplish this.


```

NODE 4
  IF 41 Y LE 14000
    TRUE
    GOTO N 5
  LABEL 41
END

```

If $Y \leq 14000$, then GOTO N 5 will be executed. If not, no additional commands will be executed at node 4.

IF commands may be nested and may share labels, e.g.,

```

IF L1 GV(N) GT SIN(X)**2 + COS(X)**2
  FALSE
  IF L2 .5 LE GV(N)
    TRUE
    REPLACE ONE TWO
  LABEL L2
  TRUE
  IF L1 .5 LE GV(N)
    TRUE
    REPLACE ONE ONE
  LABEL L1

```

The LOOP command permits loops similar to FORTRAN 77 DO loops. The syntax for a LOOP group is

```

LOOP, label, index variable, initial value, final value, increment
.
.
.
LABEL label

```

The "label" indicates the range of the LOOP. The "index variable" can be any secondary variable or table element. The "initial value," "final value," and "increment" can be any constant or variable. If "increment" is omitted, "1" will be used.

The commands between the LOOP and corresponding LABEL commands will be executed once for each valid value of the index variable. Valid values of the index are determined by setting the index equal to the "initial value" and incrementing by the "increment." When the index value is no longer in the closed interval whose end points are the initial and final values, looping stops. Execution will continue with the commands following the LABEL command. After looping, the index variable will have the value of the last valid index value. The loop will be executed zero times if (1) the increment is positive and the initial value is greater than the final value or (2) the increment is negative and the initial value is less than the final value. Examples of LOOP commands follow. If the LOOP is executed zero times, the index variable will be set to the initial value.

LOOP L1 GV(J) 1 2.1 0.5	The loop will be executed to LABEL L1 with values of GV(J) equal to 1, 1.5, 2.0.
LOOP L2 I -6 -10 -1	The loop will be executed with values of I equal to -6, -7, -8, -9, -10.
I = -1	
LOOP L3 J 1 10 I	The loop will be executed zero times. J = 1 following LABEL L3.

LOOP and IF commands may be nested in arbitrary fashion and may share labels and LABEL commands.

It is often desirable to terminate execution of a loop before the index has finished incrementing or to skip parts of a loop based on satisfying some condition. The JUMP and EXIT commands permit this.

The JUMP command has the following syntax:

JUMP, label

where "label" is the label on a LOOP command. When a JUMP command is executed, any remaining commands in the loop are not executed, and the index is incremented to its next value. For example, in the following LOOP

```

LOOP LAB1 I 1 5
  IF LAB2 I EQ 3
    TRUE
    JUMP LAB1
    LABEL LAB2
.
.
.
.
LABEL LAB1

```

the commands after LABEL LAB2 would be executed only for values of I equal to 1, 2, 4, and 5.

The EXIT command has the syntax

```
EXIT, label
```

where "label" is a label on a LOOP command. When an EXIT command is executed, the loop whose label is specified is immediately terminated and execution continues with the command immediately following the appropriate LABEL. Consider

```

I = .1
LOOP LL J 0, I, .1
  I = I + .1
  Z = EXP(J)
  IF LL Z GE 26
    TRUE
    EXIT LL
  LABEL LL

```

The above commands illustrate three things. First, the LOOP and the IF share the LABEL command. Second, the final value is reset within the loop so the loop will never end except through the IF command. IOSYM permits resetting the final value, the increment, and even the index value within the loop. Finally, the EXIT command ends the loop when $Z \geq 26$.

If nested loops share labels, then a JUMP or EXIT will pertain to the "innermost" loop. For example,

```
LOOP L1  I 1 10
  LOOP L1  J 1 10
    LOOP L2  K GV(1) NV(J)
      IF L3  Z GE 26
        TRUE
        EXIT L1
        LABEL L3
        .
        .
        .
        .
      LABEL L2
    LABEL L1
```

In the above series, the EXIT L1 will end the loop with index J and will cause I to be incremented. The loop with index I will continue. If it is desired to end the "I loop," then the "I and J loops" must have different labels.

Finally, note that JUMP and EXIT may appear only within the range of a LOOP command.

The ACTIVATE, DEACTIVATE, and SWITCH commands relate to the status of arcs. The syntaxes are

```
ACTIVATE, arc name
DEACTIVATE, arc name
SWITCH, arc name
```

If an arc is active, it is eligible for selection from its emanating node. If the arc is inactive, it is effectively removed from the network and may not be selected. The ability to activate and deactivate arcs from within the network permits network modifications that complement the GOTO and REPLACE capabilities.

The three commands are mnemonic. ACTIVATE makes the named arc active, DEACTIVATE makes it inactive, and SWITCH changes its status to the opposite condition. The three commands may appear anywhere executable commands are acceptable. The status change takes place immediately. If an arc deactivates itself, the current operation will continue, but the arc will not be selected again so long as it is deactivated.

A DEACTIVATE command could have been used as an alternative to ending the simulation for Example 2. (Refer to Figure 8 and Appendix B.) Node 3 could be written as

```
NODE 3
  IF N3 DEPTH LE Z2
    TRUE
      DEACTIVATE D
    LABEL N3
  END
```

where D is the arc name of the arc from node 3 to node 5 which performs the operation CHGBIT. The simulation would end because there would be no active outgoing arcs from node 3.

The EVENT command simply passes the values of specified parameters to a user written subprogram. The syntax is

```
EVENT, event number, parameter list
```

For example,

```
EVENT 6 X Y GV(I) I
```

would pass the values of X, Y, GV(I), and I to the subroutine UEVENT and specify that is event number 6. EVENT is discussed more completely in subsection 3.10.

The PRINT, PUT, GET, and STAT commands are also executable. However, they will not be discussed in this section because the required background has not been introduced. PRINT and STAT commands have to do with statistics and will be discussed in subsections 4.1 and 4.5, respectively. PUT and GET are for use in accessing the GASP IV file storage areas and will be discussed in subsection 4.5 and Appendix C.

3.6 OPERATIONS

In IOSYM, an operation is a set of differential equations which describe a phase of the system operation. Each operation is given a name, so the set of equations can be referenced simply. Also, IOSYM will, on request, collect certain statistics for operations.

The syntax for specifying an operation is as follows:

OPERATION, operation name

```
  .  
  .  
  .  
  .  
  .  
  } commands  
END
```

IOSYM will associate all of the commands between the OPERATION command and the END with the "operation name."

The commands must assign values to the derivative variables for use in numerically integrating the differential equations. These commands are executed at each time step of the integration and are called "type I" commands. It is also sometimes convenient to have sets of commands that are executed only once, either at the beginning or at the end of the operation. Commands executed at the beginning are "type S" commands, and those at the end are "type E" commands. An operation having all these types would be entered like this:

OPERATION, operation name

TYPE S

```
  .  
  .  
  .  
  } type S commands  
END
```

TYPE I

```
  .  
  .  
  .  
  } type I commands  
END
```

TYPE E

```
  .  
  . } type E commands  
  .  
END
```

END

Note that TYPE is an IOSYM command and that each TYPE group must terminate with an END statement. All of the type S, I, and E commands must be executable commands presented in the last section. Type S, I, and E may appear in any order.

Type S commands can be used to initialize variables for an operation. For instance, in Example 1 (see Appendix B) the initialization for the entire simulation is done by type S commands on the operation EJECT.

Type I commands must assign values to all derivative variables with nonzero values. Any derivative variable not assigned a value by type I commands will be set to zero by IOSYM.

Commands are performed sequentially as in FORTRAN. To the user, it appears that the first type I command sets all derivative variables equal to zero. In other words, IOSYM internally zeroes the derivative variables before executing the type I commands. Therefore, only derivatives that differ from zero need be set explicitly.

Type S, I, and E commands can also be specified on arcs and under IOSYM GLOBAL commands. When specified on arcs, these commands become part of the operation only when that arc is selected (an operation can be performed on more than one arc). Under GLOBAL, the commands are part of all operations. These features are discussed in subsections 3.7 and 3.8.

OPTIME is an IOSYM-provided secondary variable that is set automatically on operations. Prior to executing the first type S command, IOSYM performs the equivalent of

OPTIME = TIME.

So, during the operation, OPTIME is equal to the start time of the operation. Then, prior to the first type E command, IOSYM performs the equivalent of

OPTIME = TIME - OPTIME.

Therefore, during type E commands, OPTIME is the time interval spent doing the operation.

Suppose there are three primary variables: COST, DIST, SPEED. Consider the following operation:

```
OPERATION, CLOSE
  TYPE S
    COST = COST + 10
    SPEED = 0
  END S
  TYPE I
    DCOST = AV(1)**2 * SPEED
    DSPEED = SPEED + DIST
  END I
  TYPE E
    DIST = DIST + OPTIME * GV(7)**2
  END E
END OP
```

Each time CLOSE is performed, a fixed cost of 10 is incurred, and SPEED is initialized to zero. The derivative variable DCOST and DSPEED are set. If DDIST is not set on the arc or by GLOBAL, then IOSYM assigns zero to it. After the operation, DIST is incremented by the square of GV(7) times the length of time spent doing CLOSE.

Note that the operation contains no information about how it terminates. This information is specified on arcs.

3.7 ARCS

Arcs are the most complex structures in IOSYM and are made up of several commands. Information specified on an arc determines how it is selected, which operation is performed, and how the operation terminates. Arcs are made up of the following commands which will be discussed in order.

ARC

PICK

OPR

TYPEs

TERM

CROSS

The ARC command begins the information for a new arc and has syntax:

ARC, arc name, start node, end node, initial status.

Arcs are identified by name. The start and end nodes are identified, and the initial status is specified. Arcs may be active or inactive. An active arc is eligible for selection. An inactive arc cannot be selected and is effectively removed from the network. The status of the arc at the beginning of the simulation run is specified as A, for active, or I, for inactive. If no status is specified, A is assumed.

Examples are

ARC, ONE, A, B

ARC, ONE, A, B, A

ARC, 3, SELECT, BOXED, I

The first two ARC commands are equivalent; the third names the arc "3" and specifies start and end nodes SELECT and BOXED, respectively. The arc is initially inactive.

The PICK command specifies the criterion for selecting the arc. The syntax is

PICK, pick number, logical operator, expression.

The logical operator is one of LT, LE, EQ, GE, or GT. The expression will be evaluated and compared to zero with the logical operator to obtain a true or false condition as follows:

(expression) operator zero

For example, the condition

$$x^2 \geq 0$$

could be specified by

PICK 3 GE X**2

The "pick number" specifies the order in which the arcs leaving a node are examined. The arcs are examined in increasing order of pick number. Pick numbers need not be consecutive, must be positive, and will be truncated to integer values.

Only one arc can be selected, so a conditional-take-first logic is followed. In other words, the first arc encountered whose condition is true is selected.

The following four rules completely characterize arc selection:

1. If only one arc leaves a node, no PICK command is required. If more than one arc leaves, then every arc must have a PICK.
2. PICK commands may be abbreviated if no condition is associated with it. The abbreviated form is

PICK, pick number

An abbreviated PICK has a default true condition associated with it, so the first abbreviated PICK encountered will be selected. Figure 9 contains an example. PICK 2 will be selected if $X \geq 2$. If not, PICK 4 will be taken. PICK 7 will never be selected (unless PICK 4 has been deactivated). This feature permits a default arc if no other PICK condition is satisfied and allows different selections based on the status of the outgoing arcs.

3. Regardless of PICK condition, if only one outgoing arc is active, it will be selected even if its PICK condition is false.
4. If no outgoing arcs are active, the simulation run will terminate normally. If more than one arc is active and none is selected because all of the PICK conditions are false, a fatal error occurs.

Figure 9. Example of Abbreviated PICK Commands

The OPR command is used to specify the operation to be performed. The syntax is

OPR, operation name, list of substitutable parameters

Type S, I, and/or E commands can follow immediately after the OPR command. These commands will be added to the commands of the same type specified under GLOBAL and on the named operation. The order of execution is GLOBAL, operation, and arc. In other words, for commands of type X, all GLOBAL type X commands will be executed first, then all type X commands on the operation will be performed. Finally, the type X commands specified on the arc will be executed.

Type S, I, and E commands under GLOBAL, operations, and arcs may all be written with dummy parameters. The parameters are denoted by the dummy table DUMM(.). Up to 10 dummy variables may be used, and they are specified by DUMM(1), DUMM(2), etc. Subscripts must be numbers, not variables.

The real values to be used in place of the DUMMs are specified in the "list of substitutable parameters" on the OPR command. The parameters on the OPR card are order dependent. The first one becomes DUMM(1), the second becomes DUMM(2), and so on.

Two substitute parameters are used in Example 1. Drag is calculated in type I commands under GLOBAL (see Appendix B). The statement is

$$\text{DRAG} = .5 * \text{DUMM}(1) * V^{**2} * \text{DUMM}(2)$$

Here, DUMM(1) represents the drag coefficient, and DUMM(2) is the area.

On each arc, the correct values for these quantities are placed on the OPR command. For example, on arc D where FREEFAL is performed, the OPR command is

OPR FREEFAL CDP SP

Substitutable parameters may be numbers or variables. Expressions are not permitted. Default values may be specified on the GLOBAL command as discussed in the next section.

An example of additional commands on an arc can be seen at arc B in Example 1 where type I commands are needed to detect the pilot hitting the tail of the airplane.

Occasionally it is useful to use an arc as a connector only. No operation is desired but some commands may be needed. For this case, the OPR command is omitted. TYPE commands immediately follow the PICK (or the ARC command if there is no PICK). The arc will consume no simulated time. In other words, the arc will start and stop at the same simulated time. Any type S and E commands (including those on GLOBAL) will each be executed once. Type I commands will not be performed.

Operations may be ended in two ways: (1) after a specified elapsed time or (2) when some condition on specified variables is satisfied. The latter case is called a crossing and refers to a condition when the value of some function crosses a specified value.

The TERM command is used to specify an elapsed time condition. The syntax for a TERM group is

TERM, termination number, expression for the end time

```
.  
.  
.  
.  
.  
.
```

} commands to be executed only if the operation
is ended by this TERM

END

More than one TERM group may be present on an arc. Each TERM is numbered by a termination number that must be a positive integer. The numbering is arbitrary and need not be consecutive.

After type S commands are performed on the arc, the expression on each TERM will be evaluated. The result will be used as an end time for the operation. The arc will actually terminate on the minimum of the end times specified on all the TERM commands. Then any additional commands to be executed will be performed (after all the type E commands on the GLOBAL, OPERATION, and ARC have been completed). Only the commands immediately following the TERM which caused termination will be executed.

Consider the following example.

```
TERM 4 TIME + GV(6)  
  GOTO N ALARM  
  END  
TERM 2 TIME + NORMTIM  
  END
```

Two time intervals are specified, GV(6) and NORMTIM. The end times are specified as the current time plus the appropriate time interval. If $\text{NORMTIM} < \text{GV}(6)$, then TERM 2 will end the operation and control will pass to the end node of the arc as specified on the ARC command. If, on the other hand, $\text{GV}(6) < \text{NORMTIM}$, TERM 4 will end the arc and the GOTO N ALARM command will be executed. The next transfer will be to node ALARM and not to the normal end node of the arc.

The tolerance is specified for use by the numerical integration algorithm. It is possible that a time step will be taken which causes the crossing to occur but which exceeds the tolerance (if, for example, the pilot moves from 14,050 to 13,700 feet in one time step). The program will reduce the time step to attempt to determine the crossing to the specified tolerance. The tolerance is an indication of how accurately the user wants the crossing detected. Very small tolerances require small minimum step sizes and consume additional computer time.

Note that a crossing must occur. Suppose X has an initial value zero and decreases. The condition

CROSS 6 LE 1 X

will never cause termination because X does not cross the value zero. If X increased above zero and decreased below zero, it would be detected.

The crossing condition EQ will detect a crossing of zero in either direction, but still a crossing must occur. GE is analogous to LE but in the increasing direction.

There are no limitations on the expression other than that it be a smooth function of its variables. Smoothness is a sufficient condition, not a necessary one. Fundamentally, the only requirement is that the expression be compatible with the tolerance specified (e.g., discrete jumps larger than the tolerance will cause an error because the crossing condition cannot be satisfied).

The first crossing which is detected ends the arc. CROSS expressions are evaluated in increasing order of their termination number. If two crossings occur simultaneously, only the one with the lowest termination number will be detected, and only the additional commands in its CROSS group will be executed.

As many TERM and CROSS groups as needed may appear on an arc. The earliest termination time or the first crossing will end the arc.

IOSYM statistics are presented in Section 4, and discussion of that topic will be deferred until then.

Type S, I, and/or E commands are specified in the GLOBAL group just as they are on operations and arcs. Such commands are executed on every arc (if the arc has no operation, type S and E commands are executed along with any such commands specified on the arc). In every case, the GLOBAL commands of the specified type are executed, then the commands of the same type on the operation are performed, and finally any such commands on the arc are executed.

If DUMM(.) dummy variables are used on any command (in GLOBAL, on operation, or on an arc), then default values may be specified on the GLOBAL command. As an example, suppose we have the following:

```
GLOBAL, GV(6), 3.4, 8
```

```
.
```

```
.
```

```
.
```

```
.
```

```
END
```

```
.
```

```
.
```

```
.
```

```
ARC A 1 2
```

```
  OPR, LIFT, X,, THUD
```

```
.
```

```
.
```

On arc A, DUMM(1) would be replaced by X, DUMM(2) would have the default value 3.4, and DUMM(3) would be replaced by THUD. If the OPR command had been

```
  OPR, LIFT
```

then all three DUMMs would use the default values. If no default values are specified and DUMMs are used, then every OPR command must specify a complete list of substitutable parameters.

Example 1 (see Appendix B) is an example using default values. No list of parameters is specified on arc A, so the values of the DUMMs are taken from the GLOBAL command.

3.9 LIBRARY, MENU, AND PROBABILITY FUNCTIONS

IOSYM provides most standard FORTRAN library functions as well as a group of more complex "menu" functions for use in SET commands and other arithmetic expressions. In addition, samples can be drawn from a group of probability distributions.

The library functions recognized by IOSYM are the following:

ATAN	EXP
AMAX1	ALOG
AMIN1	SIN
AMOD	SQRT
SIGN	TAN
ABS	IFIX
COS	ATAN2

All of the functions except AMAX1 and AMIN1 operate as they do in FORTRAN. Only two arguments are permitted for these functions. The arguments for any of the functions may be expressions.

Also, 29 menu functions are available. The general syntax for accessing a menu function is shown below.

Fi(list of variables/list of tables/list of parameter sets/list of accessible entities/truth condition)

The "i" is the number of the menu function desired. For example, F1, F20, and F28 would access functions 1, 20, and 28, respectively. The list of variables is a list of values to be passed to the function. An example is

F1(Z, -K, 202.7, (-Q + 4), (AV(7) + TIME))

Note that the variables may be expressions.

The list of tables indicates which tables should be passed to the function. Some menu functions require tables; however, a default table will be used if none is specified. The default table is an imaginary table all of whose values are equal to one. Suppose a function requires three tables, and the first two are desired to be defaulted. The third table is GV. The list would look like this:

Fi(• • •/, ,GV)

The following

Fi(• •/GV)

Fi(• •/GV, ,AV)

Fi(• •/)

would default the second and third tables, the second table, and all the tables, respectively.

Parameter sets are data sets consisting of four numbers. GASP IV provides the storage for the parameter sets, and they are used primarily in the probability functions that will be discussed later. The syntax for menu and probability functions is the same, however, so the parameter sets are mentioned here. They are discussed in more detail later in this section and in subsection 5.5. No menu functions require parameter sets. Parameter sets may be defaulted in the same way as tables. The value 1.0 will be used for defaults.

Accessible entities are required by some menu functions. They also may be needed with user functions which are discussed in subsection 3.10.

The truth condition is one of LT, LE, EQ, GE, or GT. If no truth condition is specified, the value returned is the value computed by the menu function. If a truth condition is specified, the computed value is compared to zero with the condition. If a true results, the value one is returned. If false results, the function returns zero.

A few examples will be helpful. Let t_{ij} be the j^{th} element of table i , and let X_j be the value of the j^{th} variable in the variable list. Let the computed value of the menu function be y .

Menu function one computes the following (a complete list of menu functions is given in Appendix D):

$$y = \sum_{j=1}^{n_x} t_{1j} x_j$$

Where n_x is the number of variables. Clearly, F1 computes linear combinations of the variables using the table elements as coefficients.

Let $GV(1) = 1$, $GV(2) = -1$, and $Z = 2$. The following examples illustrate the use of menu functions.

	<u>Expression</u>	<u>y</u>	<u>F1</u>
1.	F1(Z, -Z/GV)	4	4
2.	F1(Z, Z/GV)	0	0
3.	F1(Z, Z)	4	4
4.	F1(Z, Z////GE)	4	1
5.	F1(Z, Z////LT)	4	0
6.	F1(Z, Z/GV////EQ)	0	1

In the first two examples, the linear combination is calculated, and no truth condition is specified. Hence, $F1 = y$. In Example 3, the table is defaulted so $y = Z + Z$, and, since no truth condition is specified, $F1 = y$. Example 4 is the same as Example 3 except the GE condition is imposed. The condition $y \geq 0$ is tested with $y = 4$. A "true" results, and $F1 = 1$. Example 5 is the same as 4 with LT. A "false" results, so $F1 = 0$. In Example 6, $y = 0$. The truth condition is $y = 0$, which is true, so $F1 = 1$.

