

AD-A135 761

AISIM (AUTOMATED INTERACTIVE SIMULATION MODEL) USER'S
MANUAL(U) HUGHES AIRCRAFT CO FULLERTON CA GROUND

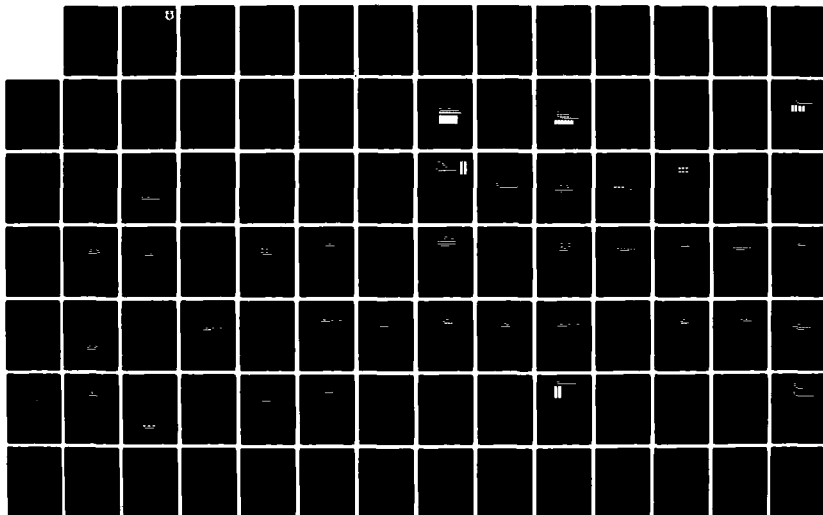
1/4

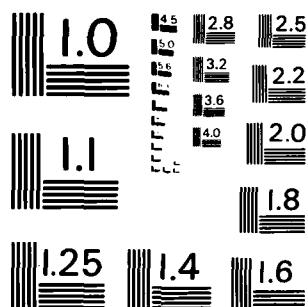
UNCLASSIFIED

SYSTEMS GROUP W AUSTELL ET AL. 26 FEB 82 ESD-TR-83-218
F19628-79-C-0153

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A135761

AISIM USER'S MANUAL

W. Austell
M. Deshler
J. Hearne
S. Kneeburg
M. Mabry

Hughes Aircraft Company
Ground Systems Group
Box 3310
Fullerton, California 92634



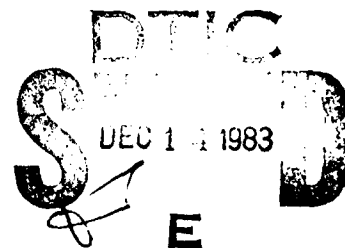
Approved for public release; distribution unlimited

26 February 1982

UNC FILE COPY

Prepared for

ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
DEPUTY FOR ACQUISITION LOGISTICS
AND TECHNICAL OPERATIONS
HANSCOM AFB, MASSACHUSETTS 01731



LEGAL NOTICE

When U. S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

OTHER NOTICES

Do not return this copy. Retain or destroy.

Review and Approval

This technical report has been reviewed and is approved for publication.

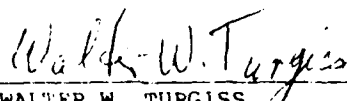


WILLIAM J. LETENDRE
Program Manager, Computer Resource
Management Technology (PE 64740F)



ARTHUR G. DECELLES, CAPT, USAF
Project Officer, Requirements
Analysis

FOR THE COMMANDER



WALTER W. TURGISS
Director, Engineering and Test
Deputy for Acquisition Logistics
and Technical Operations

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-83-218	2. GOVT ACCESSION NO. 41 722 218	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AISIM User's Manual		5. TYPE OF REPORT & PERIOD COVERED CDRL No. 104 Final
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) W. Austell S. Kneeburg M. Deshler M. Mabry J. Hearne		8. CONTRACT OR GRANT NUMBER(s) F19628-79-C-0153
9. PERFORMING ORGANIZATION NAME AND ADDRESS Hughes Aircraft Company Ground Systems Group Box 3310 Fullerton, CA 92634		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Hq Electronic Systems Division (ALEE) Hanscom AFB Massachusetts 01731		12. REPORT DATE 26 February 1982
		13. NUMBER OF PAGES 292
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) Unclassified
15a. DECLASSIFICATION DOWNGRADING SCHEDULE		
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) AISIM Analysis Entities Architecture User's Modeling Primitives Processes Design Simulation Systems		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This document is the User's Manual for the Hughes developed Automated Interactive Simulation Model (AISIM). This manual provides the user with a comprehensive guide for using this system to perform high level simulation of operational and distributed data processing systems.		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

CONTENTS

1.	IDENTIFICATION.....	1
1.1	PURPOSE.....	1
1.2	SCOPE.....	1
1.3	ORGANIZATION.....	1
1.4	DOCUMENTATION CONVENTIONS.....	1
1.5	APPLICABLE DOCUMENTS.....	3
2.	AISIM CONCEPTS.....	4
2.1	CHARACTERISTICS OF SYSTEMS MODELLED BY AISIM....	4
2.2	MODELING.....	5
2.3	DESIGNING MODELS.....	5
2.4	CONSTRUCTING AN AISIM MODEL.....	6
2.4.1	CHARTING A PAPER MODEL.....	7
2.4.2	DEFINING THE AISIM MODEL.....	7
2.5	AISIM MODELING ENTITIES.....	7
3.	AISIM ENTITES AND OTHER MODELING CONSTRUCTS.....	10
3.1	SCENARIO.....	12
3.2	LOAD.....	14
3.3	ITEM.....	18
3.4	USER DEFINED QUEUES.....	20
3.5	SYSTEM DEFINED QUEUES.....	23
3.5.1	QUEUES ASSOCIATED WITH RESOURCES.....	23
3.6	RESOURCE.....	25
3.7	ACTION.....	27
3.8	PROCESS.....	28
3.9	PRIMITIVES.....	32
3.9.1	ACTION.....	34
3.9.2	ALLOC.....	35
3.9.3	ASSIGN.....	36
3.9.4	BRANCH.....	38
3.9.5	CALL.....	39
3.9.6	COMPARE.....	41
3.9.7	CREATE.....	43
3.9.8	DEALLOC.....	44
3.9.9	DESTROY.....	45
3.9.10	ENTRY.....	46
3.9.11	EVAL.....	47
3.9.12	FILE.....	50
3.9.13	FIND.....	52
3.9.14	LOCK.....	53
3.9.15	LOOP.....	54
3.9.16	PROB.....	55
3.9.17	REMOVE.....	56
3.9.18	RESET.....	58
3.9.19	RESUME.....	59
3.9.20	SEND.....	60
3.9.21	SUSPEND.....	61
3.9.22	TEST.....	62
3.9.23	TRACE.....	63
3.9.24	WAIT.....	65

3.9.25	UNLOCK.....	66
3.10	LEGAL PATH TABLE - NODE - LINK.....	67
3.11	TABLES.....	69
3.11.1	DISCRETE TABLES.....	69
3.11.2	CONTINUOUS TABLES.....	69
3.11.3	ALPHANUMERIC TABLES.....	69
3.12	ATTRIBUTES.....	71
3.13	CONSTANTS AND GLOBAL VARIABLES.....	73
3.14	LOCAL VARIABLES.....	75
3.15	ALPHA LITERALS.....	77
3.16	KEYWORDS.....	77
3.17	MESSAGE ROUTING SUBMODEL.....	79
3.17.1	ITEM: MSG.....	80
3.17.2	USING THE MESSAGE ROUTING SUBMODEL.....	81
4.	AISIM SYSTEM OVERVIEW AND SYSTEM INITIALIZATION.....	82
4.1	REACHING THE AISIM READY LEVEL.....	82
5.	AISIM READY LEVEL - LEVEL 3.....	84
5.1	AISIM READY LEVEL COMMAND SUMMARY.....	86
5.1.1	INITIATING AN ANALYSIS SESSION.....	89
5.1.2	BACKING UP A DATABASE.....	91
5.1.3	CHANGING THE CURRENT PARAMETERS.....	92
5.1.4	INITIATE A DESIGN SESSION.....	93
5.1.5	VIEWING OUTPUT REPORTS.....	95
5.1.6	TERMINATION OF AISIM READY LEVEL.....	96
5.1.7	REGISTERING A COMPLAINT.....	97
5.1.8	HARDCOPY OUTPUT OF THE PROCESS FLOW-CHARTS.....	98
5.1.9	OBTAINING HELP FROM THE SYSTEM.....	99
5.1.10	EXERCISING THE LIBRARY FACILITY.....	100
5.1.11	LISTING THE CURRENT OPTIONS.....	101
5.1.12	PRINTING OUTPUT REPORTS.....	102
5.1.13	INITIATING A REPLOT SESSION.....	103
5.1.14	RESTORING A DATABASE (AFTER A CATASTRO- PHE HAS OCCURRED).....	104
6.	DESIGN USER INTERFACE (DUI).....	105
6.1.1	COMMAND: ARCH.....	108
6.1.2	COMMAND: COPY.....	109
6.1.3	COMMAND: DELETE.....	111
6.1.4	COMMAND: EDIT.....	113
6.1.5	COMMAND: END.....	114
6.1.6	COMMAND: HELP.....	115
6.1.7	COMMAND: LIST.....	116
6.1.8	COMMAND: SAVE.....	117
6.1.9	TERMINATION OF A DUI SESSION.....	118
6.2	PROCESS EDITOR INTERFACE (PEI).....	119
6.2.1	USE OF THE PEI.....	119
6.2.2	COMMAND: BOTTOM.....	122
6.2.3	COMMAND: CHANGE.....	123
6.2.4	COMMAND: DELETE.....	124
6.2.5	COMMAND: DOWN.....	125

6.2.6	COMMAND: END.....	126
6.2.7	COMMAND: HELP.....	127
6.2.8	COMMAND: HOLD.....	128
6.2.9	COMMAND: MENU.....	129
6.2.10	COMMAND: PLACE.....	130
6.2.11	COMMAND: TOP.....	131
6.2.12	COMMAND: UP.....	132
6.2.13	TERMINATING A PEI SESSION.....	133
6.3	ARCHITECTURE DESIGN EDITOR (ADE).....	134
6.3.1	CONCEPTS FOR USING ADE.....	136
6.3.2	USE OF THE ADE.....	136
6.3.3	ADE SYMBOLS.....	140
6.3.4	COMMAND: CHANGE.....	142
6.3.5	COMMAND: CONNECT.....	143
6.3.6	COMMAND: DEFINE.....	145
6.3.7	COMMAND: DELETE.....	149
6.3.8	COMMAND: END.....	150
6.3.9	COMMAND: LIST.....	151
6.3.10	COMMAND: MOVE.....	152
6.3.11	COMMAND: PLACE.....	153
6.3.12	COMMAND: RECON.....	154
6.3.13	COMMAND: SAVE.....	155
6.3.14	COMMAND: WINDOW.....	156
6.3.15	TERMINATION OF AN ADE SESSION.....	158
7.	USE OF THE ANALYSIS USER INTERFACE.....	165
7.1	COMMAND: CANBREAK.....	171
7.2	COMMAND: DEFLOT.....	172
7.3	COMMAND: EDIT.....	173
7.4	COMMAND: END.....	174
7.5	COMMAND: GET.....	175
7.6	COMMAND: GO.....	176
7.7	COMMAND: HELP.....	177
7.8	COMMAND: INFRES.....	179
7.9	COMMAND: LIST.....	180
7.10	COMMAND: LISTVAL.....	181
7.11	COMMAND: PLOT.....	182
7.12	COMMAND: SAVE.....	184
7.13	COMMAND: SETBREAK.....	185
7.14	TERMINATION OF AN AUI SESSION.....	187
8.	REPLOT USER INTERFACE.....	188
8.1	COMMAND: CLEAR.....	190
8.2	COMMAND: DELETE.....	191
8.3	COMMAND: END.....	192
8.4	COMMAND: GET.....	193
8.5	COMMAND: LIST.....	194
8.6	COMMAND: PLOT.....	195
9.	HARDCOPY USER INTERFACE.....	196
10.	LIBRARY USER INTERFACE.....	198
10.1	COMMAND: CHECKIN.....	200
10.2	COMMAND: CHECKOUT.....	201

10.2.1	COMMAND: DELETE.....	203
10.2.2	COMMAND: END.....	204
10.2.3	COMMAND: EXTRACT.....	205
10.2.4	COMMAND: HELP.....	206
10.2.5	COMMAND: LIST.....	207
10.3	COMMAND: MERGEIN.....	208
10.3.1	COMMAND: END.....	211
10.3.2	COMMAND: HELP.....	212
10.3.3	COMMAND: IGNORE.....	213
10.3.4	COMMAND: INFO.....	214
10.3.5	COMMAND: RENAME.....	215
10.3.6	COMMAND: REPLACE.....	216
10.4	COMMAND: MERGEOUT.....	217
10.4.1	COMMAND: HELP.....	220
10.4.2	COMMAND: LIST.....	221
20.4.3	COMMAND: SELECT.....	222
11.	AISIM SIMULATION RESULTS REPORTS.....	223
11.2	REPORT RESULTS AND HOW TO OBTAIN THEM.....	224
11.2.1	CONSTANT REPORT.....	230
11.2.2	VARIABLE REPORT.....	231
11.2.3	ITEM REPORT.....	233
11.2.4	RESOURCE REPORT.....	234
11.2.5	ACTION REPORT.....	236
11.2.6	QUEUE REPORT.....	238
11.3	COMMANDS RELEVANT TO VIEWING OUTPUT REPORTS.....	243
11.3.1	TOP, BOTTOM.....	243
11.3.2	UP, DOWN.....	243
11.3.3	FIND.....	243
11.3.4	LIST.....	243
APPENDIX A - OPERATIONAL PROCEDURES AND IMPORTANT INFORMATION.....		244
A.1	IMPORTANCE OF DATABASE BACKUP AND ALLOCATION....	244
A.2	ABNORMAL TERMINATION OF A DUI OR AUI SESSION....	244
A.3	AISIM PLOTS.....	245
APPENDIX B - AISIM ERRORS.....		247
APPENDIX C - GLOSSARY.....		264
APPENDIX D - QUEUES ASSOCIATED WITH ENTITY NAMES.....		268
APPENDIX E - MESSAGE ROUTINE SUBMODEL PROCESSES.....		269
E.1	PROCESS: REQ-I/O.....	269
E.2	PROCESS: ESR-CALL.....	272
E.3	PROCESS: ROUTER.....	273
E.4	PROCESS: CONTROL.....	276
E.5	PROCESS: CHLIO.....	279
E.6	PROCESS: IHANDLER.....	281

FIGURES

FIGURE		PAGE
1	AISIM Modeling Constructs.....	11
2	Resource States.....	24
3	Flow Chart Representation of a Process.....	31
4	Sample Legal Path Table Entries.....	67
5	Workspace/Viewspace Contrast.....	135
6	Report Giving Definitions of Constants, Tables, Items and Queues.....	226
7	Resource Report and Listing of Legal Path Table....	227
8	Report Giving Definitions of Actions and Processes..	228
9	Report Giving Load and Scenario Definitions.....	229
10	Sample Resource Utilization Report.....	235
11	Sample Queue Utilization Report.....	240
12	Sample Process Utilization Report.....	240

Accession For	
NTIS OFAI	X
DTIC TAB	
Unannounced	
Justification	
By	
Date	
Approved	
Dist	
A-1	



1. INTRODUCTION

1.1 PURPOSE

The Automated Interactive Simulation Model (AISIM) System provides the user with the ability to do high level simulation of complex operational and distributed data processing systems. The purpose of this manual is to provide the AISIM user with a comprehensive guide for the use of the system.

1.2 SCOPE

This manual describes the use of the AISIM software primarily from the user's point of view. It contains information necessary to operate AISIM. This manual discusses the hardware and software environments for AISIM as well as the function of the AISIM software. It is intended that this manual will be used as a reference for the AISIM user.

1.3 ORGANIZATION

This manual is organized to provide a straightforward reference document for the AISIM user. Chapter 1 introduces this document, detailing the organization of this document, the document conventions and applicable documents. Chapter 2 is an overview of the concepts used in modeling and simulation of systems using AISIM. Chapter 3 contains a detailed description of entities used in AISIM modeling. Chapter 4 describes the interface between the AISIM software and the host computer's time sharing system. Chapters 5 through 10 present information of the various system user interface levels, including detailed descriptions of prompts and commands. Chapter 11 discusses AISIM simulation results and how to interpret them. Appendix A presents operational procedures and other information which is useful for the user to know but not mandatory for using the system. Appendix B lists error messages which the user may see with a description of their meaning. Appendix C is a glossary of AISIM terms. Appendix D contains a description of AISIM data structures which can be manipulated by the user. Appendix E contains a detailed description of the Message Routing model for communication modelling.

1.4 DOCUMENTATION CONVENTIONS

The descriptions of AISIM commands given in this manual use the following notations to define the syntax and format of the AISIM commands:

1. Commands shown in the format below are equivalent:

DESIGN

D

Typing either DESIGN or D--the latter being an abbreviation for the former--followed by a carriage return will take the user from the AISIM Ready Level to the AISIM Design User Interface (DUI) sublevel.

2. Required parameters are enclosed in braces:

{language}

3. Optional parameters are enclosed in brackets:

[NOXLATE]

Default values exist for all optional parameters:

4. The brace and bracket symbols are used only to define the format. They should never be typed in the actual command statement.

braces { }

brackets []

5. The symbols listed below should be typed in a command statement exactly as shown in the command statement definition.

apostrophe '

comma ,

parentheses ()

period .

6. Words in lower-case appearing in a command definition represent variables for which the user should substitute specific information in the actual command.

EXAMPLE: If "database" appears in a command definition, the user should substitute a specific name of a database (for example, CONTACT) for the variable when the command is entered on the terminal.

7. All upper-case words and letters in a command definition, such as a command name or a parameter name (if the parameter is used), must be typed as part of the command statement.
8. All command names and associated parameters must be separated from each other by an appropriate delimiter. Acceptable delimiters are a comma or a blank. A blank is entered on the terminal by pressing the space bar at the bottom of the terminal keyboard.

EXAMPLE: BACKUP [PROJECT(database)] [USER(userid)]

If either of the optional parameters is used it must be separated from the command name BACKUP by a blank (). If both optional parameters are used they are to be separated by a blank (), i.e.

BACKUP PROJECT(CONTACT) USER(TF01234)

When a comma is to be used as the delimiter it will be specified as part of the command definition.

EXAMPLE: DEFLOT {entity-type},{entity-name}

In this example the command name DEFLOT would be separated from the required parameter {entity-type} by a blank and the two required parameters would be separated from each other by a comma i.e.

DEFLOT R,RESOURCE

9. The references in this document to specific words which are AISIM entities, will appear with an initial capital. This is to distinguish the reference to an AISIM specific concept from a common interpretation of the word.

EXAMPLE: Process - occurrences of this refer to the AISIM entity

1.5 APPLICABLE DOCUMENTS

The following documents provide additional information on the operation and use of AISIM:

IBM TSO User's Manual

Hewlett-Packard 2647A User's Manual

AISIM Training Manual

AISIM Training Examples

AISIM Product Specification

2. AISIM CONCEPTS

The Automated Interactive Simulation System (AISIM) provides a tool for the analysis of complex systems. The tool is designed for the operations analyst or engineer as a workbench for investigating the impact of system alternatives. AISIM provides a graphics language for the expression of systems, a database for storing a system's design and a simulation capability for analyzing the system. AISIM is applicable to design analysis of systems proposed hypothetically as well to the operations analysis of existing systems.

AISIM is a computer program that allows for the simulation of complex systems by a user without additional programming. The program can be executed interactively by a user communicating with a host computer through a terminal. By using the host computer in an interactive mode, an AISIM user can use AISIM to obtain timely data to support decisions on how a system is to function.

2.1 CHARACTERISTICS OF SYSTEMS MODELED BY AISIM

AISIM supports the design and analysis of systems that have the following characteristics.

1. Procedural operations -- Processes in the system can be described by a sequence of steps that describe the logic of every operation (e.g. operator actions, operating system logic, applications logic, man-machine interface, real time input processing)
2. Parallel Processing -- Any number of processes can occur simultaneously.
3. Shared Resources -- Some processes require resources that are contended for by other processes (e.g. two I/O requests contending for a single channel). Queueing is reflected in the degradation of the time required to complete processes suffering resource contention (e.g., large queues behind bottlenecks in a network).
4. Operational loading -- The operation of the system is a function both of its internal structure and of the environmental pressures on it.
5. Process communication -- Processes transfer data and materials to other processes in the system (e.g., both message routing and network control information communication can be easily represented).
6. Interconnected network -- Network architectures consisting of interconnected nodes can be represented in

AISIM. System constructs allow the user to define the routing of messages through the described architecture. AISIM also allows for the modelling of systems abstracted from any particular architecture.

These characteristics are generic to a large class of systems including military, computer, and industrial systems.

2.2 MODELING

In scientific and engineering usage, a model is a simplified (or idealized) representation of a system that is advanced as a basis for calculations, predictions or further investigation. AISIM modeling fits comfortably under this general characterization, but AISIM is especially designed for the modeling of systems which incorporate parallel processing (simultaneous activity) and networks. AISIM is particularly suited to the modelling of embedded computer systems for command, control and communication applications.

There are many applications of simulation modelling in this problem area. AISIM models are representative, discrete event simulation models used for predictive operations analysis. What this means is that entities in a real system are mapped onto AISIM entities which have a very close functional relationship. AISIM entities respond to simulated conditions much like the real entities do under actual conditions. This is in contrast to functional modelling where the real system is described in terms of equations in differential calculus. The emphasis in representative modelling is in describing the system.

Generally, determining and clearly describing the system is the first major obstacle a modeller must confront. If a system exists is in the design phase, then no data is available on how it will perform or what the major bottlenecks will be. For existing systems these characteristics may known but the combination of events that cause problems may not be understood. In both cases , much can be learned from modelling the system.

A key concept to keep in mind is that models are a simplified description of a system. This implies that elements of the real system may not be represented in the model. The challenge in modelling is to represent all the elements of critical interest to the system dynamics in the model. This requires some thought to the development of the model.

2.3 DESIGNING MODELS

A model should be carefully designed before being built. The key activities addressed during the design phase are the following:

Understand the Model and Collect Relevant Data -- To model any system effectively, a modeler has to know something about the system. Building an executable simulation model requires that the system have a complete description. A modeler must be aware of the functions performed in a system which effect the dynamics of the operation. A modeler must also know the characteristics of all the elements that perform work, create data, control processing, interrupt normal operations and produce output. This data can be obtained from design specifications, hardware specifications, previous studies or empirical testing. It is important to collect good data because that data becomes the foundation of the model.

Determine Model Boundaries -- Systems modeled by AISIM generally consist of many subsystems. The problems caused by the combination of subsystem activities are of interest to the analyst. AISIM provides varying levels of detail in modelling a subsystem. Sometimes the activity can be viewed as a black box. The flow of control through this box can be simply represented by a delay. This type of phenomena is modelled by AISIM with an Action. Other times, the characteristics of a subsystem can be represented by a mathematical function. AISIM has such a functional capability with the EVAL primitive and Table construct. If an activity is more complicated, it can be described by logic. In this case, AISIM allows the modeler to go to his own level of detail by building a Process. Setting the boundaries of an AISIM model is precisely what the modeler does in deciding which of these constructs will be used to model the elements of a system. A method of paper modeling developed for software design is known as "structured design". This method uses structure charts, heirarchical charts showing calling sequences, to describe functional processing. This method has been used successfully with AISIM. An alternate method would be to create flow charts of the various system functions.

Determine Experimental Method -- A model allows an analyst to run experiments on a system to predict how an operation will behave. Before any effort is expended in building a model, the output of simulation runs must be considered. Monitors can be designed to provide data on the system's operation. Experiments can be designed to validate the model.

2.4 CONSTRUCTING AN AISIM MODEL

2.4.1 CHARTING A PAPER MODEL In building a model, a modeler maps the elements of a system onto the constructs of the simulation language. To do this, the modeler must be familiar with the characteristics and relationships of both the simulation tool and the real-world system. The flow chart representations provide a definition of the system activities and data flow. AISIM represents the interaction of the system's activities over time. The mapping is not always clear-cut and usually requires iteration. The modeler charts out what processing takes place in a system, where resources are allocated, how processes communicate and where activities initiate. This chart is referred to as a paper model. It may be derived from an understanding of the system's functions and a graphical representation of its network. On the paper model, the modeler names the entities in the system that will be modelled by AISIM entities - Processes, Resources, Items, Queues, Tables, et cetera.

2.4.2 DEFINING THE AISIM MODEL An AISIM model is built by defining the AISIM entities which represent the system entities. This is done interactively on the computer. AISIM solicits relevant data for defining all design entities. From this point, the modeler uses the automated tool to build his model.

2.5 AISIM MODELING ENTITIES

As mentioned earlier, a model is a description or abstraction of a real or proposed system. To build a model with the intention of simulating its operation, we must describe the model in terms which can be interpreted, and operated upon, by the simulation system. That is, a system can be modelled using a prose description; but unless it has some systematic relation to a computer language it would be useless as a computer model because prose is ambiguous. AISIM uses a special set of terms to describe system structure and operation called AISIM entities. A modeller must understand the meaning and use of these entities to build successful models using AISIM. A detailed discussion of each of these entities is provided in Section 3. The following brief list names each AISIM entity. Definitions of entities important to our discussion are included. Figure 1 also provides further insights to the meaning, use, and relationships between entities and other modeling constructs.

Constant - A Constant is a term whose value does not change during a simulation exercise of a model. Constants are used to represent parameters that do not vary with time or in response to the workings of the system being modeled.

Item - An Item is a transient data element and is used to represent messages (or materials or even physical objects) flowing through the system.

Load - A Load is used to represent aspects of the world outside the system that trigger Processes within it; Loads represent the normal burden, i.e. cluster of Process triggerings, on a system.

Primitive - Primitives are logical constructs that represent steps in a logical sequence of (see Process). There are 25 different primitives each representing a different logical function. A sequence of primitives compose a Process. All primitives are listed below. The ACTION primitive has an Action entity associated with it. The Action entity is defined below. AISIM provides a language for describing the dynamics of a system with an expandable set of primitives (i.e. if the user wishes to model at a high level, a smaller number of these primitives are used; however, if more detail is desired or if several models are to be integrated, more primitives are available).

ACTION - (See below)

ALLOC
ASSIGN
BRANCH
CALL
COMPARE
CREATE
DEALLOC
DESTROY
ENTRY
EVAL
FILE
FIND
LOCK
LOOP
PROB
REMOVE
RESET
RESUME
SEND
SUSPEND
TEST
TRACE
UNLOCK
WAIT

Process - A Process is a description (using Primitives) of the logic of the operations, decisions or activities of the system being modeled.

Action - An Action, which is associated with the ACTION Primitive, is used to represent any action, activity, decision, etc. that consumes time. The Action entity is the only one that updates the simulation clock.

Queue - The Queue entity is used to model an ordered holding area for one or more items. A Queue may be used to model, for example, a job queue or a memory buffer. A Queue may be defined with a maximum size parameter to model, for example, such limits as the maximum number of messages that a buffer can hold before it is overloaded. Queues bear a default size of infinite.

Resource - The Resource entity is used to model the mechanisms (people, CPU, communication lines, etc.) necessary to perform a Process. Resources generally have the property of being shared among Processes. The sharing of a Resource may affect the performance of a Process. Performance of a Process can be degraded due to contention for Resources.

Scenario - The Scenario entity is used to model the various environments in which a system must perform. A Scenario specifies the number of periods of a simulation run as well as their length (which is uniform). A Scenario will determine which Processes and Loads are to be initiated and when.

Table - A Table is a user-definable function with up to fifteen pairs of data points. Tables may be defined as either continuous, discrete or alpha. A continuous Table interpolates linearly between numeric points. A discrete Table is a step function connecting numeric points. Alpha Tables are used for structuring data over non-numeric ranges and domains.

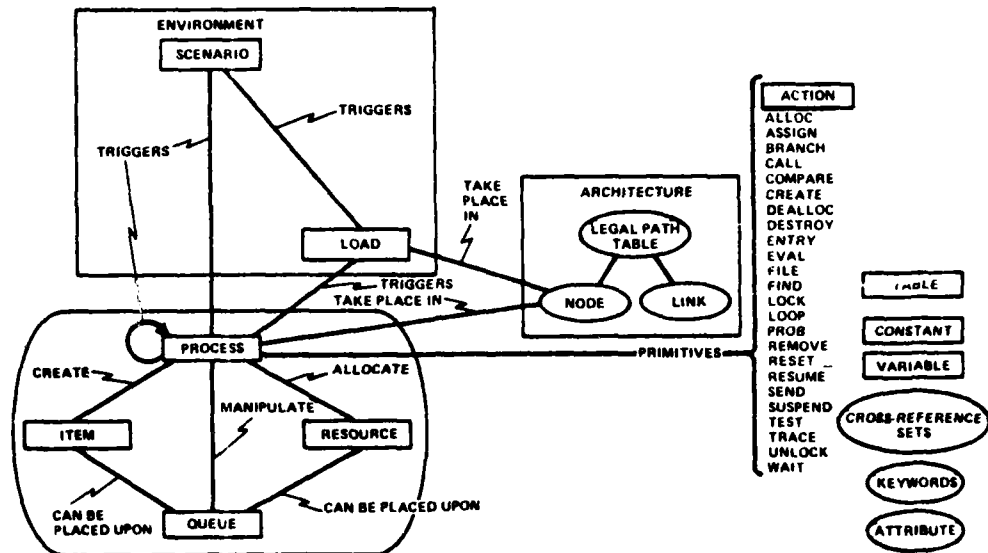
Variable - A Variable is a term whose value can change during a simulation run, either through use in calculations or through reassignment by the user between stages of a simulation. As with Constants, the value of a Variable can be set either during the design of a model (in the DUI) or just before a simulation exercise (in the AUI). Variables can also be changed during a breakpoint in the simulation.

Using these entities, the user must map the system to be modeled into an AISIM model.

3. AISIM ENTITIES AND OTHER MODELING CONSTRUCTS

In this section AISIM's entities and other modeling constructs are described. For each entity, the form the user is presented with when interacting with AISIM is given and its required parameters and meaning are explained. Included is mention of relations between the various AISIM entities, where such mention is deemed helpful. Similar treatment is given to other modelling constructs which are not AISIM entities but are representative of modelling concepts. The difference in the description of these constructs is that no AISIM forms are described. Figure 1 diagrams the main relationships between AISIM modeling constructs.

Figure 1. AISIM Modeling Constructs



SCENARIO

3.1 SCENARIO

The Scenario entity is used to represent the various environments in which the system being modeled must perform. Together with the Load entity it represents the external stimuli on a modeled system.

Scenarios are divided into periods whose length and number are chosen by the user. These periods provide break points at which the user can stop a simulation to alter a variable or inspect the results up to that point. There may be up to 14 periods in a given Scenario.

For each Scenario the user defines a collection of Loads and/or Processes, together with schedule time and triggering priorities for each. Each scenario calls for the initiation of activity over time by activating a Process or Load.

The form for Scenario is:

SCENARIO: [REDACTED] PERIOD LENGTH: [REDACTED]

DESCRIPTION: [REDACTED]

PERIODS: [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED]

CALLS:	TRIGGER	SCH TIME	PRIORITY	TRIGGER	SCH TIME	PRIORITY
	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]

where:

NAME: Scenario name (1 to 8 characters)

PERIOD LENGTH: Number of Timeunits in each simulated period.

DESCRIPTION: Any Scenario description (0 to 53 characters)

PERIODS: Mnemonic names can be entered in these fields consisting of up to 8 characters per name. The number of fields containing characters determines the number of periods in a simulation, i.e., for each of the 14 fields in which an entry is made a period is added to the total simulation run. A Scenario can have a maximum of 14 periods.

TRIGGER: 1 to 20 Process names or Load names; each Load or Process named causes the initiation of that Process or Load at the scheduled time.

SCH TIME: The time--in terms of absolute time units from the start of simulation (time = 0) at which the the Load or Process specified is to be initiated.

PRIORITY: The priority with which the a Process is to be triggered. (Priority is inverse, priority 1 pre-empts priority 2.) If a Load is scheduled from the Scenario, the priority field is ignored.

Operation - For each AUI session, a Scenario is specified by name. The Scenario specified will define simulation period length (in Timeunits), Loads and Processes. The number of periods specified in the Scenario is the number by which period length is multiplied to derive the total simulation time.

3.2 LOAD

The Load entity is used with the Scenario to initiate Process triggerings. These will include what may normally be thought of as the environment of the modeled system. It models the Process triggerings that are due periodic or random, whether or not such causes are outside of the system being modeled.

The Load entity enables batches of Process triggerings to be defined at specific nodes in the architecture. The Load describes which Processes will be initiated at which nodes. Loads trigger simultaneously at all nodes. This entity can be described in the following way; for each Process in the Load, initiate up to the maximum number with an interval given by the schedule method and mean between each triggering. Do this at all nodes simultaneously.

A Load definition defines a cluster of Process triggerings to take place at a certain point in the Scenario (exactly which point is given in the Scenario definition). This schedule specifies up to five Processes in up to eight architectural Nodes, together with a scheduling procedure the triggerings will follow.

The form for Load is:

```

LOAD: [REDACTED]

      NODE1  NODE2  NODE3  NODE4
      [REDACTED] [REDACTED] [REDACTED] [REDACTED]
      NODE5  NODE6  NODE7  NODE8
      [REDACTED] [REDACTED] [REDACTED] [REDACTED]

DESCR: [REDACTED]

PROCESS  RATE  SCHED  MEAN  DELTA  PRIORITY
[REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED]

```

where:

NAME: Load name (1 to 8 characters)

NODES: If architecture used, one to eight Nodes in which the Processes specified will take place.

DESCRIPTION: Load description (0 to 53 characters)

PROCESS: 1 to 5 names of Processes which the Load triggers according to schedule.

MAX #: Maximum number of times this Process is to be initiated in each execution of the Load.

SCHMTD: Statistical function to be used to determine the distribution of Process initiations over time. Schedule method can be any of those described under SCHEDULE METHOD.

MEAN: Depending upon schedule method, MEAN is used to determine the interval between each initiation of a Process. (In general this is the mean inter-arrival time.)

DELTA: Depending upon schedule method, DELTA is used to determine the deviations about the mean for for the interval between initiations of a Process.

PRIORITY: Priority with which the Process is to be executed. (Priority is inverse, priority 1 pre-empts priority 2). Priority is used to determine which Process will be allowed to allocate a Resource when it is contended for by two or more Processes.

SCHEDULE METHODS:

START - MEAN: inapplicable
DELTA: inapplicable

All Processes up to the maximum number are initiated at the same clock time, the start of the Load. This can be used to simulate pre-loading.

INTERVAL - MEAN: time between initiations
DELTA: undefined

One Process is initiated at the interval defined by MEAN. The first starts at the time given by MEAN with respect to the starting time of the Load.

POISSON - MEAN: mean number in a PERIOD
DELTA: undefined

Processes are scheduled randomly by a POISSON Process. Inter-arrival times are distributed exponentially. The MEAN parameter defines

the mean number for a PERIOD. PERIOD length is defined in the SCENARIO.

EXPONENT - MEAN: mean inter-arrival time
DELTA: undefined

Processes are scheduled randomly with exponentially distributed inter-arrival times.

LOGNORMAL - MEAN: mean inter-arrival time
DELTA: standard deviation of arrival time

Processes are scheduled randomly with LOG NORMAL inter-arrival times. The first Process initiates at some time after the start of the Load defined in the Scenario.

NORMAL - MEAN: mean inter-arrival time
DELTA: standard deviation of arrival time

Processes are scheduled randomly with NORMAL inter-arrival times. The first Process initiates at some time after the start of the Load defined in the Scenario.

UNIFORM - MEAN: mean inter-arrival time
DELTA: range about the MEAN

Processes are scheduled randomly with a rectangular distribution. The first Process initiates at some time after the start of the Load defined in the Scenario.

ERLANG - MEAN: mean inter-arrival time
DELTA: order of the distribution function

Processes are scheduled randomly with ERLANG distribution. The order "k" is given by the DELTA. The first Process initiates at some time after the start of the Load defined in the Scenario.

WEIBULL - MEAN: scale parameter.
DELTA: shape parameter

Processes are scheduled randomly with a WEIBULL distribution of inter-arrival times. The first Process initiates at some time after the start of the Load defined in the Scenario.

GAMMA - MEAN: mean inter-arrival time
DELTA: k

/

Processes are scheduled randomly with a GAMMA distribution for inter-arrival times. The first Process initiates at some time after the start of the Load defined in the Scenario.

Operation - a Load specifies a cluster of Processes to be triggered according to a scheduling method and a priority.

Relationships - Loads are part of Scenarios, and specify Processes to be triggered, and nodes in which they are to be triggered.

3.3 ITEM

The Item entity is used to model transient data elements that "flow" through a system. The Item declaration permits the modeling of data items which, by the nature of their varying attribute values, permit data-dependent decision making and timing.

Items can be originated, terminated and passed through the system from one Process to another through the Primitives CREATE, DESTROY, and SEND. Items can also be placed on and removed from Queues via the Primitives FILE and REMOVE.

The form for Item is:

:TEM NAME: [REDACTED]

DESCRIPTION: [REDACTED]

NAME	VALUE	NAME	VALUE

where:

ITEM NAME: 1 to 8 character name of Item

COMMENT: 0 to 53 character definition

NAME: 1 to 8 character name of attribute of Item.
An Item can have up to 15 user-defined attributes.

VALUE: The value to be assigned to attribute (integer, decimal, or character); If a name, it must be a defined Process, Resource, Variable, Constant, or Item name.

NOTE: All Items have implicitly defined attributes TAIL numbers and PRIORITY with values equal to the number of the Item created and the priority of the Process that created it, respectively. The tail attribute can be used for Item matching (see SEND Primitive).

Operation - An Item is created for each occurrence of the following:

- a. a CREATE Primitive that is executed - used to model transient data elements
- b. a SEND Primitive that is executed in a Process which does not have an Item of the specific name attached at the time.

An Item is automatically terminated only when the DESTROY Primitive is executed .

Attribute values are assigned at the time of creation.

Relationship - Item attributes are used by Process Primitives and attribute values can be modified by the ASSIGN and EVAL Process Primitives.

3.4 USER DEFINED QUEUES

A Queue is a global entity used to represent an ordered holding area. User-defined Queues may hold either Items or Resources.

When a Queue is defined, a maximum size parameter is specified (the default is "infinite"). This allows Queues to model finite storage devices that have a limited capacity (e.g., a storage bin, a computer job scheduler). Once the value is defined, it may not be changed and thus this parameter must be either a numeric value or a defined Constant. The maximum size parameter interacts with Process Primitives FILE and REMOVE to model queue blocking.

Queues are manipulated by Processes through the use of the FILE, FIND, and REMOVE Primitives. An Item or Resource unit may be placed on a Queue by using the FILE Primitive, specifying one of four location parameters: FIRST, LAST, BEFORE and NEXT. The former two parameters denote the end points of a Queue, the first and last slots. The latter two are location parameters relative to the Queue pointer.

A Queue may be "traversed" by using the FIND Primitive, which will alter the Queue pointer (see below). It can then be used by the FILE Primitive to place another Item or Resource in the Queue. This is done by specifying the location of the new Item or Resource relative to the Queue pointer, as either BEFORE or NEXT, where NEXT is the current value of the Queue pointer. In this way, a Queue could be ranked by one or more attributes of an Item (e.g., Item size or color).

An Item may be taken off the Queue by using the REMOVE Primitive. One may remove the first, last, or current Item (i.e. an Item that was found) from the Queue. After an Item is removed from a Queue, it may be sent, destroyed, or otherwise modified.

An Item may not be modified, sent, or destroyed while it is on a Queue. The same Item instance may not exist on more than one Queue. Multiple Processes may access the same Queue.

A Queue pointer is maintained for each Process which references a Queue. This pointer contains the address of the Item that the Process is addressing in a Queue. The contents of the Queue pointer is determined by rules described below and in the sections on the Primitives FILE, FIND and REMOVE:

1. The pointer contains the address of the last entity found with a FIND Primitive; otherwise,

2. The pointer contains the address of the last entity filed with a FILE Primitive; otherwise,
3. The pointer contains the address of the successor of the last entity removed with a REMOVE Primitive with a NEXT option.

The FILE Primitive places an Item or Resource on a Queue. This is done in one of the following two ways:

If space exists on the Queue, that is, the number of Items on the Queue is less than the maximum Queue size specified for the Queue, the Item is inserted into the Queue at the correct location specified by the option (FIRST, LAST, NEXT, BEFORE). If no space exists on the Queue, the Process which is executing the FILE Primitive is suspended. This condition is known as Queue blocked. In this state the Process waits until space becomes available on the Queue. Waiting for space on a Queue is by a first come first served discipline. This feature allows AISIM to model finite storage resources such as buffers and disks very easily. The default size of all Queues is INFINITE.

The REMOVE and FIND Primitives access a Queue and change the value of the Item referenced in the Primitive. This means that when a FIND or REMOVE Primitive is executed, the value of the Item could be set to 0. This occurs under the following circumstances:

1. A REMOVE Primitive attempts to remove an entity from an empty Queue.
2. A FIND Primitive accesses an empty Queue.
3. The NEXT or BEFORE Item in a Queue does not exist.

The form for Queue is:

```

QUEUE: [REDACTED]  SIZE: [REDACTED]
DESCR: [REDACTED]

```

where:

```

QUEUE:      1 to 8 character name of Queue
SIZE:       an integer value of 1 to 8 digits, or the
             word INFINITE
DESCRIPTION: 0 to 53 character description

```

Relationships - Queues are used to hold Items and Resources.

Queues are manipulated by the FILE, FIND, and REMOVE Primitives.

SYSTEM DEFINED STATES

3.5 SYSTEM DEFINED QUEUES

3.5.1 QUEUES ASSOCIATED WITH RESOURCES Associated with each Resource are four system defined "states" upon which statistics are kept during simulation time. The structures which maintain the Resource states are called Resource queues because the logic which manipulates these structures in the AISIM simulator follows queueing rules. The queues are (1) the Resource idle queue, (2) the Resource busy queue, (3) the Resource inactive queue (4) and the wait queue. The first three of these queues hold Resources and the fourth--the wait queue-- holds Processes. Entities are placed on these queues during simulation as follows:

Resources are on the idle queue while they are unallocated but available to Processes. Resources are placed on the idle queue (1) at the initialization of the simulation, (2) from the inactive queue (by the RESET Primitive) or (3) from the busy queue (by the DEALLOC Primitive).

Resources are placed on the busy queue while they are allocated by some Process through the ALLOC Primitive. They may be removed from the busy queue only by being deallocated with the DEALLOC Primitive.

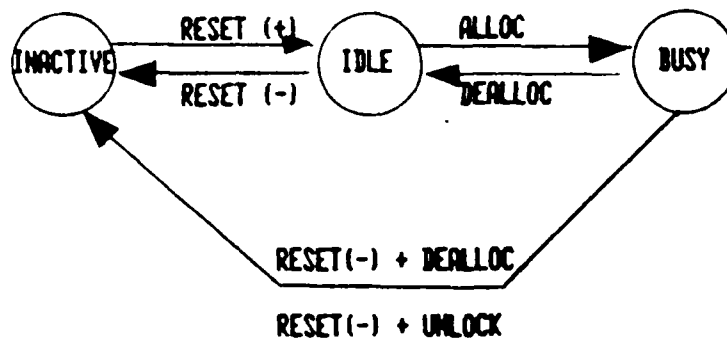
The inactive queue holds Resource units that are not available to be allocated by Processes. Resources may be placed on this queue (1) at the initialization of the simulation and (2) from the idle queue by means of the RESET Primitive.

The fourth queue associated with a Resource, the wait queue, holds Processes that are suspended for lack of an available unit of the needed Resource. A Process is placed on this queue when either (1) it attempts to allocate the Resource (with the ALLOC Primitive) that is held by another Process of equal or "higher" priority or (2) another Process of "higher" priority allocates the Resource it holds. (Priority is inverse, priority 1 pre-empts priority 2.)

The relation between these queues is illustrated in the accompanying figure.

During a simulation run statistics are kept on the activity of these queues. These results are presented in the simulation report.

Figure 2. Resource States



3.6 RESOURCE

The Resource entity is used to model the mechanisms required to perform a Process. "Mechanisms" in this context can be computer processors, memory, communications channels, support personnell, documents ,etc.

Queuing for a Resource occurs only within a Process and, in particular, only where an ALLOC Primitive is used. In other words, if no ALLOC Primitive is used there will be no queuing . If no Resource is used (allocated) within a Process, the Process can be executed in parallel (simultaneously) by any number of concurrent requests and the model will represent only time delays associated with the ACTION Primitive.

When a Resource is used (allocated) by a Process, there can be only as many concurrent executions of the Process as there are Resource units available. For example, if the capacity of a Resource is one (1), then any Processes which allocate that Resource will be executed serially (one at a time). Execution concurrency is controlled only between the allocation and deallocation of the Resource (i.e., if the ALLOC Primitive is the second Primitive in a Process, the first Primitive can be executed concurrently by any number of requests, whereas the second, ALLOC Primitive can be executed concurrently by only as many requests as the Resource has units available).

If a Resource has no units idle when an ALLOC Primitive is attempted, the request to execute the Process is merged onto a system structure associated with the Resource. How the request is merged depends on the priority given to the Process that is requesting the Resource. The request is merged and sorted by inverse priority (priority 1 pre-empts priority 2). Within priority the sorting is done first-in-first-out. When deallocation of the Resource (by some other Process) has resulted in enough units to satisfy the requests , and the request has moved to the top of the waiting requests queue, then the request is removed from the queue, the allocation is performed, and the Process is executed. Note that a deallocation of several units may result in several requests being removed from the Queue simultaneously.

The Resource entity provides the most interesting and useful simulation results; eg., delays, bottlenecks, utilization percentages, and traffic statistics. Therefore, the use of Resources should be carefully designed from both the standpoint of model credibility and the specification of required simulation output.

The form for Resource is:

		ATTRIBUTES	
		NAME	VALUE
RESOURCE NAME:	██████	████████████████████	████████████████████
TOTAL NUMBER OF UNITS:	██████		
INITIAL NUMBER OF UNITS:	██████		
ATTRIBUTES PRESENT (YES OR NO)	██████		
COST:	██████		
DESCRIPTION:	██		

where:

RESOURCE: 1 to 8 character name of Resource

TOTAL NUMBER: Maximum number of units of Resource that can be allocated (integer or named Constant).

INITIAL NUMBER: Number of units available for allocation at the start of the simulation (integer or named Constant).

ATTRIBUTE PRESENT: Indicates whether defined attributes are associated with this Resource.

COST: A default attribute to document the cost of the Resource.

DESCRIPTION: 0 to 53 character description

ATTRIBUTE: 1 to 8 character name of user defined attribute

VALUE: Initial value to be assigned to an attribute; can be single precision real or integer number, named Variable, named Constant, named Process, named Item, named Resource or named Queue.

Operation - Resources are initialized at beginning of simulation to the values given above. The interaction of Resources with Processes is dependent on the Primitives that effect them.

Relationships - Resources are used by Processes with the ALLOC, DEALLOC, RESET, LOCK, UNLOCK, TEST, FILE, FIND and REMOVE Primitives.

ACTION

3.7 ACTION

The Action entity represents any activity, decision, et cetera, that consumes time. This entity functions in conjunction with the ACTION Primitive. The Action entity of a given name is the summary point for statistics generated by the Action primitives with the same name. For this reason, each Action named in an ACTION Primitive is given a separate definition outside the Processes in which they appear.

In the form for this definition, the ACTION field contains a designation identical with one that appears in a Process. The field CLASS is optional and is intended as a means to document what kind of activity is taking place or who/what is performing the action (viz., man/machine). DESCRIPTION is used for any mnemonic.

The form for the Action entity is:

ACTION: [REDACTED]

CLASS: [REDACTED]

DESCRIPTION: [REDACTED]

where:

ACTION: 1 to 8 character name of action

CLASS: user defined class

DESCRIPTION: 0 to 53 character description

Relationships - Actions are used by the ACTION Primitive.

PROCESS

3.8 PROCESS

The Process entity is used to represent the sequential logic and activities, operations, functions, et cetera, of the modeled system. Processes are composed of Primitives, each of which represents a step in the function being modeled by the Process. It is at the Primitive level that Resources are allocated and deallocated, time is consumed, decisions take place, etc.

In the graphic representation of a Process, the Primitives are flanked at the top and bottom by figures labeled START and END. These figures represent the logical entry and exit points for the Process.

Processes are initiated by (1) Scenarios and Loads (within Scenarios) and (2) by other Processes through the CALL and SEND Primitives. Once initiated, the execution of the Process depends on the availability of the Resources that the Process references through the ALLOC and DEALLOC Primitives. The initial form for Process is:

```
START  
  
PROCESS NAME  NODE   
  
ATTRIBUTES ATTACHED (YES OR NO)   
  
PROCESS DESCRIPTION  
  
  
START BLOCK TYPE   
  
ENTER "PARM" FOR PARAMETER PASSING  
ENTER "ITEM" FOR ITEM PASSING  
ENTER "STD" FOR STANDARD PROCESS
```

where:

NAME:	1 to 8 character name of Process
NODE:	architecture node in which this Process is to execute (if it's execution is restricted to a specific node, ALL in this field indicates Process may execute in any node)
TYPE:	how the Process is to be initiated (STD, ITEM, PARM)

DESCRIPTION: 1 to 53 alphanumeric character description
used with the Process name in generating the
Process report

The three types of Processes (Item,PARM,STD) correspond to the three methods by which they can be triggered: (1) Item passing, (2) parameter passing and (3) standard.

An Item passing Process is one that is triggered by having Items delivered to it from another Process through the SEND Primitive. The graphic representation of an Item-passing Process will have a parallelogram to the left of the Start figure which will be labeled "Receive" and will contain the names of up to six Items required by the Process in order to execute. The required Items need not be delivered from a single Process; the sending Processes may be as many as six, but the Process will not execute until all of the Items indicated in the definition (and in the Receive figure) are delivered.

To define an Item passing Process the user enters "Item" in the form originally offered in the PEI. The user will then be presented with this form,

ITEM PASSING START

ITEMS RECEIVED:

██████, ██████, ██████, ..

MUST ALL THE ITEM SERIAL NUMBERS MATCH (Y/N) ██████

on which the needed Items are listed. The Items received by each must be of distinct types.

The field concerning the matching of serial numbers asks whether the TAIL numbers (which is a default attribute of every Item) must be the same for all the Items in the Process. If the user enters "Yes" in this field, the Process will not execute until it has received Items of the specified type to which the same TAIL number attribute has been assigned.

A parameter passing Process is one that takes values of local variables from another Process as inputs and/or returns the values of local variables to another Process as outputs. Such Processes can be triggered only by a CALL Primitive and it is the calling Process which sets up the relation for the values given and returned. (see CALL Primitive).

Such a Process is created by entering "PARM" in the form originally offered to the user. The secondary form is,

PARAMETER PASSING START

GIVEN:

--	--	--

RETURN:

--	--	--

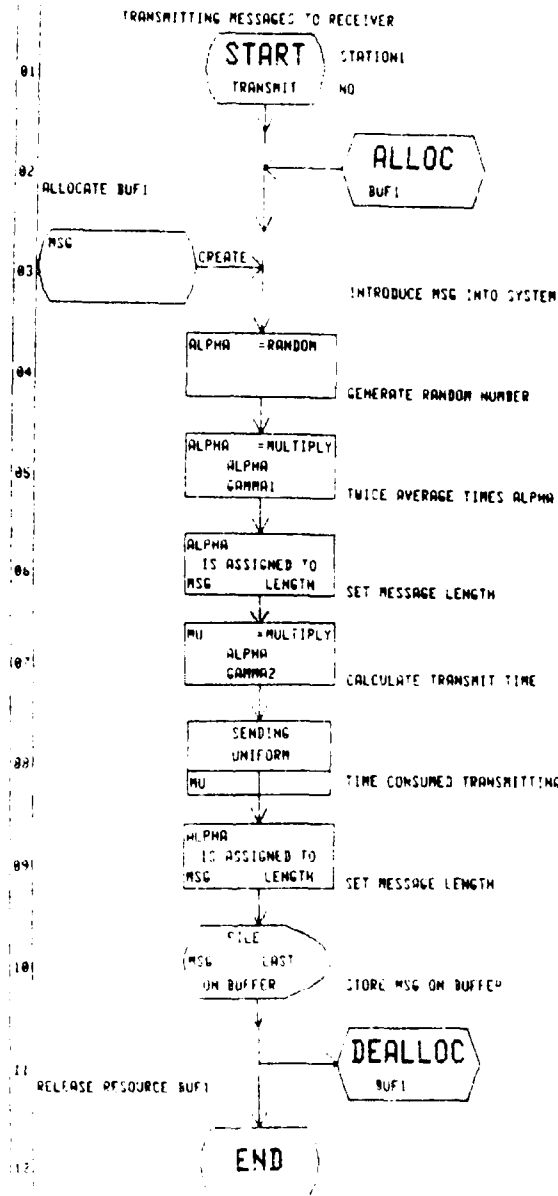
on which one writes the names of the local variables to be Given and Returned to any Process that calls it through the CALL Primitive. A parameter passing Process must have all the values in the local variables in this form to execute. If the CALL Primitive does not give or return all the necessary values, an execution error will occur indicating a disagreement.

A standard Process is one which neither requires Items nor is given (or returns) values of local variables. It can be triggered either by a CALL Primitive from another Process or through Scenarios and Loads. Since no inputs are relevant to its execution, there is no secondary form for the definition of a standard Process.

In the accompanying figure is a typical flow-chart representation of a Process. This graphical representation of the logic of a Process is presented to the user during the design of an AISIM model.

Relationships - Processes are constructed from Primitives. Resources are used by the Process through the ALLOC, DEALLOC, RESET, LOCK, UNLOCK, and TEST Primitives. Processes are initiated by Loads, Scenarios and other Processes through the CALL and SEND Primitives.

Figure 3. Flow Chart Representation of a Process



PRIMITIVES

3.9 PRIMITIVES

Primitives are the constituent elements of Processes and are used to characterize procedural steps by sequential logic. AISIM offers a list of 25 Primitives. Although limited in number the Primitive have been shown to represent all logical operations for computer system modelling. The Primitives can be grouped into nine functional categories. These categories are:

Process Execution Control

CALL
SEND
SUSPEND
RESUME
WAIT

These Primitives control the initiation and sequencing of Processes.

Branch Control

COMPARE
BRANCH
ENTRY
PROB
LOOP

These Primitives govern the internal branching in the logic of a Process.

Item Handling

CREATE
DESTROY

These two Primitives govern the introduction and elimination of a system's transient data elements.

Time Consumption

ACTION

This Primitive represents the consumption of time through some activity, decision, et cetera.

Mathematical Operations

EVAL

This Primitive governs calculations, invoking standard mathematical functions and operations or user-defined Tables.

Resource Allocation

ALLOC
DEALLOC
RESET
TEST
LOCK
UNLOCK

These Primitives govern the use of Resources.

Queue Manipulation

FILE
FIND
REMOVE

These Primitives govern storage and retrieval on Queues.

Variable Assignment

ASSIGN

This Primitive governs the assignment of values to variables (both numerical and non-numerical).

Debugging

TRACE

This Primitive has the special function of creating a record of the sequence of Process Primitive executions which takes place during simulation. It is used for debugging and validating a model.

Following is a description of the meaning of each Primitive and the parameters necessary to define it.

3.9.1 ACTION The ACTION Primitive indicates the performance of an action that consumes time. The action Primitive is used to model the time (in terms of mean, plus or minus delta) to perform some real work event such as a man's activity or a machine's activity. The variation in time is accomplished by inputting a random number into a distribution function. The distribution function can be changed by designating the desired function in the METHOD field. The choices for the parameter are described in greater detail in the section on the Load entity.

The form for ACTION is:

PARAMETERS FOR ACTION:

ACTION NAME: METHOD:
 MEAN TIME: DELTA-TIME:
 COMMENT:

where:

ACTION NAME:	A reference to a defined Action entity
METHOD:	Distribution function type, which may be: CONSTANT, EXPONENT, LOGNORML, NORMAL, UNIFORM, GAMMA, ERLANG or WEIBULL. (The random number seed used for statistical functions can be controlled by the user in the AUI.)
MEAN TIME:	Must be used to specify average duration times of Action. This parameter varies in meaning depending on the METHOD selected. In general, the Action duration times will average to the value specified.
DELTA TIME:	Depending upon METHOD, this is used to determine Action duration rates. This parameter varies in meaning depending on the METHOD selected. In general, it is the deviation parameter for the statistical function and must be specified for all METHODS except CONSTANT.

3.9.2 ALLOC The ALLOC Primitive indicates the allocation of (request to use) one Resources which is needed by the Process. ALLOC is used to model the need for a Resource in order to continue some operation. Whether a Resource requested by the ALLOC Primitive is actually obtained by a Process depends on a number of conditions. If a Resource unit is in the idle state it is available to be allocated to the requesting Process. If none is idle then allocated resources are checked to see if a process can be pre-empted by priority (priority is inverse - priority 1 pre-empts priority 2) unless the Resource is protected with a LOCK primitive.

The form for the ALLOC Primitive is:

PARAMETERS FOR ALLOCATE:

ALLOCATE RESOURCE NAME: XXXXXXXXXX

COMMENT: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

where,

RESOURCE NAME: A reference to a Resource

COMMENT: Any user comment.

3.9.3 ASSIGN The ASSIGN Primitive is used to set the value of the following references (V2 and Q2 parameters):

1. a global Variable
2. a local variable (to the executing Process)
3. the attribute of an Item (which has arrived at the Process)
4. the attribute of a Resource
5. \$CNODE
6. the attribute of a Process

Values that can be accessed for the assignment (V1 and Q1 parameters) are:

1. signed, single precision, real or integer numbers
2. \$CLOCK
3. global Variables or Constants
4. local variables
5. Resources with any of the qualifiers NWAITQ, NBUSYQ, NINACTQ or NIDLEQ
6. Item attribute values
7. Queue attribute values
8. Resource attribute values
9. Process attribute values
10. an Item name
11. a Resource name
12. a Process name
13. a Queue name
14. a Table name
15. an Action name

- 16. \$NODE
- 17. \$NXTNODE
- 18. \$LINK
- 19. \$TASK
- 20. an alpha literal (first character is \$)

The form for ASSIGN is:

PARAMETERS FOR ASSIGN

V1: [] Q1: []

TO

V2: [] Q2: []

COMMENT: []

where the entries may be any meaningful permutation and combination of the entries listed above. Some typical ones are:

V1: Item	V1: Item	V1: Variable
Q1: attribute	Q1: attribute	Q1:
V2: Item	V2: Variable	V2: Item
Q2: attribute	Q2:	Q2: attribute
V1: Variable	V1: Constant	V1: Constant
Q1:	Q1:	Q1:
V2: Variable	V2: Item	V2: Variable
Q2:	Q2: attribute	Q2:

COMMENT: Any user comment.

3.9.4 BRANCH The BRANCH Primitive indicates an unconditional branch to a named entry point. It is used for Process execution sequence control.

The form for BRANCH is:

PARAMETERS FOR BRANCH:

BRANCH TO LABEL: [REDACTED]

COMMENT: [REDACTED]

where:

LABEL: The entry point to which the Process execution is to go (which is represented by an ENTRY Primitive).

COMMENT: Any user comment.

PRIMITIVES / CALL

3.9.5 CALL The CALL Primitive indicates a call to another named Process and is the method by which one Process requests another Process to execute.

A CALL has one of three options (1) WAIT, (2) NOWAIT and (3) BLOCK. If a Process is called with the option WAIT, the calling Process will suspend execution until the called Process is completed. If a Process is called with the NOWAIT option, both called and calling Processes will execute simultaneously and will have no further communication. If a Process is called with the Block option the two Processes will execute in parallel until a WAIT is reached in the execution of the calling Process. When the WAIT Primitive is reached the calling Process suspends execution until the called Process completes itself. The principal purpose of the BLOCK option is to allow the calling of several different Processes, all of which must be completed before the calling Process will continue. If several Processes are called with the BLOCK parameter, the calling Process will suspend at the WAIT Primitive--whose presence somewhere below such a CALL Primitive is obligatory-- until all of them have completed execution.

Two of the three kinds of Processes can be triggered via the CALL Primitive, parameter passing Processes and standard Processes. In triggering a parameter passing Process with a CALL Primitive, values of local variables are given to the called Process and/or values of local variables are returned to the calling Process. Parameter passing Processes with return parameters can be called only with the WAIT option. Standard Processes, which neither give nor return information may be called with any of the three options WAIT, NOWAIT and BLOCK.

The CALL also requires that a priority (zero being highest) be established for the called Process; this priority is used by the called Process when competing with other Processes for available Resources (through the ALLOC Primitive).

The form for the CALL Primitive is:

PARAMETERS FOR CALL

CALLED-PROCESS NAME: [REDACTED]

WAIT/NOWAIT/BLOCK: [REDACTED] PRIORITY: [REDACTED]

GIVEN: [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED]

RETURNS: [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED]

COMMENT: [REDACTED]

where:

CALLED-PROCESS NAME: The Process to be triggered.

WAIT/NOWAIT/BLOCK: Explained above.

PRIORITY: The priority associated with the triggering.

GIVEN: Up to six local variables, literals or keywords whose values are to be communicated to the called Process.

RETURN: Up to six local variables, literals or keywords whose values are to be returned to the calling Process.

COMMENT: User defined comment

PRIMITIVES / COMPARE

3.9.6 COMPARE The COMPARE Primitive is used to model decisions based on user-controlled Variables. The COMPARE performs the following operation:

IF P IS TRUE, THEN GO TO A

where:

"A" is an ENTRY label which is branched to if P is true.

"P" is a predicate which can be TRUE or FALSE. It consists of a phrase,

X1 OP X2

X1,X2 can be

- (1) signed, single precision, real or integer numbers,
- (2) global Variables or Constants,
- (3) local variables
- (4) Resources with either NWAITQ, NBUSYQ, NINACTQ or NIDLEQ attributes (which cannot be modified by the user)
- (5) \$CLOCK
- (6) a value specified by an Item name and attribute
- (7) a value specified by a Resource name and attribute
- (8) a value specified by a Process name and attribute
- (9) an Item name
- (10) a Resource name
- (11) a Process name
- (12) a Queue name
- (13) a Table name
- (14) an Action name
- (15) \$NODE
- (16) \$NXTNODE
- (17) \$LINK
- (18) \$TASK
- (19) \$CNODE
- (20) an alpha literal (first character is \$)
- (21) a Queue with either NQUEUE or TQUEUE as an attribute (which cannot be modified by the user)

"OP" is one of the following operators: EQ, NE, GE, GT, LE, LT, (i.e., "equal to," "not equal to," "greater than or equal to," "greater than," "less than or equal to," and "less than," respectively).

Operation - "X1" is compared to "X2" using real, single precision arithmetic. If the comparison results in the same relation as "OP", then "P" is set TRUE and a branch is made to label "a"; otherwise, no branch is made (the next Process Primitive is executed).

The form for COMPARE is:

PARAMETERS FOR COMPARE

IF OPERAND 1: [REDACTED] QUALIFIER 1: [REDACTED]

RELATION: [REDACTED]

OPERAND 2: [REDACTED] QUALIFIER 2: [REDACTED]

BRANCH TO: [REDACTED]

COMMENT: [REDACTED]

where the parameters are filled in as indicated above.

3.9.8 DEALLOC The DEALLOC Primitive indicates the release of previously allocated Resources. It is used to represent the release of a Resource upon completion of a job.

The form for DEALLOC is:

PARAMETERS FOR DEALLOCATE:

DEALLOCATE RESOURCE NAME: [REDACTED]

COMMENT: [REDACTED]

where:

RESOURCE NAME: A reference to the Resource to be released.

COMMENT: Any user comment.

3.9.10 ENTRY The ENTRY Primitive is used to define entry points from the branching Primitives, BRANCH, PROB, COMPARE, TEST and LOOP.

The form for ENTRY is:

PARAMETERS FOR ENTRY:

ENTRY LABEL: [REDACTED]

COMMENT: [REDACTED]

where:

ENTRY LABEL: The name of the entry point used by the branching Primitive(s) which transfer control to it.

COMMENT: Any user comment.

PRIMITIVES / EVAL

3.9.11 EVAL The EVAL Primitive is used to perform simple arithmetic functions within a Process so that model logic and timing can be a function of Variables rather than a constant. EVAL operates in the following manner:

$$X = f(a,b)$$

where:

X is any Variable that is changed to the value $f(a,b)$

"a" and "b" are arguments that can be

1. signed, single precision, real, or integer number, or
2. named Variable or Constant, or
3. named local variable, or
4. \$CLOCK (simulation clock value)

where f is one of 27 functions below. All calculations are carried out in a single precision, real arithmetic.

<u>FUNCTION NAME</u>	<u>RESULT</u>
1. ADD	a+b
2. SUBTRACT	a-b
3. MULTIPLY	a*b
4. DIVIDE	a/b
5. ABSOLUTE	a
6. INTEGER	returns the integer part of a number
7. POWER	a**b
8. COSINE	cos(a) (a in radians)
9. SINE	sin(a) (a in radians)
10. TANGENT	sin(a)/cos(a) (a in radians)
11. SQRT	sqrt(a)
12. RANDOM	random fraction (random number between 0 and 1.0)
13. ARCOSINE	arccosine(a) (in radians)
14. ARCSINE	arcsine(a) (in radians)
15. ARCTAN	arctangent(a/b) (in radians)
16. BETA	random sample of the beta function with a = power of x; b = power of 1-x
17. BINOMIAL	random sample of the binomial function with a = number of trials b = probability of success
18. ERLANG	random sample of erlang function with a = mean

19. EXPONENT	b = k (integer order of function) random sample of exponential function with a = mean
20. GAMMA	random sample of gamma function with a = mean b = k
21. LOGE	natural logarithm of a
22. LOGNORMAL	random sample of log-normal function with a = mean b = standard deviation
23. LOG10	common logarithm of a
24. NORMAL	random sample of normal function with a = mean b = standard deviation
25. POISSON	random sample of poisson function with a = mean number
26. UNIFORM	random sample of a uniform function with a = mean b = delta
27. WEIBULL	random sample of the weibull function with a = scale parameter b = shape parameter

In addition to these functions the user modeller may define his own functions through the Table entity. A Table is accessed with the EVAL Primitive. A reference to the Table is used as the EVAL function name.

The form for EVAL is:

PARAMETERS FOR EVALUATE

SET VARIABLE: FUNCTION:

OPERAND1: OPERAND2:

COMMENT:

where:

VARIABLE: The Variable whose value is to be set.

FUNCTION: The operation used to calculate the value of the Variable.

OPERAND1: The first operand in the calculation of the new Variable ("a" parameter). This may be blank, depending on the function.

OPERAND2: The second operand in the calculation of the new Variable ("b" parameter). This may be blank, depending on the function.

COMMENT: Any user comment

3.9.12 FILE The FILE Primitive is used to place an Item on a user-defined Queue . It also is used to place a Resource on an idle-queue or Resource idle queue.

The effect of filing an Item on a user-defined Queue is to keep it in storage after the Process from which it is filed has ceased execution.

The effect of filing a Resource unit on the Resource idle-Queue is to make it accessible to any Process that requests it through the ALLOC Primitive. The effect of filing a Resource on a user-defined Queue is to make it accessible to another Process that wishes to take from the user-defined Queue and return it to the idle-Queue where it will be subject to allocation.

The form for FILE is:

PARAMETERS FOR FILE:

FILE ITEM NAME: [REDACTED] OPTION: [REDACTED] ON QUEUE: [REDACTED]
COMMENT: [REDACTED]

where:

OPTION:	The location in the Queue at which the entity is to be filed relative to the queue pointer. The following can be used :
FIRST	- The entity is placed first and the queue pointer is set to it.
LAST	- The entity placed last and the queue pointer is set to it.
NEXT	- The entity is placed after the current entity in the Queue and the queue pointer is set to it.
BEFORE	- The entity is placed before the current entity in the Queue and the queue pointer is set to it.
NAME:	The name of the Item or Resource unit to be filed
QUEUE:	The Queue on which the Item or Resource unit is to be filed. If the Resource idle-Queue is intended this field is entered with the name of the Resource to be filed.

COMMENT: Any user comment.

3.9.13 FIND The FIND Primitive is used to set the queue pointer on a user-defined Queue, a cross-reference set or a Resource idle-Queue, and to assign to a variable a "locator" pointer to a current position in the Queue. The rules governing queue pointers are covered above in the section on Queues.

The form for FIND is:

PARAMETERS FOR FIND:

FIND OPTION: ☐ ITEM NAME: ☐ ON QUEUE: ☐

COMMENT:

where:

NAME:	The local variable which will refer to the Item, Resource or member of a cross-reference set.
OPTION:	The location of the Item, Resource or member of the cross-reference set to be assigned to the variable relative to the present queue pointer.
QUEUE:	The name of the Queue, Resource idle-Queue or cross-reference set that is to be traversed. If the Resource idle-Queue is intended, the name of the Resource is entered. If the cross-reference set is intended the entity type whose cross-reference set is to be traversed is entered.
COMMENT:	Any user comment

The effect of locating an element with the find Primitive is (1) to set the queue pointer to the beginning or end of the ordered holding area (i.e., FIRST or LAST) or relative to the previous location of the queue pointer (i.e., NEXT or BEFORE), and (2) to assign the local variable to the element in the position then indicated.

PRIMITIVES / LOCK

3.9.14 LOCK The LOCK Primitive prevents a Process from being suspended by losing Resources to a "higher" priority Process (priority is inverse , priority 1 pre-empts priority 2). LOCK is used to represent uninterruptable work. If LOCK is not used, Process execution can be suspended by a higher priority Process. When a Process loses any one of the Resources it has allocated it stops execution and is placed on a system-defined Queue (the wait queue) until the Resource is again available. The LOCK Primitive overrides this suspension.

The form for LOCK is:

PARAMETERS FOR LOCK:

COMMENT: XXXXXXXXXX

where:

COMMENT: Any user comment.

3.9.15 LOOP The LOOP Primitive causes a branch to a named entry point a specified number of times.

The form for LOOP is:

PARAMETERS FOR LOOP:

LOOP TO LABEL:

LOOP TIMES

COMMENT:

where:

LABEL:	The name of the ENTRY label (defined by an ENTRY Primitive) to which execution is to branch.
LOOP:	Indicates the number of times the branching is to be repeated.
COMMENT:	Any user comment.

PRIMITIVES / PROB

3.9.16 PROB The PROB Primitive is used to model stochastic decision making. It indicates a probabilistic branch to a named entry point. (Random number selection for the probabilistic branch can be controlled by the use of the edit stream command in the AUI. Ten different branch streams are available.)

The form for PROB is:

PARAMETERS FOR PROBABILISTIC BRANCH:

BRANCH TO LABEL:

PROBABILITY OF BRANCH:

COMMENT:

where:

LABEL:	The ENTRY label to which the branching is to take place.
PROBABILITY:	The probability with which the branching is take place, expressed in (integer) percentages.
COMMENT:	Any user comment.

3.9.17 REMOVE The REMOVE Primitive is used to remove an Item or Resource unit from a user-defined Queue or a Resource unit from the Resource idle-Queue.

The effect of removing an Item or Resource unit is to make it inaccessible to other Processes until it has been placed on another Queue (through the FILE Primitive) or--in the case of Items--delivered to another Processes through the SEND Primitive.

The form for REMOVE is:

PARAMETERS FOR REMOVE:

REMOVE OPTION: ☐ ITEM NAME: ☐ FROM QUEUE: ☐

COMMENT: ☐

where:

OPTION:

The location in the Queue for the queue pointer after the primitive is executed. The option can be one of the following:

FIRST - The first entity is removed and the queue pointer is set to the new first element.

LAST - The last entity is removed and the queue pointer is set to the new last element.

NEXT - The current entity in the Queue is removed and the queue pointer is set to the succeeding element to it in the Queue.

BEFORE - The current entity in the Queue is removed and the queue pointer is set to the preceding element to it in the Queue.

NAME:

A local variable that will refer to the Item or Resource unit removed.

QUEUE:

The Queue from which the Item or Resource unit is to be removed. If the Resource idle-Queue is intended the name of the Resource is entered.

COMMENT: Any user comment.

PRIMITIVES / RESET

3.9.18 RESET The RESET Primitive redefines the number of available units of a named Resource to plus or minus the indicated value. It is used to represent the increase or decrease of the available number of Resources.