A menu function uses no more information than it needs, so specifying additional information such as tables does no harm. In particular,

F1(Z, Z/GV)

and

F1(Z, Z/GV, NV)

are equivalent since F1 requires only one table. NV will simply not be used.

Variable lists may be arbitrarily long. An error will result if a specified table does not have sufficient elements to evaluate the function.

Truth conditions are useful for testing logical conditions. For example, suppose an arc is to be selected only if $Q + R > 6$ and $X \leq 7$. The following PICK command would evaluate as true for this case.

```
PICK 1 EQ F1(Q + R, - 6//// GT) + F1(X - 7//// LE) - 2
```

Only if both F1 functions evaluate as true will the PICK be satisfied.

Probability functions use the same syntax as menu functions except the "F" is replaced by "P". Twelve distributions are available for sampling. They are listed with their parameter set requirements in Appendix D. Also, truth conditions are not operative with probability functions.

Most probability functions require one parameter set. Parameter sets are referenced by number. The normal distribution is probability function 6 and requires a parameter set with the following information

1. Mean value
2. Minimum value
3. Maximum value
4. Standard deviation

Values sampled from the distribution will never be less than the minimum nor more than the maximum values. If these values are stored in parameter set 4, then

```
Z = P6(ISTR//4)
```

would return a sample from the normal distribution with specified parameters. Random number stream ISTR ($1 \leq \text{ISTR} \leq 6$) would be used to generate the sample.

Some probability functions do not require a parameter set. For example,

```
P2(5, 10, ISTR)
```

will sample uniformly between 5 and 10 with random number stream ISTR.

3.10 USER-WRITTEN FORTRAN PROGRAMS

IOSYM permits a flexible array of methods for the user to supplement the capabilities of the language with user-written FORTRAN programs. Several of the methods are for use in interacting with the underlying GASP IV language. These capabilities require a familiarity with GASP IV and will be discussed in Appendix C. Others can be used to perform special functions in addition to those provided automatically by IOSYM. These last cases will be discussed here.

FORTRAN FUNCTION U is a way the user can supplement the menu and probability functions provided by IOSYM. It is accessed with the same syntax as "F" and "P" functions but with a leading "U".

Suppose the user desires functions in addition to the menu functions. He would then write a FORTRAN function whose initial statements are

```
FUNCTION U(IFF, NX, X, NT, ITAB, NP, IPAR, NA, IAC)
  DIMENSION X(NX), ITAB(NT), IPAR(NP), IACC(NA)
  .
  .
  .
  GO TO (100, 200, ....) IFF
  .
  .
  .
```

In the IOSYM input, statements similar to the following might appear.

```
X = U1(. . . .)
GV(I) = U2(. . . .)
```

When IOSYM encounters such statements, it will call FUNCTION U with IFF equal to the user function number (1 or 2 in the example above). The user must write FUNCTION U in such a way (the computed GO TO, for example) that U returns the appropriate value. If a truth condition is specified, it will be processed by IOSYM after return from FUNCTION U.

The parameters to FUNCTION U have the following meanings:

IFF	The user function number.
NX	The length of the list of variables; $NX \geq 0$.
X	An array of length NX containing the <u>values</u> , in order, of the variables. <u>None of the values of array X may be changed in U.</u>
NT	The length of the list of tables; $NT \geq 0$.
ITAB	An integer array containing the <u>internal</u> number of the tables specified. IOSYM internally keeps track of tables by number (e.g., 1, 2, 3, etc.). This is the number in the ITAB array. A defaulted table is represented by a zero value. IOSYM provides utility subprograms with which the user can access tables. These are presented in the next section. The statement $X = U1(/, GV)$ would result in $ITAB(1) = 0$, $ITAB(2) = 1$ (since GV is the first IOSYM table), and $NT = 2$ (and $NX = 0$).
NP	The length of the list of parameter sets; $NP \geq 0$.
IPAR	An integer array of length NP containing the parameter set numbers in the order specified.
NA	The length of the list of accessible entities; $NA \geq 0$.
IACC	An integer array of length NA containing the <u>internal</u> number of the entity.

Accessible entities may be variable names (e.g., X, GV(1), NV(1)), node names, or arc names. IACC contains the following for each type.

Variable name	An integer number which uniquely identifies (to IOSYM) the variable
Node name	IOSYM node number
Arc name	IOSYM arc number

IOSYM numbers the arcs and nodes as they are encountered. The command "NODE 4" will, in general, not be numbered four internally. The "4" on the NODE command is a label. The same is true of arcs.

FORTTRAN utility functions (described in the next section) are provided for accessing and changing the values of variables, tables and parameter sets, and for performing REPLACE and GOTO commands. The information in ITAB, IPAR, and IACC permit such capabilities to be used in FUNCTION U.

IOSYM searches for accessible entities in the order shown above (e.g., variables, nodes, then arcs). Therefore, if an arc and a node have the same name, it may not be possible to pass the arc as an accessible entity. Also, IOSYM cannot recognize arc and node names before they have been defined. Hence a NODE or ARC command with the specified name must have appeared in the network prior to the use of the name in a list of accessible entities.

Examples of the use of accessible entities in FUNCTION U will be given in the next section.

SUBROUTINE UEEROR provides the user with the ability to produce error-induced output to supplement IOSYM output. When IOSYM detects errors during input or execution, it calls the IOSYM error processing program EEROR. The calling sequence is

```
SUBROUTINE EEROR(IERR1, ISUB, IERO)
```

IERR1 is a logical variable that is set TRUE in EEROR if a fatal error has occurred. ISUB is the number of the calling subprogram, and IERO is the error code. A complete list of IOSYM error codes and a complete description of error processing are given in Appendix D.

Even error codes produce only warning messages. Odd error codes represent fatal errors. Whenever a fatal error occurs, EEROR calls UERROR which has the following calling sequence:

```
SUBROUTINE UERROR(IERO)
```


The user may produce any output he desires in UERROR. He should not, however, execute a STOP or CALL EXIT statement; IOSYM will perform any actions needed to terminate execution and will attempt to produce as much output as possible.

The user may also create error codes of his own. If the user detects errors in any of his user-written subprograms, he may call EEROR himself to obtain standard IOSYM diagnostics. IERR1 need not be a defined variable but it must be a variable, not a number or expression in the CALL statement. ISUB may be an arbitrary integer, and IERO is the user's error code. If the user wishes to generate a fatal error that ends execution, IERO must be odd and negative. Only odd, negative error codes result in a call to UEEROR and job termination (during input, IOSYM will attempt to finish reading the entire network, even if a fatal syntax error has occurred). Odd, positive error codes are used for fatal syntax errors. IOSYM terminates the job after the entire network is read. If the error code is negative, execution is terminated by EEROR and no RETURN is executed from that subroutine.

SUBROUTINE UEVENT is needed to implement the EVENT commands. The calling sequence and program schematic are given below.

```
SUBROUTINE UEVENT(IV, NP, PAR)
  DIMENSION PAR(NP)
  .
  .
  GO TO (100, 200, . . .) IV
```

The parameters are

```
IV -- Event number
NP -- Length of the parameter list NP  $\geq$  0
PAR -- Array of length NP that contains the values of the
      parameters
```

The values of the array PAR may not be changed in UEVENT. A computed GOTO or equivalent branching logic can be used to execute the required actions for each EVENT command in the network.

In Example 1, an EVENT command (without parameters) is used to do the node replacement required to skip the operation FREEFAL. The utility routine RREPLC is used in UEVENT to accomplish this (See Appendix B).

SUBROUTINE USTAT can be used to collect additional statistics. USTAT is called on each arc before the first type S command and after the last command on a TERM or CROSS. USTAT is also called at the end of each simulation run. The calling sequence is

SUBROUTINE USTAT(IS)

The first call on an arc is made with IS = 1. The second call is with IS = 2. The call at the end of the simulation run has IS = 3.

SUBROUTINE UINTLC is called at the beginning of the simulation run to permit the user to do any input and/or initialization needed. The calling sequence is

SUBROUTINE UINTLC

UINTLC is called after the first node has been processed in each simulation run. Every simulation begins at a node as explained in subsection 5.2, so UINTLC permits initialization at the beginning of the simulation run after any commands at the first node have been executed.

UINTLC might be used to initialize variables to be used in other user-written subprograms. Such information can be stored in user-created COMMON storage. Also, input prior to each run can be performed.

SUBROUTINE UOTPUT is called after each simulation run and can be used to produce user output. The calling sequence is

SUBROUTINE UOTPUT

SUBROUTINE UREAD is provided to do user input before each simulation. The calling sequence is

SUBROUTINE UREAD(ISIM)

where ISIM is the number of the simulation to be performed next. A simulation consists of one or more simulation runs. UREAD is called

before each simulation. UINTLC is called before each simulation run. Placement of data cards for reading from UREAD and UINTLC is discussed in subsection 5.2.

SUBROUTINE UPLOT is provided to allow the user to produce plots in addition to the standard IOSYM line printer plots. The calling sequence is

```
SUBROUTINE UPLOT(NUMPLT, IVAR, IMPLTNM, ITP)
```

where the parameters are

NUMPLT -- Number of dependent variables

IVAR -- Array with identifiers of variables. IVAR(1) is the independent variable. IVAR(2) to IVAR(NUMPLT + 1) are the dependent variables

IPLTNM -- Plot number, IPLTNM > 0

ITP -- File number on which plot information is stored.

To use UPLOT, the user must be familiar with the internal storage of plot information. Further discussion of UPLOT can be found in subsection 4.4.

Several additional capabilities are also available to the user familiar with GASP IV and who is using features of GASP not directly available in IOSYM. These capabilities are accessible through the subprograms entitled UEVNTS, UTERM, USCOND, USSAVE, and USTATE which are discussed in detail in Appendix C.

3.11 UTILITY SUBPROGRAMS FOR USE IN USER PROGRAMS

IOSYM provides several subprograms that the user can access in user-written programs. These utility programs allow access to IOSYM variables, permit performing certain commands, and provide certain input, output, and data manipulation capabilities.

1. FUNCTION VVARBL(I12, IDV)

I12: 1 = current value

2 = initial value

IDV = variable identifier

IOSYM assigns a unique integer number to each variable and table element. It is referred to as a variable identifier (IDV). When accessible entities are used in FUNCTION U, the values in the IACC array for variables are the IDVs of the variables and table elements. VVARBL will return the value of the variable specified by IDV. If I12 = 2, the value at the beginning of the simulation (initial value) will be returned. If I12 = 1, the current value will be returned.

2. SUBROUTINE PPIDF(IDV, VAL)

IDV = variable identifier

VAL = value to be stored

PPIDF will change the current value of the variable IDV to VAL.

VVARBL and PPIDF permit accessing and changing the values of accessible entities that are variables. VVARBL is used in subroutine USSAVE in Example 2 to access variables to be plotted with the GASP IV plot program (See Appendices B and C).

3. FUNCTION GGVAR(I0123, IVAR)

I0123: 0 = independent variable

1 = primary variable

2 = secondary variable

3 = derivative variable

IVAR = variable number

IOSYM also numbers variables within type. For example, I is the first secondary variable, J is the second, etc. GGVAR returns the value of the simple variable numbered IVAR of type I0123. If IVAR < 0, the initial value will be returned. For I0123 = 0, only the sign of IVAR is significant. If IVAR > 0, the current value of the independent variable will be returned; if IVAR < 0, the initial value will be returned.

IVAR can be determined by examining the network summary printed by IOSYM after it has read the network. All variables are listed by

internal number (not IDV). User programs may then be written using GGVAR instead of VVARBL if desired.

4. SUBROUTINE PPVAR(I123, IVAR, VAL)

I123: 1 = primary variable
2 = secondary variable
3 = derivative variable

IVAR = variable number

VAL = value to be stored

PPVAR will set the specified variable equal to VAL. I123 = 0 is not permitted; the independent variable is set automatically by GASP IV and the user should not change it. PPVAR affects only current values, not initial values. I123 $\leq$ 0 is not permitted.

5. FUNCTION GGTab(ITABL, LMENT)

ITABL = table number

LMENT = $\pm$ element number

GGTAB returns the value of the LMENT element of table numbered ITABL. IOSYM numbers the tables and prints the table name/table number correspondences in the network summary. For example, GV is always the first table. If LMENT < 0, the initial value is returned; otherwise the current value is returned. If ITABL $\leq$ 0, GGTab returns a value of 1.0, and LMENT is not used.

6. SUBROUTINE PPTAB(ITABL, LMENT, VAL)

ITABL = table number

LMENT = element number

VAL = value to be stored

PPTAB will change the current value of the specified table element to VAL. Only current values may be changed.

7. FUNCTION GGPAR(IPAR, LMENT)

IPAR = parameter set number

LMENT = element number; $1 \leq \text{LMENT} \leq 4$

GGPAR will return the value of the LMENT element of parameter set IPAR. Parameter sets do not have initial values. If $\text{IPAR} \leq 0$, GGP

AR returns 1.0, and LMENT is not used.

8. SUBROUTINE PPPAR(IPAR, LMENT, VAL)

IPAR = parameter set number

LMENT = element number

VAL = value to be stored

PPPAR will change the specified parameter set value to VAL. Since parameter sets do not have initial values that are stored separately, any changes to parameter sets will remain not only through the remainder of the simulation run, but also during all subsequent runs.

9. FUNCTION IIACST(IARC)

IARC = arc number

IIACST will return the current status of the specified arc. IOSYM numbers the arcs, and IARC is the internal arc number. It is the value passed in IACC for an arc that is an accessible entity. IIACST returns 1 if the arc is active, -1 if it is inactive.

10. SUBROUTINE SSWARC(IDO, IARC)

IDO: 1 = activate

2 = deactivate

3 = switch

IARC = arc number

SSWARC may change the current status of the specified arc.

11. SUBROUTINE RREPLC(NODOUT, NODIN)

NODOUT = number of node to be replaced

NODIN = number of node replacing NODOUT

RREPLC will cause a node replacement command to occur. The node numbers are the IOSYM internal node numbers. They are available in the network summary and are passed in the IACC array when nodes are accessible parameters.

12. SUBROUTINE GGOTO(IARCND)

IARCND = arc number or negative of node number

If IARCND > 0, GGOTO executes the equivalent of a GOTO A IARCND command. If IARCND < 0, GGOTO executes GOTO N ABS(IARCND). The node and arc numbers are the internal representations and are passed in IACC to FUNCTION U.

13. SUBROUTINE ZZSET(Z, N, VAL)

Z = real array

N = length of Z

VAL = value to be stored

All elements of Z are set to equal to VAL.

14. SUBROUTINE IISSET(IX, N, IVAL)

IX = integer array

N = length of IX

IVAL = value to be stored

All elements of IX are set equal to IVAL.

15. SUBROUTINE VVMOV(X1, X2, N, XMUL)

X1 = real array

X2 = real array

N = length of X1 and X2

XMUL = multiplier

The array X1 is set equal to $X2 * XMUL$, term by term.

16. SUBROUTINE IIMOV(IX1, IX2, N, IMUL)

IX1 = integer array

IX2 = integer array

N = length of IX1 and IX2

IMUL = multiplier

The array IX1 is set equal to $IX2 * IMUL$, term by term.

17. SUBROUTINE FFRJ(X)

X = variable to be read

IOSYM provides a suite of subprograms that will perform format free data input for the user. FFRJ is one such program. FFRJ will read a single datum from a data record and interpret it as a real FORTRAN variable. It will be returned stored in the variable X.

Data is entered without reference to column positions. Data items are separated by blanks or commas. As many items as desired may be entered on a line.

Eighty columns of information are read and stored in an internal buffer array. When a call is made to FFRJ or any of the routines to be described subsequently, the next datum is interpreted as a value of the appropriate type. When the buffer is exhausted, it is renewed by reading another record. A dollar sign (\$) will cause the record to be terminated. In other words, nothing after the "\$" will be used as data. Everything after the "\$" is a comment.

Any form of the data acceptable to FORTRAN can be read. Thus, the following FORTRAN statements,


```

CALL FFRJ(X)
CALL FFRJ(Y)
DO 10 I = 1, 3
10 CALL FFRJ(Z(I))

```

will read $X = 1$, $Y = 1.E - 3$, $Z(1) = 13.5$, $Z(2) = -100.$, and $Z(3) = .333$ from any of the following data streams.

<u>Stream 1</u>	<u>Stream 2</u>	<u>Stream 3</u>
1	1.0, 1.E-3, 13.5	1. .001
.001	-100 .333	13.5, -100.0
13.5		3.33E-1
-1E2		
.333		

18. SUBROUTINE FFIJ(IX)

IX = variable to be read

FFIJ is analogous to FFRJ for integer variables.

19. SUBROUTINE FFAJ(IALPH)

IALPH = variable to be read

FFAJ will read alphanumeric information and store it in IALPH. To be compatible with FFAJ, IALPH must be an integer variable. FFAJ will read the first to the last nonblank characters in the next datum. If blanks are to be embedded in the datum, the entire field must be enclosed in double quote marks (or the equivalent of the 8 - 4 punch in 026 code). For example, to store SITE 1 in IALPH requires a datum of "SITE 1".

20. SUBROUTINE FFLJ(LX)

LX = variable to be read

FFLJ performs a read of a logical variable. LX must be type logical. If the first character of the datum is a "T", then LX will be set TRUE. Otherwise, LX is FALSE.

21. SUBROUTINE FFDJ(DX)

DX = variable to be read

FFDJ performs a double-precision read. DX must be a double-precision variable.

22. SUBROUTINE FFPUDJ

FFPUDJ clears the internal buffer. In other words, the next call to an FF routine will force a new record to be read even though the present record may not have been exhausted.

23. SUBROUTINE PVECJ(X, N, LAB, LINE, LEN)

X = real array to be printed

N = length of X

LAB = word containing one alphanumeric character

LINE = integer array containing an alphanumeric label,
four characters per word

LEN = length in words of LINE

PVECJ prints the array X using the label in LAB. The output is preceded by the contents of LINE. If $LEN \leq 0$, LINE will not be printed.

24. SUBROUTINE PIVECJ(IX, N, LAB, LINE, LEN)

PIVECJ is analogous to PVECJ for integer arrays.

25. SUBROUTINE PMATJ(A, IA, M, N, LINE, LEN)

A = real, two-dimensional array to be printed

IA = maximum first dimension of A

M = number of rows to be printed

N = number of columns to be printed

LINE = integer array, as in PVECJ

LEN = integer array, as in PVECJ

The principle M-by-N submatrix of A will be printed.

26. SUBROUTINE PKMATJ(KA, IA, M, N, LINE, LEN)

PKMATJ is analogous to PMATJ for the integer array KA.

27. SUBROUTINE PVARJ(X, LINE, LEN, ICRTL)

X = real variable to be printed

LINE = array as in PVARJ, but no more than 5 words

LEN = array as in PVARJ, but less than or equal to 5

ICRTL = 1 print a blank in column 1 (single space)
= 2 print a 0 in column 1 (double space)

PVARJ will print the variable X.

28. SUBROUTINE PIVARJ(JJ, LINE, LEN, ICRTL)

PIVARJ is analogous to PVARJ for the integer variable JJ.

29. SUBROUTINE PLOGJ(TF, LINE, LEN, ICRTL)

PLOGJ is analogous to PVARJ for the logical variable TF.

30. SUBROUTINE PSTRJ(LINE, LEN)

LINE = integer array of not more than 20 words containing
alphanumeric information; 4 characters to each word

LEN = length of LINE in words, LEN > 0

PSTRJ will print the alphanumeric information in LINE.

31. SUBROUTINE LNFDJ (LF)

LF = number of printer lines to advance

LNFDJ will advance the printer LF lines. LF may be zero.

32. SUBROUTINE UUPRNT(ITAB, NV, IVAL)

ITAB = tabulation number

NV = length of IVAL; $NV \leq 4$

IVAL = array containing IDV's of variables

UUPRNT permits the user to create entries on IOSYM tabulation. UUPRNT is discussed further in subsection 4.1.

3.12 RESERVED NAMES AND CONVENTIONS IN IOSYM

The following rules should be followed when selecting names. Violating any of them may cause IOSYM to be unable to interpret input correctly.

1. Avoid names like F_i , U_i , or P_i where i is a number. Also, do not use names that are the same as library functions.
2. Do not use names that are the same as IOSYM commands. Command names are listed in Appendix A.
3. Do not use names that can be confused with derivative variables. For example, if a primary variable has name X , do not create another variable with name DX .

In general, it is good practice to select unique names for arcs, nodes, variables, tables, etc.

4. IOSYM STATISTICS

IOSYM will, on command, automatically produce certain informative statistics. These are described in the following sections. All statistics commands are placed under GLOBAL in the network.

4.1 TABULATIONS

IOSYM tabulations are requested with the TABULATE command. The syntax is shown below.

TABULATE, tabulation number (time parameter), list of variables and report types.

When tabulations are requested, a message is printed at the end of each operation giving the start and end times, the duration, the operation and arc names, and the values of the variable types specified.

Up to 10 tabulations may be requested. The "tabulation number" is an integer from 1 to 10. The tabulations are printed, in order, after each simulation run. Multiple tabulations need not have consecutive tabulation numbers. The time parameter is the character "T". If present, all times will be printed as

dd.hh.mm

where dd, hh, and mm are days, hours, and minutes, respectively. This assumes that the units of the variable "TIME" are hours. If the time parameter is absent, the value of the independent variable is printed directly as a decimal number.

Up to four variables may be specified in the list of variables. Each variable has a "report type" which is one of the following:

S -- Print value at start of operation (before type S commands)

- E -- Print value at end of operation (after last commands on TERM or CROSS)
- C -- Print the change in the value that occurred during the operation.

A variable list with report types has syntax

variable name(type), variable name(type)

Examples are

TABULATE 2 X(E) GV(J)(S)

TABULATE 4(T) COST(S) COST(C) DEPTH(E) DDEPTH(E)

In the first case, only two variables are specified and the time option is "off." Four variables are specified for tabulation 4, and the time option is "on." No variables need be specified, but if they are, the report types must be provided. Examples of tabulation output are given in subsection 6.2.

Tabulations automatically produce one entry for each operation, and the same variables are printed in each entry. Occasionally it is convenient to print special entries on tabulations. IOSYM provides two ways to do this. The first is with the PRINT command. The syntax is

PRINT, print variable, list of variables

PRINT is an executable command as discussed in subsection 3.5. The values of the list of variables are printed whenever the PRINT is encountered. The "print variable" determines where the information is printed. The values and meanings of the print variable are as follows (n is the print variable):

<u>Value</u>	<u>Meaning</u>
$1 \leq n \leq 10$	Print on tabulation n
999	Print on all tabulations
0	Print immediately on the output file
$n < 0$	Skip this print command

The first two cases are self explanatory. The last two cases are for use in debugging a network.

Tabulation output is not printed as it is generated. It is stored out of core until the simulation run is over; then the tabulations are printed. In case of execution errors, IOSYM will attempt to produce whatever statistics are available before job termination. If this is not possible, little output may be available for diagnostics. PRINT commands of the following form may be inserted at strategic places in the network

PRINT, PV, list of variables

with PV assigned to zero. Printing will occur as the PRINT statements are executed. When the network has been debugged, setting PV = -1 eliminates unwanted output without removing the PRINT commands.

No report type is associated with the list of variables on PRINT. The value at the time of execution is printed. In Example 1, on node 7 the command

PRINT 1 TAIL V

is used to print velocity and distance from the tail in the case where the pilot strikes the tail of the airplane.

Entries that emulate PRINT commands can be generated by the user with subroutine UUPRNT (see subsection 3.11). The IDVs of the variables are passed in the IVAR array. NV may be zero. ITAB has the same meanings as the print variable on the PRINT command.

Additional debugging information can be obtained by use of the TRACE command. TRACE is also placed under GLOBAL and has syntax

TRACE, variable list

The current values of up to five variables specified by the variable list are printed immediately as they are generated. TRACE prints a complete sequence of arcs and nodes in the network as it is simulated. An entry is printed as the first action when beginning node or arc processing. The entry is similar to that produced by a PRINT command.

As an example, to trace Example 2 and know the values of BIT, DBIT, BEARING, and DBEARING, the following statement would be required:

```
TRACE BIT DBIT BEARING DBEARING
```

Each time an arc or node is processed, the above variables will be printed.

4.2 REPORT STATISTICS

Report statistics are requested with the REPORT command which has syntax

```
REPORT, variable list
```

The REPORT command is entered under GLOBAL. At the end of each simulation, the mean and standard deviation of the variables specified will be printed. The samples used to compute these statistics are the values of the variables at the end of the runs that make up the simulation. If there is only one run, then the mean will equal the value at the end of the run and the standard deviation will be zero. For more than one run, the values at the ends of the runs will be used to compute the statistics.

In Example 2, the command

```
REPORT NBITS COSTPF
```

is used to determine the total number of bits and the cost per foot for the well.

The variable TIME is added to the variable list automatically by IOSYM, so the command

```
REPORT
```

will print statistics on the end times of the simulations.

4.3 COLLECT STATISTICS

Collected statistics are requested with the COLLECT and FOR commands. FOR immediately follows COLLECT. The syntaxes are

COLLECT, variable list

FOR, list of operation names

COLLECT and FOR are entered under GLOBAL.

Output for COLLECT statistics is in two parts. The first part produces information identical to REPORT statistics for the variables indicated. TIME is automatically added to the COLLECT variable list.

An example will make the explanation of COLLECT statistics easier to understand. Suppose we have the following command:

COLLECT X

and the simulation consists of one run. During the run, X will change value from its initial value to its final value. Let this total change be ΔX . Some part of that total change will be due to each operation. The second part of COLLECT statistics prints the total change of X that occurred during each operation. It will also print the percentage that change represents of ΔX . If there is more than one run, the mean and standard deviation of the changes and the mean and standard deviation of the percentages for each run will be printed.

The FOR command may be absent, in which case COLLECT statistics are printed for all operations. If such statistics are desired for only some of the operations, include the FOR command and list the operations for which statistics are to be reported.

The total change ΔX includes only changes that occur on operations. Changes in the value of variables that occur at nodes will not be recognized by COLLECT.

4.4 IOSYM PLOTS

Plots are requested with the PLOT command whose syntax is

PLOT, plot number, plot-independent variable, list of
plot-dependent variables

Up to five plots may be requested, and up to five variables may be included in each list of dependent variables. The absolute value of the plot number must be an integer from one to five. Negative numbers indicate special user processing and are described in the final paragraph of this section.

Line printer plots are produced with the plot-independent variable on the horizontal axis. The dependent variables are plotted on the vertical axis. Only one vertical scale is used, and it will be selected so all points of all dependent variables fall within the plot.

Data points for plots are saved every DTSAV increments of TIME. DTSAV is a GASP IV variable and is discussed in subsection 5.5. If less than 1000 points are plotted, all will be plotted on a single page, and the independent variable need not be monotonically increasing. If more than 1000 points are to be plotted, IOSYM produces additional plot frames on subsequent pages. Each frame has the same scale on the horizontal axis, and no more than 1000 points will be plotted per frame. For this case, the independent variable must monotonically increase at successive points.

The user may wish to produce plots on some device other than the line printer. Subroutine UPLOD, as discussed in subsection 3.11, is provided. UPLOD is called by the subprogram that produces IOSYM plots at the end of each run if the plot number on the plot command was negative. The calling sequence for UPLOD is

```
SUBROUTINE UPLOD(NUMPLT, IVAR, IPLTNM, ITP)
DIMENSION IVAR(6)
NUMPLT -- Number of dependent variables
IVAR    -- Array with IDV's of variables. IVAR(1) is the
           independent variable. IVAR(2) to IVAR(NUMPLT + 1) are
           the dependent variables.
IPLTNM  -- Plot number, IPLTNM > 0
ITP     -- File number on which plot information is stored. ITP
           is assumed to be a sequential file.
```

IOSYM creates one plot record on ITP for each plot point. In UPLOT, the user may read the data on ITP and plot on another device. Records may be read with the following FORTRAN statement:

```
READ (ITP) INUM, T, NV, (VAL(K), K = 1, NV)
```

where

INUM = plot number

T = value of independent variable

NV = number of dependent variables in the plot

VAL = values of dependent variables.

Records of all plots are on ITP, so the user must compare IPLTNM with INUM as each record is read. Only when they are equal do T and VAL belong to plot IPLTNM. ITP should be rewound before using it. Also, the end of the file is marked by a special record. The last record has INUM = 191919 and T = 191919.

UPLOT is called once for each negative plot number specified by the user. The user requests IOSYM to call UPLOT by making the plot number negative on the PLOT command.

```
PLOT 2 TIME X Y
```

```
PLOT -4 TIME Z
```

Only a line printer plot will be produced for plot 2. UPLOT will be called for plot 4, so additional plotting can be done if desired.

4.5 GASP IV STATISTICS

Four types of GASP IV statistics can be recorded directly from an IOSYM command. That command is STAT and the syntax is as follows:

STAT, statistics type, number within type, sample value, time

The statistics type will be referred to by the name of the GASP IV subprogram that records the statistics.

<u>Statistics Type</u>	<u>GASP IV Subprogram</u>
1	COLCT
2	TIMST
3	HISTO
4	TIMSA

Each of these types will be discussed subsequently. Statistics on several variables of the same type may be recorded. The variables are numbered, and "number within type" is this value. Each execution of STAT records one sample of the variable. This is the "sample value." TIMST and TIMSA statistics require the current time, so for those types, time is usually "TIME". The time parameter is not required for COLCT and HISTO types.

As an example, if cost accrued on an operation is proportional to time, and COLCT statistics on the cost of the operation are wanted, the following commands on the operation could be given:

```
TYPE E
  COST  = OPTIME * VARCOST
  STAT 1 2 COST
END E
```

The above assumes that the cost of the operation is the second COLCT variable (number within type = 2).

The COLCT type report statistics based on observations. Each STAT command of this type records one observation of the variable. Appropriate variables to record with COLCT include the duration of telephone calls, the number of flips of a coin to obtain the first head, the amount of gasoline in a car's tank when it pulls into a service station, and the number of people in a household.

TIMST statistics record "time-persistent" variables. TIMST integrates the sample values over time assuming the value to remain constant between calls. An appropriate variable for TIMST statistics is the number of customers in a queue. The number of people in the queue is constant between times when someone enters or leaves the queue, and the average number in the queue is the total man-hours spent in the queue (integral of number in queue) divided by the total time.

TIMSA statistics are similar to TIMST except TIMSA assumes that the variable varies linearly between calls. Figure 10 illustrates the difference between TIMST and TIMSA.

Figure 10. The Difference between TIMST and TIMSA Integration

The HISTO statistics collect histogram data. At the end of each run an histogram of the sample values is printed. The number of calls and the width of the calls are specified separately through GASP IV input.

All of the statistics collected with STAT require additional information that is specified through GASP IV input. Instructions for doing this are contained in subsection 5.5. At the end of each simulation, the COLCT, TIMST, and TIMSA statistics automatically print the mean, standard deviation, standard deviation of the mean, number of observations, minimum sample, and maximum sample for each variable.

5. DATA INPUT AND RUNNING IOSYM

Thus far the IOYSM commands have been discussed without relation to the sequence in which major groups of data are input. This section will discuss the preparation of input in a form acceptable to IOSYM.

5.1 CREATING A NETWORK

The general sequence of data input for creating a new network is shown on the following page.

The NETWORK command causes IOSYM to prepare for the new network. If GASP IV-formatted, free-form input is required, it must immediately follow NETWORK. Its presence is signaled by the IOSYM command GASP. The syntax is

GASP

Control will transfer to GASP IV subprograms and GASP IV data cards will be read until the GASP command FIN or SIM is encountered. Then IOSYM input will resume. GASP IV input is discussed in subsection 5.5.

VARIABLE and TABLE commands follow next. As many of these as needed can be used, and they may appear in any order.

If needed, the command GLOBAL follows next. GLOBAL is needed if any IOSYM statistics commands or any global TYPE commands are present. Statistics commands and type S, I, and/or E groups may appear as needed and in any order. The entire GLOBAL group must be terminated with an END command.

Operations must be input next. As many operation groups as needed can be input.

Node and arc groups follow next. As many as needed may be specified, and they may appear in any order. NODE and ARC commands may be intermixed.

NETWORK	
GASP	
.	}
.	
.	
FIN or SIM	GASP IV input if required
VARIABLE	}
TABLE	
GLOBAL	as many as needed of each
TRACE	}
TABULATE	
COLLECT	
FOR	
REPORT	
PLOT	
TYPE S, I, or E	}
.	
.	
.	
END	as needed in any order
END GLOBAL	
OPERATION	}
.	
.	
END	as many groups as needed
NODE	}
.	
.	
END	
ARC	}
.	
.	
END	as many groups as needed
END NETWORK	

The end of the network must be signified with an END command. At this point, IOSYM is ready to accept commands that will cause it to simulate the network.

5.2 SIMULATING THE NETWORK

The simplest way to execute the network simulation is with the two commands

DO, n

GO, name of start node

DO specifies that n simulations are to be performed. It does not cause the simulations to begin, however. The GO command directs the simulations to occur and specifies the node at which simulation begins. So the commands

DO 1

GO START

will cause one simulation beginning at node START. All variables will have initial values specified on VAR and TABLE commands.

It is frequently desirable to start the simulation with certain variables having different initial values. The WITH command accomplishes this; WITH must immediately follow DO. Its syntax is

WITH, list of variable specifications

The "list of variable specifications" can take two forms. The first is
variable name(low value, high value)

In the n simulations specified on DO, the initial value of the named variable will be varied from the low value to the high value in equal increments. So the commands

DO 3

WITH X(5, 15) Y(0, 20)

GO START

will cause three simulations with initial X values of 5, 10, and 15 and initial Y values of 0, 10, and 20. If only one simulation is requested, then only one value may be specified. For example

```
DO 1
  WITH X(5) Y(0)
GO START
```

Internally, the initial values specified on VAR and TABLE commands are never destroyed by WITH. Initial values are simply temporarily assigned for the simulations requested. After the simulations, the network is in the same condition as before the DO command.

A second form for the list of variable specifications is
variable 1, variable 2, ...etc.

Values for the variables are not specified on the WITH command. Instead they are read prior to each simulation using the IOSYM READ command. READ has the following syntax:

```
READ, value 1, value 2, ...etc.
```

Value 1 is the value for variable 1; value 2 is the value for variable 2, etc.

The two types of variable lists may be intermixed, so a complete data record might look like this:

```
DO 3
  WITH V, W (5,15), X, Y, Z(10,10)
GO ONE
READ 1 2 3 $ VALUES FOR V, X, Y - SIMULATION 1
READ 4 5 6 $ VALUES FOR V, X, Y - SIMULATION 2
READ 7 8 9 $ VALUES FOR V, X, Y - SIMULATION 3
```

Values for the "READ variables" V, X, and Y are placed on the READ command in their order of occurrence on the WITH command. Note that Z will have the same initial value in all three simulations because the "low" and "high" values are equal. The values of the variables in each simulation are

<u>Simulation</u>	<u>V</u>	<u>W</u>	<u>X</u>	<u>Y</u>	<u>Z</u>
1	1	5	2	3	10
2	4	10	5	6	10
3	7	15	8	9	10

Only one READ command is permitted per simulation. The command can be continued to additional data cards if needed, however.

Additional GASP IV input may be desired between simulations. If the command GASP is placed after WITH (or DO if there is no WITH), then GASP IV input must be specified for each run. The sequence is

```
DO, n
  WITH . . . .
  GASP
GO, i
GASP  } GASP data for simulation 1
      }
FIN   }
READ . . . . READ command for simulation 1
GASP  }
      } GASP command for simulation 2
FIN   }
READ . . .
.
.
```

The GASP command after WITH indicates that GASP IV input is present for each simulation. Each group of GASP IV input must be preceded by a GASP command. The GASP IV input must precede the READ command for each simulation.

Also, after any GASP IV and READ information has been processed, IOSYM will call subroutine UREAD(ISIM). ISIM will be equal to the current simulation number. The user may do any needed input by writing UREAD himself and loading the program with IOSYM. UREAD data is placed after the READ command. UREAD is called once before each simulation.

Finally, it may be desirable to perform data input before each run of a simulation. Subroutine UINTLC can be used for this purpose. UINTLC is called once at the start of each run. It is called after initial node processing but before the first arc is processed. For

example, in Example 2 (see Figure 7), node START is the starting place of the simulation (i.e., of each run in the simulation). After the command GO START, all commands at node START will be executed. One of these is GOTO N 1. When node START is complete, the transfer to node 1 will be made, and any commands there will be executed. UINTLC will be called after node 1 is complete and before the arc with EJECT is begun. Thus, the user can revise standard initialization done at the node specified on the GO command. The order of data cards when performing simulations is summarized as follows.

```

DO, n
  WITH . . . .
    GASP
GO, i
GASP  }
      } GASP IV input for simulation 1
FIN   }
READ  READ command for simulation 1
      } UREAD input for simulation 1
      } UINTLC input for run 1 of simulation 1
      } UINTLC input for run 2 of simulation 1
      .
      .
      } UINTLC input for run NNRNS of simulation 1
GASP  }
      } GASP input for simulation 2
FIN   }
      .
      .
      .
etc. for each simulation

```

Several other commands are available to regulate the response of IOSYM to the input. ECHO will cause printing of data cards on the output file. NOECHO will suppress printing. The normal (default) condition is ECHO.

INFO causes IOSYM warning messages to be printed. NOINFO suppresses warnings. The default condition is INFO.

DUMP will cause a print of IOSYM's internal storage arrays at the beginning of the first simulation run. DUMP is for debug purposes and the user must be familiar with Reference 1 to interpret the resulting output. NODUMP suppresses the DUMP and is the default condition.

The conditions set by ECHO, NOECHO, INFO, NOINFO, DUMP, and NODUMP remain in effect until changed by the corresponding antithetical command. Thus, data cards are printed, for example, until a NOECHO is encountered. Then they are suppressed until an ECHO is found.

The QUIT command causes execution to terminate. IOSYM will execute a FORTRAN STOP statement when a QUIT is encountered.

None of the commands DO, WITH, GO, READ, ECHO, NOECHO, INFO, NOINFO, DUMP, NODUMP, or QUIT may appear between a NETWORK command and its associated END command.

IOSYM commands can be thought of as consisting of three modes. "Network" mode includes all commands needed to describe a network (e.g., OPERATION, GLOBAL, TERM, ARC, GOTO, etc.). Network mode is entered when a NETWORK command is found and terminates when the END command associated with NETWORK is encountered. The normal mode for IOSYM is "instruct" mode. All of the commands discussed in this section are instruct mode commands. Note that some commands, like GASP, can be entered under more than one mode.

The third mode, "change" mode, is discussed in the next section. Whenever IOSYM is not in network or change mode, it is in instruct mode.

5.3 MAKING CHANGES TO A NETWORK

Change mode is entered with the CHANGE command. The syntax is

```
CHANGE, P
  .
  .   } change commands
  .
END, CHANGE
```

If "P" is present on CHANGE, a network summary is printed after the END command. If "P" is not present, no summary is printed.

Changes made in change mode are permanent in that they become part of the network. All changes made by WITH, READ, and GASP are "temporary" since they affect only the simulations to which they refer. Any changes to GASP IV or IOSYM variables made by UREAD and UINTLC will also be temporary (of course, the user is responsible for re-initializing user-defined COMMON blocks and variables).

Change mode changes become part of the network, however. A description of commands that can be given in change mode is given below.

1. NETWORK, network name

The name of the network is changed to that specified on the command.

2. ACTIVATE, arc name

DEACTIVATE, arc name

The initial status of the named arc is changed as specified.

3. VARIABLE, name, type, initial value

The initial value of the named variable is set to that specified. If the type does not agree with the existing type, a warning is printed. Note that changing type is probably an error.

4. TABLE, name(size), initial values

The initial values of the named table are changed to those specified. The size specified must agree with the existing size.

5. COLLECT, variable list
NOCOLLECT, variable list

COLLECT replaces the current COLLECT variables with those specified. NOCOLLECT will delete the named variables from being COLLECTed. If NOCOLLECT has no variable list, no COLLECT statistics will be maintained. All FOR operations are also deleted.

6. REPORT, variable list
NOREPORT, variable list

REPORT replaces the current REPORT variables with those specified. NOREPORT will delete the named variables from the REPORT statistics. If NOREPORT has no variable list, all REPORT variables will be deleted.

7. TABULATE, n(T), list of variables and report types
NOTABULATE, n

TABULATE replaces the current tabulation n (or adds it if it does not currently exist). NOTABULATE deletes tabulation n.

8. PLOT, n, list of variables
NOPLOT, n

Plot replaces plot n or adds it if it does not exist. NOPLOT deletes plot n.

9. FOR, list of operations
NOFOR, list of operations

FOR replaces current FOR operations, if any. If FOR has no list, all operations become COLLECT operations. NOFOR deletes

the named operations. If NOFOR has no list of operations, then all operations become COLLECT operations.

- 10. ECHO
NOECHO

These commands work as usual.

- 11. TRACE, list of variables
NOTRACE, list of variables

TRACE replaces the current trace variables with those specified. If no trace is in effect, TRACE causes tracing to occur. NOTRACE deletes the named variables from the trace. NOTRACE with no variables ends all tracing.

5.4 THE SAVE AND RESTORE COMMANDS

Once network data cards have been read by IOSYM, that network is represented completely by the state or "mindset" of the IOSYM and GASP IV storage arrays. If that same network is to be used in a subsequent computer job, and if, in some way, the mindset could be restored, then the network data cards would not need to be read again. The SAVE and RESTORE commands provide this capability. The syntax for the two commands is given below.