The form for RESET is:

PARAMETERS FOR RESET:

RESOURCE NAME:

RESET TO AVAILABLE UNITS

COMMENT:

where:

RESOURCE: A reference to a Resource whose available units are increasing or decreasing.

RESET TO: The number of units to be added to or subtracted from those presently available.

COMMENT: Any user comment.

PRIMITIVES / RESUME

3.9.19 RESUME The RESUME Primitive is used to control explicitly the resumption of a Process which has been suspended through the SUSPEND Primitive. The RESUME causes the reallocation of Resources to the Process.

The form for RESUME is:

RESUME PROCESS REFERENCED BY

V1: [REDACTED] Q1: [REDACTED]

COMMENT: [REDACTED]

where the fields V1 and Q1 constitute a reference to the task that is begin resumed and the COMMENT field has any user comment.

PRIMITIVE / SEND

3.9.20 SEND The SEND Primitive is used to send up to six Items to an Item passing Process. If an Item to be sent is not currently attached to the sending Process, it is automatically created. When the Items are sent, the receiving Process determines whether all the Items required by its definition have been sent. If they have, the Process then initiates; if not, it will wait until all of the necessary Items have been sent before executing.

The form for SEND is:

PARAMETERS FOR SEND

SEND ITEMS TO [REDACTED]

ITEMS TO BE SENT ARE:

[REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED]

COMMENT: [REDACTED]

where:

SEND:	A reference to the Process to which Items are to be sent.
ITEMS:	References to up to six Item types, instances of which are to be sent.
COMMENT:	Any user comment.

PRIMITIVES / SUSPEND

3.9.21 SUSPEND The SUSPEND Primitive is used to suspend the Process in which it appears. A Process that suspend itself with this Primitive may only be resumed by another Process which uses the RESUME Primitive. Since the resume Primitive must be able to refer to the task instant to be resumed, the suspending Process instance must save a reference to itself for later access by a RESUME Primitive. This reference will be the value of the \$TASK keyword. The SUSPEND Primitive causes the deallocation of the Resource corresponding to the current Node (\$CNODE).

The form for suspend is:

PARAMETERS FOR SUSPEND:

COMMENT: XXXXXXXXXXXXXXXXXXXX

where:

COMMENT Any user comment.

PRIMITIVES / TEST

3.9.22 TEST The TEST Primitive indicates a branch to a named ENTRY Primitive if a Resource or Queue is not available. It is used to model decision making based on the availability of needed Resources or Queues.

The form for TEST is:

PARAMETERS FOR TEST:

RESOURCE NAME: [REDACTED]

BRANCH TO LABEL [REDACTED] IF NOT AVAILABLE

COMMENT: [REDACTED]

where:

NAME:	A reference to the Resource or Queue being tested for availability.
BRANCH:	The name of the ENTRY Primitive to which execution is to branch if the Queue or Resource is not available.
COMMENT:	Any user comment.

PRIMITIVES / TRACE

3.9.23 TRACE The TRACE Primitive starts a debugging mechanism that is useful for analyzing the dynamics of an AISIM model. The effect of the TRACE Primitive is to create a file that records every execution of the following Primitives:

1. Start
2. CALL
3. ALLOC
4. DEALLOC
5. End
6. RESUME
7. RESET
8. SUSPEND
9. TRACE (on or off)

These Primitives are traced because they introduce major changes in the state of the system into a simulation run.

When the TRACE Primitive is operating every instance of these Primitives in every Process is recorded either for the remainder of the simulation or until TRACE is turned off. The trace line writes out the simulation clock time, the Node in which the Primitive is executed, and the Process executing the Primitive. The format for a trace line is the following:

T = clock time N = node name P = Process name Primitive parameter

The form for the TRACE Primitive is,

PARAMETERS FOR TRACE:

P1		P2		P3	
P4		P5		P6	
COMMENT					

but only the first field (P1 in the upper left) is used. To

enable the TRACE one enters "ON" in that field; to disable it one enters "OFF".

PRIMITIVES / WAIT

3.9.24 WAIT The WAIT Primitive used in conjunction with the CALL Primitive when the BLOCK option is used. The WAIT Primitive indicates that this Process is to be suspended until all Processes that began as a result of a CALL with the BLOCK parameter have completed and returned control to that Process. It is generally used to model phenomena such as assembly points, executive schedulers, and other events in which progress cannot continue until several parallel activities are completed. Resources currently in possession of the calling Process are not deallocated.

The form for WAIT is:

PARAMETERS FOR WAIT:

COMMENT: XXXXXXXXXX

where:

COMMENT Any user comment.

PRIMITIVES / UNLOCK

3.9.25 UNLOCK The UNLOCK Primitive cancels the effect of a previously executed LOCK Primitive. It is used to represent the conclusion of the uninterruptable phase of a Process.

The form for UNLOCK is:

PARAMETERS FOR UNLOCK:

COMMENT: XXXXXXXXXX

where:

COMMENT: Any user comment.

3.10 LEGAL PATH TABLE - NODE - LINK

The Legal Path Table (LPT) entity is the means by which the user can model physical communication paths between Resources. Typically, this is referred to as inter-node communication. When the LPT is not used, the communication mechanisms are implicit in the Process logic and do not usually have explicit Resources that cause communication queueing and transfer delays.

Two other model elements need to be discussed as part of the LPT entity; these are nodes and links. Nodes represent the points in an architecture where processing occurs. Links are the communication paths between nodes. Each node and link is actually a model Resource -- the name of the Resource being the name of the node or link. Full duplex links (denoted by ".F" after the link name) are two Resources. One will be named the name of the link with ".A" appended to it and the other a ".B".

The LPT consists of a four part list that specifies the FROM node, a TO node, a NEXT node, and a link. An example of Legal Path Table Entries is given in the accompanying figure.

FROM NODE	TO NODE	NEXT NODE	VIA LINK
-----	-----	-----	-----
A	C	C	C1
B	C	C	C2
C	A	A	C1
C	B	B	C2
C	D	D	C3
D	C	C	C4
D	E	E	C6
D	F	F	C5
E	D	D	C4
E	G	G	C8
F	D	D	C5
F	G	G	C7
G	E	E	C8
G	F	F	C7
G	H	H	C9
G	I	I	C10
H	G	G	C9
I	G	G	C10

Figure 4. Sample Legal Path Table Entries

The headings indicate that to move from the FROM node to the TO node one must first go to the NEXT node via the link.

The LPT is a passive entity in that it does not contribute directly to the simulation statistics but, instead, is simply a table of values used by a model to effect data flow through a system. It is only changed through the Architecture Design Editor and therefore remains constant for any specific simulation run. Processes reference the LPT through the ASSIGN Primitive using \$CNODE (current node), \$NXTNODE (next node as specified in

LPT), and \$LINK (the Link for the transfer) keywords.

Operation - Every Process in AISIM is assumed to be executing in a specific node. Using the LPT through the keywords and the ASSIGN Primitive, a Process can locate itself in the network and reference other nodes. The referencing is done symbolically so that a Process can do this when executing. This allows AISIM to model different architectures without changing the model Processes.

Relationships - Associated with each entry in the LPT is a Resource. Each symbol in an architecture is represented in one or more entries in the LPT. The LPT is referenced by the ASSIGN and COMPARE Primitives using keywords.

TABLES

3.11 TABLES

Tables are user definable functions with 1 to 15 entries. Each entry consists of an X-VALUE and a Y-VALUE. The following may be used for these parameter values: (1) both numeric, (2) one numeric and the other alphanumeric or (3) both alphanumeric.

Tables are accessed by using the EVAL Primitive. The EVAL FUNCTION parameter is the name of the desired Table. Operand 1 is the X-VALUE. Operand 2 is not used. The SET VARIABLE will be set to the Y-VALUE which maps from the X-VALUE.

3.11.1 DISCRETE TABLES If the Table accessed is discrete (TYPE is D), the Table entry's X-VALUE must be numeric, and the X-VALUE entries must be in increasing order. The Y-VALUE extracted from the Table is that value that is associated with the X-VALUE that is equal to or less than the X-VALUE given in OPERAND 1. For example, if an X-VALUE of 3.5 is given in OPERAND 1 and the nearest X-VALUES in the Table are 3 and 4, the Y-VALUE associated with the X-VALUE of 3 will be extracted and placed in the given SET VARIABLE name. If OPERAND 1 is less than the smallest X-VALUE, the value returned is the Y-VALUE associated with the largest X-VALUE.

3.11.2 CONTINUOUS TABLES If the Table accessed is continuous (TYPE is C), all X-VALUE and Y-VALUE entries must be numeric. The SET VARIABLE of EVAL is set by the following rules :

- a. the Y-VALUE associated with the X-VALUE that equals OPERAND 1, or
- b. the interpolation of the Y-VALUE associated with the X-VALUE which is less than OPERAND 1 and the X-VALUE greater-than OPERAND 1, or
- c. the Y-VALUE associated with the largest X-VALUE, if no interpolation is possible.

3.11.3 ALPHANUMERIC TABLES If the Table is defined as alphanumeric (TYPE is A), one or both X-VALUE and Y-VALUE for each entry must be a name of a model entity. The SET VARIABLE is set to the Y-VALUE corresponding to the X-VALUE.

If OPERAND 1 in the EVAL Primitive does not correspond to an X-VALUE in the Table referenced, an execution error message will be printed in the analyze report and the value of the SET VARIABLE will remain unchanged.

The form for Table is:

TABLE: [REDACTED] TYPE: [REDACTED]

COMMENT: [REDACTED]

X VALUE	Y VALUE
[REDACTED]	[REDACTED]

NAME: 1 to 8 character name of table

TYPE: C - continuous, D - discrete or A - alphanumeric.

X VALUE: x-axis value

Y VALUE: y-axis value

COMMENT: 0 to 53 character description

Relationships - Tables are made up of Table Entries. Tables are used by the EVAL Primitives. Entries consist of an X-VALUE and a Y-VALUE. The values for these may be numeric, alpha literals, or the names of model entities Constants, Variables, Items, Queues, Processes, Resources, or Tables.

ATTRIBUTES

3.12 ATTRIBUTES

Certain AISIM constructs have associated attributes which can take as values, (1) numerics, (2) alpha literals or (3) entity names. Attributes of entities can be referenced by Processes. Some attributes are user defined. Others are dynamic attributes which are recognized and modified by the AISIM simulator.

The values of attributes may be accessed by a Process with the ASSIGN and COMPARE Primitives. The forms for both of these Primitives uses two fields to indicate the value accessed. The first field contains the name of the entity and the second the name of an attribute associated with it.

Three AISIM entities, Processes, Resources and Items, may have attributes specified by the user. These attributes allow the modeler to define a unique set of characteristics for certain entities. An example is a channel. Channels have a physical attribute of maximum transfer rate. This characteristic is assigned to the AISIM Resource by specifying an attribute of RATE for the channel Resource.

Simulation experience has shown that some logic in a system is dependent on the system's dynamics. That is, some activity is dependent on queue lengths or the number of busy Resources. Since this phenomenon is fairly common, AISIM has embedded features to model this. The following attributes are built into the AISIM simulator for each instance of an entity:

Entity	Attribute	Description
Resource	NIDLEQ	the number of units of the Resource which are in an idle state
	NBUSYQ	the number of units of the Resource which are in a busy state
	NINACTQ	the number of units of the Resource which are in an inactive state
	NWAITQ	the number of Processes executing which are waiting for a Resource unit to be deallocated
Item	TAIL	the sequential creation number of the Item
	PRIORITY	the priority of the Item

Queue	NQUEUE	the number in the Queue
	TINQUE	the average time entities are in the Queue

Relationships - Attributes may be accessed with the ASSIGN and COMPARE Primitives.

CONSTANTS AND VARIABLES

3.13 CONSTANTS AND GLOBAL VARIABLES

Constants and Variables are entities used to define global parameters of a model, that is, values which may be accessed by all Processes. There is an implicit caution which must be used when using these entities. Because AISIM simulates multi-processing, global parameters can be accessed "concurrently" by more than one Process. This can cause some anomalies because of "race" conditions when two Processes modify a global Variable simultaneously.

A Constant is given a numeric value before a model starts a simulation. The value must be numeric and can not be changed by the simulation. A Variable may be set to (1) an alphanumerical, (2) the value of a keyword, or (3) to any other AISIM entity that may be accessed by the EVAL and ASSIGN Primitives. A Variable's value may vary throughout the simulation.

The initial values of both Constants and Variables are set in the design portion of AISIM. The value of both entities may also be set before the simulation is started in the analysis function.

While the value of a Constant may not be changed during the simulation, the initial value of a Variable may be changed by the user (between periods or at break points) or by the model itself (by use of the ASSIGN and EVAL Primitives).

Constants and Variables may be used in place of a numeric value anywhere a numeric value is required with the following exceptions:

1. The number of units of a Resource may only be a Constant or a numeric value;
2. The initial value of a Constant must be a numeric value;

The forms for Variables and Constants are :

CONSTANT: [REDACTED]

VALUE: [REDACTED]

DESCRIPTION: [REDACTED]

VARIABLE: [REDACTED]

VALUE: [REDACTED]

DESCRIPTION: [REDACTED]

NAME: 1 to 8 character name of Variable or Constant

VALUE: 8 digit floating point or any AISIM variable reference.

DESCRIPTION: 0 to 53 character description

Relationships - Constants may be used with Variables, Resources, Items, Loads, Scenarios and Process Primitives. Variables may be used with Items, Loads, Scenarios, and Process Primitives.

LOCAL VARIABLES

3.14 LOCAL VARIABLES

AISIM has two kinds of variables: local and global. Global Variables are those explicitly defined for the model and given initial values. Local variables are ones that appear in Process Primitives but are not otherwise defined. Local variables enable Processes to execute in parallel without interfering with each other because each Process has an independent set.

At the beginning of the execution of a Process all local variables are initialized to zero. They will remain so unless other values are explicitly assigned to them. Local variables may be assigned values with the ASSIGN Primitive or through parameter passing. Local variables may be assigned the following values:

- Numeric - a floating point or integer number

- Global Constant or Global Variable value

- Another local variable

- A Resource name

- A Process name

- An Item name

- A Queue name

- An alpha literal (first character \$)

- The value of a keyword evaluation

Although "local," the values of such variables can be communicated from one Process to another through parameter passing (i.e., through the CALL Primitive). Local variables can be used to fill in any parameter slot in any Primitive that is not an option, a label, a distribution or function, and including:

- Item attribute

- Resource attribute

- Process attribute

- CALL given parameter

- CALL return parameter

- Process given parameter

Process return parameter
ALLOC Resource name
DEALLOC Resource name
CALL Process name
CALL Priority name
ASSIGN set variable (variable 2)
COMPARE variable
FILE Item name
FILE Queue name
FIND Queue name
FIND Item name
REMOVE Queue name
REMOVE Item name
RESUME task reference

3.15 ALPHA LITERALS

An alpha literal is a character string. It consists of a \$ followed by up to seven other characters, as in

and

\$WAT

\$JONES

that do not make up the name of a keyword (see below). Alpha literals can be used to compare strings for identity or non-identity with the COMPARE Primitive. They can be used as attributes. This is useful for making AISIM models more readable.

3.16 KEYWORDS

The following keywords are defined in the AISIM simulator and may be used in Process logic in any Primitive in which the evaluation of the keyword results in a value which is correct in context.

Like alpha literals, these terms begin with the character '\$'. However, keywords function differently from alpha literals. Keywords evaluate to a value. In that sense they can be considered intrinsic functions.

- \$CLOCK - The value of the current simulation clock during the execution of a simulation run. This keyword may be placed in any field of a Process Primitive which may contain a numerical value.
- \$CNODE - The reference to the current node in which a Process is executing. All Processes are considered to be executing in a node in the architecture. The node corresponds to a Resource. This keyword evaluates to the Resource. This keyword allows a modeler to control allocation and deallocation of a Node from within the execution of a Process. This keyword can be assigned a value. This, in effect, changes the node in which a Process is logically executing. This is the only keyword that may be assigned a value in the Process logic.
- \$TASK - The current instance of the Process in which this keyword appears. A Process executing in a simulation can set a variable to the task instance of itself through the \$TASK keyword. This allows one Process to suspend itself and another Process to resume it by allowing the user to store the current value of \$TASK where it can later be accessed.

The following keywords directly access the legal path table and architecture structure. Each keyword evaluates to the name of a

node or link Resource.

- \$NODE - \$NODE takes one argument, a reference to a Process. Given a Process, \$NODE evaluates to the name of the node in which the Process has been defined to execute. This is the name of a Resource. This keyword allows a Process in AISIM to determine a destination for messages which request a specific Process to be executed. The node specification for a Process is defined by a user and is associated with the START symbol for the Process.
- \$NXTNODE - \$NXTNODE takes one argument, a reference to a destination node. Given a destination node, \$NXTNODE assumes the current node (\$CNODE) of the executing Process is the source (FROM) node. Accessing the legal path table, \$NXTNODE returns the name of the next node along the path to the destination node. This is the name of a Resource. This keyword allows the AISIM modeler to write Processes that perform message forwarding through a network.
- \$LINK - \$LINK takes one argument, a reference to a destination node. Given a destination node, \$LINK assume the current node (\$CNODE) of the Process is the source (FROM) node. Accessing the legal path table, \$LINK evaluates to the name of the link to the next node along the path to the destination node. This is the name of a Resource.

MESSAGE ROUTING

3.17 MESSAGE ROUTING SUBMODEL

When one Process triggers another through a CALL Primitive the called Process is initiated in the same node as the calling Process. This is implicit in the AISIM simulator and is true even if the called Process is associated with a different node.

In order to model the functional distribution of Processes throughout a network a logical Process communication feature had to be incorporated into AISIM. One requirement for this feature is that the delays inherent in the network communications be accurately represented in the model so that if a Process resident in one node initiates a Process resident in another node, the delays and queuing effecting this communication are taken into account. Also, AISIM is required to enable the analysis of different architectures performing the same functions with a minimum of change to the AISIM.

To satisfy these requirements a special sub-model has been devised to represent the routing of messages through an AISIM architecture to initiate remote Process triggering. Since different protocols for network communication are conceivable the AISIM message routing function has been implemented as an AISIM model and included in the AISIM model library. This enables an AISIM user to select and merge this model into his own. The advantage of this approach is that the user can review the logic in this sub-model, determine its appropriateness to his problem and modify the message routing sub-model if necessary. This will not often be the case because it has been shown that the message routing sub-model applies to most all communications networks.

The message routing sub-model uses the architecture and Legal Path Table of a model through the use of the system-defined keywords and the Process Primitives..

The message-routing submodel consists of one Item representing the message dispatched through the system architecture, six Processes representing the activities required for the inter-node communication and other supporting entities. Everything required for this model is included in the AISIM system library and can be merged into a user's model in a simple operation. (See LIBRARY User Interface)

Although intra-node communication is modeled by means of a collection of six Processes, the user need explicitly invoke (with a CALL primitive) only one of them. To represent the intra-Node triggering of a Process one calls the first Process in the sub-model called "REQ-I/O". It is a parameter passing Process with six values for the GIVEN list. No values in the RETURN list are specified. The parameters given are:

- (1) a name reference of the Process to be triggered;
- (2) the priority associated with the Process;
- (3) the option for "wait" or "nowait" (the option to have the calling Process suspend until receiving a message declaring that the called Process has executed, or to proceed in parallel), this is specified with the literals "\$WAIT" and "\$NOWAIT".
- (4) the length of the message which communicates the remote triggering. (This corresponds to channel rates normally expressed in bytes.)
- (5) the Node in which the triggered Process is to execute.
- (6) a name of the Item which is to be created and passed through the system for intra-Node communication.

The entities contained in the Message Routing Submodel are described in the following sections.

3.17.1 ITEM: MSG The Item MSG is the prototype for all communication messages which interact with the Message Routing Submodel. If the user does not want specific point to point transit times, all statistics for message routing will be accumulated for this one entity. If specific point to point transit times are desired, the AISIM user copies MSG to another Item name (through the Design User Interface) and provides the unique name as parameter six in the call to the REQ-I/O Process. All of the attributes of the Item MSG are essential to the Message Routing Submodel. The attributes are explained below.

DEFINITION OF ATTRIBUTES

<u>Attribute Name</u>	<u>Default Value</u>	<u>Description</u>
CNODE	\$CNODE	The current node in which the MSG resides in the network.
FNODE	\$CNODE	The source node of the MSG in the network for a communications path.
LENGTH	.99999999	The length of the message MSG in bytes.
PTASK	ERROR	The pointer to the instant of the requesting Process.
RESPONSE	\$WAIT	The option of the requesting Process. \$WAIT indicates the requesting Process waits for

a response. \$NOWAIT indicates the requesting Process continues.

RTASK	ERROR	The requested Process name.
TASKPRI	99999999	The priority for the requested Process.
TNODE	\$CNODE	The destination node of the requested Process.
TYPE	\$REQ	The message type of MSG. \$REQ indicates a request. \$RESP indicates a response.

3.17.2 USING THE MESSAGE ROUTING SUBMODEL The Message Routing Submodel is a very powerful feature of AISIM because it allows the user the capability of modelling network configurations and logical processing independently. That is, once the user has built the models of the functions performed in a system, the functions can be embedded in different architectures without modifications. As mentioned previously in this chapter, in order to use this feature it is necessary to understand the use of the Library User Interface, the architecture and the Legal Path Table. They all work together to provide the network modelling capability in AISIM.

The Message Routing Submodel requires that the user model interface with it properly to function correctly. The requirements for this interface are that the Message Routing Submodel be merged into the user model with the AISIM Library commands and that the entities in the user's model have the attributes referenced in the six Processes. This specifically applies to the nodes and links in the user's architecture and the Items which pass through the six Processes.

Items must have all the attributes described in the item MSG. Channels modelled with links must have a BAUDRATE attribute. Switching nodes in the architecture must have a CS-TIME (operating system overhead time for switching), a M.CS attribute (mean time to switch) and a D.CS attribute (delta time to switch).

4. AISIM SYSTEM OVERVIEW AND SYSTEM INITIALIZATION

The AISIM user interface consists of five levels of operation. They are:

- Level 1 - Not connected level
- Level 2 - TSO Ready level
- Level 3 - AISIM Ready level
- Level 4 - Level 4A - Design User Interface (DUI) Sublevel
 - Level 4B - Analysis User Interface (AUI) Sublevel
 - Level 4C - Replot User Interface (RUI) Sublevel
 - Level 4D - Hardcopy User Interface (HUI) Sublevel
 - Level 4E - Library User Interface
- Level 5 - Level 5A1 - Process Edit Interface (PEI) Sublevel
 - Level 5A2 - Architecture Design Editor (ADE) Sublevel
 - Level 5E1 - Mergein (MI)
 - Level 5E2 - Mergeout (MO)
 - Level 5E3 - Checkin (CI)
 - Level 5E4 - Checkout (CO)

Note that levels 4 and 5 are composed of a number of sublevels. The relationship of these different levels and sublevels is shown in the accompanying figure. The current level or sublevel of operation determines the system's response to a given command. For example, the command EDIT LOAD has a different effect when entered while at the DUI sublevel than at the AUI sublevel. Each level or sublevel prompts the user for input with a specific symbol or phrase. For example, the AISIM READY level prompts with the phrase "AISIM READY" on the screen when it expects a command to be entered from the keyboard. The DUI sublevel, on the other hand, prompts with an "**". The prompt for each level and sublevel is shown in the figure in its box. The commands used to go from one level or sublevel to another are shown next to the arrows indicating the direction of transfer.

4.1 REACHING THE AISIM READY LEVEL

The procedure for logging on is specific to a given computer system and the user is referred to local references for gaining access to the top level of the system on which AISIM is hosted. (This section assumes an IBM compatible host. For other installations please refer to installation specific instructions). When prompted with,

READY

the user has reached Level 2 of AISIM operation. To reach Level 3, the user enters the command,

EX AISIM

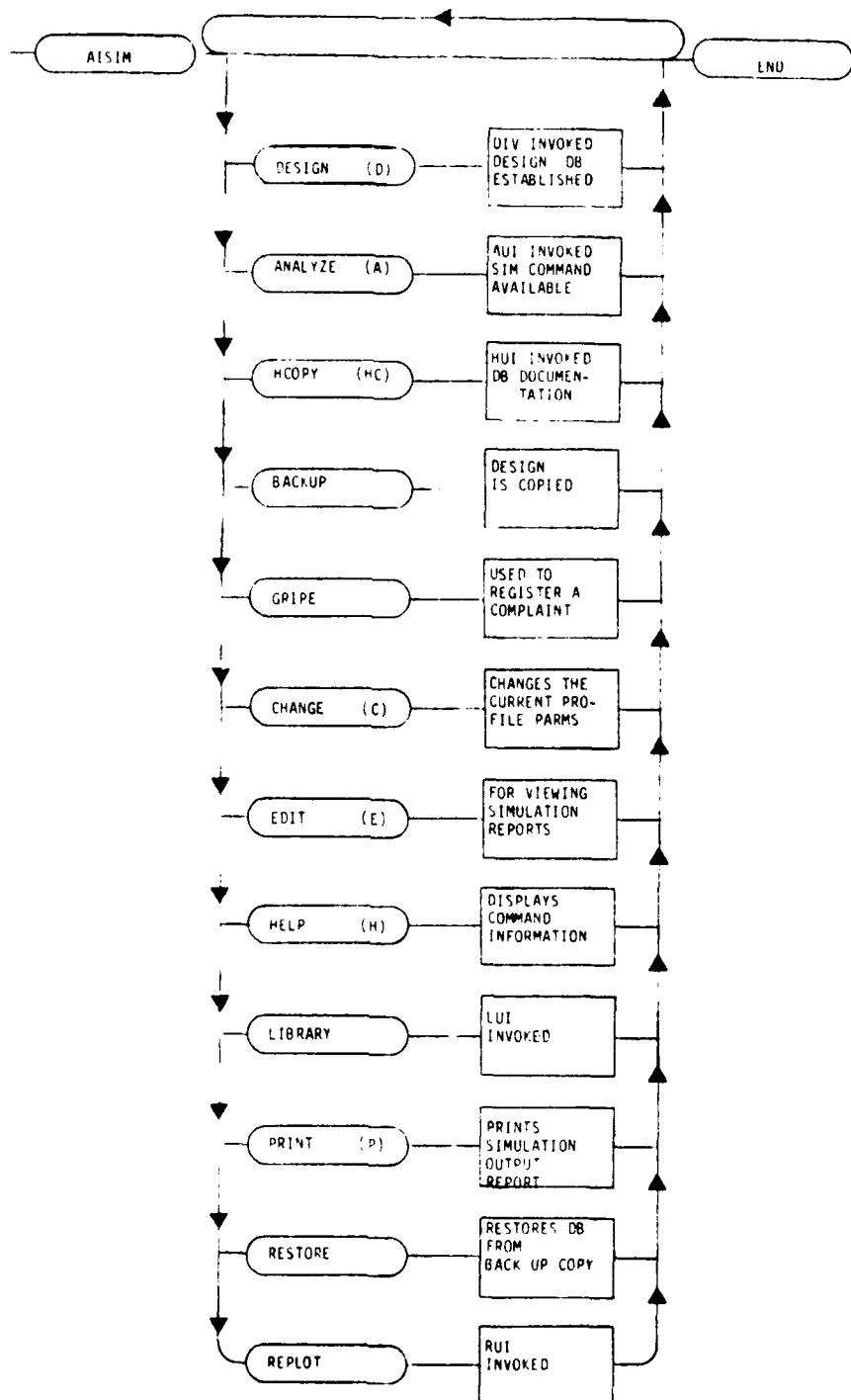
When the disk pack containing the AISIM system has been loaded, an audible 'beep' will be heard at the terminal.

5. AISIM READY LEVEL - LEVEL 3

The following sections describe the operation and use of the AISIM READY level. The system prompt at this level is the following:

AISIM READY

This level is the first level at which the user interacts with AISIM. This level also represents an AISIM control or monitor mode. No modeling is done at this level. Instead, the user must descend to Level 4 by issuing one of the commands shown the accompanying figure before any modeling, simulation, or other major system function can be exercised. The other commands shown in the figure are available at the AISIM READY level and perform various system level functions as described in the figure and in the command summary. When descending to level 4, the user will receive new prompts at the Design User Interface (DUI) level (level 4A), the Analysis User Interface (AUI) level (level 4B) or the Replot User Interface (RUI) level (level 4C) These levels are discussed in the appropriate sections. When the user descends to the Hardcopy User Interface (HUI) level (level 4D) or the Library User Interface (LUI) level (level 4E) the user will not receive new prompts.



5.1 AISIM READY LEVEL COMMAND SUMMARY

At the AISIM Ready Level of operation a number of commands are available to the user for directing the course of the session (ANALYZE, CHANGE, DESIGN, END) for manipulating the database (BACKUP, EDIT, RESTORE), for requesting information about AISIM operation (HELP), for requesting model data (PRINT, HCOPI) and for sending information to the support group (GRIPE). These commands are summarized in the command summary below. They may be entered only while in the AISIM Ready Level of operation (i.e., when the user has received an AISIM READY prompt).

A summary of the meaning of the parameters at this level follows:

[PROJECT(project)]
[P(project)]

The parameter indicating that the desired database file to be acted upon by the command is "project", where "project" is a standard alphanumeric file label containing 1-8 characters beginning with an alpha character and containing no special characters or imbedded blanks.

If omitted, the default value of "project" is assumed. Upon entry to the AISIM READY level from the TSO READY level the default value is blank and must be set by the use of an ANALYZE, BACKUP, CHANGE, DESIGN or RESTORE command with the parameter PROJECT. If an AISIM control command is used and no default value for "project" has been set an error message will appear prompting the user to set the "project" default value. Every use of ANALYZE, BACKUP, CHANGE, DESIGN and RESTORE with the PROJECT parameter changes the default value of "project" for subsequent commands.

[USER(userid)]

An optional parameter indicating the TSO "userid" of the owner of the reference PROJECT (project) if the owner is someone other than the current user. Default for "userid" is the current TSO logon "userid".

[VERSION(xx)]

[V(xx)]

An optional parameter indicating which version of AISIM software is to be invoked by subsequent commands. The default value of VERSION is the last VERSION parameter specified. If never specified explicitly by a VERSION parameter the default value is "PROD", the current production version.

Every use of ANALYZE, BACKUP, CHANGE, DESIGN and RESTORE with the VERSION parameter changes the default value of "xx" for subsequent commands.

The value of xx is either "PROD" to select the current production Version, or a 2 digit integer to select an alternate version.

[COMMAND(command name)]

An optional parameter indicating the AISIM control command for which the user needs help information. If omitted, the user will receive summary help information on all commands.

[DEST(location)]

An optional parameter indicating where the listing of the database elements is to be printed. "Location" is one of the legal destinations as specified in the TSO User Manual. The default value for "location" is the system printer.

[LIBRARY(library)]

[L(library)]

A parameter used in the CHECKIN and CHECKOUT facilities, where library is an 8 character library database name.

[BUFFER(buffer)]

[B(buffer)]

A parameter used with the CHECKIN, CHECKOUT, MERGEIN and MERGEOUT facilities. The parameter buffer denotes a temporary holding area for storing entity definitions.

[NOXLATE]

An optional parameter indication that for the current ANALYZE session, no translation from the PROJECT (project) is to be performed, and simulation input from a previous translation is to be used. The "previous translation" must have been performed. Translation is the process whereby the design database specified by the user is made ready for simulation.

Each of the AISIM READY commands is discussed on the following pages.

COMMAND SUMMARY

COMMANDS:

ANALYZE [PROJECT(project)] [USER(userid)] [VERSION(xx)] [NOX-LATE]

BACKUP [PROJECT(project)] [USER(userid)]

CHANGE [PROJECT(project)] [USER(userid)] [VERSION(xx)]

C

DESIGN [PROJECT(project)] [USER(userid)] [VERSION(xx)]

D

EDIT [PROJECT(project)] [BUFFER(buffer)] [TRACE]

E

END

GRIPE

HCOPY [PROJECT(project)] [USER(userid)] [VERSION(XX)]
[DEST(location)] [GDEST(location)]

HELP [COMMAND(command name)]

LIBRARY

LUI

LIST

L

PRINT [PRINT(project)] [DEST(location)]

P

REPLOT [PROJECT(project)] [USER(userid)] [VERSION(XX)]

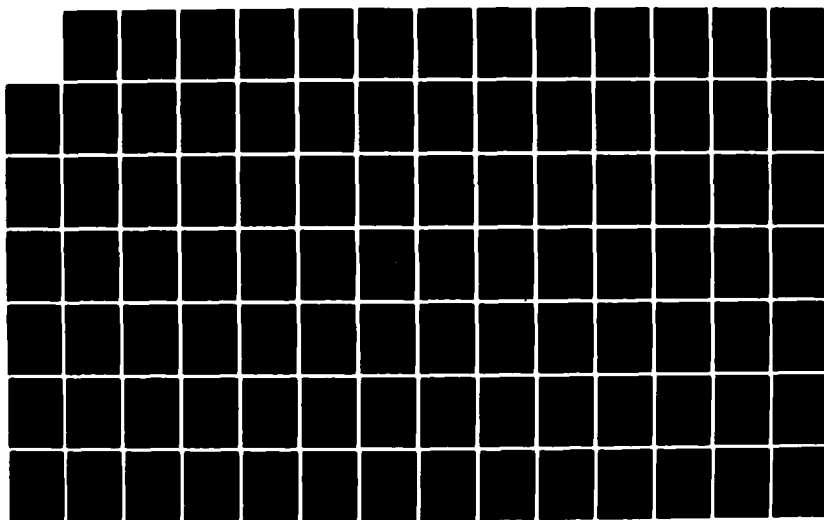
R

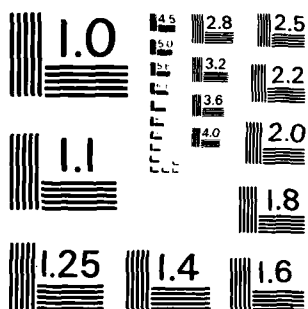
RESTORE [PROJECT(project)] [USER(userid)]

AD-A135 761 AISIM (AUTOMATED INTERACTIVE SIMULATION MODEL) USER'S
MANUAL(U) HUGHES AIRCRAFT CO FULLERTON CA GROUND
SYSTEMS GROUP W AUSTELL ET AL. 26 FEB 82 ESD-TR-83-218
UNCLASSIFIED F19628-79-C-0153 F/G 9/2

2/4

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

ANALYZE

5.1.1 INITIATING AN ANALYSIS SESSION Simulation of the model developed under the DUI level is accomplished through commands available in the AUI level. The AUI is accessed from the AISIM Ready level (level 3) by issuing the ANALYZE command as follows:

```
ANALYZE [PROJECT(project)] [USER(userid)] [VERSION(version)] [NOXLATE]  
A      [P(project)]
```

where:

[PROJECT(project)] is an optional parameter indicating that the analyze session is to involve database "project", where "project" should be 1-8 alpha or numeric characters, beginning with an alpha character (NO SPECIAL CHARACTERS OR IMBEDDED BLANKS).

Default if omitted is the last "project" specified in a previous DESIGN, ANALYZE, CHANGE, BACKUP, or RESTORE command. If this parameter is included but the USER keyword is omitted the system assumes the "project" belongs to the current user.

[USER(userid)] is an optional parameter indicating the TSO "userid" of the owner of "project".

If the PROJECT keyword is included, and USER keyword is omitted, the default for USER is the current logon "userid" PROFILE PREFIX value. If both PROJECT and USER keywords are omitted, the CURRENT PROJECT and USER remain unchanged.

[VERSION(version)] is an optional parameter indicating the software version to be invoked by this and all subsequent commands. Default if omitted is the last "version" specified in a previous AISIM, DESIGN, ANALYZE, or CHANGE command, or if never specified, the current PRODUCTION version, or a 2 digit integer "XX" to select TEST version "XX".

[NOXLATE] is an optional parameter indicating that, FOR THIS ANALYSIS SESSION ONLY, no translation from the "project" database is to be performed, and simulation input from a previous translation is to be used. The "previous translation" must have been performed.

The system will respond with the following output:

```
CURRENT PARAMETERS IN EFFECT:  
VERSION:      PRODUCTION  
PROJECT:      GERA  
USER:         TF00000  
XLATE/NOXLATE: XLATE  
ENTER YES TO PROCEED, NO TO ABORT....
```

Typing yes will cause the system to complete the transfer to the AUI level. Typing no returns the user to the AISIM Ready level.