SAVE, network name, P
RESTORE, network name, P

The SAVE command causes the entire mindset to be written to the file whose unit number is eight (TAPE8). The network name on the SAVE command is assigned to the mindset as it is written. This name need not be the same as the current name of the network. SAVE changes the internal network name to the name specified on the SAVE command regardless of what network name is specified on the NETWORK command. If "network name" is not present on the SAVE command, the current network name is used. Because the network name does not have to be present on the SAVE and RESTORE commands, the name "P" should be avoided since it might conflict with the P parameter.

The network is written to TAPE8 with unformatted FORTRAN WRITE statements. TAPE8 may contain as many mindsets as desired. Each must have a unique network name, however. If a SAVE command specifies a network name already present on TAPE8, then the network currently on TAPE8 is replaced by the mindset currently resident in core. If "P" is present on the SAVE command, a network summary will be printed after the SAVE.

The RESTORE command will cause a search of TAPE8 (the "SAVE file") for a network whose name matches the name specified on the command. The mindset is then read from the file. The internal mindset will become the same as when it was originally created on TAPE8 with a SAVE command. Any internal mindset existing at the time of the RESTORE will be destroyed. If "P" is present on RESTORE, then a network summary will be printed after the named network is read into memory.

CHANGE and SAVE/RESTORE can be used to maintain several variants of a network readily at hand. SAVE and RESTORE are instruct mode commands as the following example illustrates.

```
NETWORK, NET
  }
END NETWORK
SAVE
CHANGE
  }
END CHANGE
SAVE NET1
CHANGE
  }
END CHANGE
SAVE NET2
QUIT.
```

On subsequent computer runs, if the same TAPE8 is attached to the job, then the commands

RESTORE, NET
RESTORE, NET1
RESTORE, NET2

would restore the networks NET, NET1, or NET2, respectively.

5.5 GASP INPUT

GASP IV input will not be covered in detail here. Input for those GASP IV parameters that frequently need attention for IOSYM will be discussed. Additional capabilities are presented in Appendix C. A complete description of GASP IV variables is contained in Reference 2.

All GASP commands consist of three letters. Data fields are separated by commas and are numbered from left to right. Data field number one is the command name. Each data field has a default value which will be used if that field is skipped. Fields may be skipped by specifying the number, in parentheses, of the next data field for which data is specified. For example,

GEN, (7) 20 *

has field 1 equal to GEN, field 7 equal to 20, and all other fields defaulted. An asterisk ends GASP data cards. The "\$" sign will not be recognized as a line terminator by GASP IV.

An explanation will be given for commonly needed GASP IV commands. For the most part, the order in which these commands appear is not significant. Exceptions are noted below.

1. More than one run per simulation -- Use

GEN, (7) n *

where n is the desired number of runs. The default is one, so no GEN card is required if this is sufficient.

2. Numerical integration parameters -- GASP IV has three parameters that frequently need to be set: DTMIN, DTMAX, and DTSAV. DTMIN is the minimum step size, DTMAX is the maximum step size, and DTSAV is

the interval for collecting GASP IV and IOSYM plot data. DTMAX is less than or equal to DTSAV. The command for setting these variables is

```
CON, (6) DTMIN, DTMAX, DTSAV *
```

The default values are

```
DTSAV -- 5
```

```
DTMAX -- min (DTSAV, 1.E20)
```

```
DTMIN -- 0.01 * DTMAX
```

3. Setting parameter sets -- Recall that parameter sets are groups of four numbers. They are specified by

```
PAR, i, val1, val2, val3, val4 *
```

where i is the parameter set number and val1, val2, etc. are the values. The default for val3 = 1.E20. The others default to zero. Additional PAR cards can be included to define as many parameter sets as needed (within the restriction $1 \leq i \leq 20$).

4. Limits for the event file -- IOSYM and GASP IV both store information in blank COMMON. The GASP internal filing system is in blank COMMON as is the entire IOSYM network description. It is necessary that sufficient space be allocated for each use.

The only direct use that IOSYM makes of the GASP IV filing system is with the TERM command. TERM causes an end of operation entry to be stored in the file. This corresponds to one entry, and three words of storage are needed for this entry. GASP requires two more words, so the total length needed for GASP blank COMMON is five. Default values are sufficient to provide this storage. However, if 2 TERMS appear on the same arc, then 10 words will be needed; if 3 TERMS, then 15 words, etc. The command needed to expand this storage is

```
CON, (4) NNTRY,,,NNSET *
```

where NNTRY is the maximum number of entries which equals the maximum number of TERMS that appear on any arc (default = 1), and NNSET is the length of required GASP blank COMMON (NNSET = 5 * NNTRY; default = 5).

The three commas between NNTRY and NNSET are required.

5. GASP IV statistics -- GASP statistics of type 1 (COLCT) are specified by

COL, ICLCT, name *

where ICLCT is the "number within type" on the STAT command and "name" is an eight-character or less alphanumeric string which will be printed with the statistics. One COL card is needed for each COLCT "number within type."

Type 2 (TIMST) and type 4 (TIMSA) statistics are requested by

TIM, ISTAT, name, initial value *

where ISTAT is the "number within type," "name" is as in COL above, and "initial value" is the initial value for integration (default = 0). One TIM card is needed for each type 2 or 4 statistic. Types 2 and 4 are numbered within type together. In other words, there cannot be a number three type 2 and a number three type 4. They are physically stored in the same location. It is up to the user to correctly associate the type and number within type for TIMST and TIMSA.

Type 3 (HISTO) statistics are requested by

HIS, IHIST, name, NNCEL, HHLOW, HHWID *

where IHIST is the number within type, "name" is the same as above, NNCEL is the number of cells in the histogram (default = 10), HHLOW is the value of lower limit of first cell (default = 0), and HHWID is the width of each cell (default = 1).

6. Random number seeds -- Six random number streams are provided. Each has a default seed. These can be overridden with the following command:

SEE, seed 1, seed 2, seed 3, ... seed 6 *

where seed 1, etc. are integer values.

6. IOSYM OUTPUT

This section describes the output from IOSYM. Examples 1 and 2 will be used to illustrate the information that can be requested from the language.

6.1 NETWORK SUMMARY

IOSYM prints a summary of the network after leaving network mode. The summary cannot be suppressed and should be scrutinized when debugging a new network. In addition, a network summary can be requested with the "P" parameter on the CHANGE, SAVE, and RESTORE commands.

The network summary consists of eight parts, each of which is discussed below.

1. The concise summary lists the network identifier and the number of nodes, arcs, operations, primary variables, secondary variables, and tables. Figure 11 is the concise summary for Example 1.

2. The node summary lists the internal node number, the node name, and the number of outgoing arcs for each node. Figure 12 is the node summary for Example 1. Nodes are numbered internally in the order in which they are referenced in the data. Thus, the node whose name is "7" is internally numbered 4. If the node name "7" were passed as an accessible entity to a user function, the corresponding IACC value would equal 4. Note that nodes that are not explicitly defined on NODE commands are flagged in the node summary.

3. The arc summary lists the start node name, the user-assigned PICK number, the internal arc number, the arc name, the operation, the

end node name, and the initial status. Figure 13 is the arc summary for Example 2. The "arc number" is the value that is passed to FUNCTION U if an arc name is used as an accessible entity.

4. The operation summary is next. Figure 14 is the operation summary for Example 1. The internal operation number, the operation name, and the names of all arcs that use the operation are given.

5. The variable summary follows the arc summary. Figure 15 is the first page of the variable summary for Example 1. The variable type, the number within type, the variable name, and the initial value are given for each variable.

Note that the variables I, J, K, L, M, N, EEE, PII, and OPTIME are always the first eight secondary variables. Unlike arcs and nodes, the variable "number" in the variable summary is not the value passed in the IACC array to FUNCTION U. The IACC value is an IDV which contains the "number" but which also indicates the type. This IDV may be used directly in all of the utility programs listed in subsection 3.11 that require such a value. Any variables that are not initialized on VAR commands are assigned initial values of zero and flagged in the variable summary (e.g., DRAG).

6. The table summary for Example 1 is shown in Figure 16. The table number, name, size, and initial values are given. Four tables are always present (GV, NV, AV, and MV). Two user tables exist in EJECTN. They are ALT and DENSITY which hold the altitudes and corresponding air densities for use in menu function 20.

The tables are printed in five columns. The first row, from left to right, holds elements 1 through 5. The second row has elements 6 through 10, etc.

7. The statistics summary indicates what TABULATION, REPORT, and COLLECT statistics have been requested. Figure 17 is the statistics summary for Example 1.

NETWORK SUMMARY

NETWORK IDENTIFIER	EJECTN
NUMBER OF NODES	8
NUMBER OF ARCS	6
NUMBER OF OPERATIONS	6
NUMBER OF PRIMARY VARIABLES	8
NUMBER OF SECONDARY VARIABLES	39
NUMBER OF TABLES	6

Figure 11. Concise Summary for Example 1

NODES

NUMBER	NAME	NUMBER OF OUTGOING ARCS	
-----	-----	-----	
1	1	1	
2	2	1	
3	3	1	*NODE NOT USER DEFINED*
4	7	0	
5	4	1	
6	5	1	*NODE NOT USER DEFINED*
7	6	1	
8	8	0	*NODE NOT USER DEFINED*

Figure 12. Node Summary for Example 1

ARCS

PAGE 3

START NODE	PICK NUMBER	ARC NUMBER	ARC NAME	OPERATION	END NODE	INITIAL STATUS
-----	-----	-----	-----	-----	-----	-----
1	0	1	A	EJECT	2	ACTIVE
2	0	2	B	FREEFLY	3	ACTIVE
3	0	3	C	SNAP	4	ACTIVE
4	0	4	D	FREEFAL	5	ACTIVE
5	0	5	E	OPEN	6	ACTIVE
6	0	6	F	CHUTE	8	ACTIVE

Figure 13. Arc Summary for Example 2

OPERATIONS

NUMBER	NAME	ARC USED ON
1	EJECT	A
2	FREEFLY	B
3	SNAP	C
4	FREEFAL	D
5	OPEN	E
6	CHUTE	F

Figure 14. Operation Summary for Example 1

VARIABLES

TYPE	NUMBER	NAME	INITIAL VALUE
INDEPENDENT	0	TIME	0.
PRIMARY	1	X	0.
	2	Y	0.
	3	V	0.
	4	THETA	0.
	5	XT	0.
	6	YT	0.
	7	DELTA	0.
	8	VEELTA	0.
SECONDARY	1	I	0.
	2	J	0.
	3	K	0.
	4	L	0.
	5	M	0.
	6	N	0.
	7	EEE	2.718282
	8	PII	3.141593
	9	OPTIME	0.
	10	SURFACE	0.
	11	THETAA	-5.000000
	12	VA	400.0000
	13	THETAR	15.000000
	14	VR	45.000000
	15	HR	4.000000
	16	MP	5.000000
	17	MS	2.500000
	18	G	32.170000
	19	H	22000.00
	20	LT	35.000000
	21	HT	9.000000
	22	SSP	12.000000
	23	SP	8.000000
	24	SC	315.0000
	25	CCSP	1.000000
	26	CDP	1.000000
	27	CDC	1.000000
	28	KSNAP	200.0000
	29	DELMAX	1.000000
	30	TSEAT	1.000000
	31	TCHUTE	.50000000
	32	TAIL	0.
	33	RHO	0.
	34	XP	0.
	35	YP	0.
	36	FSNAP	0.
	37	CRAG	0.

NOT USER INITIALIZED

Figure 15. Variable Summary for Example 1

NUMEER	NAME	SIZE	INITIAL VALUES				
1	GV	50	0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
2	NV	25	0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
3	AV	50	0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
4	MV	10	0.	0.	0.	0.	0.
			0.	0.	0.	0.	0.
5	ALT	6	0.	10000.00	20000.00	30000.00	40000.00
			50000.00				
6	DENSITY	6	2.3769000E-03	1.7553000E-03	1.2664000E-03	8.8930000E-04	5.8513000E-04
			3.6180000E-04				

Figure 16. Table Summary for Example 1

EJECTA

STATISTICS

TABULATIONS

NUMBER	TIME OPTION	VARIABLE (REPORT TYPE)			
1	NO	X (END)	Y (END)	V (END)	THETA (END)

REPORT VARIABLES

TIME	X	Y	V	THETA
------	---	---	---	-------

COLLECT

THESE VARIABLES ARE COLLECTED

TIME

ON THESE OPERATIONS

EJECT	FREELY	SNAP	FREEFAL	OPEN	CHUTE
-------	--------	------	---------	------	-------

Figure 17. Statistics Summary for Example 1

8. The plot summary for Example 1 is shown in Figure 18. The plot number, independent variable, and dependent variables are shown.

6.2 TABULATIONS

Tabulations are printed for each run of the simulation. Figures 19 and 20 contain the tabulation for Example 1 and the first page of the tabulation for Example 2, respectively.

Example 1 is a "one pass" network where each arc is executed once. The tabulation has only six entries indicating the start time, end time, and duration of the operations. Also the arc name, operation name, and the values of the requested variables are shown. Example 2, on the other hand, has much more network logic that requires each operation to be performed repeatedly. The tabulation is a complete record of the sequence in which operations are performed. Note that neither nodes nor arcs that have no operations on them are listed. The tabulation presents an operation sequence. If a complete network sequence is needed, TRACE is used.

6.3 PLOTS

Standard plots from IOSYM are printed on the line printer. Figure 21 is plot 1 from Example 1, and Figure 22 is plot 3 from Example 2. At Sandia National Laboratories, a user plot program has been implemented using the commercial DISSPLA⁴ programs. The five plots from Example 1 are shown as Figures 23 through 27. The three plots from Example 2 are Figures 28 through 30.

Figure 25 is interesting in that only part of the plot is meaningful. Plot 3 is intended to show the approach distance of the pilot and seat to the tail of the aircraft during FREEFLY. The approach distance, TAIL, is not computed during EJECT, and so it has the value zero until $X = 34.73$ (see Figure 19). The plot immediately jumps up to the correct value of TAIL at the beginning of FREEFLY. After FREEFLY ends at $X = 366.8$, TAIL is no longer computed, and so it retains its last value for the rest of the simulation. Between $X = 34.73$ and $X = 366.8$, the plot shows the pilot's distance from the tail.

EJECTA

PLOTS

NUMBER	INDEPENDENT VARIABLE	DEPENDENT VARIABLES
1	X	Y
2	TIME	X
3	X	TAIL
4	X	V
5	X	THETA

Figure 18. Plot Summary for Example 1

EJECTA	TABULATION NUMBER	1 OF SIMULATION NUMBER	2 AND RUN NUMBER	1	PAGE	1			
						REPORT TYPES AND VARIABLES			
						END X	END Y	END V	END THETA
TIME									
START	END	DURATION	ARC	OPERATION					
1	0.	8.8889E-02	8.8889E-02	A	EJECT	34.73	2.2001E+04	390.8	2.4155E-02
2	8.8889E-02	1.089	1.000	B	FREEFLY	366.8	2.1994E+04	285.4	-7.3144E-02
3	1.089	1.232	.1434	C	SNAP	408.4	2.1991E+04	291.5	-8.9145E-02
4	1.232	51.80	50.57	D	FREEFAL	1972.	1.3997E+04	161.6	-1.571
5	51.80	52.30	.5000	E	OPEN	1972.	1.3917E+04	158.5	-1.571
6	52.30	658.4	606.1	F	CHUTE	1972.	-8.2139E-03	20.72	-1.571

Figure 19. Tabulation for Example 1

DRILL	TABULATION NUMBER	1 OF SIMULATION NUMBER	2 AND RUN NUMBER	1	PAGE	1			
REPORT TYPES AND VARIABLES									
TIME				END	END	END			
START	END	DURATION	ARC	OPERATION	BIT	D DEPTH	TOOTH	BEARING	
1	0.	14.15	14.15	A	DRILL	-3500.	-29.50	.4343	.3613
2	14.15	14.68	.5299	B	SURVEY	-3500.	0.	.4343	.3613
3	14.68	35.00	20.32	A	DRILL	-4002.	-20.90	.7499	.8804
4	35.00	35.60	.5979	B	SURVEY	-4002.	0.	.7499	.8804
5	35.60	40.28	4.685	A	DRILL	-4097.	-19.61	.8083	1.000
6	40.28	42.33	2.049	C	TRIPOUT	0.	0.	.8083	1.000
7	42.33	44.83	2.500	D	CHGBIT	0.	0.	0.	0.
8	44.83	46.88	2.048	E	TRIPIN	-4095.	0.	0.	0.
9	46.88	64.11	17.23	A	DRILL	-4500.	-19.74	.4920	.4401
10	64.11	64.78	.6718	B	SURVEY	-4500.	0.	.4920	.4401
11	64.78	86.71	21.92	A	DRILL	-4871.	-14.67	.8083	1.000
12	86.71	89.14	2.436	C	TRIPOUT	0.	0.	.8083	1.000
13	89.14	91.64	2.500	D	CHGBIT	0.	0.	0.	0.
14	91.64	94.08	2.436	E	TRIPIN	-4871.	0.	0.	0.
15	94.08	100.6	6.471	A	DRILL	-5000.	-18.40	.2580	.1653
16	100.6	101.3	.7520	B	SURVEY	-5000.	0.	.2580	.1653
17	101.3	134.0	32.68	A	DRILL	-5472.	-11.71	.8083	1.000
18	134.0	136.7	2.737	C	TRIPOUT	0.	0.	.8083	1.000
19	136.7	139.2	2.500	D	CHGBIT	0.	0.	0.	0.
20	139.2	142.0	2.737	E	TRIPIN	-5472.	0.	0.	0.
21	142.0	143.6	1.642	A	DRILL	-5500.	-16.59	5.1168E-02	4.1939E-02
22	143.6	144.4	.8343	B	SURVEY	-5500.	0.	5.1168E-02	4.1939E-02
23	144.4	182.0	37.51	A	DRILL	-5962.	-9.743	.8083	1.000
24	182.0	184.9	2.981	C	TRIPOUT	0.	0.	.8083	1.000
25	184.9	187.4	2.500	D	CHGBIT	0.	0.	0.	0.
26	187.4	191.4	2.982	E	TRIPIN	-5962.	0.	0.	0.
27	191.4	193.1	2.708	A	DRILL	-6000.	-13.44	.1360	6.9166E-02
28	193.1	194.1	.9307	B	SURVEY	-6000.	0.	.1360	6.9166E-02
29	194.1	230.5	36.45	A	DRILL	-6376.	-8.341	.8083	1.000
30	230.5	233.7	3.189	C	TRIPOUT	0.	0.	.8083	1.000
31	233.7	236.2	2.500	D	CHGBIT	0.	0.	0.	0.
32	236.2	239.4	3.188	E	TRIPIN	-6376.	0.	0.	0.
33	239.4	250.7	11.34	A	DRILL	-6500.	-9.879	.3765	.2896
34	250.7	251.8	1.029	B	SURVEY	-6500.	0.	.3765	.2896
35	251.8	279.6	27.82	A	DRILL	-6734.	-7.290	.8083	1.000
36	279.6	282.9	3.368	C	TRIPOUT	0.	0.	.8083	1.000
37	282.9	285.4	2.500	D	CHGBIT	0.	0.	0.	0.
38	285.4	288.8	3.367	E	TRIPIN	-6734.	0.	0.	0.
39	288.8	320.5	31.75	A	DRILL	-7000.	-6.914	.7142	.8108
40	320.5	321.7	1.133	B	SURVEY	-7000.	0.	.7142	.8108
41	321.7	329.1	7.410	A	DRILL	-7050.	-6.475	.8083	1.000
42	329.1	332.6	3.525	C	TRIPOUT	0.	0.	.8083	1.000
43	332.6	335.1	2.500	D	CHGBIT	0.	0.	0.	0.
44	335.1	338.6	3.525	E	TRIPIN	-7050.	0.	0.	0.
45	338.6	377.8	39.16	A	DRILL	-7332.	-5.823	.8083	1.000
46	377.8	381.5	3.666	C	TRIPOUT	0.	0.	.8083	1.000
47	381.5	384.0	2.500	D	CHGBIT	0.	0.	0.	0.
48	384.0	387.6	3.666	E	TRIPIN	-7332.	0.	0.	0.
49	387.6	411.5	23.85	A	DRILL	-7500.	-6.060	.6014	.6092
50	411.5	412.7	1.243	B	SURVEY	-7500.	0.	.6014	.6092

Figure 20. First Page of Tabulation for Example 2

Figure 21. Plot 1 from Example 1 -- Line Printer Plot

Figure 22. Plot 3 from Example 2 -- Line Printer Plot

Figure 23. Plot 1 -- Example 1

Figure 24. Plot 2 -- Example 1

Figure 25. Plot 3 -- Example 1

Figure 26. Plot 4 -- Example 1

Figure 27. Plot 5 -- Example 1

Figure 28. Plot 1 -- Example 2

Figure 29. Plot 3 -- Example 2

Figure 30. Plot 4 -- Example 2

Figure 28 clearly indicates the bit change times, and Figure 29 shows the survey depths (vertical lines without a sawtooth) and the bit changes (sawtooth points). From Figure 30, it can be seen that bearing wear caused each bit change.

The position of the bit (BIT) and the rate of penetration (DDEPTH) are also plotted by GASP IV through the use of SUBROUTINE USSAVE and GASP data cards (CARD 5, 6, and 7 in Appendix B). The first page of the GASP IV plot is shown in Figure 31. The use of USSAVE is discussed in more detail in Appendix C.

6.4 REPORT STATISTICS

The REPORT statistics for Example 2 are shown in Figure 32. Recall that the variable TIME is always given when REPORT statistics are requested. Since there was only one run in this simulation, all standard deviations are zero. The value of TIME at the end of the run was 524. Eleven bits were used, and the cost per foot was \$37.64.

6.5 COLLECT STATISTICS

Part one of the COLLECT statistics is shown in Figure 33 for Example 2. Part one is equivalent to REPORT statistics for the COLLECT variables. Part two is given in Figure 34. Here the amount of the change of each variable is distributed over the operations. For example, of the 524 hours spent working on the well, 423.9 were spent performing the operation DRILL. This represented 80.9% of the total time. Similarly, of the total cost of \$188,200, \$163,192 was incurred on DRILL. This represented 86.7% of the cost. Note that if COST had not been initially zero, the COLLECT statistics would reflect the percent of the change in cost that occurred during the simulation.

6.6 MULTIPLE RUNS AND SENSITIVITY ANALYSIS

In Example 2, the sensitivity of the cost per foot to rate of penetration is of interest. A simple way to do this is to use a multiplier on the ROP equation. In the listing for the network data for Example 2 in Appendix B, the variable ROPMULT is created at CARD 22 and used at CARD 105.

Figure 31. First Page of GASP IV Plot for Example 2

DRILL	REPORT STATISTICS FOR SIMULATION NUMBER 2, THERE WERE 1 RUNS IN THIS SIMULATION		
VARIABLE	MEAN	STANDARD DEVIATION	
TIME	524.001	0.	
NBITS	11.3000	0.	
COSTPF	37.6491	0.	

Figure 32. REPORT Statistics for Example 2

DRILL COLLECT STATISTICS-PART ONE FOR SIMULATION NUMBER 2, THERE WERE 1 RUNS IN THIS SIMULATION

VARIABLE	MEAN	STANDARD DEVIATION
TIME	524.001	0.
COST	188200.	0.

Figure 33. COLLECT Statistics -- Part 1, Example 2

DRILL COLLECT STATISTICS-PART TWO FOR SIMULATION NUMBER 2, THERE WERE 1 RUNS IN THIS SIMULATION PAGE 2

VARIABLE	OPERATION	AVERAGE	STANDARD DEVIATION	AVERAGE PERCENT	STANDARD DEVIATION PERCENT
TIME	DRILL	423.968	0.	80.9098	0.
	TRIPOUT	35.6557	0.	6.80450	0.
	TRIPIN	31.6525	0.	6.04055	0.
	SURVEY	7.72494	0.	1.47422	0.
	CHGBIT	25.0000	0.	4.77098	0.
COST	DRILL	163192.	0.	86.7119	0.
	TRIPOUT	8913.91	0.	4.73640	0.
	TRIPIN	7913.14	0.	4.20463	0.
	SURVEY	1931.24	0.	1.02616	0.
	CHGBIT	6250.00	0.	3.32093	0.

Figure 34. COLLECT Statistics -- Part 2, Example 2

Three simulations were performed with the commands

```
DO 3
  WITH ROPMULT (.5, 1.5)
GO START
QUIT
```

Plot, REPORT, COLLECT, and TABULATE statistics shown earlier for this example are all for simulation 2 which has ROPMULT = 1.

Before each simulation, IOSYM prints a message indicating which variables have had their initial values changed by WITH and READ commands. Figure 35 is an example.

The costs per foot for the three simulations are shown in Figure 36. The sensitivity analysis shown in Figure 36 could have been obtained with the following sequence of commands also.

```
DO 3
  WITH ROPMULT
GO START
READ .5
READ 1
READ 1.5
QUIT
```

This section concludes the discussion of IOSYM. The appendices which follow contain reference material for use in preparing IOSYM simulations and a brief tutorial on advanced GASP applications using IOSYM.

START SIMULATION NUMBER 2
RUNS IN THIS SIMULATION 1
WITH NETWORK INITIAL VALUES

EXCEPT AS FOLLOWS

RCPMULT = 1.000000

Figure 35. IOSYM Message at Start of a Simulation

Figure 36. Plot of Cost per Foot versus ROPMULT

References

1. Dave Smith, "Programmer's Manual for IOSYM," 2 vols., SAND80-7154/1, SAND80-7154/2 (Albuquerque: Sandia National Laboratories, to be published).
2. A. A. B. Pritsker, The GASP IV Simulation Language (New York: John Wiley and Sons, 1974).
3. A. T. Bourgoyne, Jr., F. S. Young, Jr., and Martin Chenevert, Applied Drilling Engineering (Dallas: Society of Petroleum Engineering of AIME).
4. DISSPLA (Display Integrated Software System Plotting Language), Vols. I and II (San Diego: Integrated Software Systems Corporation, 1975).

APPENDICES

APPENDIX A

IOSYM COMMANDS, SYNTAX, AND NETWORK SYMBOLISM

The IOSYM commands are listed below with their syntax. Optional parameters are enclosed in square brackets. Default values are shown by equal signs (e.g., initial status = A). Figure A-1 depicts the symbolism used in preparing IOSYM networks.

ACTIVATE, arc name
ARC, start node, end node [, initial status = A]
CHANGE [, P]
CLEAR
COLLECT [, variable list]
CONSTRAINT -- no syntax, not implemented yet
CROSS, termination number, crossing condition, tolerance,
expression
DEACTIVATE, arc name
DO, number of simulations
DUMP
ECHO
EDIT -- no syntax, not implemented yet
END
EVENT, event number [, parameter list]
EXIT, loop label
FALSE
FOR [, list of operations]
GASP
GET, file number, indicator, table element
GLOBAL [, list of defaults for substitutable parameters]
GO, node name
GOTO, N, node name or GOTO, A, arc name
IF, label, variable, truth condition, expression
INFO
JUMP, loop label
LABEL, label
LOOP, label, index variable, initial value, final value
[, increment = 1]

MINIMIZE -- no syntax, not implemented yet
 NETWORK, name
 NOCOLLECT [, list of variables]
 NODE, node name
 NODUMP
 NOECHO
 NOFOR [, list of operations]
 NOINFO
 NOPLOT, plot number
 NOREPORT [, list of variables]
 NOTABULATE, tabulation number
 NOTRACE [, list of variables]
 OPERATION, operation name
 OPR, operation name [, list of substitutable parameters]
 PICK, pick number [, logical operator, expression]
 PLOT, plot number, plot independent variable, list of dependent variables
 PRINT, print identifier [, list of variables]
 PUT, file number, table element
 QUIT
 READ, list of values
 REPLACE, node out, node in
 REPORT [, list of variables]
 RESTORE, network name [,P]
 SAVE [, network name, P]
 [SET,] variable name [truth condition] = expression
 STAT, statistic type, number within type, sample value [, time]
 SWITCH, arc name
 TABLE, name(length) [, initial values = 0]
 TABULATE, tabulation number [(T)], list of variables and report types
 TERM, termination number, expression
 TRACE [, list of variables]
 TRUE
 TYPE, command type
 VAR[IABLE], name, type [, initial value = 0]
 WITH, list of variable specifications

Figure A-1. IOSYM Network Symbolism

APPENDIX B

IOSYM NETWORKS AND DATA INPUT FOR EXAMPLES 1 AND 2

Figures B-1 and B-5 are the networks for Examples 1 and 2 drawn with the symbolism shown in Figure A-1. The data cards for Example 1 are shown in Figure B-2. FUNCTION U and SUBROUTINE UEVENT for this example are given in Figures B-3 and B-4, respectively.

Figure B-1. IOSYM Network for Example 1

Several notable features are demonstrated in Example 1:

1. Card 49 shows an example of an IOSYM warning message. The message is printed immediately after the card to which it pertains. A check in Appendix F will reveal that warning number 1012 means that no initial value was specified for TAIL. The value will default to zero.
2. Default substitutable parameters are given on the GLOBAL command (CARD 74). These are default values for drag coefficient and area. The defaults are used on arc A (CARD 181). All other arcs specify other variables to be used in place of the defaults (CARDS 188, 209, 220, 225, and 233).
3. FUNCTION U is used to test whether or not the pilot hits the tail of the airplane (CARDS 200 and 201). If so, node 3 is replaced by node 7 (CARD 202).

4. The EVENT command is used to change GASP variables DTMAX and DTSAV at CARD 230. These variables control the maximum step size and the frequency of taking plot points. CHUTE represents most of the simulated time and is relatively smooth, so a longer interval between plot points is recorded. This reduces computer run time.

Figure B-6 is the data input for Example 2. Figures B-7, B-8, and B-9 are FUNCTION U, SUBROUTINE UEVENT, and SUBROUTINE USSAVE for Example 2, respectively. The salient features of this simulation are

1. The GASP input (CARDS 2 to 8) includes COLCT statistics (CARDS 3 and 4) and GASP plot specifications (CARDS 5 to 7). This plot data is recorded in SUBROUTINE USSAVE which is called automatically every DTSAV time units (see Appendix C).
2. A user function (U2) with accessible parameters is used to gain access to the variable IDVs needed in USSAVE and to the node numbers needed in UEVENT. The user function is called at CARD 170. In FUNCTION U, the accessible parameters are stored in the user-defined, labeled-COMMON block, UCOM.
3. In SUBROUTINE USSAVE, the IDVs of the variables to be plotted are used with FUNCTION VVARBL to access the current values of the variables. These are used as arguments to the GASP IV plotting program GPLOT.
4. At node 3 (CARD 149) the current depth is compared to the end depth. If the final depth has been reached, node 3 is replaced by node 4. This is done by an EVENT command (CARD 152). In SUBROUTINE UEVENT, the stored node numbers are used with SUBROUTINE RREPLC to perform the replacement.
5. A user function is used to compute the rate of penetration. All required parameters are passed as arguments to FUNCTION U. FUNCTION U uses FUNCTION GGTab to access the required table values.
6. Four crossing conditions are present on arc A (CARD 193) to represent the four termination criteria for DRILL.

7. PICK commands are present on arcs B and C (CARDS 211 and 219) to select the correct operation.
8. Initialization is done at a dummy node, START (CARD 159). A GOTO command (CARD 169) at this node causes a jump into the main network to perform the simulation. The initial call to FUNCTION U to save the accessible parameters is done at this node (CARD 170).
9. GASP COLCT statistics are collected with the STAT command at CARDS 222, 224, and 226.

```

CARD 1 - NETWORK EJECTN
CARD 2 - GASP
CARD 3 - CON,(7).2,.2*
CARD 4 - FIN*
CARD 5 - $
CARD 6 - $ INPUT VARIABLES
CARD 7 - $
CARD 8 - VAR SURFACE 2 0
CARD 9 - $
CARD 10 - VAR THETAA 2 -5
CARD 11 - VAR VA 2 400
CARD 12 - VAR THETAR 2 15
CARD 13 - VAR VR 2 40
CARD 14 - VAR HR 2 4
CARD 15 - VAR MP 2 5
CARD 16 - VAR MS 2 2.5
CARD 17 - VAR G 2 32.17
CARD 18 - $
CARD 19 - VAR H 2 22000
CARD 20 - VAR LT 2 35
CARD 21 - VAR HT 2 5
CARD 22 - VAR SSP 2 12
CARD 23 - VAR SP 2 8
CARD 24 - VAR SC 2 315
CARD 25 - VAR CDSP 2 1.0
CARD 26 - $
CARD 27 - VAR CDP 2 1.0
CARD 28 - VAR CDC 2 1.0
CARD 29 - VAR KSNAP 2 200
CARD 30 - $
CARD 31 - VAR DELMAX 2 1.0
CARD 32 - $
CARD 33 - VAR TSEAT 2 1.0
CARD 34 - $
CARD 35 - VAR TCHUTE 2 .5
CARD 36 - $
CARD 37 - TABLE ALT(6) 0 10000 20000 30000,
CARD 38 - 40000 50000
CARD 39 - TABLE DENSITY(6) .0023769 .0017553,
CARD 40 - .0012664 .0008893,
CARD 41 - .0005851 .0003618
CARD 42 - $
CARD 43 - $ OUTPUT VARIABLES
CARD 44 - $
CARD 45 - VAR X 1 0
CARD 46 - VAR Y 1 0
CARD 47 - VAR V 1 0
CARD 48 - VAR THETA 1 0
CARD 49 - VAR TAIL 2
CARD 50 - $
CARD 51 - $ OTHER VARIABLES
CARD 52 - $
CARD 53 - VAR RHO 2 0
CARD 54 - VAR XT 1 0
CARD 55 - VAR YT 1 0
CARD 56 - VAR XP 2 0
CARD 57 - VAR YP 2 0
CARD 58 - VAR DELTA 1 0
CARD 59 - $
CARD 60 - VAR VDELTA 1 0
CARD 61 - $
CARD 62 - $
CARD 63 - VAR FSNAP 2 0
CARD 64 - $
CARD 65 - $
CARD 66 - $
CARD 67 - $ USE OF GV TABLE
CARD 68 - $
CARD 69 - $ GV(1)- COS(THETAA)
CARD 70 - $ (2)- SIN(THETAA)
CARD 71 - $ (3)- VA*GV(1)
CARD 72 - $ (4)- VA*GV(2)
CARD 73 - $
CARD 74 - GLOBAL,1.0,10
CARD 75 - $
CARD 76 - $
CARD 77 - $ STATISTICS
CARD 78 - $
CARD 79 - $ REPORT X Y V THETA

```

```

$ SAMPLE PROBLEM 1
$ ALTITUDE ABOVE MSL OF GROUND-
$ FT
$ FLIGHT ANGLE-DEGREES
$ AIRCRAFT VELOCITY-FT/SEC
$ ANGLE OF RAIL-DEGREES
$ VELOCITY OF RAIL-FT/SEC
$ LENGTH OF RAIL-FT
$ MASS OF PILOT-SLUGS
$ MASS OF SEAT-SLUGS
$ GRAVITATIONAL CONSTANT-FT/SEC
$ **2
$ INITIAL ALTITUDE-FT(MSL)
$ DISTANCE FROM PILOT TO TAIL-FT
$ HEIGHT OF TAIL-FT
$ AREA OF PILOT AND SEAT-FT**2
$ AREA OF PILOT-FT**2
$ AREA OF CHUTE-FT**2
$ DRAG COEFFICIENT OF PILOT AND
$ SEAT
$ DRAG COEFFICIENT OF PILOT
$ DRAG COEFFICIENT OF CHUTE
$ SPRING COEFFICIENT OF SEAT
$ SNAPPER-LE/FT
$ DISTANCE THROUGH WHICH SNAPPER
$ ACTS-FT
$ TIME FROM EJECTION TO SEAT
$ SEPARATION-SEC
$ CHUTE OPENING TIME-SEC
$ ALTITUDES FOR DENSITY TABLE
$ FUNCTION-FT(MSL)
$ AIR DENSITIES FOR TABLE
$ FUNCTION-SLUGS/FT**3

```

WARNING 1C12 IN SUBROUTINE

```

$ AIR DENSITY-SLUGS/FT**3
$ X POSITION OF TIP OF TAIL-FT
$ Y POSITION OF TIP OF TAIL-FT
$ X POSITION OF BASE OF TAIL-FT
$ Y POSITION OF BASE OF TAIL-FT
$ DISTANCE BETWEEN PILOT AND
$ SEAT DURING SNAP-FT
$ VELOCITY OF PILOT AND SEAT
$ SEPARATION DURING SNAP-
$ FT/SEC.
$ FORCE OF SNAPPER-LE

```

```

$ PARAMETERS ARE DRAG COEFFICIENT
$ AND AREA

```

Figure B-2. Data Cards for Example 1


```

CARD 80 - TABULATE 1 X(E) Y(E) V(E) THETA(E)
CARD 81 - PLOT -1 X Y
CARD 82 - PLOT -2 TIME X
CARD 83 - PLOT -3 X TAIL
CARD 84 - PLOT -4 X V
CARD 85 - PLOT -5 X THETA
CARD 86 - COLLECT
CARD 87 - $
CARD 88 - TYPE I
CARD 89 - RHO=F20(Y/ALT,DENSITY) $ TABLE FUNCTION FOR DENSITY
CARD 90 - DRAG=.5*DUMM(1)*RHO*V**2*DUMM(2) $ DRAG USES DUMMY CD AND AREA
CARD 91 - DX=V*COS(THETA)
CARD 92 - DY=V*SIN(THETA)
CARD 93 - END I
CARD 94 - END GLOBAL
CARD 95 - $
CARD 96 - $ OPERATIONS
CARD 97 - $
CARD 98 - OPERATION EJECT
CARD 99 - TYPE S
CARD 100 - GV(1)=COS(THETAA)
CARD 101 - GV(2)=SIN(THETAA)
CARD 102 - GV(3)=VA*GV(1) $ CALCULATE DXT AND CYT. SAVE
CARD 103 - GV(4)=VA*GV(2) $ IN GV(3) AND GV(4)
CARD 104 - AV(1)=GV(3)-VR*SIN(THETAA+THETAR)
CARD 105 - AV(2)=GV(4)+VR*COS(THETAA+THETAR)
CARD 106 - $ CALCULATE VALUES FOR DX AND
CARD 107 - $ DY. STORE IN AV(1) AND AV(2)
CARD 108 - V=SQRT(AV(1)**2+AV(2)**2)
CARD 109 - IF S1 1.E-8 GE ABS(AV(1))
CARD 110 - TRUE
CARD 111 - AV(1)=0
CARD 112 - THETA=PII
CARD 113 - FALSE
CARD 114 - THETA=ATAN(AV(2)/AV(1))
CARD 115 - LABEL S1
CARD 116 - AV(3)=HR/VR $ CALCULATE END TIME FOR EJECT
CARD 117 - XT=-LT*GV(1)-HT*GV(2) $ INITIALIZE XT AND YT
CARD 118 - YT=Y-LT*GV(2)+HT*GV(1)
CARD 119 - END S
CARD 120 - TYPE I
CARD 121 - DX=AV(1) $ OVERRIDE DX AND DY FROM GLOBAL
CARD 122 - DY=AV(2)
CARD 123 - DXT=GV(3)
CARD 124 - DYT=GV(4)
CARD 125 - END I
CARD 126 - END OP
CARD 127 - $
CARD 128 - OPERATION FREEFLY
CARD 129 - TYPE I
CARD 130 - DV=-DRAG/(MP*MS)-G*SIN(THETA)
CARD 131 - DTHETA=-G*COS(THETA)/V
CARD 132 - DXT=GV(3)
CARD 133 - DYT=GV(4)
CARD 134 - XP=XT+HT*GV(2) $ CALCULATE POSITION OF BASE OF
CARD 135 - YP=YT-HT*GV(1) $ TAIL
CARD 136 - TAIL=SQRT((X-XT)**2+(Y-YT)**2) $ CALCULATE DISTANCE TO TAIL
CARD 137 - END I
CARD 138 - END CP
CARD 139 - $
CARD 140 - OPERATION SNAP
CARD 141 - TYPE I
CARD 142 - FSNAP=KSNAP*(CELMAX-DELTA) $ FORCE OF SNAPPER
CARD 143 - CV=-DRAG/(MP*MS)-G*SIN(THETA)+FSNAP*
CARD 144 - COS(THETAR+THETAA-THETA)
CARD 145 - DTHETA=-G*COS(THETA)/V+FSNAP*
CARD 146 - SIN(THETAR+THETAA-THETA)/V
CARD 147 - DDELTA=VDELTA
CARD 148 - DVDELTA=FSNAP*(1/MS+1/MP)
CARD 149 - END I
CARD 150 - END OP
CARD 151 - $
CARD 152 - OPERATION FREEFAL
CARD 153 - TYPE I
CARD 154 - DV=-DRAG/MP-G*SIN(THETA)
CARD 155 - DTHETA=-G*COS(THETA)/V
CARD 156 - END I
CARD 157 - END OP
CARD 158 - $