A 'beep' will be given at the terminal when the system is ready to accept commands at the AUI level.

BACKUP

5.1.2 BACKING UP A DATABASE

To provide a backup of a project database, especially useful when performing saving the model design at a specific time, enter the following command:

```
BACKUP [PROJECT(project)] [USER(userid)]
```

where:

Parameters are described in paragraph 5.2.

CHANGE

5.1.3 CHANGING THE CURRENT PARAMETERS

The "CURRENT PARAMETERS" (PROJECT, USER, and VERSION) can be changed by using the appropriate keywords in a DESIGN or ANALYZE command, or via the following command:

```
CHANGE [PROJECT(project)] [USER(userid)] [VERSION(version)]  
C
```

where:

Parameters are described in paragraph 5.2.

DESIGN

5.1.4 INITIATE A DESIGN SESSION

```
DESIGN [PROJECT(project)] [USER(userid)] [VERSION(XX)]  
D      [P(project)]          [V(XX)]
```

where:

[PROJECT(project)] is an optional parameter indicating that the desired project file to be acted upon by the command is "Project", where "Project" is a standard alphanumeric file label containing 1-8 characters beginning with an alpha character and containing no special characters or imbedded blanks.

[USER(userid)] is an optional parameter indicating the TSO "userid" of the owner of the reference PROJECT (project) if other than the current user. Default for "userid" is the current TSO logon "userid".

[VERSION(XX)] is an optional parameter indicating which version of AISIM software is to be invoked by subsequent commands. The default value of VERSION is the last VERSION parameter specified. If never specified explicitly by a VERSION parameter the default value is "PROD", the current production version.

Every use of ANALYZE, BACKUP, CHANGE, DESIGN and RESTORE with the VERSION parameter changes the default value of "XX" for subsequent commands.

The value of XX is either "PROD" to select the current production Version, or a 2 digit integer to select an alternate version.

FUNCTION: Initiate a system design session

DESCRIPTION: This command transfers the user from the AISIM Ready level to the DESIGN User Interface (DUI) Level.

The following information and prompt are displayed for this command:

```
AISIM READY  
d p(test)
```

CURRENT PARAMETERS IN EFFECT:

VERSION	VERSION 1
PROJECT:	Project name from project specified in the command or default. In this case it is test.

USER: User id from command or default

Enter YES to proceed, NO to abort....

Typing YES causes the completion of the level transfer;
NO returns the user to the AISIM Ready Level.

COPYING DATABASE.....

.....COPY COMPLETE

Level transfer is then completed.

Use of the parameters PROJECT or VERSION changes
the respective default values for subsequent commands. Upon
completion of the level transfer the user is prompted with
a * which signifies the DUI Ready state.

EDIT

5.1.5 VIEWING OUTPUT REPORTS To access the model simulation report interactively on the terminal (via the TSO EDITOR), enter the following command:

```
EDIT [PROJECT(project)] [BUFFER(buffer)]
```

E

or

```
EDIT [PROJECT(project)] [TRACE]
```

E

where:

Parameters are described in paragraph 5.2.

Result:

The TSO EDITOR is entered with the database file to be edited set according to the PROJECT. All TSO EDITOR commands can be used on this file. The file is either the PROJECT report file (this is the default), or the PROJECT trace file, or the PROJECT buffer file.

END

5.1.6 TERMINATION OF AISIM READY LEVEL After system interaction at Level 3 and lower levels, the user may return to Level 2 by typing the command:

END

The system will return to the TSO Ready Level and the screen will display,

READY

GRIPE

5.1.7 REGISTERING A COMPLAINT

To record a complaint, comment, etc. in the AISIM system log, enter the following command:

GRIPE

The system will prompt the user for all required input.

HCOPY

5.1.8 HARDCOPY OUTPUT OF THE PROCESS FLOW-CHARTS To obtain hardcopy graphics of all or some of the Process flow-charts for an AISIM Model enter the following command:

```
HCOPY [PROJECT(project)] [USER(userid)] [VERSION(XX)]  
[DEST(location)] [GDEST(location)]
```

where:

Parameters are described in paragraph 5.2.

Result:

Depending on responses to system queries, database information is printed or plots are generated from one or more of all Processes.

Examples of Use:

User types: hcopy

System responds: ** 'PROJECT' IS UNDEFINED **
ENTER PROJECT: gera

```
CURRENT PARAMETERS IN EFFECT:  
VERSION:          PRODUCTION  
PROJECT:          GERA  
USER:             TF01508  
ENTER YES TO PROCEED, NO TO ABORT.....
```

Note that the entry "gera" in response to "ENTER PROJECT" was entered by the user.

5.1.9 OBTAINING HELP FROM THE SYSTEM

To obtain help from the system, type the following command:

HELP [COMMAND(command)]

where:

Parameters are described in paragraph 5.2.

LIBRARY

5.1.10 EXERCISING THE LIBRARY FACILITY

The Library User Interface is entered by issuing the command,

LIBRARY

This enables access to the MERGEIN, MERGEOUT, CHECKIN and CHECKOUT sublevels.

LIST

5.1.11 LISTING THE CURRENT OPTIONS

To list the CURRENT OPTIONS in effect, type the following command:

LIST

L

The system will display the CURRENT OPTIONS in effect, including VERSION, PROJECT and USER.

PRINT

5.1.12 PRINTING OUTPUT REPORTS

To request printing of the model output report, type the following command:

```
PRINT [PROJECT(project)]
```

```
P      P(project)
```

where:

Parameters are described in paragraph 5.2.

Result:

The output report (Project.REPORT.DATA) of a project is printed. This is a report of the standard results of a simulation run.

Example of Use:

User types: print p(gera)

System Responds: GERA.REPORT.DATA PRINTED (UNLESS OTHERWISE NOTED)

REPLOT

5.1.13 INITIATING A REPLOT SESSION

The Replot Function allows the user to display plots which were saved during previous Analyze Function runs. The command to invoke the Replot Function is as follows:

```
REPLOT [PROJECT(project)] [USER(userid)] [VERSION(XX)]  
      P(project)          U(userid)          V(XX)
```

where:

[PROJECT(project)] is an optional parameter indicating that the Replot session is to use the database "project" where "project" is a one to eight character name of a data base used in a previous analyze session.

[USER(userid)] is an optional parameter indicating the TSO "userid" of the owner of "project" if other than the current user. Default for "userid" is the current TSO logon "userid".

[VERSION(XX)] is an optional parameter indicating which version of AISIM software is to be invoked by subsequent commands. The default value of VERSION is the last VERSION parameter specified. If never specified explicitly by a VERSION parameter, the default value is "PROD", the current production version.

The system will respond with the following display:

```
CURRENT PARAMETERS IN EFFECT:  
VERSION:      PRODUCTION  
PROJECT:      data base  
USER:         TF00000  
ENTER YES TO PROCEED, NO TO ABORT...
```

A "yes" response will cause the Replot Function to be invoked and a "\$" prompt to be displayed.

A "no" response will cause the user to be returned to the AISIM READY level.

RESTORE

5.1.14 RESTORING A DATABASE (AFTER A CATASTROPHE HAS OCCURRED)

To restore a previously backed-up database (only necessary if a catastrophe has occurred which altered the PROJECT database ,or it is desirable to restart a model from a known configuration), enter the following command:

```
RESTORE [PROJECT(project)] [USER(userid)]
```

where:

Parameters are described in paragraph 5.2.

This command is used in conjunction with the BACKUP command. If the user was editing the original database and he had issued a BACKUP command against this database, then a copy of the original database exists. The RESTORE command causes the damaged original database to be replaced with the backup copy.

6. DESIGN USER INTERFACE (DUI)

The DUI and its lower levels are used to define the system model by creating, modifying, or deleting AISIM model entities. This is accomplished through the use of commands available to the user and by use of the "forms mode" which is a function of the HP-2647A terminal. The accompanying figure shows the commands available at the DUI Level and the results of using these commands.

The Process entities which represent operations in the modeled system are created and edited at a sublevel of the DUI Level called the Process Editor Interface (PEI). A system architecture and its related Legal Path Tables, nodes, and links are defined and edited in the Architecture Design Editor (ADE) level. The Clock and Timeunit entities are defined indirectly through the Scenario entity and are otherwise invisible to the user. The Constant, Item, Load, Process, Queue, Resource, Scenario, Table and Variable entities are all created and edited at the DUI level.

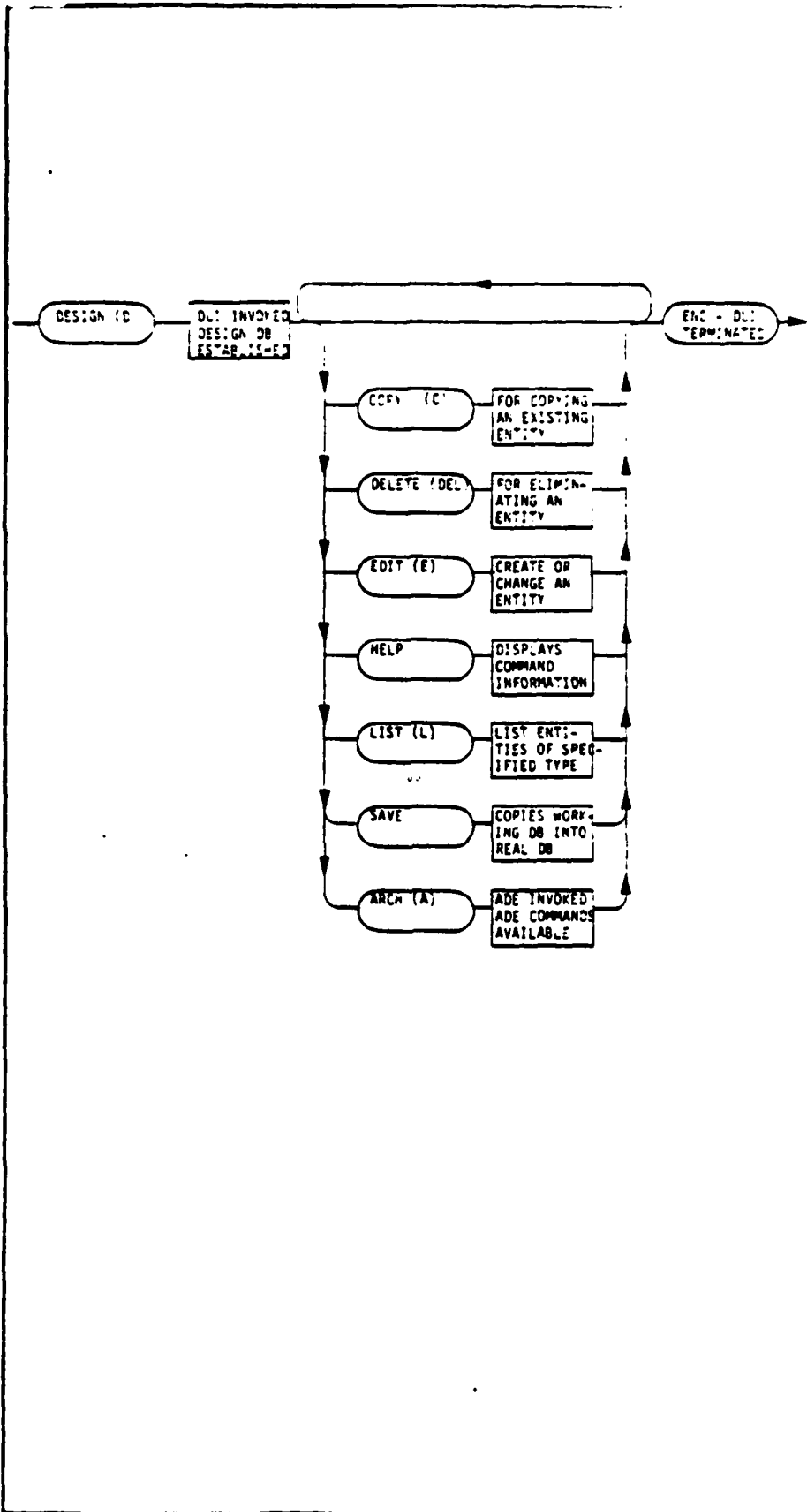
When creating and editing entities in the DUI level, the system prompts the user for further information by use of the "forms mode", which is a function of the HP-2647A terminal. Each form specifies the required and optional attributes of its respective entity-type. The areas on which information is to be entered appear in "reverse video" (dark characters on a light background), and indicate the attributes that are to be supplied by the user.

Each time the user presses the keyboard TAB key, the character cursor is positioned to the start of another designated area. The user enters parameters requested by the form by keying in the desired alphanumeric information. If the user changes his mind about the parameters previously keyed in, he may alter them by merely writing over the old information. When the user is satisfied with the contents of the form, he inputs it to the computer by pressing the ENTER key (as opposed to the RETURN key).

The AISIM DUI commands, used to input, modify, and delete entities from the model, are illustrated in the accompanying figure and described on the pages that follow it.

HUGHES

DATE	TIME	DATE	TIME
1981	10	1981	10
1	2	3	4
5	6	7	8
9	10	11	12



DATE	TIME
1981	10
1	2
3	4
5	6
7	8
9	10

DUI COMMAND SUMMARY

COMMANDS:

ARCH

A

COPY {entity-type},{existing-name},{new-name}

C

DELETE {entity-type},{entity-name}

DEL

EDIT {entity-type},{entity-name},{old/new}

E

END

HELP

LIST {entity-type}

L

SAVE

6.1.1 COMMAND: ARCH

The ARCH command is used to invoke the Architecture Design Editor (ADE).

This command is valid only in the DUI Ready Level.

COMMAND SYNTAX:

ARCH

A

FUNCTION RESULT:

The ADE is invoked so that the architecture is built under the project designated by the DESIGN command. A # prompt is provided for the user to input ADE commands.

6.1.2 COMMAND: COPY

The COPY command is used to create a copy of an existing entity.

COMMAND SYNTAX:

COPY {entity-type},{existing-name},{new name}

C

where:

{entity-type} is a required parameter indicating any valid entity type.

{existing-name} is a required parameter identifying the existing entity whose parameters are to be duplicated.

{new-name} is a required parameter which specifies the name of the new entity whose parameters are duplicates of the "existing entity".

DESCRIPTION: ENTITY-TYPE may be any of the following:

<u>Entity-type</u>	<u>Acceptable Abbreviation</u>
Action	A
Constant	C
Item	I
Load	L
Process	P
Queue	Q
Resource	R
Scenario	S
Table	T
Variable	V

If entity type, existing-name or new-name is missing or invalid, the user is prompted.

A carriage return entered in response to any prompt aborts the command and returns the user to the DUI Ready state - * prompt.

6.1.3 COMMAND: DELETE

The DELETE command is used to eliminate a named entity of a given type from the user database. The only restriction on the use of this command is that Resources associated with architectural nodes or links cannot be deleted outside of the Architecture sub-level.

This command is valid only at the DUI Ready Level.

COMMAND SYNTAX:

```
DELETE {entity-type},{entity-name}
      {entity-type},*
      {entity-type},{entity-name},...,{entity-name}
```

DEL

where:

{entity-type} is a required parameter indicating any valid entity type.

{entity-name} is a required parameter indicating the name of the entity to be deleted. It is permissible to give a list of entity-names--of the same type-- each member of which is separated by a comma.

* is a parameter used indicate all of the entities of the specified type are to be deleted.

If entity-type or entity-name is missing or invalid, the user is prompted for a valid parameter.

A carriage return in response to the prompt aborts the command, and the user is returned to the DUI Ready state - * prompt.

FUNCTION RESULT:

If the named entity is not a Resource associated with a architectural node or link, the entity will be deleted from the user's "working" database. If the entity is a Resource associated with a node or link, the user will be given the message:

" entity " IS ASSOCIATED WITH THE ARCH. AND CAN ONLY BE DELETED IN ADE

where " entity " is the name of the entity to have been deleted.

When there is more than one such Resource listed in the command to delete the user will be given the above message for each one.

6.1.4 COMMAND: EDIT

The EDIT command is used either to create an entity, or to change an existing entity. If the entity requested is Process the Process Editor Interface (PEI) is invoked.

This command is valid only at the DUI Ready Level.

COMMAND SYNTAX:

EDIT {entity-type},{entity-name},{old/new}

E

where:

{entity-type} is a required parameter indicating any valid entity type, as described in sections 3.2 and 5.

{entity-name} is a required parameter indicating the name of the entity to be edited.

{old/new} is an optional parameter indicating that the named entity is to be created (NEW), or that the named entity exists (OLD) and is to be changed. If the [OLD/NEW] parameter is entered incorrectly, the user is prompted for a valid entry. The default for this optional parameter is old.

FUNCTION RESULT:

If the entity-type specified is Process, the PEI level is automatically invoked. The user is then presented with a form to describe the Process. The user must fill out the form, and then use the "ENTER" key to input the Process into the "working" database. The user is left in the PEI level in order to complete the Process definition.

If any other valid entity type is specified, the user is presented with a form to describe that entity. The user must fill out the form, and then use the "ENTER" key to input the completed entity into the "working" database. The user is then returned to the DUI Ready state - * prompt.

6.1.5 COMMAND: END

The END command is used to terminate a DUI session.

The command for the above purpose is valid only in the DUI Ready Level.

COMMAND SYNTAX:

END

FUNCTION RESULT:

The DESIGN session is ended. The "working" database is closed. If a SAVE command has not been given since the last EDIT command the user is asked if the working database is to be saved. The query is:

SAVE (Y/N)

If the user answers "Y", the working database is saved into the "real" database and the session is ended. Control is passed to the AISIM Ready level (level 3). If the user answers "N", the session is ended and the working database is not saved. Control is passed to the AISIM Ready level (level 3). Depressing the RETURN key in response to the SAVE query aborts the END command, and returns the user to the DUI Ready State - * prompt.

6.1.6 COMMAND: HELP

The HELP command lists the commands currently available to the user during a DUI session.

This command may be used any time during a DUI session.

COMMAND SYNTAX:

HELP

FUNCTION RESULT:

The acceptable commands (i.e., the ones valid at the current level) are listed following the last line appearing on the terminal.

HELP displays the following commands:

LIST	L	DELETE	DEL	EDIT	E	SAVE
ARCH	A	COPY	C			

6.1.7 COMMAND: LIST

The LIST command displays all entities of a specified type. Included with each entity is its name and a description.

This command is valid only at the DUI Ready state - * prompt.

COMMAND SYNTAX:

LIST {entity-type}

L

where:

{entity-type} is a required parameter indicating any valid entity type.

If {entity-type} is missing or invalid, the user is prompted for a valid entity type.

A carriage return entered in response to the prompt aborts the command, and the user is returned to the DUI Ready state - * prompt.

FUNCTION RESULT:

The user is presented with a list of all existing entities of the requested type.

6.1.8 COMMAND: SAVE

The SAVE command copies the contents of the working database into the users's permanent database.

This command is valid only at the DUI Ready state - * prompt.

COMMAND SYNTAX:

SAVE

FUNCTION RESULT:

The real database is replaced with the contents of the work database, and the user is returned to the DUI Ready state - * prompt. The command is useful when the user is defining a large system. With the SAVE command the user saves the model design up to the point at which the command is given. This protects that portion of the design from computer failures.

1

6.1.9 TERMINATION OF A DUI SESSION As mentioned earlier, a DUI session is terminated by issuing the END command. Syntax and results are described in the preceeding section. The DUI session is ended. The "working" data base is closed. If a SAVE command has not been given since the last EDIT command, the user is asked if the working data base is to be saved. The query is:

SAVE (Y/N)

If the user answers "Y", the working database is saved into the "real" database and the session is ended. If the user answers "N", the session is ended and the working database is not saved. Depressing the RETURN key in response to the SAVE query aborts the END command, and returns the user to the DUI Ready state. When the SAVE query is answered, control is returned to the AISIM Ready Level and the AISIM READY prompt is displayed.

6.2 PROCESS EDITOR INTERFACE (PEI)

The PEI, coupled with the capabilities of the HP2647A terminal, allows the user to describe graphically the logical flow of an operation which he wishes to model. The PEI is used to build "Processes" which model man-machine interaction as well as data Processing functions (software logic). A Process is composed of "Primitives" which are symbols that represent the individual steps in an operation. Using the PEI commands which are described below the user arranges the Primitives in an order that describes the Process. The Processes represent operations or activities in the modeled system and are interpreted by the simulator. Simulation of the Processes over time is controlled by Loads and Scenarios. Depending on the Primitives composing the Processes, Resources may be allocated and deallocated. Given limited Resources and the use of common Resources by Processes, contention for Resources is modeled. The simulator evaluates all these factors of timing, events and Resource contention which are contained in the Processes, Loads and Scenarios, and dynamically controls the simulation.

6.2.1 USE OF THE PEI The system transfers control to the PEI when the user issues the following command:

```
EDIT {Process},{entity-name},{old/new}
```

```
E {P},{entity-name},{old/new}
```

where:

[Process] is a required parameter.

[entity-name] is required parameter indicating the name of the Process to be edited.

[old/new] is an optional parameter indicating that the named entity is to be created [new] or that the named entity exists [old] and is to be changed. If the [old/new] parameter is entered incorrectly, the user is prompted for a valid entry. The default for this optional parameter is old.

To enter the PEI, the user must issue the EDIT PROCESS command while at the DUI level.

When the PEI is entered the screen is blanked. A set of reference lines are drawn on the screen. If a Process has already been created, the graphic for the Process is displayed. If the Process is new, the screen is again blanked and a form is displayed which requests information about the Process. The user must complete the form and then use the "ENTER" key on the HP2647A to input the task into the database. The form is cleared from the screen and the START and END Primitives of the new

Process are displayed on the right side of the screen. At this point, a pound sign (#) prompt will be displayed indicating that the user may issue any of the PEI commands. The PEI commands, which are described on the following pages, are used to select, position, and describe the Primitives to create a Process.

PEI COMMAND SUMMARY

COMMANDS:

BOTTOM

B

Change {position}

C

DELETE {position},[number of consecutive positions]

DEL

DOWN [number of positions]

D

END

HELP

HOLD {position}

H

MENU

M

PLACE {primitive-type},{position}

TOP

T

UP [number of positions]

U

NOTE: The positioning commands (Bottom, Down, Top, Up) are necessary because only six Primitives of a Process can be displayed on the screen at a time and a Process could contain many more primitives.

6.2.2 COMMAND: BOTTOM

The Bottom command is used to display the last six positions of the current Process structure.

This command is valid only in the PEI level.

COMMAND SYNTAX:

BOTTOM

B

FUNCTION RESULT:

The last six positions of the Process structure being edited are drawn from the END symbol up. The END symbol is always the last position of a Process structure.

6.2.3 COMMAND: CHANGE

The CHANGE command is used to modify the user defined attributes of a Primitive at a specific position within the current Process structure.

This command is valid only in the PEI level

COMMAND SYNTAX:

CHANGE {position}

C

where:

{position} is a required parameter indicating the position of the Primitive, within the Process structure, whose attributes are to be changed.

The Primitive which is to be changed must be on-screen.

FUNCTION RESULT:

When the CHANGE command is invoked, the user is presented with a form corresponding to the Primitive at the indicated position. The user may change any or none of the attributes of the Primitive, and then must use the "ENTER" key to input the contents of the form. The Process structure is then redisplayed with the changes made.

NOTE: Even if no changes are made to a form, the user must depress the ENTER key to continue.

6.2.4 COMMAND: DELETE

The DELETE command allows the user to delete a single Primitive, or a range of Primitives, from the current Process structure.

This command is valid only in the PEI Level.

COMMAND SYNTAX:

DELETE {position},[for number of consecutive positions]

DEL

where:

{position} is a required parameter indicating the position of the Primitive to be deleted.

[for number of consecutive positions] is an optional parameter indicating the range of positions to be deleted, starting with the Primitive indicated by the [position] parameter. If this parameter is omitted the default condition is to delete the Primitive at the position indicated by the required parameter.

FUNCTION RESULT:

The Primitives indicated by the position parameter and the optional parameter are deleted from the Process structure. The remaining Primitives in the structure are scrolled up. Neither the START nor the END symbol may be deleted, and the remaining symbols may only be deleted if they are on-screen.

6.2.5 COMMAND: DOWN

The DOWN command allows the user to scroll the current Process structure down an indicated number of positions.

This command is valid only in the PEI Level.

COMMAND SYNTAX:

DOWN [number of positions]

D

where:

[number of positions] is an optional parameter indicating the number of positions that the structure is to be "scrolled". If this parameter is not used, the default condition is to scroll the Process structure down six positions, which is analogous to displaying the next page.

FUNCTION RESULT:

The Process structure is scrolled down the number of positions indicated by the optional parameter, if given. Otherwise the structure is scrolled down six positions or to the bottom of the structure if less than six positions follow the last position currently displayed.

6.2.6 COMMAND: END

The END command is used to terminate and exit the PEI session.

COMMAND SYNTAX:

END

FUNCTION RESULT:

The PEI session is ended, the graphics display is erased, and the user is returned to the DUI Level.

6.2.7 COMMAND: HELP

The HELP subcommand displays a list of the valid PEI commands.

FUNCTION RESULT:

The list of valid commands (see PEI Command summary) is displayed at the end of in-use alphanumeric memory.

6.2.8 COMMAND: HOLD

The HOLD command allows the user to insert any valid Primitive, which is already a part of the current Process structure, into the menu item "HOLD" so that it may be replicated.

This command is valid only in the PEI Level.

COMMAND SYNTAX:

HOLD {position}

H

where:

[position] is a required parameter indicating the position of the Primitive which is to be placed in hold for the purpose of replication.

FUNCTION RESULT:

The Primitive indicated by the parameter is placed in hold. This item may then be replicated by using the PLACE command and using HOLD as the menu-item to be placed. The menu, if displayed, will contain a description of the Primitive currently in hold. When a primitive is stored in Hold, it remains there, accessible to the user throughout the DUI session, and thus primitives may be moved from one Process to another,

6.2.9 COMMAND: MENU

MENU is used to display the valid Primitives of a Process.

COMMAND SYNTAX:

MENU

M

FUNCTION RESULT:

The menu is drawn to the left of the Process flowchart. After the MENU command has been given, the menu will automatically appear whenever a Process is edited. Once issued, the menu command is irreversible and will appear for the remainder of the Design session.

6.2.10 COMMAND: PLACE

The PLACE command is used to put a Primitive at a position in a Process. The Primitive may be placed in any position within the Process structure except prior to the START symbol or after the END symbol. The default placement is immediately prior to the END symbol.

COMMAND SYNTAX:

PLACE {symbol},[position]

P where:

{symbol} is a required parameter, indicating any valid Primitive menu item.

[position] is an optional parameter indicating any valid position within a Process structure, (i.e. after the START symbol and before the END symbol). The position of a Primitive in a Process is indicated by the numbered column to the left of the flow-chart representation of a Process.

FUNCTION RESULT:

The user is presented with a form that corresponds to the Primitive to be placed.

When the form has been completed, the Process is redrawn, if necessary. The Primitive is placed at the position indicated by the position parameter, if given, and all following Primitives are moved down one position. If the position parameter is omitted, the Primitive is placed immediately prior to the END Primitive. If the Primitive was placed off-screen, the Process is not redrawn, and the user does not see the placement of the Primitive.

/

PEI / TOP

6.2.11 COMMAND: TOP

The TOP command is used to display the first six positions of the current Process structure.

This command is valid only in the PEI level.

COMMAND SYNTAX:

TOP

T

FUNCTION RESULT:

The first six positions of the Process structure being edited (or the entire Process if the structure consists of no more than six primitives) are drawn from the START symbol down. The START symbol is always the first position of a Process structure.

6.2.12 COMMAND: UP

The UP command allows the user to scroll the current Process structure up an indicated number of positions.

This command is valid only in the PEI level.

COMMAND SYNTAX:

UP [number of positions]

U

where:

[number of positions] is an optional parameter indicating the number of positions that the structure is to be "scrolled". If this parameter is not used, the default condition is to scroll the Process structure up six positions, which is analogous to displaying the previous page.

FUNCTION RESULT:

The Process structure is scrolled up the number of positions indicated by the optional parameter, if given. Otherwise the structure is scrolled up six positions or to the top of the structure if less than six positions precede the first position currently displayed.

6.2.13 TERMINATING A PEI SESSION Only one Process can be created or edited during a PEI session. To create or edit other Processes or change to another level the user must terminate the current PEI session and return to the DUI level. This is accomplished by giving the END command described earlier. The current working database is left open and control is transferred to the DUI level. The DUI ready and prompt is displayed.

6.3 ARCHITECTURE DESIGN EDITOR (ADE)

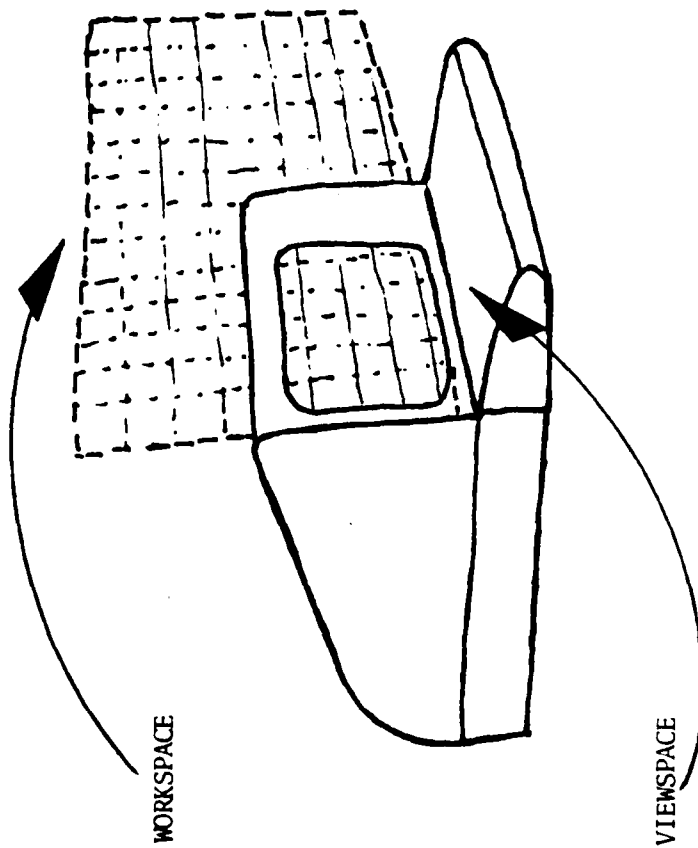
The ADE is used to define the layout and interconnection of the physical aspect of a data processing network. It is not necessary to develop an architecture model if the user wishes to model operations without regard to where these operations take place. However, if items are routed through a system or if Processes at one location trigger Processes in another then an architecture model is necessary.

The ADE, with the capabilities of the 2647A terminal, allows the user to create graphically a picture of the system architecture by positioning symbols and connections. It also allows the user to define the legal paths of communication between the connections (and along the connections).

Even if a user has defined a Legal Path Table while creating an architecture, the system offers the option of automatically building a Legal Path Table. The user is queried to resolve any ambiguities. The Legal Path Table is used during the simulation to control the routing of items that are being passed through the system.

It is important to note that each node and link represented in the architecture is intended to represent some system Resource such as a CPU, disk drive, tape drive, or channel. The system automatically creates model Resources for these system Resources. The parameters of such Resources can be altered both in ADE--though the DEFINE command--and in the DUI --with the EDIT command.

USED AT:	AUTHOR:	DATE REV.	WORKING	READER	DATE	CONTEXT:
	PROJECT:					
NOTES: 1 2 3 4 5 6 7 8 9 10			DRAFT			
			RECOMMENDED			
			PUBLICATION			



NODE:	TITLE:	NUMBER:
	VIEWSPACE/WORKSPACE CONTRAST. Figure 5.	

6.3.1 CONCEPTS FOR USING ADE This section is intended to familiarize the user with the capabilities of the ADE so that he may better understand the description of its use in sections further below.

The view space is divided by vertical and horizontal grids. Grid lines running vertically mark off the position and are numbered starting with one at the left side. Grids running horizontally mark off the Y position and are identified with numbers, starting with 1 at the bottom. Another aid to building the architecture is variable symbol size. The user can specify the size of symbols as he positions them in the view space. The user is provided with commands to change his view screen position, to position nodes which represent system Resources, to delete nodes, and to change symbol names and sizes. A command is provided which allows the user to specify connections between nodes. These connections (or links) are defined as system Resources. Any two nodes may be connected by more than one link. (Exception: When using Method A, B, or C algorithms to define the Legal Path Table, two node types "TTY" and "LOD" are considered "leaf-nodes" and should have only one connection to one other node. See section 6.3.15.) The architecture developed using the ADE becomes the basis for generating the Legal Path Table which is used to route Items through a system.

The view screen on the HP-2647A terminal is approximately five inches high by eight and one-half inches wide. This workspace is too small for some systems. The ADE, therefore, gives the user a workspace which is thirteen and two-tenths inches high by 20 inches wide and allows the user to move the viewspace anywhere in this workspace to construct the architecture. The contrast between viewspace and workspace is illustrated in the accompanying figure.

6.3.2 USE OF THE ADE The ADE can only be accessed from the DUI level. The ADE level is entered by issuing the following command:

ARCH

A

where:

Only one architecture is allowed per design database. This prevents the user from specifying an architecture structure that does not relate to the Processes and Resources that have been defined. Experiments using common Processes, Resources, etc. with different architectures can be run by following the procedure listed below:

- 1) While in the TSO Ready Level, COPY the project.DBF data file

to newproject.DBF data file where: project and newproject are names of PROJECT databases for AISIM models.

2) Enter AISIM and use the ADE to edit the architecture contained in newproject.DBF.

3) Simulations can now be run using the newproject database.

If there is no architecture defined in the design database, the system will provide a blank grid on the screen and a pound sign (#) prompt for the user to enter commands. If an architecture has already been defined, then the old architecture will be displayed and the user will be provided a pound sign (#) prompt for entering commands. The following pages give a summary of commands available in the ADE and their use. These commands are legal in only the ADE level.

/

ADE COMMAND SUMMARY

COMMANDS:

CHANGE NAME {node1},{name2}

CHANGE TYPE {node1},{type}

CHANGE SIZE {node1},{size}

CHG

CONNECT {node1},{node2},{link}

CON

DEFINE {symbol-type},{Resource-name}

PATH {node1},{node2},{Link1},...,{Linkn}

DEF

DELETE {name1},{name2},...,{name3}

DEL

END

LIST PATH

LPT

L

MOVE {node1},{x-position},{y-position}

M

PLACE {node-type},{node},{x-position},{y-position},{size}

P

RECON {link}

R

SAVE

WINDOW {direction},{number-of-grids]

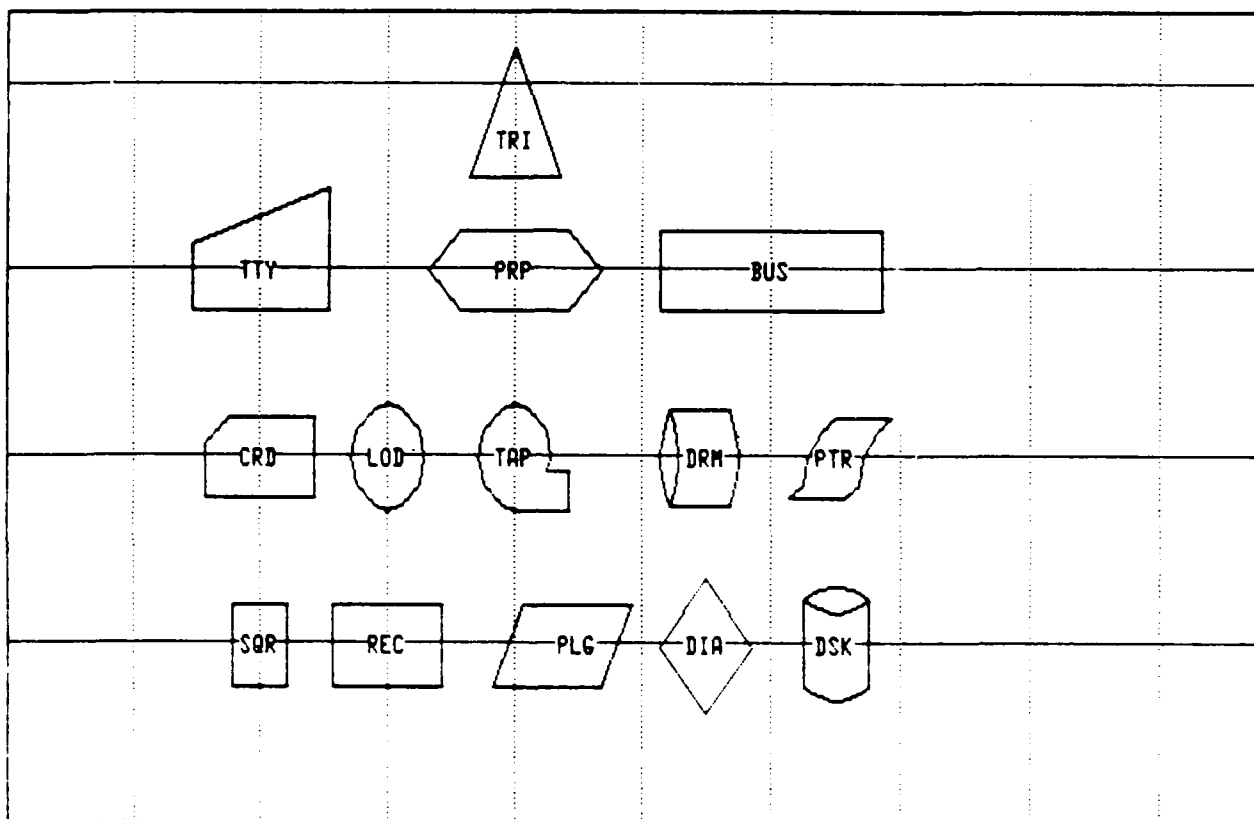
W

6.3.3 ADE SYMBOLS Symbols used to construct an architecture are generic in nature. The shape associated with symbols is representative of a computer system's hardware elements although no implicit attributes of computer hardware elements are given to the symbols. Attributes defined for a symbol which make it represent an actual physical device must be defined by the user. Attributes are attached to symbols by the DEFINE command.