```

Figure B-2 (Continued)

```

CARD 159 - OPERATION OPEN
CARD 160 - TYPE I
CARD 161 - DRAG=1.2*DRAG
CARD 162 - DV=-DRAG/MP-G*SIN(THETA)
CARD 163 - DTHETA=-G*COS(THETA)/V
CARD 164 - ENC I
CARD 165 - END GP
CARD 166 - $
CARD 167 - OPERATION CHUTE
CARD 168 - TYPE I
CARD 169 - DV=-DRAG/MP-G*SIN(THETA)
CARD 170 - DTHETA=-G*COS(THETA)/V
CARD 171 - ENC I
CARD 172 - END GP
CARD 173 - $
CARD 174 - $
CARD 175 - NODE 1
CARD 176 - Y=H $ INITIALIZE Y,V,AND THETA
CARD 177 - THETAA=THETAA*PII/180 $ CONVERT TO RADIANS
CARD 178 - THETAR=THETAR*PII/180
CARD 179 - END
CARD 180 - ARC A 1 2
CARD 181 - OPR EJECT
CARD 182 - TERM 1 TIME+AV(3)
CARD 183 - END
CARD 184 - END ARC
CARD 185 - NODE 2
CARD 186 - END
CARD 187 - ARC B 2 3
CARD 188 - OPR FREEFLY,CDSP,SSP $ PASS CORRECT CD AND AREA
CARD 189 - TYPE I
CARD 190 - IF B1 EQ THETAA $ COMPUTE TEST VALUE FOR HITTING
CARD 191 - TRUE $ TAIL. CHECK FOR THETAA=0
CARD 192 - TEST=X-XT
CARD 193 - FALSE
CARD 194 - NV(1)=(YT-YP)/(XT-XP)
CARD 195 - TEST=Y-NV(1)*X-YT+NV(1)*XT
CARD 196 - LABEL B1
CARD 197 - END I
CARD 198 - TERM 1 TIME+TSEAT
CARD 199 - END
CARD 200 - CRGSS 2 LE 1.1 TEST* $ USE USER FUNCTION
CARD 201 - UI(X,Y,XT,YT,XP,YP,HT)+.1 $ TO TEST IF HIT TAIL
CARD 202 - REPLACE 3 7 $ REPLACE NCCE 3 IF HIT TAIL
CARD 203 - END
CARD 204 - END ARC
CARD 205 - NODE 7
CARD 206 - PRINT 1 TAIL V $ PRINT IF HIT TAIL
CARD 207 - END
CARD 208 - ARC C 3 4
CARD 209 - OPR SNAP,CDSP,SSP
CARD 210 - CROSS 1 EQ .01 DELTA-DELMAX $ CHECK ON MAX EXTENSION OF
CARD 211 - END $ SNAPPER
CARD 212 - END ARC
CARD 213 - NODE 4
CARD 214 - IF 41 Y LE 14000 $ IF LESS THAN 14000 FT, OPEN
CARD 215 - TRUE $ CHUTE IMMEDIATELY
CARD 216 - GOTO N 5
CARD 217 - LABEL 41
CARD 218 - END
CARD 219 - ARC D 4 5
CARD 220 - OPR FREEFAL,CDP,SP
CARD 221 - CROSS 1 LE 100 Y-14000 $ OPEN CHUTE WHEN BELOW 14000 FT
CARD 222 - END
CARD 223 - END ARC
CARD 224 - ARC E 5 6
CARD 225 - OPR OPEN,CDP,SP
CARD 226 - TERM 1 TIME+TCHUTE
CARD 227 - END
CARD 228 - END ARC
CARD 229 - NODE 6
CARD 230 - EVENT 1 10 $ RESET DTMAX AND DTSV TO 10
CARD 231 - END
CARD 232 - ARC F 6 8
CARD 233 - OPR CHUTE,CDG,SC
CARD 234 - CROSS 1 LE 1 Y-SURFACE $ TERMINATE WHEN PILOT LANDS
CARD 235 - END
CARD 236 - END ARC
CARD 237 - END NETWORK

```

Figure B-2 (Concluded)

```

C
C
C      X(1) - X
C      (2) - Y
C      (3) - XT
C      (4) - YT
C      (5) - XP
C      (6) - YP
C      (7) - HT
C
C      IF SQRT( (X-XT)**2+ (Y-YT)**2 ) .LE. HT .AND.
C        SQRT( (X-XP)**2+ (Y-YP)**2 ) .LE. HT
C
C        THEN U=1.  OTHERWISE U=0
C
C      U=0.
C      IF(SQRT((X(1)-X(3))**2+(X(2)-X(4))**2) .LE.X(7) .AND.
C      -  SQRT((X(1)-X(5))**2+(X(2)-X(6))**2) .LE.X(7)) U=1.
C      RETURN
C      END

```

Figure B-3. FUNCTION U for Example 1

```

SUBROUTINE UEVENT(IV,NP,PAR)
DIMENSION PAR(NP)
COMMON/GCOM3/ Z(11)
EQUIVALENCE (DTMAX,Z(2)),(DTSAV,Z(4))
C
C      SET DTMAX AND DTSAV TO PAR(1)
C
C      DTMAX=PAR(1)
C      DTSAV=PAR(1)
C      RETURN
C      END

```

Figure B-4. SUBROUTINE UEVENT for Example 1

Figure B-5. IOSYM Network for Example 2

CARD	1 -	NETWORK DRILL			\$ EXAMPLE 2
CARD	2 -	GASP			
CARD	3 -	STA,3*			
CARD	4 -	COL,1,ROP,2,FOOTAGE,3,LIFE*			
CARD	5 -	PLO,1,TIME,12,2,0*			
CARD	6 -	VAR,1,1,B,BITPOS,2,2,1000,1000*			
CARD	7 -	VAR,1,2,R,ROP,2,2,10,10*			
CARD	8 -	FIN*			
CARD	9 -	\$			
CARD	10 -	\$ INPUT VARIABLES			
CARD	11 -	\$			
CARD	12 -	VAR Z1	2	-3000	\$ INITIAL DEPTH-FT
CARD	13 -	VAR Z2	2	-8000	\$ FINAL DEPTH-FT
CARD	14 -	VAR BITCOST	2	5200	\$ BIT COST-\$
CARD	15 -	VAR BITDIA	2	12.25	\$ BIT DIAMETER-IN
CARD	16 -	VAR RPM	2	60	\$ RPM
CARD	17 -	VAR RIGRATE	2	250	\$ RENTAL RATE FOR DRILLING RIG
CARD	18 -				\$ -\$/HR
CARD	19 -	VAR SURVINT	2	500	\$ DEPTH INTERVAL FOR DIRECTIONAL
CARD	20 -				\$ SURVEYS-FT
CARD	21 -	VAR WOB	2	60	\$ WEIGHT ON BIT-1000 LBS
CARD	22 -	VAR ROPMULT	2	1	\$ RATE OF PENETRATION
CARD	23 -				\$ MULTIPLIER
CARD	24 -	VAR TRIPRT	2	.5	\$ TRIPING RATE-HR/1000 FT
CARD	25 -	\$			
CARD	26 -	\$ OUTPUT VARIABLES			
CARD	27 -	\$			
CARD	28 -	VAR COST	1	0	\$ TOTAL COST-\$
CARD	29 -	VAR COSTPF	2	0	\$ COST PER FOOT-\$/FT
CARD	30 -	VAR NBITS	2	0	\$ NUMBER OF BITS USED
CARD	31 -	\$			
CARD	32 -	\$ OTHER VARIABLES			
CARD	33 -	\$			
CARD	34 -	VAR DEPTH	1	0	\$ DEPTH OF WELL-FT
CARD	35 -	VAR BIT	1	0	\$ POSITION OF BIT-FT
CARD	36 -	VAR TOOTH	1	0	\$ FRACTION OF BIT TEETH WORN
CARD	37 -	VAR BEARING	1	0	\$ FRACTION OF BIT BEARINGS WORN
CARD	38 -	VAR NDEPTH	2	0	\$ NEGATIVE OF DEPTH
CARD	39 -	TABLE A(12),			\$ CONSTANTS FOR ROP EQUATION
CARD	40 -			26.1,	\$ 1-DRILLABILITY OF ROCK
CARD	41 -			.00007,	\$ 2
CARD	42 -			.00005,	\$ 3
CARD	43 -			.0002,	\$ 4
CARD	44 -			1.0,	\$ 5
CARD	45 -			.5,	\$ 6
CARD	46 -			.5,	\$ 7
CARD	47 -			.3,	\$ 8
CARD	48 -			12,	\$ 9-PORE PRESSURE GRADIENT-LBS/
CARD	49 -				\$ GAL
CARD	50 -			12.5,	\$ 10-EQUIVALENT CIRCULATING
CARD	51 -				\$ DENSITY-LBS/GAL
CARD	52 -			1200,	\$ 11-HYDRAULIC IMPACT FORCE-LBS
CARD	53 -			0	\$ 12-THRESHOLD BIT WEIGHT-KLBS
CARD	54 -	TABLE B(2),			\$ CONSTANTS FOR BEARING EQUATION
CARD	55 -			2.8,	\$ 1
CARD	56 -			41.42	\$ 2-BEARING CONSTANT
CARD	57 -	TABLE H(5),			\$ CONSTANTS FOR TOOTH EQUATION
CARD	58 -			1.84,	\$ 1
CARD	59 -			6,	\$ 2
CARD	60 -			.8,	\$ 3
CARD	61 -			8,	\$ 4-MAX(WOB/BITDIA)-KLBS/IN
CARD	62 -			22.8	\$ 5-FORMATION ABRASIVENESS
CARD	63 -				\$ CONSTANT-HRS
CARD	64 -	\$			
CARD	65 -	\$ USE OF THE GV TABLE			

Figure B-6. Data Cards for Example 2

```

CARD 66 - $
CARD 67 - $   GV(1)  - =1 IF INSTALLING A NEW BIT
CARD 68 - $           =0 IF DOING A SURVEY
CARD 69 - $   (2)  - RATE OF BEARING WEAR
CARD 70 - $   (3)  - CONSTANT IN TOOTH WEAR EQUATION
CARD 71 - $   (4)  - NEXT SURVEY DEPTH
CARD 72 - $   (5)  - DEPTH AT START OF BIT RUN
CARD 73 - $   (6)  - ACCUMULATED TIME A BIT IS RUN
CARD 74 - $
CARD 75 - $
CARD 76 - $   GLOBAL
CARD 77 - $
CARD 78 - $   STATISTICS
CARD 79 - $
CARD 80 - $   REPORT NBITS,COSTPF
CARD 81 - $   COLLECT COST
CARD 82 - $   PLOT -1 TIME BIT
CARD 83 - $   PLOT -3 NDEPTH DDEPTH
CARD 84 - $   PLOT -4 NDEPTH TOOTH BEARING
CARD 85 - $   TABULATE 1 BIT(E) DDEPTH(E) TOOTH(E) BEARING(E)
CARD 86 - $
CARD 87 - $
CARD 88 - $   TYPE I
CARD 89 - $       NDEPTH=-DEPTH
CARD 90 - $       DCOST=RIGRATE
CARD 91 - $       AV(4)=ABS(DEPTH-21)
CARD 92 - $       COSTPF=COST/AMAX1(1,AV(4))
CARD 93 - $       END I
CARD 94 - $   END GLOBAL
CARD 95 - $
CARD 96 - $   OPERATIONS
CARD 97 - $
CARD 98 - $
CARD 99 - $   OPERATION DRILL
CARD 100 - $   TYPE S
CARD 101 - $       COST=COST+BITCOST*GV(1)
CARD 102 - $       GV(1)=0
CARD 103 - $       END S
CARD 104 - $   TYPE I
CARD 105 - $       DDEPTH=-U1(NDEPTH,WOB,BITDIA,RPM,TOOTH/A)*ROPMULT
CARD 106 - $                                           $ USER FUNCTION SETS ROP
CARD 107 - $       DBIT=DDEPTH
CARD 108 - $       DTOOTH=GV(3)/((1+H(2))*TOOTH)
CARD 109 - $       DBEARING=GV(2)
CARD 110 - $       END I
CARD 111 - $   END OP
CARD 112 - $
CARD 113 - $   OPERATION TRIPOUT
CARD 114 - $   TYPE I
CARD 115 - $       DBIT=F7(1000,TRIPRT)
CARD 116 - $       END I
CARD 117 - $   TYPE E
CARD 118 - $       BIT=0
CARD 119 - $       END E
CARD 120 - $   END OP
CARD 121 - $   OPERATION TRIPIN
CARD 122 - $   TYPE I
CARD 123 - $       DBIT=-F7(1000,TRIPRT)
CARD 124 - $       END I
CARD 125 - $   TYPE E
CARD 126 - $       BIT=DEPTH
CARD 127 - $       END E
CARD 128 - $   END OP
CARD 129 - $
CARD 130 - $   OPERATION SURVEY
CARD 131 - $   TYPE S

```

Figure B-6 (Continued)

```

CARD 132 -      GV(4)=GV(4)-SURVINT      $ SET NEXT SURVEY DEPTH
CARD 133 -      END S
CARD 134 -      END OP
CARD 135 - $
CARD 136 -      OPERATION CHGBIT
CARD 137 -      TYPE S
CARD 138 -      NBITS=NBITS+1            $ INCREMENT BIT COUNTER
CARD 139 -      GV(1)=1
CARD 140 -      TOOTH=0                  $ ZERO TOOTH AND BEARING WEAR
CARD 141 -      BEARING=0
CARD 142 -      GV(5)=DEPTH              $ INITIALIZE STATISTICS FOR THIS
CARD 143 -      GV(6)=0                  $ BIT
CARD 144 -      END S
CARD 145 -      END OP
CARD 146 - $
CARD 147 - $      NODES
CARD 148 - $
CARD 149 -      NODE 3
CARD 150 -      IF N31 DEPTH LE Z2      $ IF FINISHED, EVENT COMMAND
CARD 151 -      TRUE                    $ REPLACES NODE 3 WITH NODE 4
CARD 152 -      EVENT 1
CARD 153 -      LABEL N31
CARD 154 -      END
CARD 155 - $
CARD 156 -      NODE 4
CARD 157 -      END
CARD 158 - $
CARD 159 -      NODE START
CARD 160 -      DEPTH=Z1
CARD 161 -      BIT=Z1
CARD 162 -      NBITS=1
CARD 163 -      NDEPTH=-DEPTH
CARD 164 -      GV(1)=1                  $ INITIALIZE GV TABLE
CARD 165 -      GV(2)=(WOB/(4*BITDIA))*B(1)*RPM/B(2)/100 $ BEARING CONSTANT
CARD 166 -      GV(3)=H(3)/H(5)*(RPM/100)**H(1)*(H(4)-4)/ $ TOOTH WEAR CONSTANT
CARD 167 -      (H(4)-WOB/BITDIA)*(1+H(2)/2)
CARD 168 -      GV(4)=Z1-SURVINT        $ FIRST SURVEY DEPTH
CARD 169 -      GOTO N 1
CARD 170 -      AV(1)=U2(//3,4,
CARD 171 -
CARD 172 -
CARD 173 -
CARD 174 -
CARD 175 -      TIME,BIT,DDEPTH)
CARD 176 -
CARD 177 -      GV(5)=DEPTH *
CARD 178 -      GV(6)=0
CARD 179 -      END
CARD 180 - $
CARD 181 -      NODE 2
CARD 182 -      IF N21 DEPTH EQ Z2
CARD 183 -      TRUE
CARD 184 -      GOTO A C
CARD 185 -      LABEL N21
CARD 186 -      END
CARD 187 - $
CARD 188 -      NODE 5
CARD 189 -      END
CARD 190 - $
CARD 191 - $      ARCS
CARD 192 - $
CARD 193 -      ARC A 1 2 A
CARD 194 -      OPR DRILL
CARD 195 -      TYPE E
CARD 196 -      GV(6)=GV(6)+OPTIME      $ ACCUMULATE TIME FOR THIS BIT
CARD 197 -      END E

```

Figure B-6 (Continued)

CARD	198 -	CROSS 4 LE 5 DEPTH-GV(4)	\$ DETECT SURVEY DEPTH
CARD	199 -	DEPTH=GV(4)	
CARD	200 -	END	
CARD	201 -	CROSS 3 EQ .01 TOOTH-1	\$ DETECT WORN TEETH
CARD	202 -	END	
CARD	203 -	CROSS 2 EQ .01 BEARING-1	\$ DETECT WORN BEARING
CARD	204 -	END	
CARD	205 -	CROSS 1 LE 5 NDEPTH-Z2	\$ DETECT FINAL DEPTH
CARD	206 -	DEPTH=Z2	
CARD	207 -	END	
CARD	208 -	END ARC	
CARD	209 -	\$	
CARD	210 -	ARC B 2 1	
CARD	211 -	PICK 1 EQ DEPTH-GV(4)	\$ SELECT THIS ARC IF A SURVEY
CARD	212 -	OPR SURVEY	\$ DEPTH IS REACHED
CARD	213 -	TERM 1 TIME+(.00125*NDEPTH+15+1.25*	\$ COMPUTE SURVEY TIME
CARD	214 -	(NDEPTH/1000)*+1.8326)/60	
CARD	215 -	END	
CARD	216 -	END ARC	
CARD	217 -	\$	
CARD	218 -	ARC C 2 3 A	
CARD	219 -	PICK 2	
CARD	220 -	OPR TRIPOUT	
CARD	221 -	TYPE S	
CARD	222 -	STAT 1 3 GV(6)	\$ RECORD BIT LIFE STATISTICS
CARD	223 -	AV(1)=GV(5)-DEPTH	
CARD	224 -	STAT 1 2 AV(1)	\$ RECORD FOOTAGE STASTICS
CARD	225 -	AV(1)=AV(1)/GV(6)	
CARD	226 -	STAT 1 1 AV(1)	\$ RECORD AVERAGE ROP STATISTICS
CARD	227 -	END S	
CARD	228 -	CROSS 1 GE 1 BIT	
CARD	229 -	END	
CARD	230 -	END ARC	
CARD	231 -	\$	
CARD	232 -	ARC D 3 5	
CARD	233 -	OPR CHGBIT	
CARD	234 -	TERM 1 TIME+2.5	
CARD	235 -	END	
CARD	236 -	END ARC	
CARD	237 -	\$	
CARD	238 -	ARC E 5 1	
CARD	239 -	OPR TRIPIN	
CARD	240 -	CROSS 1 LE 1 BIT-DEPTH	
CARD	241 -	END	
CARD	242 -	END ARC	
CARD	243 -	\$	
CARD	244 -	END NETWORK	

Figure B-6 (Concluded)

```

      FUNCTION U(IFF,NX,X,NT,ITAB,NP,IPAR,NA,IAC)
      DIMENSION X(NX),ITAB(NT),IPAR(NP),IAC(NA)
      DIMENSION F(8)
      COMMON/UCOM/ ISAVE(5)

C
C      SELECT FUNCTION
C
      IF(IFF.EQ.2) GO TO 200

C      COMPUTE RATE OF PENETRATION
C
      X(1)=NDEPTH
      (2)=WOB
      (3)=BITDIA
      (4)=RPM
      (5)=TOOTH

      ITAB(1)=A

      ITB=ITAB(1)
      F(1)=GGTAB(ITB,1)
      F(2)=EXP(2.303*GGTAB(ITB,2)*(10000.-X(1)))
      F(3)=EXP(2.303*GGTAB(ITB,3)*X(1)*(1.69)+(GGTAB(ITB,9)-9.))
      F(4)=EXP(2.303*GGTAB(ITB,4)*X(1)*(GGTAB(ITB,9)-GGTAB(ITB,10)))
      F(5)=((X(2)/X(3)-GGTAB(ITB,12))/(4.-GGTAB(ITB,12)))*GGTAB(ITB,5)
      F(6)=(X(4)/100.)*GGTAB(ITB,6)
      F(7)=EXP(-GGTAB(ITB,7)*X(5))
      F(8)=(GGTAB(ITB,11)/1000.)*GGTAB(ITB,8)
      U=1.
      DO 10 I=1,8
10  U=U*F(I)
      GO TO 500

C      SAVE THE ACCESSIBLE PARAMETERS
C
      200 DO 210 I=1,5
      210 ISAVE(I)=IAC(I)
      U=1.
      GO TO 500

C
500 RETURN
      END

```

Figure B-7. FUNCTION U for Example 2

```

      SUBROUTINE UEVENT(IV,NP,PAR)
      DIMENSION PAR(NP)
      COMMON/UCOM/ ISAVE(5)

C
C      ISAVE(1) - NODE 3
C      (2) - NODE 4
C
C      REPLACE NODE 3 WITH NODE 4
C
      CALL RREPLC(ISAVE(1),ISAVE(2))
      RETURN
      END

```

Figure B-8. SUBROUTINE UEVENT for Example 2

```

      SUBROUTINE USSAVE
      DIMENSION VAL(2)
      COMMON/UCOM/ ISAVE(5)

C
C      ISAVE(3) - TIME
C      (4) - BIT
C      (5) - DDEPTH
C
C      OBTAIN THE VALUES OF THE VARIABLES WHOSE IIDS ARE STORED
C      IN ISAVE
C
      T=VVARBL(1,ISAVE(3))
      VAL(1)=VVARBL(1,ISAVE(4))
      VAL(2)=VVARBL(1,ISAVE(5))

C
C      CALL GPLOT
C
      CALL GPLOT(VAL,T,1)
      RETURN
      END

```

Figure B-9. SUBROUTINE USSAVE for Example 2

APPENDIX C

ADVANCED GASP IV APPLICATIONS

This appendix assumes that the user is familiar with the GASP IV simulation language and desires to use its capabilities in addition to those incorporated in IOSYM. It is also assumed here that the user is familiar with GASP IV free-form input so he/she is aware of the input changes needed to use the features described below.

C.1 EVENT CODE DEFINITION AND ATTRIBUTES

IOSYM TERM and CROSS commands are implemented as time and state events, respectively. All state events must be denoted by event code = 1 (IIEVT = 1), and all end of arc events are denoted by event code = 2. End of arc events require three attributes, so NNATR must be greater than or equal to three. Furthermore, the event file must exist and there must be room for at least one entry in NSET/QSET.