Symbols in an architecture correspond directly with Resources . This relationship applies to nodes and links. All symbols which are directly connected correspond to an entry in the Legal Path Table.

One other implied relationship applies to the symbols in an architecture. The symbols TTY and LOD are considered to be "terminal" symbols by the Legal Path Table . Therefore , these two symbols have a constraint that they be only connected with one link to one of the other symbol types. Also, TTY and LOD symbols can not be directly connected. (These constraints are enforced by the LPT generation not the ADE.)

The complete symbol set for AISIM architecture is shown in the accompanying figure.



6.3.4 COMMAND: CHANGE

The CHANGE command allows the user to modify the name, type, or size of an ADE symbol which represents an architecture mode.

COMMAND SYNTAX:

CHANGE NAME, {name1},{name2}

CHANGE TYPE, {name1},{type}

CHANGE SIZE, {name1},{size}

CHG

where:

{name1} is a required parameter indicating the name of the symbol which is to be changed where: name1 should be 1-8 alpha numeric characters. For the commands CHANGE TYPE and CHANGE SIZE, name must designate a Node.

{name2} is a required parameter specifying a new name for the current symbol of name1 where: name2 should be 1-8 alpha numeric characters.

{type} is a parameter specifying that the symbol of name1 is to be changed from its current type to "type" which is one of the legal symbol types.

{size} is an optional parameter specifying that the symbol of name1 is to be changed from its current size to "size" where: size can be the number of letters of the name (up to 20). If no size is specified the default size for the symbol is a width equal to the number of characters in its name.

FUNCTION RESULT:

The indicated changes are made to the symbol "name1". When the user changes a symbol type or size there is no impact on the other parameters. When the name is changed, the default size is the width of the name.

6.3.5 COMMAND: CONNECT

The CONNECT command is used to show connections between architecture nodes by placing links between them.

COMMAND SYNTAX:

CONNECT {node1},{node2},{link}.[F]

CON

where: {node1} is a required parameter indicating the first symbol of a from-to pair of symbols to be connected and where: node1 is 1 to 8 alpha or numeric characters.

{node2} is a required parameter indicating the second symbol of a from-to pair of symbols which are to be connected and where: node2 is 1 to 8 alpha or numeric characters.

{link} is a required parameter indicating the link of the connection which is to be made and where: link is 1 to 8 alpha numeric characters.

[.F] is an optional parameter appended directly to link indicating that the communication link between nodes, node1 and node2, is full-duplex. The effect of this is to create two links, a "link.A" and a "link.B". Links defined without this parameter bear a half-duplex default.

FUNCTION RESULT:

If node1 is not in the viewspace when the command is issued, the user will be prompted with the message,

THE FROM NODE MUST BE ON THE SCREEN TO ESTABLISH CONNECT: COMMAND ABORTED:

If node1 is on the viewspace, a cursor (+) is displayed at node1. At this point, the user has two alternatives:

1) he may cause the system to connect the two symbols with a straight line through their centers by depressing the carriage return again (cr) or,

2) he may cause the system to produce a shaped line segment from symbol 1 to symbol 2 by:

- a) moving the cursor using the HP2647A graphics controls, to a position where he wishes to bend the line,

- b) typing a period (.),
- c) repeating a) and b) until a maximum of six corners have been created.
- d) completing the line segment from the last corner to symbol 2 by entering a carriage return (cr).

Alternative 2 allows the user to place symbols randomly and later show connections that would be obscured or confusing if generated by Alternative 1. Connections can be straightened or have corners added to them with the RECON command.

In addition, two entries are entered in the Legal Path Table. The first is an entry for the path from node1 to node2 via link, and the second entry specifies a path from node2 to node1 via link. If link is defined as full-duplex, then the path from node1 to node2 uses "link.A", while the path from node2 to node1 uses "link.B". (See section on "Define" command). Node1 is then established as the link's from node and node2 is established as the link's to node. All subsequent paths using this full-duplex link will use "link.A" if they go in the direction of the from node to the to node and will will use "link.B" if they go in the opposite direction.

6.3.6 COMMAND: DEFINE

The define serves two functions. It is used to define attributes to be associated with symbols (this allows the user to make the logical assignment of physical device characteristics to the Resource). DEFINE is also used to indicate the legal path between nodes in the architecture.

COMMAND SYNTAX:

```
DEFINE {symbol-type},(Resource-name)
```

```
    DEFINE PATH {node1},{node2},{link1},...,{linkn}
```

```
    DEF P
```

where:

{symbol-type} is the symbol type (sqr,dia,lod,ttt,etc.) for which the user wishes to define attributes.

(Resource-name) is an optional parameter that specifies the name of an existing Resource from which the symbol-type attributes are to be copied.

{node1} is the name of the node from which the Legal Path is to run.

{node2} is the name of the Node to which the path is to run.

{link1} to {con-n} are the names of the links along which the legal path between the two nodes node1 and node2 is to run.

FUNCTION RESULT:

If the DEFINE command is issued with the format

```
    DEFINE {symbol-type}
```

a form will be displayed that shows the parameters currently assigned to this symbol type. The user may modify these parameters as desired. If the ATTRIBUTES PRESENT field contains "yes" then a form will be displayed allowing the user to define attributes and their values. After symbol attributes have been defined, the Resource automatically created in association with the symbol will be given the attributes that were defined for that symbol type.

If the syntax of the command is:

1

DEFINE {symbol-type},(Resource-name)

the system will present the user with a form to be filled with the top level attributes of the named Resource. The user can check the data and/or modify it. When entered, the data last displayed in the form will be used to create the top level attributes of the symbol type. If the ATTRIBUTES PRESENT field is "yes" the attributes of the named Resource will be displayed. This data can be checked and/or modified. When the form is entered the data list displayed will be used to create the secondary attributes of the symbol type.

If the syntax of the command is,

DEFINE PATH {node1},{node2},{link1},...,{linkn}

DEF P

Entries in the Legal Path Table will be made (which can be inspected with the LIST command).

There are several rules constraining the creating of a legal path in ADE.

First, a point-to-point path is a path between two nodes that are separated from one another by a single Link, i.e., there is no other node between them. Thus, in the architecture shown, the path between node1 and node2 is a point-to-point path.

Secondly, a sub-path of a given path is any one of the segments of the path that go to the same node as the path but from any one of the nodes the original path passes through. For example, in the architecture shown, a defined legal path from node1 to node4 will have the following sub-paths: (1) from node2 through node3 to node4 and (2) the point-to-point path from node3 to node4. The path from node1 to node3 through node2 is not a sub-path of the original path because it does not go to the same node as the original.

With these two definitions, we can state five quite general rules governing the definition and deletion of legal paths. They are:

1) A Legal Path between two nodes is a collection of Legal Path Table entries of the form:

FROM	TO	NEXT	LINK
------	----	------	------

which indicates that the path from node FROM to node TO goes to node NEXT via link LINK.

- 2) There may be only one Legal Path between any two nodes.
- 3) There must be a path between any two nodes that are directly connected.
- 4) Use of the two links implied by a full-duplex name for a connection follows these rules:

a) When a connection Con.F is established (actually Con.A and Con.B) with the command,

CONNECT NODE1,NODE2,CON.F

Node1 is established as the from node for that connection and Node2 is established as the to node.

b) Any path which uses the connection CON.F in the direction from its from node to its to node will use CON.A.

c) Any path which uses the link CON.F in the direction from its to node to its from node will use CON.B d) Establishing the connection between two nodes implicitly defines a point-to-point path between them.

These five rules have a number of restriction of which the user should be aware:

- 1) Defining a path from one node to another implies defining paths from all nodes along the path to the last node in the path.
- 2) Changing a path (redefining, deleting) changes any other paths that use it as a sub-path.
- 3) A point-to-point path cannot be deleted.
- 4) When a path between two directly connected Nodes is deleted a point-to-point path is automatically restored.
- 5) Deleting a node or link from an architecture removes any paths which use the deleted entity.
- 6) Changing the name of a node or link changes the name of the entities in the Legal Path Table as well.
- 7) Cyclic paths are not allowed.

6.3.7 COMMAND: DELETE

The DELETE command allows the user to delete nodes or links in the architecture or parts (or all) of the previously defined Legal Path Table (LPT).

COMMAND SYNTAX:

DELETE {name1},...,{name-n}

PATH {node1},{node2}

DEL

where:

{name1} is a required parameter that specifies the node or link to be deleted.

{name-n} is a an optional parameter which specifies an additional node or link to be deleted.

{node1} and {node2} are required parameters indicating the nodes between which the legal path is to be deleted.

FUNCTION RESULT:

When a symbol is being deleted, the symbol and all connections to it are erased from the screen and removed from the database. If a connection is being deleted, the connection is erased from the screen and is removed from the database.

When a path between node1 and node2 is deleted from the Legal Path Table, however, any sub-paths which are in this path are unaffected.

[NOTE: if the command DEL * is issued, the entire architecture will be deleted].

6.3.8 COMMAND: END

The END command is used to terminate the ADE session.

COMMAND SYNTAX:

END

FUNCTION RESULT:

The END command terminates the edit mode of the ADE session and automatically triggers the generation of a Legal Path Table (LPT). The user will be questioned as to the method of generation for the LPT and for information necessary to clear up ambiguities in its generation before control is returned to the DUI level.

6.3.9 COMMAND: LIST

The LIST command enables the user to list the legal paths that have been defined in the architecture.

COMMAND SYNTAX:

where:

{node1} is the name of the node at which the path to be listed begins.

{node2} is the name of the node at which the path is to end.

FUNCTIONAL RESULT:

If the command syntax is LIST PATH, a format like that below is displayed:

FROM: node3 TO: node2

PATH:
link1,link2,...,linkn

If the command syntax is LIST LPT, the entire Legal Path Table is displayed.

6.3.10 COMMAND: MOVE

The MOVE command allows the user to change the location of a Node in the architecture.

COMMAND SYNTAX:

MOVE {node},{x-position},{y-position}

M

where:

{node} is the name of the node to be moved.

{x-position} is the x-coordinate of the new position, i.e., the position to which the node is to be moved.

{y-position} is the y-coordinate of the new position, i.e., the position to which it is to be moved.

FUNCTION RESULT:

The node and all links to or from it will first disappear from the screen. The node will then be redrawn at the new position and the previously defined connections with other nodes will reappear.

6.3.11 COMMAND: PLACE

The PLACE command allows the user to position a legal ADE symbol in the view space at specified coordinates.

COMMAND SYNTAX:

PLACE {type},{node},{x position},{y position},{size}

P

where:

{type} is a required parameter which specifies one of the legal ADE symbol types .

{node} is a required parameter that indicates the name that is to be displayed and associated with this placement of a symbol and where: name is 1 to 8 alpha or numeric characters.

{x position} is a required parameter that specifies the horizontal position of the symbol relative to vertical grid number position 0. The "x position" must be within the limits of the view screen.

{y position} is a required parameter that specifies the vertical position of the symbol relative to horizontal grid position 0. The "y position" must be within the limits of the view screen.

[size] is an optional parameter specifying the size of the symbol to be placed. The default size is the number of characters in name. Legal sizes are any natural number.

FUNCTION RESULT:

A symbol of specified type appears on the view screen at the x, y positions indicated in the command. The symbol name appears within the symbol and the symbol size is regulated by the size parameter.

6.3.12 COMMAND: RECON

The RECON command allows the user to alter the shape of a given link, giving it corners, decreasing the number of corners it has, or adding to the number of corners it has.

COMMAND SYNTAX:

RECON {link}

RE

where:

{link} is the name of the link to be redrawn.

FUNCTION RESULT:

The link will disappear, but the cursor (+) will be displayed at the node defined as the from node (see CONNECT command). As with the CONNECT command, the user has two alternatives:

- 1) cause the system to connect the two symbols with a straight line through their centers by hitting another carriage return (cr) or,
- 2) cause the system to produce a shaped line segment from symbol 1 to symbol 2 by:
 - (a) moving the cursor using the HP2647A graphics controls, to a position where he wishes to bend the line,
 - (b) typing a period (.) and a carriage return (cr)
 - (c) repeating (a) and (b) until a maximum of six corners have been created.
 - (d) completing the line segment from the last corner to symbol 2 by entering a carriage return (cr).

6.3.13 COMMAND: SAVE The SAVE command copies the contents of the working database into the user's permanent database.

COMMAND SYNTAX:

SAVE

FUNCTION RESULT:

The permanent database is replaced with the contents of the working database, and the user is returned to the ADE Ready state - # prompt. This command is useful when the user is defining a large system because it allows the user to protect the work done up to the point of issuing the SAVE command.

6.3.14 COMMAND: WINDOW

The WINDOW command allows the user to move the view screen to any position within the legal view space.

COMMAND SYNTAX:

WINDOW {direction1},{n},{direction2},{n}

W

where:

{direction1} is a required parameter that specifies the direction to move the view screen. Legal directions are:

U
D
L
R

where: U = up
D = down
L = left
R = right

[direction 2] is an optional parameter that specifies the direction to move the view screen. Legal directions are:

U
D
L
R

where: U = Up
D = down
L = left
R = right

[n] is an optional parameter that specifies how many grid positions the view screen is to be moved from its present position. If "n" is not given, a default of half the screen width or height is assumed.

FUNCTION RESULT:

After the command has been issued the screen is cleared, new coordinates are calculated, and the screen is redrawn as seen from the new position. View screen coordinates do not change;

only view space coordinates. If the value of "n" is too large causing the view screen to go beyond the limits of the view space, the value of "n" will be truncated to prevent the system from exceeding the view space bounds.

When the ADE is first entered, the view screen is positioned at the upper left corner of the view space.

6.3.15 TERMINATION OF AN ADE SESSION The ADE session is terminated by issuing the command,

END

This completes the edit portion of the ADE session and begins a sequence of events that leads to a return to the DUI Level. Before control is returned to the DUI Level, however, the system gives the user the option of creating a new Legal Path Table. The Legal Path Table (LPT) created by the system is based upon the architecture that was created. The LPT consists of a two dimensional array. Entries in the array represent a means of getting from one node to another.

Entries contain two pieces of information:

- 1) the next node in the path from Node 1 to Node 2
- 2) the link used to get to the next node.

There are three basic methods of generating a Legal Path Table at the end of an ADE session. Therefore in response to the END command, the system questions the user:

BY WHICH METHOD DO YOU WISH TO GENERATE THE LPT (A,B, OR C)?

IF YOU HAVE AN ESTABLISHED LPT OR IF YOU WISH TO SKIP THIS STEP, TYPE "END"

IF YOU DESIRE MORE INFORMATION ON METHODS A, B, OR C, TYPE "INFO".

After the pound sign (#) prompt, the user may enter either "A", "B", "C", HELP, or "INFO". If the user enters END and a carriage return after this or any subsequent # prompts without responding to the previous prompt question, any currently defined LPT, including none, will remain in effect, and control will return to the DUI level.

If any of the three options are chosen, previously defined LPT's will be deleted from the database. Depending on whether A,B or C

is entered, the system response varies. One response is the same. Since these algorithms may take several minutes, the user is provided with a message that lets him know the system is progressing with the LPT. The prompt initially reads "Generating LPT 1". After so many routes have been found, the message will change to "Generating LPT 2" and so on. The following paragraphs discuss the individual processing performed in response to methods A, B, and C.

METHOD A - Method A directly connects adjacent nodes in the architecture but no other paths are generated. This method is used when message routing paths is not of interest in the model. This method requires the least processing time to generate the LPT. After the user selects method A, the system will begin generation of the LPT. In general, AISIM will not solicit any further information if this method is used.

Method A detects two types of error. If the generator detects an unconnected node, the system will output the following error message:

UNREACHABLE NODE..."node name"

and control is transferred to the DUI level. If multiple links connect nodes, the system will prompt the user for resolution of ambiguous paths. The system will prompt with:

GOING FROM "Node name1" TO "Node name2" CAN GO

1. Through "next Node name" BY CHANNEL "channel name"
 2. Through "next Node name" BY CHANNEL "channel name"
- ENTER THE NUMBER OF THE ROUTE YOU WANT TO USE
#

All "Through" options will be listed. The choice of path is selected by entering the number of the path after the pound sign (#) prompt. If there are ambiguous paths for other node pairs, the user will be prompted for resolution. If the user should ABORT the LPT generation the following prompt will be displayed:

UNABLE TO SAVE LPT

Control is then passed to the DUI level. If all ambiguities are clarified, the system will complete the generation of the LPT, and issue the following message:

SAVE OF LPT COMPLETE

The user is then at the DUI level.

METHOD B - Method B should be used when there is extensive routing through the architecture. Using Method B, AISIM will algorithmically find all possible legal paths through the system.

This can involve a lot of processing in fully connected architectures because a path from every node to every other node must be defined. (If there are 20 nodes then there will be 380 paths, 20 times 19.)

The AISIM responses for method B are similar to those described in method A. Because AISIM will fully connect all nodes in the architecture there are bound to be many ambiguous paths. The user will be prompted to resolve all ambiguous paths.

METHOD C - Method C should be used when there is extensive routing through the architecture, also. Using Method C, AISIM will algorithmically find all possible legal paths through the system but will assume that the path for directly connected nodes in the architecture is the direct link. This can substantially decrease the number of paths the user must resolve.

The AISIM responses for method C are similar to those described in method A.

The HELP request causes the system to show the available commands.

The INFO request prints the following:

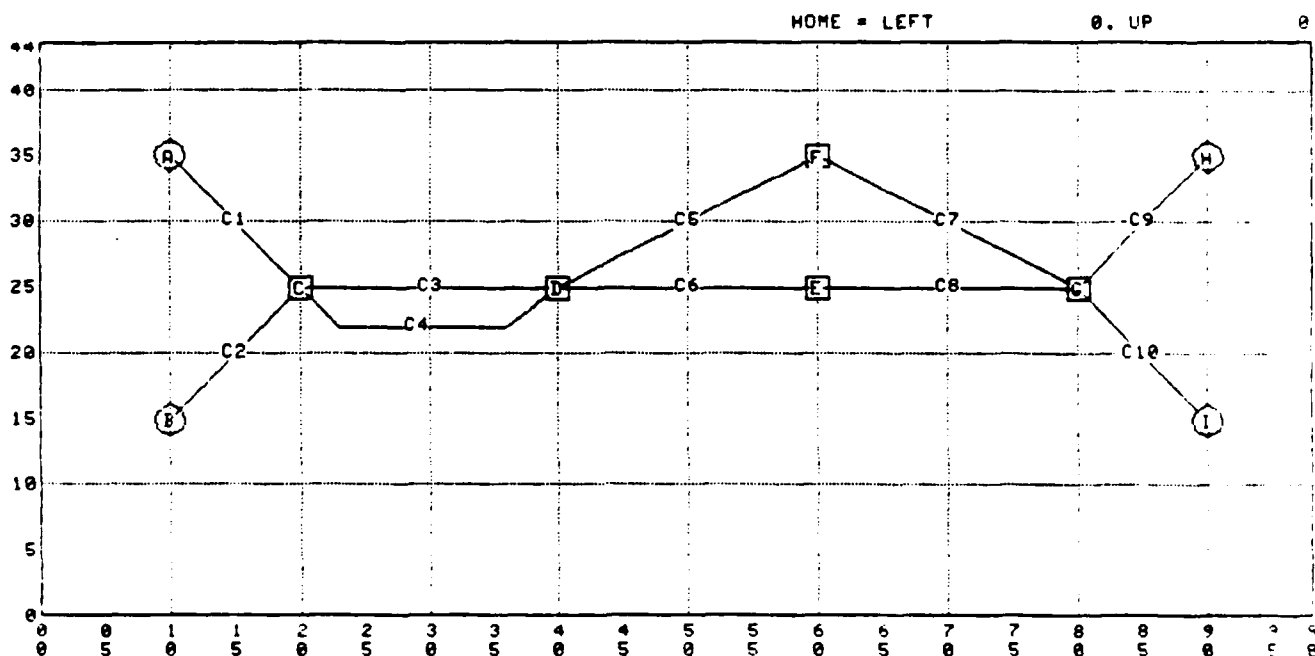
METHOD A defines as legal paths only connections directly between adjacent Nodes. Longer paths must be handled explicitly in the user Processes.

METHOD B generates all possible paths between each Node pair. You must identify default legal paths for each Node pair.

METHOD C generates all possible paths between each Node except for directly connected Nodes. In the case of adjacent Nodes, the direct connection is assumed as the legal path.

Type END and give a carriage return to exit the LPT generation.

In the accompanying figure, an AISIM architecture is shown. This architecture connects nine nodes together with 10 links. Using method A, the user is required to resolve 2 ambiguities. Using method B, the user is required to resolve 20 ambiguities. Using method C, the user is required to resolve 12 ambiguities. The Legal Path Tables using each of these methods is shown.



method A:

LPT GENERATED			
FROM	TO	NEXT	VIA
NODE	NODE	NODE	LINK
=====	=====	=====	=====
A	C	C	C1
B	C	C	C2
C	A	A	C1
C	B	B	C2
C	D	D	C3
D	C	C	C4
D	E	E	C6
D	F	F	C5
E	D	D	C6
E	G	G	C8
F	D	D	C5
F	G	G	C7
G	E	E	C8
G	F	F	C7
G	H	H	C9
G	I	I	C10
H	G	G	C9
I	G	G	C10

method B:

LFT GENERATED			
FROM NUDE	TO NUDE	NEXT NUDE	VIA LINK
A	B	C	C1
A	C	C	C1
A	D	C	C1
A	E	C	C1
A	F	C	C1
A	G	C	C1
A	H	C	C1
A	I	C	C1
B	A	C	C2
B	C	C	C2
B	D	C	C2
B	E	C	C2
B	F	C	C2
B	G	C	C2
B	H	C	C2
B	I	C	C2
C	A	A	C1
C	B	B	C2
C	D	D	C3
C	E	D	C3
C	F	D	C3
C	G	D	C3
C	H	D	C3
C	I	D	C3
D	A	C	C4
D	B	C	C4
D	C	C	C4
D	E	E	C6
D	F	F	C5
D	G	E	C6
D	H	E	C6
D	I	E	C6
E	A	D	C6
E	B	D	C6
E	C	D	C6
E	D	D	C6
E	F	G	C8
E	G	C	C8
E	H	C	C8
E	I	C	C8
F	A	D	C5
F	B	D	C5
F	C	D	C5
F	D	D	C5
F	E	C	C7
F	G	C	C7
F	H	C	C7
F	I	C	C7
G	A	E	C8
G	B	E	C8
G	C	E	C8
G	D	E	C8
G	E	E	C8
G	F	F	C7
G	H	H	C9
G	I	I	C10
H	A	C	C9
H	B	C	C9
H	C	C	C9
H	D	C	C9
H	E	C	C9
H	F	C	C9
H	G	C	C9
H	I	C	C9
I	A	C	C10
I	B	C	C10
I	C	C	C10
I	D	C	C10
I	E	C	C10
I	F	C	C10
I	G	C	C10
I	H	C	C10

method C:

TEST GENERATED			
FROM	TO	FROM	TO
NUDE	NUDE	NUDE	NUDE
-----	-----	-----	-----
A	B	C	C1
A	C	C	C1
A	D	C	C1
A	E	C	C1
A	F	C	C1
A	G	C	C1
A	H	C	C1
A	I	C	C1
B	A	C	C2
B	C	C	C2
B	D	C	C2
B	E	C	C2
B	F	C	C2
B	G	C	C2
B	H	C	C2
B	I	C	C2
C	A	A	C1
C	B	B	C2
C	D	D	C3
C	E	D	C3
C	F	D	C3
C	G	D	C3
C	H	D	C3
C	I	D	C3
D	A	C	C4
D	B	C	C4
D	C	E	C6
D	E	F	C5
D	F	F	C5
D	G	E	C6
D	H	E	C6
D	I	E	C6
E	A	D	C6
E	B	D	C6
E	C	D	C6
E	D	D	C6
E	F	G	C8
E	G	G	C8
E	H	G	C8
E	I	G	C8
F	A	D	C5
F	B	D	C5
F	C	D	C5
F	D	D	C5
F	E	G	C7
F	G	G	C7
F	H	G	C7
F	I	G	C7
G	A	E	C8
G	B	E	C8
G	C	E	C8
G	D	E	C8
G	E	F	C7
G	F	H	C9
G	H	I	C10
H	A	G	C9
H	B	G	C9
H	C	G	C9
H	D	G	C9
H	E	G	C9
H	F	G	C9
H	G	G	C9
H	I	G	C9
I	A	G	C10
I	B	G	C10
I	C	G	C10
I	D	G	C10
I	E	G	C10
I	F	G	C10
I	G	G	C10
I	H	G	C10

7. USE OF THE ANALYSIS USER INTERFACE

After completion of model design using the DUI, the model can be exercised using the AUI. At this point all entities associated with the model have been defined. Several pieces of information must be supplied, however, before the simulation can be run. It is still necessary to specify the external world stimuli; (the Scenario) under which the model is to perform. Two other categories of information must also be provided to the simulator: the user must describe the output Variables to be displayed as a result of the simulation and the user must describe how they are to be displayed.

The user must select which of possibly several available Scenarios (which were defined in the DUI) is to be used for the current simulation run. The user can modify Loads and Scenarios only by changing the values of Constants that serve as parameters in them, at the beginning of a simulation. The user may specify one of ten random number seeds for calculating random numbers. Random numbers are used to generate the stochastic times for Load triggering inter-arrival times and Action durations. Also, probabilistic branching is controlled by random sampling. By changing the random number seeds, the user can test the effect of different random number streams on simulation statistics. The user may also define a stopping point for the simulation in terms of conditions on the values of certain entities that change during the run. These entities are automatically examined at certain intervals during the run. If the conditions are satisfied, the simulation is temporarily suspended to allow the user to interact with the model. Current statistics and values of system entities can be displayed at the terminal. The user can also set new conditions and change the values of Variables.

During simulation, statistics are kept on Variable values, Item throughput, Resource utilization, Queue time, Queue lengths, Action timing, Process execution, Process timing, and Process logic execution. A set of output reports organizes these statistics for printing off line or viewing on-line while in the AISIM Ready level after completion of the simulation run. Plots of selected model parameters, however, may be drawn on the screen of the HP-2647A when simulation is halted at a breakpoint, end of period, or end of simulation.

The commands that control the simulation and the interactive and graphic capabilities are described on the following pages.

When analysis mode is entered the user is asked the following question, if there is more than one Scenario in the project database:

WHICH SCENARIO DO YOU WISH TO TRANSLATE?

The user must respond with a valid Scenario name, one that has

been defined previously in DUI level. A return in response to this question will cause AISIM to list available Scenarios.

If the Scenario name given is invalid the system will respond:

INVALID SCENARIO NAME - REENTER

The user should then enter the correct Scenario name.

When translation of the model and Scenario has completed, the simulator reads the translated database and checks for errors. If the simulator detects one or more errors, the message

ERRORS DETECTED IN MODEL TRANSLATION

is displayed, the AUI level is exited and the user is returned to the AISIM Ready level.

At this point the user should enter the command REPORT. This, automatically invokes the TSO editor on the Project report data set. The user should use the Find command of the TSO Editor to list all occurrences of "####". This will result in a list of all errors detected during initialization. Each error documents a problem detected in the model.

If no errors are detected, the following message is displayed:

NO ERRORS DETECTED IN MODEL TRANSLATION
YOU MAY NOW ENTER COMMANDS

The system provides a # prompt and is ready to accept any of the valid AUI commands. These commands are described later in this section.

During each of the three phases of analysis - 1) pre-simulation (before the first GO command is issued) 2) mid-simulation (after the first GO command is issued but before simulation termination) and 3) post-simulation (after simulation termination) the user can invoke different commands.

PRE-SIMULATION COMMANDS :

GO	G	END	LIST	L	LISTVAL	LV	
EDIT	E	SETBREAK	SET	CANBREAK	CAN	DEFPLOT	DEF
INFRES		GET					

MID-SIMULATION COMMANDS:

GO	G	END	LIST	L	LISTVAL	LV
EDIT	E	SETBREAK	SET	CANBREAK	CAN	PLOT

SAVE S

POST-SIMULATION COMMANDS:

END LIST L LISTVAL LV PLOT SAVE
S

The simulation is started with the GO command.

The SETBREAK and CANBREAK commands are used to establish and cancel stopping conditions (or breakpoints) for the simulation. EDIT is used to make temporary changes to Constant, Variable, and random number seed (the keyword is STREAM) values. A limited number of Resources in the model sometimes causes a bottleneck which is evidenced by a waiting line or queue. The effects of this queueing may be eliminated by changing the available Resources to an unlimited quantity. The INFRES command is used to do this on a temporary basis. LIST and LISTVAL are used to display model entities, their attributes, and their values. LISTVAL also allows the user to examine the current random number Seed. The END command returns AISIM to the AISIM Ready level.

The DEF PLOT and PLOT commands are used to specify what information is to be graphed at the terminal and to request the graph to be displayed at the terminal, respectively. The DEF PLOT command can only be used prior to the start of simulation since the simulator must know what statistics to sample. The user must define the graphs to be plotted before issuing the first GO command.

Simulation may be performed in periods that correspond to the Loads defined in the translated Scenario, and is suspended at the end of the number of periods specified. The number of periods to be simulated is specified as an optional period of the GO command. The user is prompted at the end of the period with the message:

END OF PERIOD
YOU MAY NOW ENTER COMMANDS

and with an audible 'beep' at the terminal.

The user can now make changes in the values of Variables, set breakpoints, define and display plots, or cancel breakpoints. By suspending the simulation at the end of a period, the system allows the user to dynamically interface with the model.

A similar result occurs at a user specified breakpoint, except that the message reads:

BREAK POINT REACHED:
(description of the condition of the breakpoint)
YOU MAY NOW ENTER COMMANDS

An audible 'beep' is also sounded at this point.

The AUI level commands are described in detail on the following pages.

[illegible]

AUI COMMAND SUMMARY

COMMANDS:

CANBREAK

CAN

DEFPLOT {entity-type},{entity-name}

DEF

EDIT {entity-type},{entity-name},{new-value}

E

END

GET {settype},{setname}

GO

G

HELP

INFRES

LIST {entity-type/def}

L

LISTVAL {entity-type},{entity-name}

LV

PLOT

SAVE {settype},{setname},{descr}

SETBREAK {entity-type},{entity-name},{rel-oper},{value}

SET

7.1 COMMAND: CANBREAK

The CANBREAK command allows the user to cancel a previously defined breakpoint. See SETBREAK.

COMMAND SYNTAX:

CANBREAK

CAN

FUNCTION RESULTS:

A previously defined breakpoint is canceled.

This command is valid prior to a simulation period only.

7.2 COMMAND: DEF PLOT

The DEF PLOT command allows the user to define a plot of the activity of a specific entity over a period of simulation.

COMMAND SYNTAX:

```
DEF PLOT {entity-type},{entity-name}
```

```
DEF
```

where:

{entity-type} is a required parameter indicating a valid ANALYSIS system entity-type (i.e., Variables, Queues, Resources, Processes, Items).

{entity-name} is a required parameter indicating the name of the entity whose value is to be plotted.

FUNCTION RESULT:

This command causes an attribute form to be displayed, from which the user must select one attribute. The list of attributes depends on the entity-type selected.

When the attribute form has been entered, a statistics form is displayed, from which the user must select one statistic. The list of statistics displayed depends on the entity-type and attribute selected.

If only one choice for either an attribute or a statistic exists, the form is not displayed.

A maximum of ten plots may be defined during any ANALYSIS session.

7.3 COMMAND: EDIT

The edit command in the ANALYSIS mode allows the user to change the value of either a Constant, a variable, or specification of the random number seed.

COMMAND SYNTAX:

```
EDIT {entity-type},{entity-name},{new-value}
```

E

where:

{entity-type} is a required parameter indicating which type of entity is to be changed (either CONSTANT, VARIABLE, or STREAM).

{entity-name} is a required parameter indicating the name of the Constant or Variable which is to be changed. When changing the random number seed this field is STREAM.

{new-value} is a required parameter indicating the new value of a Constant or variable or the number of another STREAM. The new value may be expressed in one to twelve digits, and includes the value zero. The legal values of "New Value" when specifying a random number seed are 1 through 10.

NOTE: Constants may be changed only before the start of the first simulation period. Variables and Seeds may be changed before the start of any simulation period or at a breakpoint.

FUNCTION RESULT:

The value of the Constant or Variable or Seed is changed to the new value, and remains at that value until changed by another EDIT command. This command only affects the current translation of the database; therefore, at the end of an ANALYSIS session the Constant or Variable or Seed is restored to its original value.

If the value of the SEED is not changed, default values are:

branch: 2, for calculating the probability of branching

Load : 1, for statistical scheduling methods sampling

Action: 3, for action distrobution

7.4 COMMAND: END

The END command is used to terminate an ANALYSIS session.

COMMAND SYNTAX:

END

FUNCTION RESULT:

This command causes all displays to be cleared and asks the user "Do you want to save any plots or plot definitions? (Y/N). If the answer is yes, the user remains in the AUI level. If no, the user is returned to the AISIM READY level.

7.5 COMMAND: GET

The GET command allows the user to retrieve previously saved plot definitions.

COMMAND SYNTAX:

```
GET DEF {setname}
```

where:

{setname} is the name of the set containing the plot definitions. The GET command may be issued only before the first simulation period.

FUNCTION RESULT:

The set of plot definitions is retrieved and made the current set to be used by the ANALYZE function.

7.6 COMMAND: GO

The GO command allows the user to start or resume a simulation run.

COMMAND SYNTAX:

GO [n]

G

where:

n is an optional parameter that specifies how many periods the simulation is to run. If not given, the default result is that the entire simulation defined by the selected Scenario is executed. If an n greater than the number of periods specified in the Scenario is entered, the simulation executes all periods specified in the Scenario and no more.

FUNCTION RESULT:

This command, which is valid before any simulation period or at a breakpoint, begins or resumes the simulation of the translated Scenario.

If used to resume the simulation, resumption occurs at the breakpoint or at the beginning of the next simulation period.

7.7 COMMAND: HELP

The HELP command lists, on the user's terminal, the commands that are valid during each of the three different stages of an ANALYSIS session (prior, during, or after simulation).

COMMAND SYNTAX:

HELP

FUNCTION RESULT:

The HELP command may be invoked prior to, during, or after a simulation run. When invoked, only those commands that are valid at that point in the ANALYSIS session are displayed following the last line appearing on the screen.

This command is valid at any time during an ANALYSIS session.

During each of the three phases of analysis, the user receives a different output from the HELP COMMAND.