The user may create his own time and state events as will be discussed subsequently, but he must adhere to the above limits and conventions.

One other issue deserves mention here. Each TERM command on an arc causes an entry to be filed in the event file. Enough space is available by default (NNSET = 5) for a single entry. If any arc in the network has more than one TERM, then the variables NNSET and NNTRY must be increased through GASP input to provide the extra space.

C.2 THE PUT AND GET COMMANDS

Two commands are available for directly accessing the GASP IV filing system. The PUT command files an entry into a GASP file. The syntax is

PUT, file number, table(n)

The PUT command will file the contents of table(n) to table(n + NNATR - 1) into the specified file. For example,

```
PUT 1 GV(3)
```

```
PUT J AV(I)
```

Only the event file is available by default. Other files will be created automatically by IOSYM if the file number on PUT is numeric and greater than one. An error will result if a variable file number references a nonexistent file during execution. The contents of the table are not disturbed by the PUT procedure.

The GET command will retrieve the first entry from a GASP IV file. The syntax is

```
GET, file number, indicator, table(n)
```

If at least one entry exists in the named file, the first entry will be stored in table(n) to table(n + NNATR - 1), and the indicator will be set to one. If no entry exists, table will be undisturbed and the indicator will be set to zero. Clearly, the indicator must be a variable. Examples are

```
GET 1 I GV(J)
```

```
GET GV(1) GV(2) GV(3)
```

An error will result if insufficient space is available in the table to hold NNATR attributes.

C.3 IOSYM ANALOGS TO USER-WRITTEN GASP SUBPROGRAMS

The names and some of the calling sequences of GASP user-written subprograms are changed. These are described below.

SUBROUTINE USTATE -- IOSYM automatically updates the DD variables that are known to it through the IOSYM definition of primary variables. The user may create additional DD equations by specifying the variable NNEQD to be greater than the number of IOSYM primary variables. Each time GASP calls subroutine STATE, IOSYM will call subroutine USTATE so the user can update his own primary variables. USTATE is called after all IOSYM derivative variables have been set.

SUBROUTINE USSAVE-- IOSYM plot data is saved every DTSAV time units just as in GASP. This function is performed by IOSYM in subroutine SSAVE. Each time SSAVE is called by GASP, IOSYM will call USSAVE so the user can produce GPLOT plots. See Example 2.

SUBROUTINE USCOND(IFLG, NFLG) -- Subroutine USCOND is analogous to GASP subroutine SCOND, but additional parameters are required. IOSYM tests for crossing conditions on an arc that are specified by the CROSS command. As many crossing conditions exist as there are CROSS commands on an arc. If there are k CROSS commands, then IOSYM uses the first k LFLAGS to test for them. USCOND will be called with IFLG = k + 1. In other words, IFLG is the number of the first LFLAG available to the user.

USCOND is called with NFLG = 0. It must be returned as the number of LFLAGS used in USCOND. In other words, if the user is testing for m state condition, NFLAG must equal m on return.

SUBROUTINE UEVNTS (IX, IFLG, IEND) -- Subroutine UEVNTS will be called when a user event occurs. IX is the event code. IFLG is the same parameter as for USCOND and provides the user with the information needed to decode user state events. IEND is sent as zero. If the user wishes this event to end the arc, return IEND nonzero. The value of IEND may be used as an input to subroutine UTERM described below.

SUBROUTINE UTERM (IEND) -- If IEND $\neq$ 0 from UEVNTS, the arc will be terminated, and UTERM will be called to permit the user to perform actions analogous to auxiliary commands on a TERM or CROSS command. IEND will have the value assigned in subroutine UEVNTS.

C.4 NEW GASP ERROR CODES

Five new GASP IV error codes have been created to enforce the requirements of IOSYM. They are given below.

168 -- NNFIL must be ≥ 1
172 -- NNTRY must be ≥ 1
174 -- NNATR must be ≥ 3
175 -- NNSET must be ≥ 5
195 -- IIEVT must = 1

APPENDIX D

LIBRARY, PROBABILITY, AND MENU FUNCTIONS

D.1 LIBRARY FUNCTIONS

The following functions can be referenced directly in arithmetic expressions:

ATAN(X)	EXP(X ₁)
AMAX1(X ₁ , X ₂)	ALOG(X ₁)
AMIN1(X ₁ , X ₂)	SIN(X ₁)
AMOD(X ₁ , X ₂)	SQRT(X ₁)
SIGN(X ₁ , X ₂)	TAN(X ₁)
ABS(X ₁)	IFIX(X ₁)
COS(X ₁)	

D.2 PROBABILITY FUNCTIONS

Samples from probability distributions are obtained with the following syntax.

$$p_j(X_1, X_2, \dots, X_{nx}/t_1, t_2 \dots t_{nt}/p_1)$$

In the above, distribution j is sampled with parameters X_1 through X_{nx} , tables t_1 through t_{nt} , and parameter set p_1 . The amount of information required is distribution dependent. All distributions require at least one parameter, the number of the random number stream, which is denoted ISTRM below. Six streams are available. Parameter set requirements are explained at the end of this section.

There are 12 distributions available. Each is described below.

1. Random number between zero and one

$X_1 = \text{ISTRM}$
e.g., P1(1)

2. Uniform Distribution

$X_1 \leq P2 < X_2$
 $X_3 = \text{ISTRM}$
e.g., P2(10, 20, 3)

3. Exponential Distribution

X_1 = mean value

X_2 = ISTRM

e.g., P3(10, 1)

4. Weibull Distribution

X_1 = alpha

X_2 = beta

X_3 = ISTRM

e.g., P4(.1, 1.3, 2) where the Weibull density function is given by

$$f(x) = \alpha \beta x^{\beta-1} \exp(-\alpha \beta x^{\beta})$$

5. Triangular Distribution

X_1 = ISTRM

pl = parameter set

e.g., P5(3//2)

6. Normal (Gaussian) Distribution

X_1 = ISTRM

pl = parameter set

e.g., P6(AV(2)//7)

7. Lognormal Distribution

X_1 = ISTRM

pl = parameter set

e.g., P7(GV(J)//12)

8. Erlang -- k Distribution

X_1 = ISTRM

pl = parameter set

e.g., P8(6//1)

9. Gamma Distribution

X_1 = ISTRM

pl = parameter set

e.g., P9(3//2)

10. Beta Distribution

X_1 = ISTRM

pl = parameter set

e.g., P10(4//8)

11. Poisson Distribution

X_1 = ISTRM

pl = parameter set

e.g., P11(3//5)

12. Discrete Distribution

X_1 = ISTRM

t_1 = user-defined probabilities

t_2 = values to be sampled with corresponding probability in t_1

e.g., P12(3/PROBS, VALS)

If PROB has length 2 and values 0.3 and 0.7, and VALS has values 8 and 10, then P12 will return 8 30% of the time and 10 70% of the time.

Four values may be specified in each parameter set as shown in Table D-1. A minimum and maximum value may be assigned. The P_j function will be set to the minimum value if the sampled value is less than the minimum and to the maximum value if the sample is greater than the maximum. In Table D-1, " μ " is the mean, " σ " is the standard deviation, "a" is the minimum, and "b" is the maximum. "NU" means not used.

Table D-1

Parameter Set Values

Distribution	1	2	3	4
Triangular	mode	a	b	NU
Normal	μ	a	b	σ
Lognormal	$\ln(\mu)$	a	b	$\sqrt{\ln\left(\left(\frac{\sigma}{\mu}\right)^2 + 1\right)}$
Erlang -- k	μ/k	a	b	k
Gamma	σ^2/μ	a	b	$(\sigma/\mu)^2$
Beta	$\left(\frac{\mu-a}{b-a}\right) \left(\frac{(\mu-a)(b-a)}{\sigma^2} - 1\right)$	a	b	$(1) \cdot \frac{b-\mu}{\mu-a}^*$
Poisson	λ	a	b	NU

* Here "(1)" is the value in the first location at this parameter set.

The Poisson distribution is a discrete distribution. The probability of obtaining an integer, $n < b$, greater than or equal to "a" is given by

$$p(n) = \frac{\lambda^{(n-a)} e^{-\lambda}}{(n-a)!}$$

This is not the "usual" Poisson distribution unless $a = 0$. The distribution is shifted so the mean is $\lambda + a$.

D.3 MENU FUNCTIONS

Twenty-nine menu functions are available. The following notation will be used in explaining the menu functions.

X_i -- parameter i
 n_x -- number of parameters
 t_{ij} -- element j of table i
 n_t -- number of tables

As explained in subsection 3.9, any table that is required but not specified will default all elements to one. Exceptions are noted below.

	<u>Function</u>	<u>Common Usage</u>
1.	$F1 = \sum_{j=1}^{n_x} t_{1j} x_j$	Linear combination
2.	$F2 = \prod_{j=1}^{n_x} x_j^{t_{1j}}$	Product of powers
3.	$F3 = \sum_{i=1}^{n_x} \left(\sum_j t_{ij} x_i^{j-1} \right)$	Sum of polynomials, e.g., $F3(X1/T)$ is a polynomial in $X1$. If table i is defaulted, the term in parentheses is set to zero so there is no benefit to default tables with $F3$.
4.	$F4 = \prod_{j=1}^{n_x} (t_{1j})^{(x_j t_{2j})}$	Product of exponentials

Function

$$5. \quad F5 = \prod_{j=1}^{n_x/2} x_{2j-1}^{(t_{1j} x_{2j})}$$

$$6. \quad F6 = \sum_{j=1}^{n_x} t_{1j} \ln(x_j/t_{2j})$$

$$x_j/t_{2j} > 0$$

$$7. \quad F7 = \prod_{j=1}^{n_x/2} \left(\frac{x_{2j-1}}{x_{2j}} \right)^{t_{1j}}$$

$$8. \quad F8 = \exp\left(\sum_{j=1}^{n_x} t_{1j} x_j\right)$$

$$9. \quad F9 = \prod_{j=1}^{n_x} (t_{2j} - \exp(t_{1j} x_j))^{t_{3j}}$$

$$10. \quad F10 = \prod_{j=1}^{n_x} (t_{2j} + \exp(t_{1j} x_j))^{t_{3j}}$$

Common Usage

$$F4 = (a_1^{x_1 b_1})(a_2^{x_2 b_2}) \dots$$

$$F5 = (x_1^{x_2})(x_3^{x_4})$$

$$F5(x_1, 1, x_2, -1) = x_1/x_2$$

$$F5 = (x_1^{a_1})(x_3^{a_2})$$

$$F6 = \ln(x_1) + \ln(x_2)$$

$$F6 = a \ln(x_1) + b \ln(x_2)$$

$$F7 = \frac{x_1}{x_2} \frac{x_3}{x_4} \dots$$

$$F8 = e^{x_1}$$

$$F8 = e^{(ax_1 + bx_2)}$$

$$F9 = 1 - e^{-ax_1}$$

$$F9 = (a - e^{-bx_1})(c - e^{-dx_2})$$

$$F9 = (1 - e)^{-x_1 - 1}$$

Same as F9 with "+"

FunctionCommon Usage

$$11. \quad F11 = \left(\sum_{j=1}^{n_x} \pi t_{1j} x_j \right)^{1/2}$$

$$F11 = (a_1 X_1 + a_2 X_2)^{1/2}$$

$$12. \quad F12 = \left(\sum_{j=1}^{n_x} \pi t_{1j} x_j \right)^2$$

$$F12 = (a_1 X + a_2 X_2)^2$$

$$13. \quad F13 = \sum_{j=1}^{n_x} t_{1j} \cos \frac{x_j}{t_{2j}} - t_{3j}$$

$$F13 = a \cos \frac{X_1}{3} - \phi$$

where

t_{1j} = amplitude
 $2\pi t_{2j}$ = period
 t_{3j} = phase shift.

Table three defaults to zero.

$$14. \quad F14 = \sum_{j=1}^{n_x} t_{1j} \sin \frac{x_j}{t_{2j}} - t_{3j}$$

Same as F13 with "sin"

Table 3 defaults to zero.

$$15. \quad F15 = \tan \frac{x_1 - t_{12}}{t_{11}}$$

F15 shifts the origin to $x = t_{12}$ with
 asymptotes at $t_{12} \pm (\pi/2) t_{11}$.

t_{12} defaults to zero; i.e., if the length of
 table one is one or table one is defaulted, then $t_{12} = 0$.

Function

Common Usage

16. $F16 = t_{11} \{ [x_1/t_{12}] + 1 \}$

$[\cdot]$ is greatest integer function

17. $F17 = \begin{cases} \text{sign}(\tilde{X}) t_{i1} \\ -\text{sign}(\tilde{X}) t_{i1} \end{cases} \text{ if } \begin{cases} \bar{X} < t_{12}/2 \\ \bar{X} \geq t_{12}/2 \end{cases}$

Square wave

where

$$\tilde{X} = \text{mod}(x_1 - t_{13}, t_{12})$$

$$\bar{X} = |\tilde{X}|$$

t_{i1} = amplitude

t_{i2} = period

t_{i3} = phase shift (default = 0).

	<u>Function</u>	<u>Common Usage</u>
18.	$F18 = \begin{cases} t_{11} - \left(\frac{4 t_{11}}{t_{12}}\right) \bar{x} \\ -t_{11} + \left(\frac{4 t_{11}}{t_{12}}\right) (\bar{x} - t_{12}/2) \end{cases} \text{ if } \begin{cases} \bar{x} < t_{12}/2 \\ \bar{x} \geq t_{12}/2 \end{cases}$	Sawtooth

where

$\bar{x} = |\text{mod}(x_1 - t_{13}, t_{12})|$
 t_{11} = amplitude
 t_{12} = period
 t_{13} = phase shift (default = 0).

19.	$F19 = \sum_{j=1}^{n_x} t_{1j} x_j $	Absolute value
-----	---------------------------------------	----------------

20.	F20 = table function number one	F20($x_1/T1$, $T2$)
-----	---------------------------------	------------------------

F20 interpolates between points for $t_{11} \leq x_1 \leq t_{14}$

F20 = t_{21} if $x_1 < t_{11}$

F20 = t_{24} if $x_1 > t_{14}$

t_1 and t_2 must be the same length

Function

Common Usage

21. F21 = table function number two

F21(X₁/T₁, T₂)

$$F21 = t_{2k} \text{ if } t_{1k} \leq x_1 < t_{1,k+1}$$

$$F21 = 0 \text{ if } x_1 < t_{11}$$

$$F21 = t_{24} \text{ if } x_1 \geq t_{14}$$

t₁ and t₂ must be the same length.

$$22. \quad F22 = \begin{cases} x_1 \\ 0 \end{cases} \text{ if } \begin{cases} x_i = x_j \text{ for all } i, j \\ x_i \neq x_j \text{ for some } i, j \end{cases} \text{ Dirac delta function}$$

$$23. \quad F23 = \begin{cases} t_{11} \\ 0 \end{cases} \text{ if } \begin{cases} [x_i] = [x_j] \text{ for all } i, j \\ [x_i] \neq [x_j] \text{ for some } i, j \end{cases} \text{ Kronecker delta function}$$

[·] is the greatest integer function.

$$24. \quad F24 = \max\{X_1, X_2, X_3 \dots X_{nx}\}$$

$$25. \quad F25 = \min\{X_1, X_2, X_3 \dots X_{nx}\}$$

Menu functions 26 through 29 perform operations on tables and are analogous to certain vector manipulations. In the following, "a_j" is the jth accessible entity.

26. Vector linear combinations: The function sets

$$t_1(x_2 + j - 1) = \sum_{k=2}^{\bar{n}+1} t_k(x_{k+1} + j - 1)^{x_{\bar{n}+k}} \quad \text{for } j = 1 \text{ to } x_1$$

where $\bar{n} = n_x - n_t - 1$

A generic example makes this function more clear. Suppose we wish to compute the following linear combination:

In words, multiply the values in table T2 from elements x_3 to $(x_3 + n - 1)$ by x_5 . Do the same for the elements of T3 from x_4 to $(x_4 + n - 1)$, but use x_6 as the multiplier. Adding the two resulting vectors results in T1 from location x_2 to $(x_2 + n - 1)$. The function command is

`F26(X1, X2, X3, X4, X5, X6/T1, T2, T3)`

$$F26 = \begin{cases} 0 & \text{if the function was performed} \\ 1 & \text{if an error occurred, such as one of the tables was too short} \end{cases}$$

In general, X_1 is the length of the vector to be computed. The next n_t variables are the starting elements of each table, and the remaining variables are coefficients for the tables. Note that if there are more than n_t of these remaining variables, the excess variables will be multiplied by a default table of all ones and added to the linear combinations. This effectively adds a constant vector to the sum. At least two variables and one table must be specified. Only the first table has any values changed. If $X_1 \leq 0$, no action is taken.

27. Copy table values to accessible entities.

$$a_j = t_{1k} X_2 \quad j = 1 \text{ to } n_a, k = X_1, X_1 + j - 1$$

X_1 and X_2 default to one

t_1 defaults to all ones

if $X_1 = 0$ OR $N_A = 0$ no action is taken

$$F27 = \begin{cases} 0 & \text{if the copy was performed} \\ 1 & \text{if } t_1 \text{ is too short} \end{cases}$$

Suppose we wish to set

$$A_1 = t_1 x_1$$

$$A_2 = t_1 (X_1 + 1)$$

$$A_3 = t_1 (x_1 + 2)$$

Use

$$F27(1,1/T1//A_1, A_2, A_3) \text{ or}$$

$$F27(/T1// A_1, A_2, A_3)$$

The expression

$$F27(,3/// A_1, A_2, A_3)$$

will set $A_1 = A_2 = A_3 = 3$.

28. Copy variables to a table.

$$t_{1j} = X_k \quad k = 1 \text{ to } n_x - 1; \quad j = X_{nx} \text{ to } X_{nx} + nx - 1$$

t_1 must be present.

$X_{nx} = 0$ will result in no action.

$$F28 = \begin{cases} 0 & \text{if the copy was performed} \\ 1 & \text{if } t_1 \text{ is too short.} \end{cases}$$

Suppose we wish to set

$$t_{17} = X_1$$

$$t_{18} = X_2$$

$$t_{19} = X_3$$

Use the following:

$$F28(X_1, X_2, X_3, 7/T1)$$

29. Vector product

$$F29 = \sum t_{1j} \cdot t_{2k} \quad \begin{array}{l} j = X_2 \text{ to } X_2 + X_1 - 1 \\ k = X_3 \text{ to } X_3 + X_1 - 1 \end{array}$$

$$\text{if } X_1 = 0, F29 = 0$$

Suppose we want the vector product of X_1 elements in tables T1 and T2 starting at element X_2 in T1 and X_3 in T2. Use

$$F29(X_1, X_2, X_3/T1, T2)$$

In addition, to get the sum of elements 4 through 10 in Table T1, use

$$F29(7, 4/T1)$$

APPENDIX E
GLOSSARY OF IOSYM TERMS

Accessible Entity

Accessible entities are required in certain menu functions and may be needed with user-written functions. Accessible entities may be variables, table elements, arc names, or node names. The internal code for the entity is passed to function F or U. This permits the user to access or use the entity directly. See subsections 3.9 and 3.10.

Computer Job

A computer job is one execution of IOSYM. This job may perform one or more simulations of one or more networks. The job is terminated by the QUIT command or by a fatal error.

Entity

See accessible entity.

Executable Command

Executable commands are network commands which may cause the state of the simulated system to change. Executable commands are directive in nature (such as GOTO and REPLACE) rather than instructive (COLLECT) or structural (ARC, NODE). See subsection 3.5.

Function Arguments

Arguments are the parameters of menu functions, user functions, library functions, and probability functions that assume a single numeric value. These function arguments may be specified as single constants or variables or as arithmetic expressions. See subsections 3.9 and 3.10.

IDV

An IDV is a variable identifier. Every constant, variable, table element, etc. is assigned a unique internal identifier by IOSYM. This

is the IDV. IDVs can be accessed with the accessible entity feature of user functions. See subsections 3.9 and 3.10. Function U in Example 2 uses IDVs.

Independent Variable

This is the variable of integration for the Runja-Kutta numerical integration algorithm in GASP IV. The IOSYM default name for the independent variable is TIME. See subsection 3.3.

Library Function

Library functions are IOSYM analogs of standard FORTRAN library functions. They may be used freely in IOSYM arithmetic expressions. See subsection 3.9 and Appendix D.

Menu Functions

Menu functions are complex functions that can be accessed directly from IOSYM. Twenty-nine menu functions are provided. See subsection 3.9 and Appendix D.

Parameter Set

A parameter set is a list of four values used in the generation of samples from probability functions. See subsection 3.9 and Appendix D.

Primary Variable

A primary variable is one whose values are determined internally by the Runja-Kutta numerical integration algorithm in GASP IV. Equations for the derivatives of these variables must be specified in type I commands. See subsection 3.3.

Probability Function

A function which returns samples from a probability distribution. See subsection 3.9 and Appendix D.

Secondary Variable

A variable whose value must be set explicitly by the user. IOSYM never internally changes the value of a secondary variable (with the exceptions of OPTIME, PII, and EEE). See subsections 3.3 and 3.6.