HELP INVOKED PRIOR TO SIM:

GO	G	END	LIST	L	LISTVAL	LV	
EDIT	E	SETBREAK	SET	CANBREAK	CAN	DEFPLOT	DEF
INFRES		GET					

HELP INVOKED DURING SIM:

GO	G	END	LIST	L	LISTVAL	LV	
EDIT	E	SETBREAK	SET	CANBREAK	CAN	PLOT	SAVE
S							

HELP INVOKED AFTER SIM:

END		LIST	L	LISTVAL	LV	PLOT	SAVE
S							

7.8 COMMAND: INFRES

The INFRES command causes the simulation to assume the existence of infinite available Resources.

COMMAND SYNTAX:

INFRES

FUNCTION RESULT:

This command, which is only valid before the start of the first simulation period, allows the assumption that infinite Resources available to the Scenario being simulated.

7.9 COMMAND: LIST

The LIST command displays all entities of a specified type. Included with each entity is its name, its description, and any defined parameters.

COMMAND SYNTAX:

LIST {entity-type}

L

where:

{entity-type} is a required parameter indicating any of the specific model entities listed below.

<u>ENTITY</u>	<u>ABBREVIATION</u>
CONSTANT	C
RESOURCE	R
PROCESS	P
VARIABLE	V
QUEUE	Q
PLOT	none
ITEM	I
DEF	none
TITLE	none

This command is valid at any time during an ANALYSIS session.

FUNCTION RESULT:

The user is presented with a list of all existing entities of the requested type. If the argument is not an entity-name, (as DEF) a list of all the currently defined plots is given.

7.10 COMMAND: LISTVAL

The LISTVAL command allows the user to display all statistics associated with the named entity for all attributes associated with the entity.

COMMAND SYNTAX:

LISTVAL {entity-type},{entity-name}

LV

where:

{entity-type} is a required parameter indicating a valid ANALYSIS system entity-type. STREAM is valid entity-type field.

{entity-name} is a required parameter indicating the name of the entity whose value is to be listed. When requesting the random number STREAM for Loads, branches or Actions, this field would specify Load, BRANCH or Action.

FUNCTION RESULT:

The name of the entity requested is printed out with a listing of all statistics for attributes associated with that entity and their values.

The prompt "**** (CR/NO) TO (CONTINUE/TERMINATE) ****" is displayed. If the user wishes to end the listval listing, "NO" is entered and the AUI READY prompt is displayed. If instead the carriage return is depressed, the next page of the listing is displayed, if there is one, with the prompt displayed again. If there is no further data to be displayed, the user is returned to the AUI READY level.

7.11 COMMAND: PLOT

The PLOT command allows the user to produce a graph of an entity's activity for which a plot has been previously defined.

COMMAND SYNTAX:

PLOT

FUNCTION RESULT:

This command, which is valid at the end of a simulation period or at a breakpoint, causes the display of a form containing the previously defined plot titles. From this form the user may select any, all, or none of the listed titles.

When the selected titles have been entered, the user is presented with the plot grid. The selected plots are produced and the user is prompted for more ANALYSIS mode commands.

Each of the plots is produced in a unique line pattern.

An example of the form that is displayed to allow the user to select a plot is shown .

This is the only means for viewing simulation results while in the AUI level.

PLACE AN X NEXT TO THE TITLES YOU WISH TO PLOT

CURRENT # IN WAIT QUEUE FOR RESOURCE AB1
CUMULATIVE MEAN BUSY TIME FOR RESOURCE CH1.A
CURRENT NUMBER CREATED FOR ITEM MSG
CUMULATIVE MEAN TIME IN SYSTEM FOR ITEM MSG

7.12 COMMAND: SAVE

SAVE is used to save current plot definitions or plot data and transfer them to the analysis data base.

COMMAND SYNTAX:

SAVE {settype},{setname},{descr}

where:

{settype} is

1. DEF to save plot definitions, or
2. PLOT to save plot data.

{setname} is a 1 to 8 character name to be given to the set of definitions of data.

{descr} is a 1 to 53 character description of the set. The SAVE command is valid at any analysis user interface point.

FUNCTION RESULT:

Plot definitions or plot data are saved into the analysis database. If {setname} already exists, the query "REUSE {setname}?" will appear. A "yes" response will replace the old set with the new set. A "no" response will abort the SAVE command.

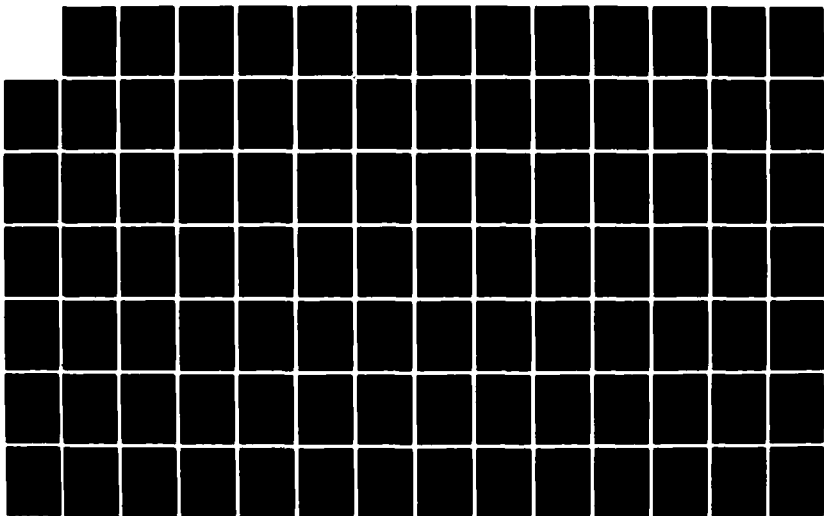
AD-A135 761

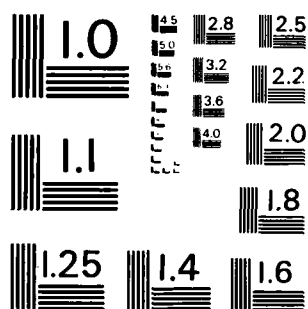
AISIM (AUTOMATED INTERACTIVE SIMULATION MODEL) USER'S
MANUAL(U) HUGHES AIRCRAFT CO FULLERTON CA GROUND
SYSTEMS GROUP W AUSTELL ET AL. 26 FEB 82 ESD-TR-83-218
F19628-79-C-0153 F/G 9/2

3/4

UNCLASSIFIED

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

7.13 COMMAND: SETBREAK

The SETBREAK command allows the user to set a single breakpoint in the simulation run that is executed when a defined relationship has been satisfied.

COMMAND SYNTAX:

```
SETBREAK {entity-type},{entity-name},{rel-oper},{value}
```

```
SET
```

where:

{entity-type} is a required parameter indicating which type of entity is to be tested (VARIABLE, RESOURCE, or TASK).

{entity-name} is a required parameter indicating the name of the entity to be tested.

{rel-oper} is a required parameter indicating the relational operator (EQ, NE, LE, GT, GE, LT) of the test.

{value} is a required parameter used to set the value for which the named entity is to be tested. This value may be expressed in one to twelve digits, and includes the value zero.

FUNCTION RESULT:

A Breakpoint is usually used in verification of a model or to examine Variable values. Typically, a simulation run executes start to finish and does not allow the user to examine the simulation state at specific times during simulation. The Breakpoint allows the user to halt the simulation and examine its state based upon the value of some system element.

This command causes an attribute form to be displayed, from which the user must select one attribute. The list of attributes depends on the entity-type selected.

When the attribute form has been entered, a statistics form is displayed, from which the user must select one statistic. The list of statistics displayed depends on the entity-type and attribute selected.

If only one choice for either an attribute or a statistic, the form is not displayed.

This command is valid at the beginning of a simulation period or

at a breakpoint.

The user should ensure that only one breakpoint is in effect at any time. Otherwise the condition that caused the break may be uncertain. When a breakpoint is reached, it is automatically cleared.

7.14 TERMINATION OF AN AUI SESSION

An AUI session is terminated by typing the command:

END

In response to this command control is transferred to the AISIM Ready level and the system responds with:

AISIM READY

8. REPLOT USER INTERFACE

The REPLOT function allows the user to (1) plot data from previous analysis runs, and (2) to delete old plot data and plot definition sets from the data base.

Data to be plotted is placed into a temporary plotset. Data from different analysis runs may be plotted on the same graph, provided that the precision units are the same for all plots requested.

REPLOT COMMAND SUMMARY

COMMANDS:

CLEAR

DELETE {settype},{setname}

DEL

END

GET plot,{plotset}

LIST {entity-type}

L

PLOT

8.1 COMMAND: CLEAR

CLEAR is used to clear out the current plotset and to clear the screen.

COMMAND SYNTAX:

CLEAR

FUNCTION RESULT:

The current plotset is cleared, and the screen is cleared.
reset.

8.2 COMMAND: DELETE

DELETE is used to delete a set of plot definitions or plot data from the analysis data base.

COMMAND SYNTAX:

```
DELETE {settype},{setname}
```

```
DEL
```

where:

{settype} is

1. DEF to delete plot definitions, or
2. PLOT to delete plot data.

{setname} is the name of the set to be deleted.

FUNCTION RESULTS:

The specified set of plot data or plot definitions are deleted from the analysis data base.

REPLOT / END

8.3 COMMAND: END

END is used to exit the PLOT function.

COMMAND SYNTAX:

END

FUNCTION RESULT: The user is returned to the AISIM READY level.

8.4 COMMAND: GET

GET is used to retrieve a set of plot data from the analysis data base and to make it part of the current set of plots to be generated by the PLOT function.

COMMAND SYNTAX:

GET plot, {plotset}

where:

{plotset} is the name of the set containing the plot data.

FUNCTION RESULT: The set of available plots is displayed. The user is then prompted for the number(s) of the plot(s) to be graphed.

8.5 COMMAND: LIST

LIST is used to list all entities of the specified type.

COMMAND SYNTAX:

LIST {entity-type}

where:

{entity-type} is a required parameter indicating a valid entity-type. It can be one of the following:

DEF to list plot definition sets

PLOT to list plot data sets

TITLE to list current plot titles

FUNCTION RESULTS: Names of all entities of the requested type are displayed.

8.6 COMMAND: PLOT

PLOT is used to produce a graph of the activity of an entity.

COMMAND SYNTAX:

PLOT

FUNCTION RESULT:

The set of available plots is displayed. The user is then prompted for the number(s) of the plot(s) to be graphed.

When the selected titles have been entered, the appropriate graph is produced. Each of the plots is produced in a unique line pattern. If only one plot is defined, the graph will be produced with no prompting.

9. HARDCOPY USER INTERFACE

The Hardcopy User Interface (HUI) is used to plot one, several, or all Processes in the specified project. In order for the Hardcopy Function to be exercised, the following conditions must be in effect:

1. An HP2631G Graphics Printer must be connected to the HP2647A Graphics Terminal with the HP-IB communications bus.
2. The HP-IB bus address of the printer must be set to one (1).
3. The printer must be turned on and set to ON LINE mode.
4. For proper formatting, the length of the paper in the printer must be either 8 1/2 inches or 11 inches long.

The HUI is entered by typing HCOPY PROJECT(project) at the AISIM Ready level. The first information that the HUI requests is:

PLOT ALL THE PROCESSES IN DATABASE? (YES OR NO)

The user responds with "NO" to specify selected Processes for plotting. A "YES" response will cause the system to automatically plot all of the Processes contained in the project data base.

The system then requests information about the printing medium:

ENTER THE LENGTH OF THE PAPER IN THE HP2631G PRINTER (A/B):
A) 8 1/2 INCHES
B) 11 INCHES.
LENGTH=

Depending on the paper in the graphics printer, the user responds by entering "A" or "B". This information is used by the HUI to center the Process graphics on the page and to insure correct form feeding. Entering any other option besides "A" or "B" causes the HUI to default to 8 1/2 inches for paper length.

The user is then instructed to:

POSITION THE PAPER PERFORATION ALONG THE T.O.F. INDICATOR
LINE ON THE PRINTER AND DEPRESS THE CARIAGE RETURN.

By doing this, the user sets up the proper alignment of the paper in the printer and initiates execution of the hardcopy plotting software.

When the carriage return has been entered, the HUI begins the plotting procedure by initializing the HP2631G printer with the correct form information. This initialization is usually characterized by a rapid movement of the print head.

If the user has requested automatic plotting of all of the Processes, they are plotted in alphabetic order.

If the user asked to select Processes, the following prompt is given:

PROCESS NAMES TO PLOT: (CR TO EXIT)

The user then supplies the name of the Process he wishes plotted followed by a carriage return. The Process is plotted and the HUI responds with:

<Process name> PLOTTED

The system will then give the selection prompt again for another Process to be plotted. The user continues entering Process names one at a time, followed by a carriage return, or exits the HUI by entering a carriage return only.

The way in which the HUI plots a Process in either of the two modes described above is as follows:

1. The first six Primitives in the Process are painted on the screen of the HP2647A terminal.
2. The name of the Process being plotted is printed in the upper left corner of the page in the printer.
3. If the first page of the Process is being plotted, the Process description is written across the top of the page.
4. The Process graphics are transferred from the terminal screen to the page in the printer and a form feed is generated.
5. If there are no more Primitives in the Process, the plotting is terminated for the Process; otherwise, the terminal screen is erased, and next six primitives are painted on the screen, and steps 2 through 5 are repeated.

When the HUI has finished plotting all of the requested Processes, the message "ALL DONE" is printed and the user is returned to the AISIM Ready level.

10. LIBRARY USER INTERFACE

The Library User Interface (LUI) enables the user (1) to store AISIM models in libraries of model components for reuse and (2) to retrieve models and include them in a newly created project database .

Two libraries are available. One is a user library in which a user can place components of models for private use. Another is an AISIM system library which contains models available for public use. There are restrictions on the placement of items in the system library because it is desirable to insure that the public models are not lost or tampered with. For this reason, general users can not modify the AISIM system library. This is done by AISIM administrator.

The library user interface allows the user to do the following:

1. Moving entities from a model project database into a storage area called here a "buffer".
2. Moving entities from a buffer into the database of another model project.
3. Moving entities from a buffer into a library of model entities.
4. Moving entities from a library to a buffer.

To reach the sub-levels in which these operations are accessible, one must issue one of the four commands MERGEIN, MERGEOUT, CHECKIN, and CHECKOUT. The facilities made accessible by these commands are described in greater detail below.

The LUI sub-level is accessible from the AISIM READY level by issuing the command,

LIBRARY

The system will then respond with the prompt

LIBRARY READY

and the user may invoke any of the four LUI sublevels listed in the LUI Command Summary.

LUI COMMAND SUMMARY

CHECKIN

CI

CHECKOUT

CO

MERGEIN

MI

MERGEOUT

MO

10.1 COMMAND: CHECKIN

To move the contents of the buffer to a library for permanent storage one issues the CHECKIN command. The user is prompted for the name of the model to be checked in, as well as an optional reference number and description.

To enter the CHECKIN sublevel, issue the command,

```
CHECKIN {BUFFER(buffer)},{LIBRARY(library)}
```

```
CI      {B(buffer)},{LIB(library)}
```

where:

{B(buffer)} is a required parameter indicating the buffer from which model entities are to be taken.

{LIB(library)} is a required parameter indicating the library into which the model entities are to be entered.

FUNCTION RESULT:

The system queries the user for a required model name and an optional document reference number and description. After getting this information, the entities in the buffer are put into the Library specified under the given model name.

10.2 COMMAND: CHECKOUT

To copy a model project stored in a library to a buffer one enters the CHECKOUT command. At this point the user can obtain a list of the models contained in the library or a list of the given entity types contained in a named model through the LIST command. Models are copied individually through the EXTRACT command which specifies the model to be copied. As in the case of MERGEOUT a HELP command is available.

To enter the CHECKOUT sublevel, issue the command,

```
CHECKOUT {BUFFER(buffer)},{LIBRARY(library)}  
CO      {B(buffer)},{LIB(library)}
```

where:

{B(buffer)} is a required parameter indicating the buffer into which model entities are to be placed.

{LIB(library)} is a required parameter indicating the library from which the model entities are to be taken.

FUNCTION RESULT:

AISIM queries the user for the name of the model to be extracted from the library. The model is written out to the buffer. From the buffer it can be included in another model with the MERGEIN command.

CHECKOUT COMMAND SUMMARY

DELETE

D

END

E

EXTRACT

HELP

LIST {model-name}

10.2.1 COMMAND: DELETE

The DELETE command instructs the system to delete a specified model from a library.

COMMAND SYNTAX:

DELETE {model-name}

where:

{model-name} is the name of the model to be deleted from the library.

FUNCTION RESULT:

The specified model is deleted from the library.

10.2.2 COMMAND: END

The END command causes the system to exit the CHECKOUT sublevel and return the user to the Library User Interface.

COMMAND SYNTAX:

END

E

FUNCTION RESULT:

The system returns to the LUI Ready level.

10.2.3 COMMAND: EXTRACT

The EXTRACT command instructs the system to copy a model in a library into a buffer.

COMMAND SYNTAX:

EXTRACT {model-name}

where:

{model-name} is the name of the model to be placed in the buffer.

FUNCTION RESULT:

The model specified is copied from the current library into the current buffer.

10.2.4 COMMAND: HELP

The HELP command enables the user to obtain a menu of the other command options available in the CHECKOUT sublevel.

COMMAND SYNTAX:

HELP

FUNCTION RESULT:

A menu of available command options is printed on the screen.

10.2.5 COMMAND: LIST

The LIST command enables the user to obtain a list of the models contained in a system or user library.

COMMAND SYNTAX:

LIST {model-name}

*

L

where:

* is a literal parameter, indicating all models in the library.

{model-name} is the name of a model in the library

FUNCTION RESULT:

If the parameter * is used, the system will display a list of the names of all the models in the library. If the parameter {model-name} is used the system will display a list of the names of the entities in the indicated model.

10.3 COMMAND: MERGEIN

To move the contents of a buffer to a project database where it will be accessible to other AISIM sub-levels, one enters the MERGEIN command, specifying the the name of the buffer and the name of the project into whose database the buffer contents are to be copied. If no entity in the buffer is the same as an entity already present in the database, the user will be prompted with

0 CONFLICTS HAVE BEEN DETECTED IN MERGEIN INITIALIZATION

in which case the copying of the buffer contents will be completed and the user will be returned to the AISIM READY level. If one or more names of entites conflict with ones already in the project database, the user will be prompted with,

n CONFLICTS HAVE BEEN DETECTED IN MERGEIN INITIALIZATION

where "n" is the number of conflicts. The system will then present the name of an entity which stands in conflict.

The user now has three command options to resolve the naming conflict. First, he may command that the entity in the database be deleted in favor of the one of the same name in the buffer. This is done by entering REPLACE. Secondly, he may command that the entity in the buffer which aroused the naming conflict be disregarded in the transferal from the buffer to the database. This is done by issuing the command IGNORE. Thirdly, one may resolve the naming conflict by giving the entity in the buffer a new name. This is done by means of the command RENAME whose one parameter is the new name the user wishes to give the entity. If the user should select as a new name one that is also being used, the system will respond with a prompt for a different name.

This cycle of naming conflict resolution will be repeated until all of the naming conflicts have been resolved. The system will then tell the user that MERGEIN initialization has been completed, do the MERGEIN and automatically return the user to the AISIM READY level.

To obtain access to the MERGEIN sublevel, issue the command, COMMAND SYNTAX:

```
MERGEIN {PROJECT(project)},{BUFFER(buffer)}
MI      {P(project)}      {BUF(buffer)}
```

where:

{P(project)} is a required parameter indicating the name of the project into which the entities are to be merged.

{BUF(buffer)} is a required parameter indicating the name of the buffer in which the entities are to be stored.

FUNCTION RESULT:

- 1) The buffer is checked for internal naming conflicts. If any are found, operation terminates.
- 2) The system checks for naming conflicts between the buffer and the project database. If any are found, the user is asked to resolve them with the sub-commands.

```
REPLACE
IGNORE
RENAME {new name}
END (i.e., give up)
```

- 3) After the resolution of conflicts, buffer entities are merged, ignored or renamed into the database.

- 4) The user is returned to the AISIM READY level.

NOTE: Resources associated with an architecture are not subject to the REPLACE command.

MERGEIN COMMAND SUMMARY

END

E

HELP

IGNORE

IG

INFO

RENAME

RN

REPLACE

RP

MERGEIN / END

10.3.1 COMMAND: END

The END command, issued at the MERGEIN sublevel cause the system to exit the MERGEIN sublevel and returns the user to the Library User Interface.

COMMAND SYNTAX:

END

E

FUNCTION RESULT:

The system returns to the LUI Ready level.

10.3.2 COMMAND: HELP

The HELP command enables the user to obtain a menu of the other command options available in the MERGEIN sublevel.

COMMAND SYNTAX:

HELP

FUNCTION RESULT:

A menu of available command options is printed on the screen.

MERGEIN / IGNORE

10.3.3 COMMAND: IGNORE

The IGNORE command enables the user to resolve any naming conflicts encountered at the MERGEIN sublevel in favor of the entities that already exist in the target database.

COMMAND SYNTAX:

IGNORE

IG

FUNCTION RESULT:

The entity indicated by the prompt is not copied into the project database. The system then prompts the user with the next naming conflict, if any, and proceeds with MERGEIN operation.

/

MERGEIN / INFO

10.3.4 COMMAND: INFO

The INFO command furnishes the user with information on the options available to resolve naming conflicts encountered in the MERGEIN sublevel.

COMMAND SYNTAX:

INFO

IN

FUNCTION RESULT:

The screen displays the following information:

IGNORE: THIS OPTION CAUSES THE NAMED ENTITY IN THE BUFFER TO BE EXCLUDED FROM THE MERGEIN OPERATION

RENAME: THIS OPTION CHANGES ALL OCCURANCES OF THE ENTITY NAME IN THE BUFFER TO THE NAME SPECIFIED BY THE USER

REPLACE: THIS OPTION DELETES THE NAMED ENTITY FROM THE USER DATA BASE, ALLOWING THE ENTITY IN THE BUFFER TO BE MERGED IN

END: THIS OPTION TERMINATES THE MERGEIN PRE-PROCESSING WITHOUT RESOLVING ANY MORE NAMING CONFLICTS AND RETURNS TO THE LUI READY LEVEL

10.3.5 COMMAND: RENAME

The RENAME command allows the user to resolve a naming conflict encountered during the MERGEIN operation by giving entities in the buffer a unique name.

COMMAND SYNTAX:

RENAME {name1}

RN

where:

{name1} is the new name the entity is to be given.

FUNCTION RESULT:

The system checks to see whether the new name given to the entity creates any naming conflicts. If it does the system will prompt the user to that effect, and await a new name. If the new name does not create any conflicts, the entity is copied into the project database under its new name. If there are naming conflicts with further entities the system then prompts the user for their resolution. If there are no remaining naming conflicts, the MERGEIN operation begins.

10.3.6 COMMAND: REPLACE

The REPLACE command enables the user to resolve a naming conflict encountered in the MERGEIN sublevel in favor of entities that exist in the buffer.

COMMAND SYNTAX:

REPLACE

RP

FUNCTION RESULT:

The entity indicated in the prompt is written into the database and the old entity of the same name is deleted. The systems then proceeds to consideration of the next naming conflict if any exist. Otherwise, the MERGEIN operation begins.

10.4 COMMAND: MERGEOUT

When the user wishes to place entities in a project database into a buffer, he does so via the MERGEOUT command, specifying the name of the project and the name of the buffer into which it is to be copied. Entities in the project are copied one at a time by name through the SELECT command. If the user needs a list of the entities of a given type, he may obtain one through the LIST command. Also available here is the HELP command which provides a menu of the other commands presently available. The END command will return the user to the AISIM READY level.

To obtain access to the MERGEOUT sublevel, issue the command,

```
MERGEOUT {PROJECT(project)},{BUFFER(buffer)}
MI      {P(project)}      {BUF(buffer)}
```

where:

{P(project)} is a required parameter indicating the name of the project into which the Entities are to be copied.

{BUF(buffer)} is a required parameter indicating the name of the buffer in which the Entities to be transferred are stored.

FUNCTION RESULT:

The user is given a "*" prompt, from which he can issue one of the following commands.

- 1) LIST {entity-type}, to list entities in project database.
- 2) SELECT {entity-type},{entity-name}, an entity to be merged out of the project database.
- 3) END, which will terminate the selection of entities to be copied.

MERGEOUT COMMAND SUMMARY

END

E

HELP

LIST

L

SELECT {entity-type},{entity-name}

S

MERGEOUT / END

The END command terminates the session at the MERGEOUT sub-level and causes entities in the current project database which have been flagged by the SELECT command to be copied into the current buffer.

COMMAND SYNTAX:

END

FUNCTION RESULT:

The user will be prompted with the question:

DO YOU WANT TO LIST YOUR SELECTIONS ON THE SCREEN?

A "NO" answer will cause the mergeout procedure to take place. When all of the flagged entities have been copied onto the database, the system will return to the LUI Ready level.

A "YES" answer will produce a list of the entities flagged in the SELECT command. The user will then be prompted as to whether he wishes to proceed with the mergeout operation. A "YES" answer to this second question will cause the flagged entities to be copied into the current buffer. A "NO" answer will return the system immediately to the LUI Ready level.

10.4.1 COMMAND: HELP

The HELP command enables the user to obtain a menu of the other command options available in the MERGEOUT sublevel.

COMMAND SYNTAX:

HELP

FUNCTION RESULT:

A menu of available command options is printed on the screen.

10.4.2 COMMAND: LIST

The LIST command enables the user to obtain a list of the names of the entities of a given type that are contained in the current model.

COMMAND SYNTAX:

LIST {entity-type}

L

where:

{entity-type} is the type of entity.

FUNCTION RESULT:

The screen will display a list of the names of the entities of the specified type in the current model.

10.4.3 COMMAND: SELECT

The SELECT command allows the user to specify which entities are to be merged out of a project database to a buffer.

COMMAND SYNTAX:

SELECT {entity-type},{entity-name}

where:

{entity-type} is the type of entity to be merged out,

{entity-name} is the name of the entity to be merged out.

FUNCTION RESULT:

The specified entity is flagged for the mergeout operation. The operation will take place only when the END command is issued.

11. AISIM SIMULATION RESULTS REPORTS

When a simulation is run, a number of Processes are initiated at various times throughout the simulation period. As their execution proceeds they contend for available resources such as machines and operators. The simulation stops at the end of a predefined period and produces output statistics.

In general, any high-level performance factor measurable on a real system in terms of time, percentages, or counts of events can be measured during the model run. Experiments that are virtually impossible to run on a real system can be constructed and easily measured in the model. Specifically, measures that may be obtained are:

- Resource utilization statistics
- Total number of Processes completed
- Average elapsed time for Process completion
- System and job delays associated with actions
- Statistics on queue sizes and timing
- Variable changes during simulation
- System and job delays associated with resources
- Execution count of Process steps

Two forms of output are available to the user as a result of the simulation. Interactive output, displayed on the screen of the HP-2647A graphics terminal, is available at any user-defined breakpoint, at the end of simulation periods, or at the end of the simulation.

The second form of output is a batch listing, obtained off-line, which lists the simulation measures mentioned above.

The following sections describe the simulation outputs and how to obtain them.

11.1 INTERACTIVE RESULTS AND HOW TO OBTAIN THEM

Interactive results can be viewed on the HP-2647A graphics terminal while in the AUI level. A review of the AUI level shows that several commands are available for viewing data after simulation periods, after breakpoints, and after simulation termination. The DEFLOT command is used before simulation is started to select the graphs that the user wishes to view after simulation (see the DEFLOT command description for attributes and statistics of entities that can be graphed). The LISTVAL command can be used at the points mentioned above to view simulation data concerning model entities (see the LISTVAL command description for attributes and statistics of entities that can be viewed). The PLOT command is also used at the points mentioned above to view graphically the statistics which were kept due to the DEFLOT plot definitions. See the PLOT command definition for examples of the forms and graphs that are displayed to the user as a

result of this command.

11.2 REPORT RESULTS AND HOW TO OBTAIN THEM

The commands to view and print batch results are available in the AISIM Ready Level. As the simulation executes, simulation results are automatically stored in a database file named:

PROJECT.REPORT.DATA

where:

PROJECT indicates that the model output report to be accessed was generated by an analyze session on the design database named PROJECT.

Two commands are available to manipulate this data file. The first we will discuss is PRINT. This command is used to print a listing of the simulation reports at the local hardcopy facility. The EDIT command allows the user to view the PROJECT.REPORT.DATA file through the use of the TSO text editor. After the EDIT command is given the system responds with EDIT indicating that the TSO text editor is ready to receive further commands. See the TSO Users Manual for TSO text editor capabilities.

Following is a description of the data contained in the PROJECT.REPORT.DATA FILE.

The PROJECT.REPORT.DATA file contains a number of reports that describe the model that was simulated and the results of the simulation. On the following pages each of these reports is described and examples of results are given.

INITIALIZATION REPORT: This report displays the contents of the model inputs as used during this simulation. Elements of this report are:

- 1) global Constant definition
- 2) Table definition
- 3) global Variable definition
- 4) Item definition
- 5) Queue definition
- 6) Resource definition
- 7) architecture definition
- 8) Action definition
- 9) Process definition
- 10) Load definition
- 11) Scenario definition

Figure 6. Report giving definitions of Constants, Tables, Items and Queues

```

*****
GENERAL FUNCTION MODEL
AISIM VERSION 1
HUGHES AIRCRAFT COMPANY
03/01/81
*****
GLOBAL CONSTANT DEFINITION.....

CONSTANT INITIAL
MNEMONIC VALUE COMMENT
*****
BUFFER 10000 NUMBER OF BYTES IN A BUFFER FOR PLOT INPUT
CHK-TM .002 WRITE CHECK TIME FOR CAMIHEL
D-INITSK .0000004 4 MICRO SECOND DEVIATION
DISKPRI 1 DISK PRIORITY
M-INITSK .000002 20 MICORSECONDS INIT SEEK TIME
N-TERM 1024 NUMBER OF CHARACTERS PER DISPLAY
READ 0 READ FLAG
TRAN-TM 800000 CHARACTERS PER SECOND OVER DAS LINK
WRITE 1 WRITE FLAG

TABLE DEFINITION....

GLOBAL VARIABLE DEFINITION.....

VARIABLE INITIAL
MNEMONIC VALUE COMMENT
*****
C-RAD-TM .00020 TIME TO FORMAT AND PROCESS RADAR DATA

ITEM DEFINITION.....

ITEM DESCRIPTION
*****
REQUEST DISK ACCESS REQUEST
      BYTES 0
      R/W 0
      UNIT 0

QUEUE DEFINITION.....

QUEUE MAXIMUM
MNEMONIC SIZE COMMENT
*****
IO-Q INFINITE QUEUE OF REQUEST FOR DISK

```

Figure 7. Resource report and listing of Legal Path Table

PAGE 2

RESOURCE DEFINITION.....

```

RESOURCE INITIAL
MEMONIC & UNITS  QUEUE  ITEMREF
=====
A                RESOURCE FOR SERIAL PORT
      1          1
      CHRS/SEC  960
B                RESOURCE FOR SERIAL PORT
      1          1
      CHRS/SEC  960
CHN              RESOURCE FOR NODE
      1          1
CP               RESOURCE FOR NODE
      1          1
C4              RESOURCE FOR SERIAL PORT
      1          1
      CHRS/SEC  960
DSK1             RESOURCE FOR DISK UNIT
      1          1
      M-LATE   .0025
      M-SEEK   .0085
  
```

ARCHITECTURE LEGAL PATH DEFINITION

FROM DEVICE	TO DEVICE	NEXT DEVICE	VIA CHANNEL
801	801	S05	CH01.8
801	806	S05	CH01.8
801	811	S05	CH01.8
801	816	S05	CH01.8
801	821	S05	CH01.8
801	826	S05	CH01.8
801	831	S05	CH01.8
801	CHQ	S05	CH01.8
801	HQ1	S05	CH01.8
801	HQ2	S05	CH01.8
801	S01	S05	CH01.8
801	S02	S05	CH01.8
801	S03	S05	CH01.8
801	S04	S05	CH01.8
801	S05	S05	CH01.8
801	S06	S05	CH01.8
801	S07	S05	CH01.8
806	801	S07	CH06.8
806	811	S07	CH06.8
806	816	S07	CH06.8
806	821	S07	CH06.8
806	826	S07	CH06.8
806	831	S07	CH06.8
806	CHQ	S07	CH06.8
806	HQ1	S07	CH06.8
806	HQ2	S07	CH06.8
806	S01	S07	CH06.8
806	S02	S07	CH06.8
806	S03	S07	CH06.8
806	S04	S07	CH06.8
806	S05	S07	CH06.8
806	S06	S07	CH06.8
806	S07	S07	CH06.8

Figure 8. Report giving definitions of Actions and Processes

ACTION DEFINITION.....

ACTION	ACTION	PHNEMONIC	CLASS	COMMENT
CHGSD.OH	MACHINE	CHQ	PROCESSING OF GRAPHICS REQUEST	
CHCHD.OH	MACHINE	CHQ	PROCESSING OF HARDCOPY REQUEST	
CS.OH	MACHINE		PROCESSING TO PERFORM A CONTEXT SWITCH	
DUMMFACT	MACHINE		ACTION TO ENABLE CYCLIC PGM S CYCLES	
HQ.OH	MACHINE	HQ	PROCESSING OF MESSAGE	
OVERHEAD	MACHINE		TIME REQUIRED TO SEND MESSAGE ACROSS A CHANNEL	
ROUTE.OH	MACHINE		PROCESSING REQUIRED TO ROUTE A MESSAGE	

PROCESS DEFINITION.....

PROCESS	PHNEMONIC	DESCRIPTION
B-ORIGIN		THIS IS A B-NODE STUB PROCESS

ENTRY	OPCODE	PARM	PARM	PARM	COMMENT
	START	ALL			
	GIVEN	MSG			
	RETURN	MSG			
	END				

ITEM REFERENCES OF THIS PROCESS

MSG

PROCESS	PHNEMONIC	DESCRIPTION
BECHO		THIS PROCESS ECHOES MESSAGE BACK TO ORIGINATOR

ENTRY	OPCODE	PARM	PARM	PARM	COMMENT
	START	ALL			
	GIVEN	MSG			
	RETURN	MSG			
	ASSIGN	MSG	FNODE		GET ORIGINATING NODE
	CALL	REQ-I/O	WAIT	0	
	GIVEN	B-ORIGIN	B-PRI	0	ONWAIT
		B-LNTH	TO.NODE		
	END				

LOCAL VARIABLES OF THIS PROCESS

TO.NODE 1

ITEM REFERENCES OF THIS PROCESS

MSG

Figure 9. Report giving Load and Scenario definitions

LOAD DEFINITION.....

LOAD MNEMONIC	DESCRIPTION
LOADNULL	MULL LOAD TO ALLOW SEVEN PERIODS OF FULL LOAD
LOAD	NODES

PROCESS MNEMONIC	NUMBER	SCHEDULE METHOD	METHOD	SCHEDULE PRIORITY

LOAD MNEMONIC	DESCRIPTION
LOADS01	AIR BASE LOAD TO 901
LOAD	NODES

PROCESS MNEMONIC	NUMBER	SCHEDULE METHOD	METHOD	SCHEDULE PRIORITY
DATAB01	125	INTERVAL 0	1440000	0
DATAB02	125	INTERVAL 0	1440000	0
DATAB03	125	INTERVAL 0	1440000	0
DATAB04	125	INTERVAL 0	1440000	0

SCENARIO DEFINITION....

SCENARIO MNEMONIC	DESCRIPTION
SCENARIO	MITRE STUDY CASE 1A - SIM UNIT IS 10 MICROSECONDS

LOAD MNEMONIC	LOAD MNEMONIC	LOAD MNEMONIC	LOAD MNEMONIC	LOAD MNEMONIC	LOAD MNEMONIC	LOAD MNEMONIC
LOADNULL	LOADNULL	LOADNULL	LOADNULL	LOADNULL	LOADNULL	LOADNULL
LOAD	LOAD	LOAD	LOAD	LOAD	LOAD	LOAD
LOADNULL	LOADNULL	LOADNULL	LOADNULL	LOADNULL	LOADNULL	LOADNULL

PROCESS MNEMONIC	TIME TO SCHEDULE	PRIORITY	PROCESS MNEMONIC	TIME TO SCHEDULE	PRIORITY
LOADS01	0	0			

0 ERRORS WERE DETECTED DURING MODEL INITIALIZATION

11.2.1 CONSTANT REPORT This report shows the value of the Constants at simulation termination. An example of this report is shown below where the labeled columns have the following significance.