Simulation

A simulation consists of one or more simulation runs. Simulations are caused by the DO command. REPORT and COLLECT statistics are printed after each simulation.

Simulation Run

A simulation run is one realization of the system to be simulated. One or more runs constitutes a simulation. Plots are printed after each run.

Substitutable Parameter

Provision may be made on the GLOBAL command to allow one or more dummy variables to be used whose value need not be specified until execution takes place. All type S, I, and E commands may use the dummy variables. The values or variables to be used in place of the substitutable parameters are specified on the OPR command. See subsections 3.7 and 3.8.

Table

A table is a one-dimensional array. See subsection 3.3.

Truth Condition

The truth condition is the logical relation on IF, PICK, and CROSS commands.

Truth Function

A truth function is the relational operator on SET commands and in menu, user, and probability functions. Truth functions cause assignments of zero or one to be made depending on whether a value bears the correct relation to zero. See subsections 3.5, 3.9, and 3.10.

User Function

User functions may be written by the user in FORTRAN to supplement menu and library functions. See subsection 3.10.

User-Written Subprograms

User-written subprograms are analogs of GASP IV user-written subprograms. See subsection 3.10 and Appendix C.

Utility Subprograms

Utility subprograms are FORTRAN callable subprograms that the user may reference in his/her user-written subprograms. See subsection 3.11.

Variable Identifier

See IDV.

APPENDIX F

IOSYM ERROR PROCESSING AND ERROR NUMBERS

IOSYM will detect certain errors and print a message indicating the error type. Errors are indicated by error number and may be fatal or nonfatal. Nonfatal errors are termed "warnings." Even error numbers are warnings; odd error numbers are fatal. IOSYM will print one of the following messages.

```
WARNING ---- IN SUBROUTINE ----  
ERROR ---- IN SUBROUTINE ----
```

A fatal error detected while in the network mode will be printed as above but will not terminate execution immediately. IOSYM will attempt to continue to the END NETWORK command. This permits more than one input error to be found for each execution.

A fatal error detected during execution of a simulation will terminate the program almost immediately. Before termination, IOSYM will attempt to print any statistics accumulated up to the time of the error.

Prior to attempting to print statistics, IOSYM will call the user-written error subroutine UEEROR. Subsection 3.10 contains a discussion of UEEROR.

Each subprogram in IOSYM is numbered. The subroutine number given in the warning and error messages above refers to the subprogram from which the error was detected. It normally will not be necessary to refer to the source code of IOSYM. If needed, however, the subprogram names and numbers are cross-referenced in Reference 1.

Table F-1 lists the IOSYM error numbers and their meanings. Additional GASP IV error codes are given in Appendix C, Section 4.

Table F-1
IOSYM Error Numbers

<u>Number</u>	<u>Fatal (F) Warning (W)</u>	<u>Error Condition</u>
1001	F	Illegal command under current command level.
1002	W	Trace option is not on.
1003	F	Exceeded the dimensions of NNPT1 array.
1004	W	Number of table elements on table card exceeds dimension, excess ignored.
1005	F	Exceeded bounds of WWRK array needed to evaluate an expression.
1006	W	Maximum length of any name is seven characters; therefore, name above has been truncated to seven characters and processing will continue. This may cause problems if another name is the same as truncated name.
1007	F	While searching for a defined variable, an alpha string that wasn't a variable or table was found.
1008	W	On NOPLOT command a plot was already inactive.
1009	F	While processing substitutable parameter on OPR command, an illegal field was found.
1010	W	While adding a plot on a plot command, the plot was already active but was replaced with new plot information.
1011	F	Unmatched parens found while processing function (F, U, or P).
1012	W	While looking for number, didn't find one.
1013	F	While processing COLLECT command, a field that is not a variable, derivative, or table was found.
1014	W	While looking for number, ran out of defined literals before finding one.
1015	F	An illegal NCODE value has been found in LITFND while looking for fields.
1017	F	On loop command, start and end values don't exist.
1018	W	While decoding a numeric representation of an alpha field, the IDV used is illegal.
1019	F	Exceeded maximum defined literal storage for a line of data input while looking for an alpha string.

Table F-1 (Continued)

1021	F	Illegal delimiter.
1022	W	While processing a number into a name, the number was truncated because maximum length for a name is seven characters.
1023	F	A number exceeds the computer's capability.
1024	W	End of the file was reached unexpectedly.
1025	F	Exponent exceeds the maximum allowed.
1026	W	Only one independent variable and five dependent variables are allowed per plot.
1027	F	Exponent is not an integer number.
1028	W	While processing table command, table dimension was not found, so a default of one was used.
1029	F	Maximum number of commands allowed per card exceeded.
1030	W	While processing TABULATE command, didn't find a report type.
1031	F	While looking for a literal in LOLFND, illegal field was found.
1033	F	While processing a function (F, U, or P), an argument starts with a * (multiply sign).
1035	F	While processing tables on function (F, U, or P), an illegal field was encountered.
1036	W	Deleting a variable in a collect list and the variable wasn't found.
1037	F	More than one arc leaves a node, and at least one doesn't have a pick.
1038	W	Changed a variable type in change mode.
1039	F	While processing input card, an illegal character was found. Valid characters are A to Z, 0 to 9, ., +, -, *, /, **, \$, =, and ,.
1040	W	A specified new tabulation already exists but will be replaced.
1041	F	Illegal truth condition.
1043	F	No start node in ARC command syntax.
1045	F	No end node name in arc command syntax.
1047	F	Illegal arc status on arc command.
1049	F	There is no expression on pick command.

Table F-1 (Continued)

1051	F	Index for a table is illegal. It should be an integer or I, J, K, L, M, or N.
1053	F	While looking for a variable or table name, none was found.
1055	F	No truth condition on pick command.
1057	F	No command on last input card.
1059	F	Command just processed was illegal.
1061	F	While looking at NCODE fields, an error field was found.
1063	F	On PICK command, there is either a truth condition and no function or vice versa.
1065	F	While processing GO command, there was a null field instead of a start node.
1067	F	A read command card is misplaced.
1069	F	Name already used as primary variable name.
1071	F	Name already used as secondary variable name.
1073	F	While processing variable command, found a null field for variable type.
1075	F	Illegal number for variable type.
1077	F	Undefined table on function (F, U, or P).
1079	F	Exceeded the dimensions of IIFPT array.
1081	F	While processing an expression into forward polish notation, an illegal IDV was encountered.
1083	F	Divide by zero.
1085	F	FORTTRAN library function (CONGJ) not operational.
1087	F	Tried to take log of zero.
1089	F	Tried to take a square root of a negative number.
1091	F	Tried to define independent variable as some other type.
1093	F	Position in NSET to start evaluating an expression was invalid; either was zero or greater than NNWRK.
1095	F	Illegal field while processing accessible entities on function (F, U, or P).
1097	F	No tabulation number on tabulation command.
1099	F	While processing TERM or CROSS, term or cross number wasn't found.

Table F-1 (Continued)

1101	F	Name has already been used as table name.
1103	F	An integer found while processing accessible entities on function (F, U, P) wasn't node or arc name.
1105	F	While looking for accessible entity on function (F, U, P), found undefined name.
1107	F	Plot number is zero on plot command.
1109	F	An equal sign was missing from an expression.
1111	F	While processing a DUMM(.) variable, found no index.
1113	F	This arc name already exists.
1115	F	This node name already exists.
1117	F	User function number may not be greater than 875.
1119	F	Requested an alpha from a command not allowed to request one; syntax error.
1121	F	Only an operator or open paren. can precede a table, variable, or number.
1123	F	Only an operator or open paren. can precede a library function.
1125	F	No matching LOOP or IF label.
1127	F	Only operations (**, *, /, -, +, =) or open paren. may precede an open paren.
1129	F	An **, *, /, -, or + may not precede a plus or minus sign.
1131	F	While processing an expression, an end of function indicator was missing.
1133	F	While processing REPORT command, table reference without integer index was encountered.
1135	F	An operator (**, *, /, -, +, =) or open paren. may not precede a "*" or "/".
1137	F	Exceeded the dimensions of NSET array.
1139	F	Exceeded the dimensions of NNCOLC array.
1141	F	Exceeded the dimensions of NNWRK array.
1143	F	An operator (**, *, /, -, +, =) or an open paren. may not precede an "***".
1145	F	Exceeded the dimensions of LITSTR array.
1147	F	Operators (**, *, /, -, +, =) or open paren. may not precede a closed paren.

Table F-1 (Continued)

1149	F	Exceeded the dimensions of CCONS array.
1151	F	Exceeded the dimensions of NNPLLOT array.
1153	F	Exceeded the dimensions of IIRPT array.
1155	F	Exceeded the dimensions of NNPTOP array.
1157	F	Exceeded the dimensions of NNPTND array.
1159	F	Exceeded the dimensions of NNPTAC array.
1161	F	Exceeded the dimensions of NNPTB array.
1163	F	Exceeded the dimensions of NCODE and NVAL arrays.
1165	F	Exceeded the dimensions of TTABL array.
1167	F	Exceeded the dimensions of NNTABU array.
1169	F	Exceeded the dimensions of NNPT2 array.
1171	F	No arc or node name on GOTO.
1173	F	Only operators (**, *, /, -, +, =) or open paren. can precede function.
1175	F	On a PUT or GET command, the file number is missing; or on a GET, the indicator is missing.
1177	F	More than four variables on tabulation command.
1179	F	While searching for a defined variable, found an illegal field.
1181	F	While sequencing the PICKs on ARCs, the number of picks found doesn't agree with number expected.
1183	F	Illegal number for statistics type.
1185	F	Have already processed this kind of "type" command on current command level.
1187	F	Have exceeded the dimensions of IIOPR array.
1189	F	While processing parameter sets on function (F, U, or P), an illegal field was encountered.
1191	F	While processing truth condition of function (F, U, or P), an illegal field was encountered.
1193	F	No truth condition on IF command.
1195	F	While processing commands under an IF, two true or two false portions were found.
1197	F	True or false command found outside IF.
1199	F	Unknown return code from PARSE to choice, so set control to stop.

Table F-1 (Continued)

1201	F	A referenced node is not defined.
1203	F	Undefined variable name on tabulation command.
1205	F	Attempted to define I, J, K, L, M, N, PII, or EEE as variables other than secondary vari- ables.
1207	F	Illegal statistic number.
1209	F	Tried to change table dimensions in change mode.
1211	F	While processing SAVE or RESTORE command, a network name wasn't found, and none was speci- fied on SAVE or RESTORE.
1213	F	While processing RESTORE command, network name could not be found.
1215	F	Table reference must be positive integer.
1217	F	Illegal field while trying to process a func- tion (F, U, or P).
1219	F	Ran out of defined literal storage while look- ing for a function (F, U, or P).
1221	F	Illegal field in function (F, U, or P).
1223	F	Illegal field while processing substitutable parameters on GLOBAL command.
1225	F	Plus and minus are the only operators allowed on GLOBAL, OPR, or LOOP commands.
1227	F	No type specified.
1229	F	Exceeded the dimensions of LLARY array.
1231	F	Not enough room in table to store NNATR values from file on PUT command.
1233	F	File specified on GET command exceeds NNFIL.
1235	F	On PRINT, there was no tabulation number or de- fined variable.
1237	F	Only defined variables may be changed by a WITH command.
1239	F	No node name on node card.
1241	F	Plot has more than 1000 points, and the X axis is not monotonically increasing. Only first 1000 points plotted.
1243	F	End of input file reached prematurely.
1245	F	No closing paren. on table reference.

Table F-1 (Continued)

1247	F	NSET array is not large enough to handle items to be read in from NETWORK SAVE file.
1249	F	No operations specified on the FOR command.
1501	F	Invalid table number encountered during execution.
1502	W	A menu function that requires an even number of variables was called with an odd number. The last variable was ignored.
1503	F	A table element number is nonpositive or too large.
1504	W	A loop has no executable commands.
1505	F	A parameter set number is nonpositive or too large.
1506	W	Incorrect arguments for menu function 26.
1507	F	A parameter set element was specified as nonpositive or greater than four.
1508	W	Incorrect arguments for menu function 27.
1509	F	An operation specified on an arc is invalid.
1510	W	Incorrect arguments for menu function 28.
1511	F	Invalid arc number or arc does not exist.
1514	W	Unrecognized variable on NOTRACE.
1516	W	Blank COMMON is too short for RESTORE. Results are unpredictable.
1517	F	GGVAR or PPVAR called with invalid variable type.
1519	F	Invalid primary variable number.
1521	F	Invalid secondary variable number.
1523	F	Invalid X**Y operation.
1525	F	Invalid menu or probability function.
1527	F	Nonpositive logarithm argument.
1529	F	Negative square root argument.
1531	F	Attempt to divide by zero.
1533	F	Second table too short for menu function 20 or 21.
1535	F	Table ordinates not monotonically increasing in menu function 20 or 21.
1537	F	Not enough variables in probability function.

Table F-1 (Concluded)

1539	F	Table for cumulative density in probability function 12 may not be defaulted.
1541	F	The probability table in probability function 12 is incorrect.
1543	F	Incorrect IDV sent to VVARBL.
1545	F	TIME may not be redefined during execution.
1547	F	A constant may not be redefined with PPIDF.
1549	F	Exceeded length of NNWRK/WWRK working array.
1551	F	Specified nonexistent default for DUMM(.) on OPR command.
1553	F	Invalid node number or nonexistent node in RREPLC or GGOTO.
1555	F	Circular node replacement detected.
1557	F	Invalid arc number, or arc does not exist in GGOTO.
1559	F	Invalid executable command type.
1561	F	Attempt to use or define an undefined DUMM(.) variable.
1563	F	No PICK command from a node yields a "TRUE" condition.
1565	F	Tried to GOTO an inactive arc.
1567	F	Invalid plot number.
1569	F	"GASP" or "READ" expected but not found.
1571	F	Invalid data found by free-form input routine FF...
1573	F	$n \leq 0$ on DO,n.
1575	F	Too many IF and/or LOOP levels.
1577	F	Duplicate loop indices in nested loops.
1579	F	A label has been encountered that is not a loop or IF terminator.
1581	F	A JUMP or EXIT has been encountered outside of a loop.
1583	F	A JUMP or EXIT has a label that does not end an active loop.
1585	F	Argument error in menu function 29.

1

2

3

4

APPENDIX G

IOSYM COMMON WITH SELECTED EXPLANATIONS

All of the information which defines the IOSYM network and controls the simulations is contained in FORTRAN COMMON storage. Several labeled COMMON blocks are used including all of the labeled COMMON for GASP IV. GASP IV COMMON is described in Reference 2, and most other labeled COMMON is of no interest to the user. Some information in four labeled COMMON blocks may be desired for writing user-written subprograms. These are shown in Figure G-1.

The IOSYM network and all GASP IV files are stored in blank COMMON. This COMMON area is referenced by both the arrays NSET and QSET (see Figure G-1), so both real and integer values can be stored. Similarly, a work array, NNWRK, is maintained in block I03. This array is equivalent to array WWRK, so both real and integer data may be stored.

A complete description of the variables shown in Figure G-1 is given in Reference 1. Selected variables are described here for use in the user-written programs.

<u>Variable</u>	<u>Block</u>	<u>Meaning</u>
NSET/QSET	blank	Do not change values in blank COMMON. Results may be unpredictable.
NNWRK/WWRK	I03	Do not change values in this array.
IIOP	I03	The number of the current operation (or next operation if between operations).
IIARC	I03	The number of the current arc. Zero, if at a node.
IINXAC	I03	The number of the next arc. This variable is usually zero. It is nonzero only if an arc has been specified with a GOTO command.
IINXND	I03	The number of the next node. This will be the end node of the arc or a node specified on a GOTO command.
NNSIM	I03	The number of simulations specified on the DO command.
IISIM	I03	The number of the current simulation. $1 \leq IISIM \leq NNSIM$.

IIGO	I03	The internal number of the node specified on the GO command.
TTIME1	I02	The initial value of TIME at the start of the simulation.

DIMENSION NSET(6000)	NDIM	2
DIMENSION WWRK(2000)	WDIM	2
COMMON/IC1/ IIRPT(50),NANCOLC(50),IITPS(5),ANFLOT(7, 5),	I01	2
- ANTABL(9,10)	I01	3
COMMON/I02/ ANDUM,TTIME1,IICPR(10,2),LLUCSL(62),NNPTGB(5),CCONS(30	I02	2
- 0),SSIN(100),DDIN(100),NAPTAC(3,50),NNPTND(4,50),	I02	3
- VVAR2(150,2),TTABL(1000,2),NNPT1(2,100),NNPT2(2,150).	I02	4
- NNPTC(2),NAPTBR(3,50),NAPTGP(6,50)	I02	5
- ,LLARY(2,20),NNGDUM,IIFPT(4,20)	I02	6
COMMON/IC3/ MMOPR,IISTAT,IICRBJ,IICP,IICAR,LLWRK,LLCROS,LLEVNT,	I03	2
- LLIST,LLTABU,LLRPT,LLCOL,LLWTH,IISIM,IIGASF,IIGO,	I03	3
- IINXAC,IINXND,NAXTB,MMV1,MMV2,MMNCDE,MMARC,MMCROS,	I03	4
- MMCCP,MMCOL,MMSIM,MMWTH,MMTBS,MMRPT,MMPLTS,MMCRBJ,	I03	5
- MMREAD,LLXSET,LLXWRK,MMCAS,MMCP,MMARY,MMFPT,MMNET(3),	I03	6
- LLTRAC,MMWRK(2000)	I03	7
COMMON/IC4/ MMSET,MMWRK,MMCAS,MMV1,MMV2,MMTAB,MMXTB,	I04	2
- MMTABL,MMRPT,MMCOLC,MMFLOT,MMREC,MMFAR,MMNCDE,	I04	3
- MMCP,MMARC,MMFUPF,MMFLNF,MMARY,MMFPT,MMLINE	I04	4
COMMON GSET(6000)	GSET	2
EQUIVALENCE(NSET(1),GSET(1))	EQA	2
EQUIVALENCE (MMWRK(1),WWRK(1))	EQW	2

Figure G-1. IOSYM Common Blocks.

DISTRIBUTION:

Amoco Production Company
Research Center
P.O. Box 591
Tulsa, Oklahoma 74102
Attn: B. Wilder

Department of Energy (10)
Division of Geothermal Energy
Mail Station 3344
12th & Penn, NW
Federal Building
Washington, DC 20461
Attn: C. McFarland
C. Carwile (2)
M. Skalka
A. Jelacic
R. Toms
R. R. Reeger
J. Salisbury
R. Black
M. Scheve

Department of Energy (5)
Office of Oil, Gas, and Shale
Technology
Mail Station D107
Washington, DC 20545
Attn: J. Smith
M. Adams
B. Dibona
D. Duke
A. Sikri

Dresser Industries, Inc.
P. O. Box 24647
Dallas, TX 75224
Attn: J. W. Langford

Dyna-Drill
P. O. Box C-19576
Irvine, CA 92713
Attn: Larry Diamond

Halliburton
Drawer 1413
Duncan, OK 75533
Attn: Dwight Smith

Loffland Brothers Company
P.O. Box 2847
Tulsa, OK 74101
Attn: H. E. Mallory

Los Alamos National Scientific
Laboratory
Mail Stop 570
Los Alamos, NM 87545
Attn: John C. Rowley

Mobil Research and Development
Corporation
Field Research Laboratory
P.O. Box 900
Dallas, Texas 75221
Attn: W. Gravley

NL Baroid Petroleum Services
City Centre Building, Suite 365W
6400 Uptown Boulevard
Albuquerque, NM 87110
Attn: Gene Polk

NL Petroleum Services
P.O. Box 1473
Houston, TX 77001
Attn: J. Fontenot

Otis
P.O. Box 34380
Dallas, TX 75243
Attn: W. D. Rumbaugh

Phillips Petroleum Company
P.O. Box 239
Salt Lake City, UT 84110
Attn: Earl Hoff

Pritsker and Associates, Inc.
P. O. Box 8345
Albuquerque, NM 87198
Attn: J. Polito (10)

Scientific Simulation Inc.
P. O. Box 9331
Albuquerque, NM 87119
Attn: D. M. Smith (5)

Smith Tool Company
P.O. Box C-19511
Irvine, CA 92713
Attn: J. Vincent

Texas A&M University
College Station, TX 77843
Attn: M. Friedman
Dept. of Geology

Union Geothermal Division
Union Oil Company of California
Union Oil Center
Los Angeles, CA 90017
Attn: D. E. Pyle

Shell Oil Co.
Two Shell Plaza
P.O. Box 2099
Houston, TX 77001
Attn: W. E. Bingman

4700 J. H. Scott
4740 R. K. Traeger
4741 J. R. Kelsey (50)
4742 A. F. Veneruso
4743 H. C. Hardee
4746 B. Granoff
4747 R. K. Traeger (Actg)
4748 B. E. Bader
4750 V. L. Dugan
4751 J. R. Tillerson
4756 H. M. Dodd
4756 B. C. Caskey (5)
8266 M. A. Pound
3141 Tech Library (5)
3151 W. L. Garner (3)
for DOE/TIC (Unlimited Release)
DOE/TIC (25)
(J. Hernandez - 3154-4)