CONSTANT: the name of the constant

CURRENT VALUE: the value (in real numbers) at the time of the end of the simulation.

SIMULATION TIME = 17999996.00000 UNITS

CONSTANT REPORT

	CURRENT
CONSTANT VALUE...	

ZERO	0.

11.2.2 VARIABLE REPORT

Variable reports are divided into the numeric and the non-numeric. A sample of the report for numerical variables is shown where the columns have the following significance.

VARIABLE: the name of the Variable

TOTAL SAMPLES: the number of times the Variable has been set to a value over the simulation period, including its initialization at the start of the simulation.

CURRENT: the value of the Variable at the start of the simulation.

MEAN: the mean of all values that the Variable was set to over the simulation (i.e., the sum of the values divided by TOTAL SAMPLES).

STD DEV: the standard deviation of the variation in the values over the simulation.

MINIMUM: the minimum value that the Variable took on during the simulation.

MAXIMUM: the maximum value that the Variable took on during the simulation.

The report for Variables taking non-numeric values is illustrated, where the labeled columns have the following significance.

VARIABLE: the name of the Variable.

CURRENT TYPE: the type of entity or construct that the Variable is set to at the end of the simulation.

CURRENT VALUE: the name of the entity or construct to which the variable is set at the end of the simulation.

SIMULATION TIME = 179999988.00000 UNITS

VARIABLE REPORT

VARIABLE	TOTAL SAMPLES	CURRENT...	MEAN.....	STD DEV...	MINIMUM...	MAXIMUM...
B-LNTH	1	750.00000	750.00000	0.	750.00000	750.00000
B-FRI	1	11.00000	11.00000	0.	11.00000	11.00000
BSLNTH	1	750.00000	750.00000	0.	750.00000	750.00000
SECHOPRI	1	11.00000	11.00000	0.	11.00000	11.00000
CHCGOVHD	1	3.00000	3.00000	0.	3.00000	3.00000
CHCHOVHD	1	1.00000	1.00000	0.	1.00000	1.00000
CHQLNTH	1	750.00000	750.00000	0.	750.00000	750.00000
CHQPRI	1	11.00000	11.00000	0.	11.00000	11.00000
GRLNTH	1	1.0000E 04	1.0000E 04	0.	1.0000E 04	1.0000E 04
HCLNTH	1	200.00000	200.00000	0.	200.00000	200.00000
HCPRI	1	11.00000	11.00000	0.	11.00000	11.00000
HCLLNTH	1	6300.00000	6300.00000	0.	6300.00000	6300.00000
HCGGLNTH	1	200.00000	200.00000	0.	200.00000	200.00000
HCGGPRI	1	11.00000	11.00000	0.	11.00000	11.00000
HCHGLNTH	1	200.00000	200.00000	0.	200.00000	200.00000
HCHGPRI	1	11.00000	11.00000	0.	11.00000	11.00000
HQLNTH	1	750.00000	750.00000	0.	750.00000	750.00000
HGOVHD	1	8.00000	8.00000	0.	8.00000	8.00000
HOPRI	1	11.00000	11.00000	0.	11.00000	11.00000
ROUTEPRI	1	0.	0.	0.	0.	0.
VO.CS	1	.00000	.00000	0.	.00000	.00000
VM.CS	1	.00000	.00000	0.	.00000	.00000
VM.RCUTE	1	8.00000	8.00000	0.	8.00000	8.00000
VRATE	1	1.00000	1.00000	0.	1.00000	1.00000

NON-NUMERIC VARIABLES...

VARIABLE	CURRENT TYPE	CURRENT VALUE
VAR	RESOURCE	CPU

11.2.3 ITEM REPORT

This report is illustrated ,where the labeled columns have the following significance.

ITEM NAME: the name of the Item type.

NUMBER CREATED: the number of instances of this Item that have been created with the CREATE or SEND Primitives over the simulation.

NUMBER DESTR'D: the number of instances of this Item that have been destroyed with the DESTROY Primitive over the simulation.

MINIMUM: the minimum time any instance of the Item was in the system.

MAXIMUM: the maximum time any instance of the Item was in the system.

AVERAGE: the average time any instance of the Item was in the system.

STD DEV: the standard deviation of the variation in the times the Item spent in the system.

The statistics MINIMUM, MAXIMUM, AVERAGE, STD DEV are based on the individual Item instances' time in the system. This statistic is calculated whenever an Item instance is destroyed (with the DESTROY Primitive) and is equal to the time of destruction minus the time of creation (with the CREATE or SEND Primitive). Therefore, Items in the system that have not been destroyed at simulation end will not be reflected in these statistics.

SIMULATION TIME = 179999968.00000 UNITS

ITEM REPORT

ITEM NAME	NUMBER CREATED	NUMBER DESTR'D	TIME IN SYSTEM			
			MINIMUM...	MAXIMUM...	AVERAGE...	STD DEV...
CH26.BI	0	0	0.	0.	0.	0.
CH31.AI	0	0	0.	0.	0.	0.
CH31.BI	0	0	0.	0.	0.	0.
HQ1I	0	0	0.	0.	0.	0.
HQ2I	0	0	0.	0.	0.	0.
MSG	668	668	6000.00000	3.0160E 05	1.4235E 05	1.3323E 05
S01I	0	0	0.	0.	0.	0.
S02I	0	0	0.	0.	0.	0.
S03I	0	0	0.	0.	0.	0.
S04I	0	0	0.	0.	0.	0.
S05I	0	0	0.	0.	0.	0.
S06I	0	0	0.	0.	0.	0.
S07I	0	0	0.	0.	0.	0.

11.2.4 RESOURCE REPORT

This report gives statistics on each Resource's presence in the idle queue, busy queue, and inactive queue as well as on the number of Processes put into a wait queue for the Resource. Four kinds of statistics are kept on each queue or state: (1) entities put into the queue (INTO), (2) entities taken out of of a queue (OUT OF), (3) the number in the queue (#) and (4) the time entities spent in the queue (TIME).

An example of the Resource Report on these queues is provided . For each row of each queue the numbers have the following significance.

The TOTAL NUMBER of the INTO and OUT OF rows indicate the number of entities that were , respectively, placed in or taken out of the queue.

The CURRENT # is the number of entities in the queue at the time the simulation run was completed.

The MEAN # is the average of the number of entities in the queue over the simulation.

The STD DEV # is the standard deviation of the variation in the number of entities in the queue over the simulation.

The MINIMUM # is the minimum number of entities in the queue at one time over the simulation.

The MAXIMUM # is the maximum number of entities in the queue at one time over the simulation.

The MEAN TIME is the average time any entity was in the queue.

The STD DEV TIME is the standard deviation of the variation in the time that any entity was in the queue at any given time.

The MINIMUM TIME is the minimum time any entity was in the queue.

The MAXIMUM TIME is the maximum time any entity was in the queue.

The field labeled "CURRENTLY ALLOCATED TO PROCESSES:" provides a list of the Processes whose task instances had allocated the Resource at simulation end.

The field labeled "PROCESSES CURRENTLY IN WAIT:" provides a list of the Process task instances which were suspended while waiting for the Resource at the end of the simulation.

Figure 10. SAMPLE RESOURCE UTILIZATION REPORT

RESOURCE	TOTAL NUMBER	CURRENT...	MEAN.....	STD DEV...	MINIMUM...	MAXIMUM...
=====						
CPU						
INTO IDLE	1					
OUT OF IDLE	0					
# IDLE		1.000	1.000	0.	1.000	1.000
IDLE TIME			0.	0.	0.	0.
INTO BUSY	0					
OUT OF BUSY	0					
# BUSY		0.	0.	0.	0.	0.
BUSY TIME			0.	0.	0.	0.
INTO INACT.	0					
OUT OF INACT.	0					
# INACTIVE		0.	0.	0.	0.	0.
INACTIVE TIME			0.	0.	0.	0.
INTO WAIT	0					
OUT OF WAIT	0					
# WAITING		0.	0.	0.	0.	0.
WAIT TIME			0.	0.	0.	0.
CURRENTLY ALLOCATED						
TO PROCESSES: NONE						
PROCESSES CURRENTLY						
WAITING: NONE						

11.2.5 ACTION REPORT

The Action Report provides the user with statistics on the time consumed by each Action. Statistics are gathered on two aspects of such time consumption, called "useful time" and "delay time".

"Useful time" is equal to the amount of time the action was being executed, whereas "delay time" is the time between the initiation and completion of an Action during which the execution of the Action (i.e., the Process in which it appears) is suspended. Both useful time and delay time are calculated only upon the completion of the Action.

An example Action Report is shown. The name immediately below the ACTION heading is the user-defined name of the Action. For the row labeled USEFUL TIME the statistics have the following significance:

TOTAL SAMPLES: the number of times the useful time was calculated (i.e., the number of times the Action was completed).

MEAN: the average useful time of this Action over the simulation (i.e., the total time taken by the Action divided by TOTAL SAMPLES).

STD DEV: the standard deviation of the variance in the useful times over the simulation.

MINIMUM: the minimum time taken in the execution of the Action over the simulation.

MAXIMUM: the maximum time taken in the execution of the Action over the simulation.

% TIME OF TOTAL: the percent of the total simulation time for which this Action was executing. Since AISIM allows for the parallel execution of the same Action, this figure can be greater than 100.

The figures in the row labeled DELAY TIME have the following significance.

TOTAL SAMPLES: the number of times the delay time was calculated (i.e., the number of times the Action was completed). This will always be equal to the TOTAL SAMPLES of USEFUL TIME.

MEAN: the average time the Action was delayed during execution over the simulation (i.e., the total time taken up in delay divided by TOTAL SAMPLES).

STD DEV: the standard deviation of the variance in the delay times over the simulation.

MINIMUM: the minimum time taken in the delay of an Action over the simulation.

MAXIMUM: the maximum time taken in the delay of an Action over the simulation.

Note that % OF TOTAL is not calculated for the delay time.

PAGE 7

SIMULATION TIME = 70.00000 UNITS

ACTION REPORT

ACTION	TOTAL SAMPLES	MEAN.....	STD DEV...	MINIMUM..	MAXIMUM...	% TIME OF TOTAL.
ACT						
USEFUL TIME	6	10.000	0.	10.000	10.000	.857
DELAY TIME	6	1.667	3.727	0.	10.000	

11.2.6 QUEUE REPORT

The Queue Report provides statistics on the utilization of user-defined Queues. An example is provided. The report contains information both on the number of entities stored on the Queue as well as information on the impact the utilization of the Queue had on Process execution and suspension. The rows labeled FILED ON, REMOVED FROM, # IN QUEUE and TIME IN QUEUE key statistics on the manipulation of the Queue itself. The rows labeled TASKS BLOCKED, TASKS RESUMED, # BEING BLOCKED, TIME BLOCKED refer to statistics on Process tasks that have been suspended because they attempted to file an entity on a Queue that was full (i.e., whose maximum number had been exceeded.)

The statistics in each category have the following significance.

The TOTAL NUMBER/FILED ON is the number of entities that have been filed on the Queue over the whole simulation.

The TOTAL NUMBER/REMOVED FROM is the number of entities that have been removed from the Queue over the simulation.

The CURRENT/# IN QUEUE is the number of entities on the Queue at the time of simulation end.

The MEAN/# IN QUEUE is the average number of entities on the Queue at any time over the simulation.

The STD DEV/# IN QUEUE is the standard deviation of the variation in the number of entities on the Queue at any time over the simulation.

The MINIMUM/# IN QUEUE is the minimum number of entities on the Queue at any time during the simulation (this statistic is always zero since the Queue will be empty at the start of the simulation).

The MAXIMUM/# IN QUEUE is the maximum number of entities residing on the Queue at any time during the simulation.

The MEAN/TIME IN QUEUE is the average time entities spent on the Queue throughout the simulation.

The STD DEV/TIME IN QUEUE is the standard deviation of the variation in times entities spent on the Queue throughout the simulation.

The MINIMUM/TIME IN QUEUE is the least amount of time any entity spent on the Queue during the simulation.

The MAXIMUM/TIME IN QUEUE is the greatest amount of time any entity spent on the Queue during the simulation.

The statistics on the blocking of tasks due to the filling of Queues have the following significance.

The TOTAL NUMBER/TASKS BLOCK is the number of Process tasks that were suspended over the simulation due to Queue blocking.

The TOTAL NUMBER/TASKS RESUMED is the number of Process tasks resumed after having been blocked due to the filling of a Queue.

The CURRENT/# BEING BLOCKED is the number of Process tasks blocked at the time of simulation end.

The MEAN/# BEING BLOCKED is the average number of Process tasks blocked over the simulation.

The STD DEV/# BEING BLOCKED is the standard deviation in the variation in the number of tasks blocked over the simulation.

The MINIMUM/# BEING BLOCKED is the fewest number of Process tasks blocked at any time during the simulation.

The MAXIMUM/# BEING BLOCKED is the greatest number of Process tasks blocked at any time during the simulation.

The MEAN/TIME BLOCKED is the average time any Process task was blocked during the simulation.

The STD DEV/TIME BLOCKED is the standard deviation of the variation in the times Process tasks were blocked during the simulation.

The MINIMUM/TIME BLOCKED is the least amount of time a Process task was blocked during the simulation.

The MAXIMUM/TIME BLOCKED is the greatest amount of time a Process task was blocked during the simulation.

Figure 11. SAMPLE QUEUE UTILIZATION REPORT

QUEUE	TOTAL NUMBER	CURRENT...	MEAN.....	STD DEV...	MINIMUM...	MAXIMUM...
MSGQ3						
FILED ON	0					
REMOVED FROM	0					
% IN QUEUE		0.	0.	0.	0.	0.
TIME IN QUEUE			0.	0.	0.	0.
TASKS BLOCKED	0					
TASKS RESUMED	0					
% BEING BLOCKED		0.	0.	0.	0.	0.
TIME BLOCKED			0.	0.	0.	0.

Figure 12. SAMPLE PROCESS UTILIZATION REPORT

PROCESS	TOTAL SAMPLES.	SUM.....	MEAN.....	STD DEV...	MINIMUM...	MAXIMUM...
TASKS						
TOTAL	2	30.000	15.000	5.000	10.000	20.000
PROCESS WAIT	0	0.	0.	0.	0.	0.
RESOURCE WAIT	5	10.000	2.000	4.000	0.	10.000
TOTAL %	% AUTO	% CALL	% OF	% NOT	% TIMES	
SCHEDULE	SCHEDULE	SCHEDULE	COMPLETE	COMPLETE	SUSPEND.	
2	2	0	2	0	2	

PROCESS	DESCRIPTION
TASKS	00000460

COUNT	ENTRY	OPCODE	PARM	PARM	PARM	COMMENT
2		START				00000480
2		ALLOC	RES2			00000490
2		ALLOC	RES2			00000500
3		ACT	CONSTANT 10			00000510
2		END				00000520

PROCESS REPORT: This report gives information on all aspects of Process exexutions. As mentioned before, Processes contend for Resources and many times must wait for another Process to complete before the current Process completes. Times spent in these states as well as other important data are recorded automatically for the user.

The Process Report provides the following statistics:

- 1) total samples - the number of times the Process began execution, the number of times the current Process waited
- 2) the sum total of time spent in all executions of this Process, sum of waits on Processes and also resources.
- 3) the mean time required for execution of the Process, for waiting on Processes, for waiting on resources.
- 4) the standard deviation of the variation of time the Process required for execution, for waiting on Processes, for waiting on resources.
- 5) the minimum time required for Process execution, minimum time spent waiting for other Processes, minimum time spent waiting for resources.
- 6) the maximum time required for Process execution, maximum time spent waiting for other Processes, maximum time spent waiting for resources.
- 7) total number of times this report was scheduled to execute.
- 8) the number of times this Process was scheduled to execute due to a Process request in a load or a Process interrupt in a scenario.
- 9) the number of times this Process was scheduled to execute due to a call from another Process.
- 10) the total number of times this Process completed execution.
- 11) the total number of times this Process did not complete execution.
- 12) total number of times the execution of this Process was suspended during execution (should equal fields 13 & 14).

- 13) total number of times the execution of this Process was suspended during execution due to a Process of higher priority "stealing" a required resource.
- 14) total number of times the execution of this Process was suspended during execution due to the level of required resources being reset below the required level through the use of a RESET primitive.
- 15) names of items used in this Process.
- 16) number of each item created by this Process.
- 17) number of each item passed to this Process.
- 18) number of each item passed out of this Process.
- 19) number of each item destroyed by this Process.
- 20) total number of items used in this Process.
- 21) mean time each item was held by this Process.
- 22) minimum time each item was held by this Process.
- 23) maximum time each item was held by this Process.
- 24) standard deviation of the variation of time each item was held by this Process.
- 25) verbal description of the Process.
- 26) how many times each primitive in the Process was executed.
- 27) any entry primitives and their names.
- 28) names of other primitives in this Process.
- 29) any parameters or items associated with each primitive in the Process.
- 30) any comment associated with each primitive in the Process.

An example of a Process Report is shown.

11.3 COMMANDS RELEVANT TO VIEWING OUTPUT REPORTS

To view output reports of simulation runs of a model from the AISIM READY level, one issues the command,

EDIT

or

E

Since the output report is a long list of information which is too long to fit on a terminal screen, to view it all, one must use some text editing commands. Below is a brief review of the commands that are most useful for this purpose. (This discussion refers to the IBM TSO text editor).

11.3.1 TOP, BOTTOM

To orient the screen to either the top or bottom of the report one should enter one of these two commands.

11.3.2 UP, DOWN To move the report either up or down on the screen n lines issue the command,

UP N

or

DOWN N

and the line n lines down from the current one will be printed. A default of 1 is assumed if n is not explicitly typed

11.3.3 FIND To find a certain sequence of characters, sequence, enter the command,

FIND /sequence/

and the screen will print the nearest line down in the text containing the characters sequence.

11.3.4 LIST To print n consecutive lines down from the one to which one is currently oriented, issues the command,

LIST * n

and the next n lines will be displayed on the screen.

APPENDIX A

A. OPERATIONAL PROCEDURES AND IMPORTANT INFORMATION

A.1 IMPORTANCE OF DATABASE BACKUP AND ALLOCATION

Processes and the other model entities are stored on disk as they are input to AISIM. Changes and additions made to this information are reflected in the current version of the database on disk. It is possible for this database to be damaged so that it is unusable if the computer system fails or if the input session is abnormally terminated while a change or addition is being made. In addition, errors made in inputting may make the stored information nonsensical if they are severe enough. For these reasons, there are two ways provided to "backup" the database with the COPY/NOCOPY option and with the BACKUP command.

The user may enter the DUI level in COPY mode. In this case, a copy of the actual database is made by AISIM before allowing the user to make any changes or additions. All subsequent changes and additions are applied to the working copy rather than to the permanent database. To render changes permanent, issue the DUI level "SAVE" command. The changes are then applied to the "real" database. In this way, if the user realizes a severe error has been made in the additions or changes applied to the working copy, the session may be abnormally terminated without a save, thus "loosing" the erroneous changes.

The other means of providing backup for the database involves NOCOPY mode and the BACKUP command. If the user enters input mode in NOCOPY mode no working copy of the database is made and all changes or additions are made directly to the actual database. In this case, it is wise to periodically create a backup copy of the database with the AISIM Ready Level command "BACKUP". Should a database be damaged, it may be recreated from the last BACKUP copy by using the "RESTORE" command.

It may happen that while making additions (or even changes) to a database the database exceeds its original allocation of space on disk memory and requires more room. If this happens help should be sought from the SED Design Analysis Section to reallocate the database, giving it more room.

A.2 ABNORMAL TERMINATION OF A DUI OR AUI SESSION

To terminate a DUI or AUI session normally the user must enter the command END. If the user becomes entwined in a situation which disallows normal system operation, the following procedures should be followed:

It should be noted that while in a DUI session, only the data entered prior to the last SAVE command will remain intact after

this procedure is executed. If the system appears to malfunction, caution should be used in issuing a SAVE command. If the database is the source of the malfunction and a SAVE command is issued, the user might destroy the entire database. It is better to lose one session's data (by not saving) than to destroy an entire database.

This procedure assumes that the user is using a Hewlett Packard HP-2647A terminal.

- 1) Strike the TERMINAL RESET key until the message 'TERMINAL READY' appears in the upper left hand corner of the screen; two strikes in a one-second period are required.
- 2) Strike the BREAK/INTERRUPT key once.
- 3) Enter the command STOP (this brings the user to the TSO command mode and results in a READY prompt).

If no response to these procedures is seen, the user should disconnect the modem, and try to log in and reinitiate the system.

If the system responds by displaying 'READY' or '/', the user should reinitiate the system.

A.3 AISIM PLOTS

The following section is intended to describe in detail how the simulation plot results produced by the AISIM Analyze function are generated. This discussion addresses the implementation of the plot function in AISIM with respect to the physical characteristics of the HP2647A display and the driving software. For a user of AISIM, it is generally not necessary to be aware of implementation specific details. This section has been included because the plot output from AISIM simulation runs is the most visible form of output produced. This data may appear to contradict other results produced by the AISIM Analyze function (output listing statistics). This explanation is intended to describe how this function works so that the AISIM user can explain apparent anomalies.

AISIM produces plotted data for many statistics. The plots represent "instantaneous" output from the simulation because in all cases, a defined statistic is plotted against time (the y-axis is the statistic value, the x-axis is the simulation clock). Time is normally considered to be continuous; therefore, it is "reasonable" to assume that AISIM plots are continuous. In reality, this is not the case. AISIM plots are produced by sampling statistics at discrete intervals during the simulation. Each sample defines a point on the plot. A couple of relationships need to be known to understand how this sampling technique produces plots.

The first relationship a user must be aware of is the resolution of the display screen. The HP2647A graphics terminal has a raster scan display. A raster is the smallest addressable unit which can be illuminated on the screen. Within the AISIM plot axis there are about 700 rasters along the x-axis. What this implies is that up to 700 points can be plotted along the x-axis without exceeding the hardware limitations of the display. When an AISIM user specified a plot be displayed which has more than 700 points, the AISIM software reduces the data sent to the terminal so that it can be displayed. This data reduction has the effect of "ignoring" some points. When points are ignored, the obvious result is that the plots lose accuracy. This can account for discrepancies between the plotted data and the simulation summary results, specifically with respect to the minimum and maximum statistics. The simulation report may indicate that a Resource queue had a maximum length of 100 when a plot of the current number in wait for a Resource over time indicates only a maximum value of 80.

Another problem which can occur with respect to plotting is that the plot sampling can miss activity occurring in the simulation because the sample interval is too long. The following default relationship is embedded in the AISIM software. One hundred data points are sampled for each period in the Scenario definition of a simulation run.

What this implies is that is a Scenario is defined to have only one period, only one hundred plot samples will be collected. The sample interval is calculated as the period length/100.0. Suppose the period length is defined to be 3000 units (where units are seconds, this is 1 hour). Plot samples are collected every 36 units (or 36 seconds). If activity occurs in the model over time intervals less than 36 units, this data will not be captured for plotting. This could occur if a user wanted to see a plot of disk utilization of a computer system over a one-hour time frame. Since disk operations occur in seconds or less, a plot of the current number busy of the Resource disk would miss most of the data points if samples were taken every 36 seconds.

It is possible to adjust the plot sampling interval in the Scenario definition. The number of samples collected for each plot is computed as the number of periods in the Scenario multiplied by 100 points.

To reiterate, AISIM plots produce graphs of statistics collected during a simulation run, and display the results over time. The data for these plots is collected by sampling discrete intervals. It is not generated by state changes detected by the simulator. Therefore, the "instantaneous" plots of "CURRENT" data over time can disagree with accumulated statistics in the simulation listing.

APPENDIX B

B. AISIM ERRORS

If there are errors detected during the initialization, an error message will be written below the invalid entry. Following is a list of the initialization error messages and their causes.

ERROR - VALUE MUST BE NUMERIC

A non-numeric value was found as the value of a Constant. The defined value of a Constant must be numeric.

ERROR - TABLE ENTRIES MUST BE NUMERIC

A non-numeric value was found as an entry in a D or C type Table. All D or C type Table entries must be numeric.

ERROR - ALPHA TABLE X ENTRY IS ILLEGAL TYPE

In an alpha Table, an x entry was a Keyword or other invalid entry. The only valid entries are references to Actions, Items, Processes, Queues, Resources, or Tables.

ERROR - ALPHA TABLE Y ENTRY IS ILLEGAL TYPE

In an alpha Table, a y entry was a Keyword or other invalid entry. The only valid entries are references to Actions, Items, Processes, Queues, Resources, or Tables.

ERROR - VARIABLE INITIALIZED TO ILLEGAL TYPE

A Keyword or other illegal type was found as the value of a Variable. Variables must be initialized to Actions, Processes, Queues, Resources, Tables, Alpha Literals, or numerics.

ERROR - ATTRIBUTE DEFINED MORE THAN ONCE

An Item, Process, or Resource attribute was defined more than once. The duplicate attribute definition should be removed.

ERROR - ***** NOT DEFINED AS A GLOBAL CONSTANT

A non-numeric value in the size field of a QUEUE was

not defined as a global Constant. A non-numeric value for the size must either be the word "INFINITE" or be a previously defined global Constant.

A non-numeric value in the total or initial units field of a Resource was not defined as a global Constant. The total and initial units of a Resource must each be either a numeric value or be a previously defined global Constant.

In the definition of a Scenario, a non-numeric value in the schedule field was not defined as a global Constant. The schedule must be a numeric value or a defined Constant.

In the definition of a Scenario, a non-numeric value in the priority field was not defined as a global Constant. The priority must be a numeric value or a defined Constant.

ERROR - INITIAL # OF RESOURCE UNITS IS GREATER THAN TOTAL # OF UNITS

In a Resource definition, the initial number of units defined was greater than the total number to units of that Resource which were to be made available.

ERROR - FROM NODE IS NOT DEFINED AS A RESOURCE

In an entry in the Legal Path Table, the node specified in the FROM DEVICE column was not the name of a defined Resource.

ERROR - TO NODE IS NOT DEFINED AS A RESOURCE

In an entry in the Legal Path Table, the node specified in the TO DEVICE column was not the name of a defined Resource.

ERROR - NEXT NODE IS NOT DEFINED AS A RESOURCE

In an entry in the Legal Path Table, the node specified in the NEXT DEVICE column was not the name of a defined Resource.

ERROR - LINK IS NOT DEFINED AS A RESOURCE

In an entry in the Legal Path Table, the link specified in the VIA LINK column was not the name of a defined Resource.

ERROR - LABEL MUST START IN COLUMN 1 OR OPCODE MUST START IN COLUMN 10

In a Process definition, a value was encountered which did not start in column 1 or in column 10. If the value is a label, it must start in column 1, or if it is an opcode, it must start in column 10.

ERROR - OPCODE MUST START IN COLUMN 10

In a process definition, a non-label value was encountered which did not start in column 10. All opcodes must start in column 10.

ERROR - ***** NODE NAME IS NOT RECOGNIZED AS A RESOURCE

An invalid value was encountered in the node field of a Process definition. This field must be blank, contain the word "ALL", or contain a value which resolves to the name of a defined Resource.

ERROR - ***** NAME IN GIVENS LIST IS IN ERROR IN THIS CONTEXT

GLOBAL NAMES, NUMBERS AND CLOCK CAN NOT BE GIVEN

The value of a given parameter for the START figure of a Process was either a numeric value or the CLOCK. Numeric values and the CLOCK cannot be used as given parameters in a Process.

ERROR - ***** ITEM IN RECEIVES LIST IS IN ERROR

This is a general message indicating an error in a START figure of type "ITEM" of a Process. This message is generally followed by one of the two following messages which more specifically describe the error.

ERROR - ITEM APPEARS TWICE IN RECEIVES LIST

In the definition of a START figure of type "Item," an Item was listed more than once. An Item should only occur once in the receives list of the START figure.

ERROR - REFERENCE IN RECEIVES LIST IS NOT DEFINED AS AN ITEM

In the definition of a START figure of type "ITEM," a value which was listed in the receives list was not defined as a Item. A Process with an ITEM START can only receive Items.

ERROR - ***** NUMERIC REFERENCE IN CALL PROCESS FIELD

In the definition of a CALL Primitive in a Process, the

process name field contained a numeric value or a keyword. This field must contain the name of a defined Process to be initiated.

ERROR - RETURN PARAMETERS NOT ALLOWED FOR CALL NOWAIT OR BLOCK

In the definition of a CALL Primitive in a Process, return parameters were defined, but the CALL option was defined as NOWAIT or BLOCK. Only Processes called with a WAIT option can return parameters.

ERROR - ***** NUMERIC OR GLOBAL MAY NOT BE USED AS RETURN

In the definition of a CALL Primitive in a Process, a numeric value, keyword, or the CLOCK was defined as a return parameter. Numeric values, keywords and the CLOCK cannot be used as return parameters.

ERROR - BRANCH CONTINUATION DOES NOT FOLLOW A BRANCH STATEMENT

In the definition of a BRANCH Primitive of a Process, the label to branch to was not given. A branch Primitive must include a label to branch to.

ERROR - KEYWORD CANNOT BE USED IN PROB

In the definition of a probabilistic BRANCH Primitive of a process, CLOCK or a keyword was used as the probability of BRANCH. These cannot be used as the BRANCH probability. Valid values for the BRANCH probability are numeric values and local and global Variables and Constants.

ERROR - ***** CHECK REFERENCE MUST BE RESOURCE OR QUEUE

In the definition of a TEST primitive in a Process, the value to be tested was defined as a numeric, a global Variable, or a global Constant. The value to be tested must be a reference to either a Resource or Queue.

ERROR - ***** NUMERIC REFERENCE INVALID IN RESOURCE FIELD

In the definition of a RESET primitive in a Process, the value to be reset was a reference to a numeric value. The value to be reset must be a reference to defined Resource whose allocation is to be changed.

In the definition of an ALLOC primitive in a process, the value in the name field was a reference to a numeric value. The value in the name field must be the name of a reference to a defined Resource which is to

be allocated.

In the definition of a DEALLOC primitive in a Process, the value in the name field was a reference to a numeric value. The value in the name field must be the name of a reference to a defined Resource which is to be deallocated.

ERROR - BRANCH LABEL ***** NOT DEFINED IN OFD

In the definition of a Process, a BRANCH primitive referenced a label for which there was no corresponding ENTRY label defined. An ENTRY primitive must be used to define the label to be BRANched to.

ERROR - LOOP LABEL ***** NOT DEFINED IN PROCESS

In the definition of a Process, a LOOP primitive referenced a label for which there was no corresponding ENTRY label defined. An ENTRY primitive must be used to define the label to be BRANched to.

ERROR - CHECK LABEL ***** NOT DEFINED IN PROCESS

In the definition of a Process, a TEST primitive referenced a label for which there was no corresponding ENTRY label defined. An ENTRY primitive must be used to define the label to be BRANched to.

ERROR - COMPARE LABEL ***** NOT DEFINED IN PROCESS

In the definition of a Process, a COMPARE primitive referenced a label for which there was no corresponding ENTRY label defined. An ENTRY primitive must be used to define the label to be BRANched to.

ERROR - ***** ALREADY DEFINED AS AN ENTRY NAME IN THIS PROCESS

In a Process definition, an ENTRY primitive was defined twice with the same label. A label can occur only once in a Process.

ERROR - '*****' KEYWORD CAN NOT BE ASSIGNED NEW VALUE

In the definition of an ASSIGN primitive in a Process, an attempt was made to assign a new value to a Keyword other than \$CNODE. Only the \$CNODE keyword can be assigned a new value.

ERROR - NUMERIC QUANTITY CAN NOT BE ASSIGNED A VALUE

In an ASSIGN primitive of a Process, an attempt was

made to assign a new value to a numeric value. The only entities which can be assigned a new value are attributes, Variables, and local variables.

ERROR - ***** GLOBAL CONSTANT CANNOT BE ASSIGNED A NEW VALUE

In the definition of an ASSIGN primitive in a Process, an attempt was made to assign a new value to a global Constant. The only entities which can be assigned a new value are attributes, Variables, and local variables.

ERROR - '*****' - NOT RECOGNIZED AS A LOGICAL RELATION

In the definition of a COMPARE primitive in a Process, the relation field was invalid. Valid relations are EQ, NE, GE, GT, LE, and LT.

ERROR - '*****' IS NOT RECOGNIZED AS AN ARITHMETIC OPERATION OR A LOCAL VARIABLE

In the definition of an EVAL primitive in a Process, the function specified was invalid. The function field can also contain the name of a local variable which is a reference to a defined Table.

ERROR - '*****' A GLOBAL CONSTANT NUMERIC OR KEYWORD CANNOT BE ASSIGNED TO

In the definition of an EVAL primitive in a Process, a global Constant, numeric or a keyword was specified in the set variable field. The only entities which can be assigned a new value by an EVAL are global Variables and local variables.

ERROR - ***** REFERENCE INVALID IN ITEM FIELD

In the definition of a SEND primitive in a Process, the list of Items to be sent to a Process contained an invalid value. Only Items can be sent to a Process.

In the definition of a CREATE primitive in a Process, the list of Items to be created included an invalid value. Only Items can be created by a CREATE primitive.

In the definition of a DESTROY primitive in a Process, the list of Items to be destroyed included an invalid value. Only Items can be destroyed by a DESTROY primitive.

In the definition of a FILE primitive in a Process, the

Item field contained an invalid value. The item field must contain the name of a reference to a defined Item or a defined Resource.

In the definition of a FIND primitive in a Process, the Item field contained an invalid value. The item field must contain the name of a local Variable to be set.

In the definition of a REMOVE primitive in a Process, the Item field contained an invalid value. The item field must contain the name of a variable to be set.

ERROR - ***** INVALID QUEUE OPTION

In the definition of a FILE primitive in a Process, the option field contained an invalid option. The valid options are FIRST, LAST, NEXT, and BEFORE.

In the definition of a FIND primitive in a Process, the option field contained an invalid option. The valid options are FIRST, LAST, NEXT, and BEFORE.

In the definition of a REMOVE primitive in a Process, the option field contained an invalid option. The valid options are FIRST, LAST, and NEXT.

ERROR - ***** REFERENCE INVALID IN QUEUE FIELD

In the definition of a FILE primitive in a Process, the queue field contained an invalid value. The queue field must contain the name of a reference to a defined Queue.

In the definition of a FIND primitive in a Process, the queue field contained an invalid value. The queue field must contain the name of a reference to a defined Queue, the name of a reference to a defined Resource, or the name of a valid cross-reference set: Action, Constant, Item, Process, Queue, Resource, Table, or Variable.

In the definition of a REMOVE primitive in a Process, the queue field contained an invalid value. The queue field must contain the name of a reference to a defined Queue or Resource.

ERROR - ***** - RESUME REFERENCE MUST NOT BE NUMERIC OR GLOBAL

In the definition of the RESUME primitive, a numeric value or a Constant or Variable was encountered in the Process field. This reference must be a local variable.

- ERROR - TRACE MODE MUST BE EITHER 'ON' OR 'OFF'

In the definition of a TRACE primitive, the first field contained a value other than "ON" or "OFF". These are the only valid values.

ERROR - '*****' DISTRIBUTION ONLY REQUIRES 1 PARAMETER

In the definition of an ACTION primitive in a Process, the specified distribution required only one parameter, but two were supplied. The extra parameter should be deleted or the distribution should be changed.

ERROR - LOAD NODE IS NOT RECOGNIZED AS A RESOURCE

In the definition of a Load entity, a value was encountered in a node field which was not a reference to a defined Resource. Nodes must be Resources.

ERROR - '*****' IS NOT DEFINED AS A PROCESS

In the definition of a Load, the name specified in the process field was not defined as a Process. The name specified in this field must be a defined Process.

ERROR - ***** IS NOT A LOAD DISTRIBUTION FUNCTION

In the definition of a LOAD, the name specified in the schedule field was not a valid Load distribution.

ERROR - ***** IS NOT DEFINED AS A CONSTANT OR VARIABLE

In the definition of a Load, a non-numeric value in the rate field was not defined as a global Constant or variable. If the rate field contains a non-numeric value, it must be a defined global Constant or Variable.

In the definition of a Load, a non-numeric value in the mean field was not defined as a global Constant or Variable. If the mean field contains a non-numeric value, it must be a defined global Constant or Variable.

In the definition of a Load, a non-numeric value in the delta field was not defined as a global Constant or Variable. If the delta field contains a non-numeric value, it must be a defined global Constant or Variable.

In the definition of a Load, a non-numeric value in the priority field was not defined as a global Constant or Variable. If the priority field contains a non-numeric

value, it must be a defined global Constant or Variable.

ERROR - NO SCENARIO DEFINED

No Scenario was defined. There must be a Scenario defined in order to run a simulation on a model.

ERROR - PERIOD NOT DEFINED

In the definition of a Scenario, the period was not defined. The period length for a Scenario can be a numeric value or a defined Constant.

ERROR - TRIGGER ***** NOT DEFINED AS A LOAD OR PROCESS

In the definition of a Scenario entity, a value in the trigger field was not a Load or Process. Scenario triggers must be either Loads or Processes.

WARNING - **** NOT LEGAL. USING D INSTEAD.

An illegal Table type was specified; The Table is being assumed to be discrete. The valid table types are continuous (c), discrete (d), and alpha (a).

WARNING - ATTRIBUTE INITIAL VALUE IS NOT DEFINED

In the definition of an Item, Process, or Resource an attribute was not assigned an initial value or was assigned an invalid value. Attributes must be initialized.

WARNING - ***** IS AN ILLEGAL OPTION. USING NOWAIT INSTEAD.

In the definition of a CALL primitive of a Process, the option field contained an invalid option; a NOWAIT option is being assumed. The valid options are BLOCK, WAIT, and NOWAIT.

WARNING - '*****' IS NOT RECOGNIZED IN THIS CONTEXT

In the definition of an ASSIGN primitive of a Process, an attempt was made to assign a numeric value or a Constant or Variable, but there was also a value in the qualifier field. The qualifier is being ignored.

In the definition of an ASSIGN primitive in a Process, an attempt was made to assign a value to the \$CNODE keyword, or an attempt was made to assign a value to a Variable, but there was also a value in the qualifier field. The qualifier is being ignored.

WARNING - '*****' - NO QUALIFICATION RECOGNIZED
FOR IDENTIFICATION

In the definition of a COMPARE primitive in a Process,
a unrecognizable qualifier for a numeric, a global
Variable, or a global Constant was encountered. Qual-
ifiers are allowed only for Items, Processes,
Resources, and certain keywords.

WARNING - ***** IS NOT RECOGNIZED IN THIS CONTEXT FOR
FUNCTION

In the definition of an EVAL primitive in a Process,
operands were specified with a random function or a
second operand was specified for a function which only
required one operand.

WARNING - ***** IS NOT AN ACTION DISTRIBUTION -
USING CONSTANT

In the definition of an ACTION primitive in a Process,
the value in the method field was not a valid Action
distribution; the distribution is being assumed to be
CONSTANT. The valid distributions are exponent, con-
stant, lognorm1, normal, uniform, weibull, gamma, and
erlang.

If an execution error occurs during the simulation, execution
will halt and an error message will be printed in the statistical
summary. In some cases there may be a Simscript II.5 traceback.
This traceback is a hexadecimal formatted report which is to be
disregarded by the user. Following the error messages, the sta-
tistical summary lists the state of the Process which was execut-
ing when the error occurred. The value of all local variables
and attached attributes for the Process are listed. All other
output reports are also generated.

Following are all of the execution errors which are produced and
an explanation of the conditions which cause each error.

EXECUTION ERROR DETECTED IN PROCESS *****

An error occurred in the specified Process which caused
an abnormal termination of the simulation.

EXECUTION ERROR - BRANCH PROBABILITY FOR CURRENT STATEMENT
IS NOT A NUMBER

The BRANCH probability in a BRANCH primitive in a Pro-
cess does not evaluate to a number.

EXECUTION ERROR - LOOP NUMBER FOR CURRENT STATEMENT IS NOT
A NUMBER

The value of the LOOP counter in a Process is not a number.

EXECUTION ERROR - TEST STATEMENT ENTITY IS NOT A RESOURCE OR QUEUE

The value to be tested by a TEST primitive in a Process is not a Resource or a Queue. The TEST primitive can only test a Resource or a Queue.

EXECUTION ERROR - VALUE OF RESET IN CURRENT STATEMENT IS NOT A NUMBER

The value for the number of units to be reset by a RESET primitive is not a number. The value for the number of units to be reset must evaluate to a number.

EXECUTION ERROR - ATTEMPT TO RESET # OF RESOURCE UNITS OUTSIDE OF LEGAL LIMITS

An attempt was made to reset a number of Resource units which would make the number of units inactive or active greater than the total number of units which were defined for this Resource.

EXECUTION ERROR - A REFERENCE IN THE CURRENT PROCESS EVALUATES TO AN ILLEGAL TYPE FOR THE CURRENT STATEMENT

The resource field in a RESET primitive did not resolve to a Resource. This field must resolve to a defined Resource entity.

The resource field in an ALLOC primitive did not resolve to a Resource. This field must resolve to a defined Resource entity.

The resource field in a DEALLOC primitive did not resolve to a Resource. This field must resolve to a defined Resource entity.

EXECUTION ERROR - AN ACTION REFERENCE DOES NOT EVALUATE TO A NUMBER

The scheduling time or the scheduling delta time for an action does not evaluate to a number.

EXECUTION ERROR PRIMITIVE REFERENCE DOES NOT EVALUATE TO AN ACTION

An undefined opcode for a primitive was encountered. The opcode was assumed to be the name of a reference to an action, but it did not resolve to a defined action name.

EXECUTION ERROR - PROCESS IN CURRENT CALL STATEMENT IS NOT
DEFINED AS A PROCESS

An attempt was made by a CALL primitive to initiate a Process which was not defined. The Process name in a CALL primitive must be a reference to an entity defined as a Process.

EXECUTION ERROR - PRIORITY IN CALL DOES NOT EVALUATE TO
A NUMBER

The priority in a CALL primitive did not evaluate to a number. The priority for calling a Process must evaluate to a number.

EXECUTION ERROR - DISAGREEMENT IN NUMBER OF GIVEN
PARAMETERS BETWEEN CURRENT CALL STMT AND CALLED PROCESS

The number of given parameters in a CALL primitive differs from the number of given parameters in the definition of the Process to be called. These parameters must correspond.

EXECUTION ERROR - DISAGREEMENT IN NUMBER OF RETURN
PARAMETERS BETWEEN CURRENT CALL STMT AND CALLED PROCESS

The number of return parameters in a CALL primitive differs from the number of return parameters in the definition of the Process to be called. These parameters must correspond.

EXECUTION ERROR - ORDER RELATIONS ARE NOT DEFINED FOR COM-
PARE TYPES

For the non-numeric types being compared, an invalid relation was specified. The only valid relations for these types is equal or not equal.

EXECUTION ERROR - EVAL VARIABLE DOES NOT EVALUATE TO
A NUMBER

One of the variables in an EVAL primitive for a function other than a Table does not evaluate to a number.

EXECUTION ERROR - EVAL FUNCTION IS NOT RECOGNIZED AS
AN ARITHMETIC OPERATOR OR A TABLE REFERENCE

The reference for the function in an EVAL primitive is not a legal arithmetic function or a reference to a defined Table.

EXECUTION ERROR - EVAL VARIABLE FOR DISCREET OR CONTINUOUS
TABLE DOES NOT EVALUATE TO A NUMBER

In an EVAL primitive which is being used to look up a value in a Table, the value used to index into the Table, the x value, does not evaluate to a number.

EXECUTION ERROR - ILLEGAL ASSIGN: CURRENT NODE MUST BE SET TO A RESOURCE

An EVAL primitive attempted to set the current node to a reference which was not a defined Resource. The current node must be a Resource.

EXECUTION ERROR - ASSIGN ATTEMPTS TO MODIFY A QUALIFIED TYPE FOR WHICH NO ATTRIBUTE IS DEFINED

An attempt was made to assign a new value to an attribute of an entity for which no attributes can be defined. Only Processes, Resources, and created Items have attributes which can be modified.

EXECUTION ERROR - ***** ATTRIBUTE NOT DEFINED FOR ITEM

An attempt was made to assign a new value to a non-existent attribute of an Item.

EXECUTION ERROR - ***** ATTRIBUTE NOT DEFINED

An attempt was made to assign a new value to a non-existent attribute of a Process or a Resource.

EXECUTION ERROR - ASSIGN ATTEMPTS TO MODIFY A TYPE WHICH CANNOT BE MODIFIED

An attempt was made to assign a new value to an entity which cannot be modified; i.e. a global Constant, a number, or a keyword other than \$CNODE.

EXECUTION ERROR - ATTEMPT TO CREATE AN ENTITY WHICH IS NOT AN ITEM

An attempt was made to create an entity which is not an Item. Only references to Items may be in the create list of the CREATE primitive.

EXECUTION ERROR - ATTEMPT TO DESTROY AN ITEM WHICH IS CURRENTLY FILED ON A QUEUE"

An attempt was made to destroy an Item which had been filed on a Queue and not removed before execution of the DESTROY Primitive.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO DESTROY AN ITEM WHICH IS NOT DEFINED OR DOES NOT EXIST

An attempt was made to destroy an Item which was not defined or created, or which has already been destroyed.

EXECUTION ERROR - PROCESS FIELD IN SEND STATEMENT IS NOT DEFINED AS A PROCESS

The reference in the process field of a SEND primitive was not resolved as a Process. Items can only be sent to a defined Process.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTING TO SEND A ENTITY WHICH IS NOT DEFINED AS AN ITEM

An attempt was made by a SEND primitive to send an entity other than an Item to a Process. Only references to Items may be specified in the SEND primitive to be sent to Processes.

EXECUTION ERROR - ITEM ***** ATTEMPT TO BE RECEIVED BY PROCESS ***** IS NOT IN PROCESS NEED LIST

An attempt was made to cause a Process to receive an Item which was not on the list of Items which the Process should receive.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO FILE AN ENTITY WHICH CANNOT BE FILED

An attempt was made by a FILE primitive to file an entity which cannot be filed. Only Items and Resource units can be filed.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO FILE AN ITEM OR RESOURCE UNIT ON AN UNDEFINED QUEUE OR IDLE QUEUE

An attempt was made to FILE an Item or Resource unit on a Queue which was not defined or on a Resource IDLE QUEUE which did not exist. The queue reference in the FILE primitive must resolve to a defined Queue or a defined Resource (to indicate its IDLE QUEUE).

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO FILE AN ITEM OR A RESOURCE UNIT IN A QUEUE OF THE WRONG TYPE

An attempt was made to FILE an Item on a Queue which had been determined to be a Resource unit queue, or to file a Resource unit on a Queue which had been determined to be an Item Queue. A Queue is determined by the first entity which is placed on that Queue and from then on, only more of the same entity can be filed in that Queue. When the Queue is emptied, its type may be redetermined by the next file onto it.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO FILE AN ITEM
ON A RESOURCE IDLE QUEUE

An attempt was made to file an Item on the IDLE QUEUE
of a Resource. Only Resource units can be filed on
Resource IDLE QUEUES.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO FILE A
RESOURCE UNIT IN THE IDLE QUEUE OF A DIFFERENT RESOURCE

An attempt was made to file a Resource unit on the IDLE
QUEUE of a Resource other than the IDLE QUEUE from
which the unit was removed. Resource units cannot be
filed on IDLE QUEUES of other Resources.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO FILE AN ITEM
OR RESOURCE UNIT WHICH IS ALREADY ON A QUEUE

An attempt was made to refile an Item or Resource unit.
An Item or Resource unit can be filed on only one Queue
at any given time.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO FILE A
RESOURCE UNIT WHICH IS ALREADY ON A RESOURCE IDLE QUEUE
OR A USER DEFINED QUEUE

An attempt was made to refile a Resource unit which had
already been filed on its IDLE QUEUE or a defined
Queue. A Resource unit can be filed in only one place
at any given time.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO FILE A
RESOURCE UNIT CURRENTLY IN AN IDLE QUEUE

An attempt was made to file a Resource unit which had
already been filed on an IDLE QUEUE. A Resource unit
can be filed in only one place at any given time.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO FILE AN
ENTITY BEFORE OR AFTER AN UNDEFINED ENTITY

An attempt was made to file an entity before or after
an entity which did not exist on the Queue.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTING TO REMOVE AN
ENTITY FROM AN UNDEFINED QUEUE

An attempt was made to remove an entity from an unde-
fined Queue or an IDLE QUEUE of a Resource which had
not been defined.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTING TO REMOVE
'NEXT' ENTITY WHICH DOES NOT EXIST

An attempt was made to remove a non-existent current Item or Resource unit from a Queue or a Resource IDLE QUEUE.

EXECUTION ERROR - A REFERENCE FOR A QUEUE IN A FIND PRIMITIVE IS NOT DEFINED AS A QUEUE, RESOURCE DEF, OR XREF SET

An invalid reference was specified in a file primitive as the queue name. Only Queues, Resources, or cross-reference sets are valid for the Queue name field.

EXECUTION ERROR - ***** ATTRIBUTE OF A RESOURCE IS NOT DEFINED

An attempt was made to reference a non-existent attribute of a Resource. Valid attributes are NIDLEQ, NBUSYQ, NWAITQ, NINACTQ, as well as user-modifiable attributes.

EXECUTION ERROR - ***** ATTRIBUTE OF A RESOURCE UNIT NOT DEFINED

An attempt was made to reference a non-existent attribute of a Resource unit. Valid attributes are NIDLEQ, NBUSYQ, NWAITQ, NINACTQ, as well as user-modifiable attributes.

EXECUTION ERROR - ***** ATTRIBUTE OF A PROCESS IS NOT DEFINED

An attempt was made to reference a non-existent attribute of a Process.

EXECUTION ERROR - ***** ATTRIBUTE OF A TASK IS NOT DEFINED

An attempt was made to reference a non-existent attribute of a Task.

EXECUTION ERROR - ***** ATTRIBUTE OF A QUEUE NOT DEFINED

An attempt was made to reference an invalid attribute of a Queue. The valid attributes are NQUEUE and TQUEUE.

EXECUTION ERROR - ***** ATTRIBUTE IS NOT DEFINED FOR CURRENT ITEM REFERENCE ***** IN EXECUTING LOGIC

An attempt was made to reference a non-existent attribute of an Item.

EXECUTION ERROR - ***** ATTRIBUTE SPECIFIED FOR A TYPE FOR WHICH NO ATTRIBUTES CAN BE DEFINED

1

An attempt was made to reference an attribute of a type which does not have attributes. Entities which have attributes are Resources, Processes, and Items.

EXECUTION ERROR - KEYWORD REFERENCE IS BLANK

When the simulator tried to resolve a Keyword, the reference field for the parameter was found to be blank.

EXECUTION ERROR - PROCESS NODE HAS NOT BEEN DEFINED

An attempt was made to reference the process node of a Process, but the node was not defined.

EXECUTION ERROR - REFERENCE FOR \$ NODE IS NOT A PROCESS

When the simulator tried to resolve the Keyword \$NODE, the reference was not a Process.

EXECUTION ERROR - ROUTE SET ERROR - NO PATH IN NETWORK

APPENDIX C

C. GLOSSARY

ACTION - A discrete event that consumes time during a simulation run.

AMDAHL/470 -A large scale IBM370-like computer system, on which the AISIM System is hosted.

ANALYSIS USER INTERFACE (AUI) - The interface between the user and the General Function Model.

ANALYSIS USER INTERFACE (AUI) READY STATE - Any time after the Analysis User Interface has been invoked, except during a simulation period. This state is indicated by the "*" prompt.

ARCHITECTURE DESIGN EDITOR (ADE) - A sublevel of the DUI which provides the user with the graphics commands to construct a system architecture.

ARCHITECTURE DESIGN EDITOR (ADE) MENU - A representation of the valid symbols available to the user during an ADE session for building an architecture. See ADE MENU.

ARCHITECTURE DESIGN EDITOR (ADE) READY STATE - The state of the system while in the ADE that allows the user to enter commands. This state is indicated by the "#" prompt.

ATTRIBUTE - The specific characteristics of a defined entity.

ATTRIBUTE FORM - A list of available attributes from which the user must select one attribute to be used for testing or data sampling.

BLOCK - Used in conjunction with the CALL symbol (see section 4.5.1) to indicate that the calling task is to call the specified task and wait until all associated tasks are complete before continuing.

BREAKPOINT - A user-specified condition, when reached, simulation is suspended.

CONSTANT - A value that is not subject to change once a simulation run has been started.

CRT - CATHODE RAY TUBE - Often used as a shorthand notation for a crt-based computer terminal.

DATABASE - The accumulation of data in a specified form related to a specific function or operation.

DATAPHONE - A registered trademark of the BELL SYSTEM for a type of data transmission device (modem).

DEFAULT CONDITION - The condition that exists if no parameters are explicitly stated.

DESIGN USER INTERFACE (DUI) - The interface that allows the user to create or modify a design database.

DESIGN USER INTERFACE (DUI) READY STATE - Any time after invocation of the DESIGN USER INTERFACE, except when utilizing the PEI or ADE sublevels of the DUI. This state is indicated by the "**" prompt.

DPN - Data Processing Network

ENTER KEY - A specific key on the HP-2647A terminal that is used to enter data when the terminal is in "forms mode".

ENTITY - A predefined set of constructs that have user defined attributes (see section 5 for valid AISIM entities). They are the "building blocks" with which the user creates his model.

ENTITY-NAME - The user-defined name of a valid entity.

ENTITY-TYPE - A type as opposed to a specific, user-defined instance of an Entity.

FORMS MODE - A specific function of the HP-2647A terminal, that provides areas which may be filled in by the user, and protected fields which define the areas to be filled in. The data is input to the computer via the Enter Key.

INFINITE RESOURCES - A feature which allows the simulator to simulate a Process as if there were no limit to the number of resources available to it.

LOAD - The amount of activity to be applied to the simulation of a process.

L-NODE - A leaf node in an architecture which typically represents an external load on the system.

NOWAIT - Used in conjunction with the CALL Primitive to indicate that a Process is to be called by a parent Process and the parent Process is to continue processing in parallel.

OFF-SCREEN - The portion of a graphics picture not visible to the user.

ON-SCREEN - The portion of a graphics picture visible to the user.

PRIMITIVE - The model entity used to model individual steps in an operation or function. A Process is constructed from a sequence of Primitives.

PROCESS - A graphical representation of a sequence of events, activities and decisions that models a real-world operation or function.

PROCESS EDITOR INTERFACE (PEI) - A sublevel of the DUI that provides the user with the graphics commands to construct Processes.

PROCESS EDITOR INTERFACE (PEI) MENU - A representation of the valid Primitives available to the user during a PEI session.

PROCESS EDITOR INTERFACE (PEI) READY STATE - The state of the system while in the PEI that allows the user to enter commands. This state is indicated by the "#" prompt.

QUERY - A request for information.

PERMANENT DATABASE (SOMETIMES REFERRED TO AS THE DESIGN DATABASE) - The master user-named database, in which the data for a modeled system resides. (opposed to the working database which temporarily holds design data while editing that data).

RELATIONAL OPERATOR - A set of mnemonics that represent a relation such as equal, not equal, less than, greater than.

RESOURCE - Representations of the real-world objects that are required by a Process to do its work.

REVERSE VIDEO - A feature of a CRT terminal in which dark characters appear on a light background, rather than a dark characters on a light background.

SCROLL - The process of moving data displayed on a CRT either up or down.

STATISTICS FORM - A form presented to the user from which one type of statistical value must be selected for use in data sampling.

SYNTAX - A set of rules that determines the structure and arrangement of words and characters.

TRANSLATOR (XLATOR) - The AISIM database translator that translates the design database into a form suitable for input to the simulator.

VARIABLE - A term whose value is subject to change.

WAIT - Used with the CALL Primitive to indicate that the calling Process is to suspend until the called Process is complete.

/

WORKING DATABASE - A copy of the user's real database, into which all work is done on a temporary basis.

APPENDIX D

D. QUEUES ASSOCIATED WITH ENTITY NAMES

In addition to the queues associated with Resource contention, there are eight system defined queues called "cross-reference sets". These queues correspond to the sets of names of the following AISISM entities:

1. Resource names
2. Queue names
3. Process names
4. Item names
5. Action names
6. Table names
7. Constant names
8. Variable names

What this means is that an AISIM modeller can write logical Processes which cycle for each entity defined in one of the above sets.

The FIND primitive accesses the set of names of an entity type by specifying the name ,e.g. RESOURCE, ITEM, PROCESS, as the QUEUE field reference in the primitive.

APPENDIX E

E. MESSAGE ROUTING SUBMODEL PROCESSES

E.1 PROCESS: REQ-I/O

This Process is the top level Process of the Message Routing Submodel. This Process is called when a user wishes to make use of this submodel. This Process causes a Process request message to be generated. When this Process is called, it is given PROCESS, which is the name of a Process to be initiated in the destination node of the message, PRIORITY, which is the priority with which PROCESS is to be initiated, RESP.OPT, which is \$WAIT or \$NOWAIT and indicates whether the parent will wait until the initiated Process completes, MSG.LNTH, which is the length in bytes of the message, TO.NODE, which is the destination node for the message and the node in which PROCESS is to be initiated. This Process does not return any parameters.

A detailed description of the parameters of this Process is given below.

PROCESS NAME: REQ-I/O - Generate a process request message on initiate I/O.

LOCATION: executes in all nodes.

GIVEN: PROCESS (DATA TYPE: PROCESS) - This Parameter is the name of the Process to be initiated in the destination node.

PRIORITY (DATA TYPE: REAL) - This parameter is the priority which the initiated Process is to have when it is started.

RESP.OPT (DATA TYPE: ALPHA) - This parameter is the option for the communication. The only legal values in this parameter are: \$WAIT - the parent Process will wait until the requested Process finishes before it resumes, \$NOWAIT - the parent Process does not wait on the requested Process.

MSG.LNTH (DATA TYPE: REAL) - This parameter is the length in bytes of the communication message routed through the network, requesting the Process to be invoked.

TO.NODE (DATA TYPE: RESOURCE or ALPHA) - This parameter is the destination node of the message which is the node in which to initiate the requested Process. If the ALPHA variable \$YES is provided for this parameter, the node of the requested Process is computed by the \$NODE keyword.

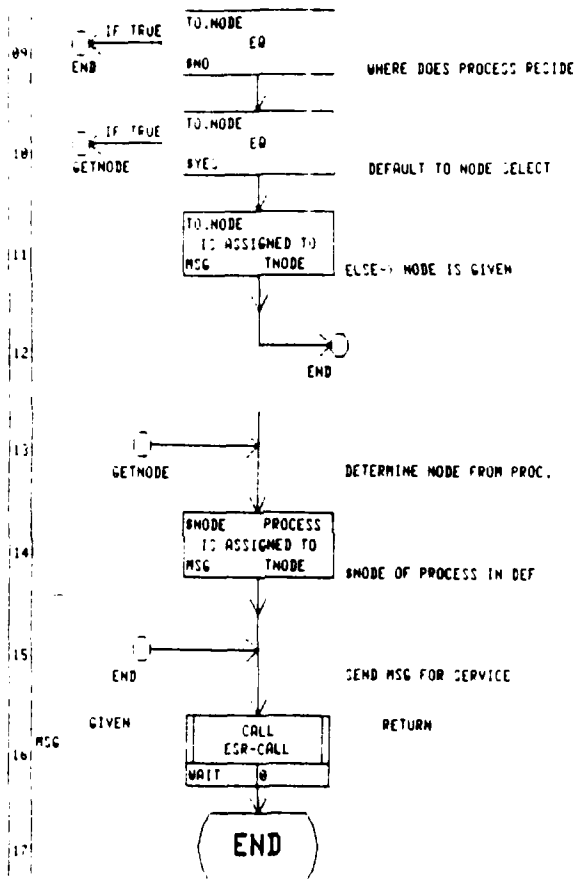
CALLS: ESR-CALL

```

    graph TD
      Start([START  
REQ-1.0]) --> Create[CREATE]
      Create --> BNode1[B.NODE  
IS ASSIGNED TO  
MSG C.NODE]
      BNode1 --> BNode2[B.NODE  
IS ASSIGNED TO  
MSG F.NODE]
      BNode2 --> PNode[P.NODE  
IS ASSIGNED TO  
MSG P.NODE]
      PNode --> Prio[PRIORITY  
IS ASSIGNED TO  
MSG TASKPRI]
      Prio --> Resp[RESP.OPT  
IS ASSIGNED TO  
MSG RESPONSE]
      Resp --> Len[MSG.LENGTH  
IS ASSIGNED TO  
MSG LENGTH]
  
```

01. GIVEN
 02. PROCESS RESP.OPT TO.NODE
 03. PRIORITY MSG.LENGTH
 04. MSG
 05. CREATE
 06. B.NODE IS ASSIGNED TO MSG C.NODE
 07. B.NODE IS ASSIGNED TO MSG F.NODE
 08. P.NODE IS ASSIGNED TO MSG P.NODE
 09. PRIORITY IS ASSIGNED TO MSG TASKPRI
 10. RESP.OPT IS ASSIGNED TO MSG RESPONSE
 11. MSG.LENGTH IS ASSIGNED TO MSG LENGTH

12. RETURN ALL NO
 13. CREATE MESSAGE DATA TO PTE
 14. INDICATE CURRENT NODE
 15. INDICATE CURRENT NODE FROM
 16. INDICATE REQUESTED PROCESS
 17. INDICATE RELATIVE PRIORITY
 18. SHOWWAIT OR * WAIT ON CALL
 19. INDICATE LENGTH IN BYTES



This Process begins by creating the message and initializing various attributes of it. The attributes CNODE and FNODE are the current node in which this Process is executing. The attribute

RTASK is set to the Process which will be executed in the destination node of the message. The attribute TASKPRI is set to the priority with which the requested Process will execute. The attribute RESPONSE is set to \$WAIT or \$NOWAIT; i.e. whether the parent is to wait for the requested Process to finish Processing. The attribute length is set to the length in bytes of the message. Next the value of the destination node is checked. The value of \$NO will not occur as the destination. If the value of the destination node is \$YES, then the TNODE attribute--i.e. the destination of the message--is set to be the node in which the requested Process executes. Otherwise, the name of the destination node is supplied, and attribute TNODE is set to this value. This Process then calls Process ESR-CALL with a WAIT option and passes it the created message.

E.2 PROCESS: ESR-CALL

This Process is called by REQ-I/O and either suspends the requesting Process if a response message is requested (WAIT option) or allows it to continue processing if no response is requested (NOWAIT option). When this Process is called, it is passed the Item MSG. This Process does not return any parameters.

PROCESS NAME: ESR-CALL - Executive Service Request (CALL)

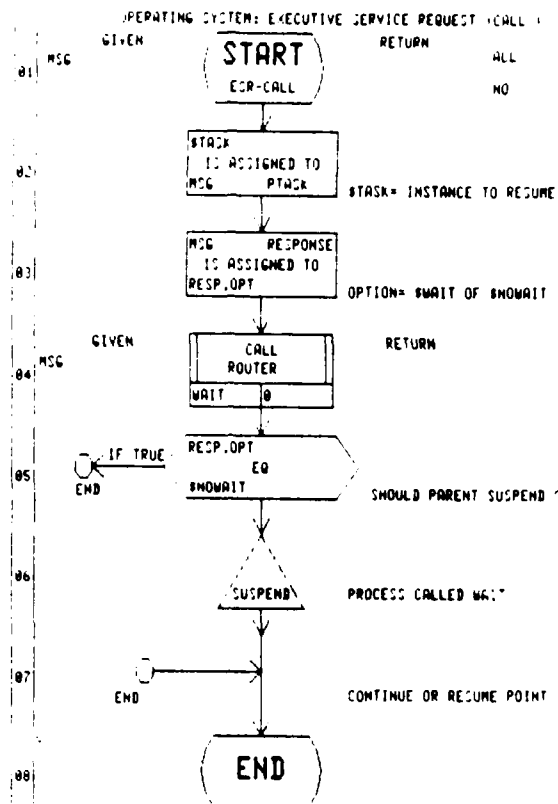
LOCATION: executes in all nodes

GIVEN: MSG (DATA TYPE: ITEM) - This parameter is the communication message created in REQ-I/O which contains the data for the Message Routing Submodel.

RETURN: N/A

CALLS: ROUTER

Following is the graphical representation of Process ESR-CALL.



This Process begins by setting the message attribute PTASK to the currently executing instance of this Process. This value is maintained in case this Process is suspended and is to be resumed by another Process. Then the response option of the requesting Process for the requested option is retrieved. Then the Process calls the Process ROUTER to begin routing the message to its destination and waits for ROUTER to complete. Finally a test is made to see if the previously retrieved request option is \$WAIT or \$NOWAIT. If it is \$NOWAIT, ESR-CALL completes. If it is \$WAIT, ESR-CALL is suspended, having the effect that the requesting Process waits until the message reaches its destination, the requested Process executes, and a response is routed back before the requesting Process finishes Processing.

E.3 PROCESS: ROUTER

This Process determines whether the message is at its destination node. When this Process is called, it is passed the message. This Process does not return any parameters.

PROCESS NAME: ROUTER - Operating System: Interrupt Handling and Routing

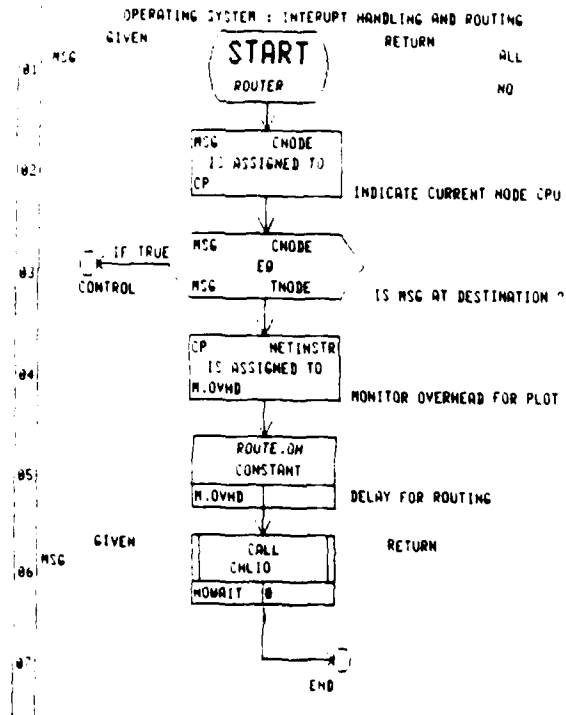
LOCATION: executes in all nodes

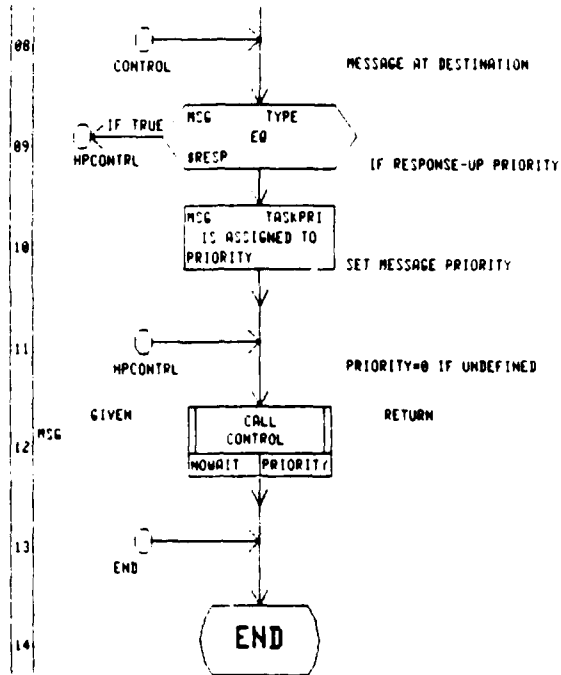
GIVEN: MSG (DATA TYPE: ITEM) - This parameter is the communication message created in REQ-I/O which contains the data for the logical process communication protocol.

RETURN: NONE

CALLS: CHLIO, CONTROL

Following is the graphical representation of Process ROUTER.





The first step of this Process is to assign to a variable the name of the node which is the message's current position. The message's current position is then compared with its destination

node. If at this point the node is at its destination, the destination node is the same node which generated the message. There is no routing overhead delay because the message did not need to be routed anywhere (i.e. M.CS is zero). The Process then tests to see if the message is a request message or a response message. If it is a request message, the routine CONTROL is called with a NOWAIT option and a priority equal to the requested priority, and the requested Process is initiated in the destination node. If the message is a response message, the routine CONTROL is called with a NOWAIT option and a priority of zero, and the requesting Process is resumed, implying that the message has reached its destination, the requested Process has been initiated, and the message has been routed back to the node from which it was generated.

If the node is not at its destination, the overhead cost for transfer of the message from its current node to its next node must be calculated. Since this model is assuming that the overhead is a constant value because the messages are all the same length, the mean context switching time (M.CS) is used; the message length (MSG.LNTH) and the route rate per length (RT.OVHD) are not used. The Action ROUTE.OH is used to simulate this delay time. The routine CHLIO is then called NOWAIT to forward the message to its next node, and the Process terminates.

E.4 PROCESS: CONTROL

This Process performs two different functions depending on when it is called. This Process is passed the message. If the message is a response message, then at this point the requesting Process has waited for the message to reach its destination, the requested Process to be initiated, and the message to be routed back. At this point the message has gone full circle and returned to the requesting Process's node. CONTROL then resumes the requesting Process. If the message is a request message, then the message is at its destination and the requested Process is initiated in that node. If the requesting Process is waiting for a response, then the attributes of the message are changed so that the original source node is now the destination node, and the message type is changed to a response type. Then the routine CHLIO is called to route the message back to the requesting Process's node. If the requesting Process is not waiting for a response, the message is destroyed.

PROCESS NAME: CONTROL - Operating System: Context Switching

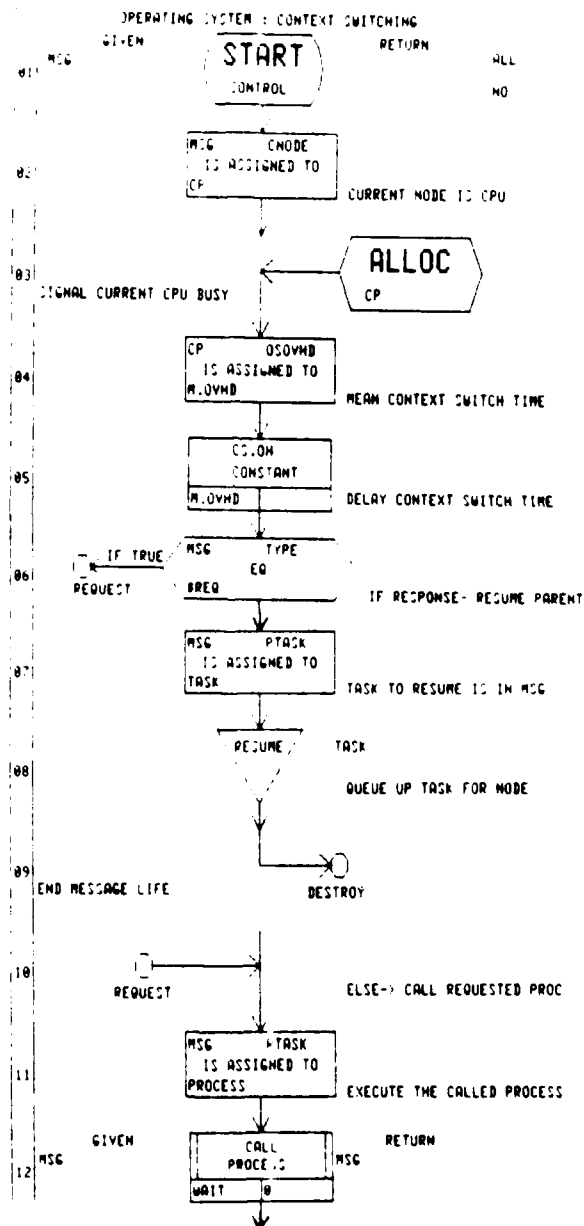
LOCATION: executes in all nodes

GIVEN: MSG (DATA TYPE: ITEM) - This parameter is the communication message created in REQ-I/O which contains the data for the logical process communication.

RETURN: NONE

CALLS: CHLIO

Following is the graphical representation of the Process CONTROL.



100: IF TRUE MSG RESPONSE
101: DESTROY EQ
102: SNOWBIRT IF WAIT -- SEND MSG BACK

151

MSG	FNODE
IS ASSIGNED TO	
MSG	TNODE

SWITCH FROM AND TO NODES

177	MCC	GIVEN	CALL		RETURN
			CHLIO		
			WAIT	0	

131 DESTROY TERMINATE MESSAGE AT DEST.

241
END

231

END

The first step is to allocate the node where the message is currently located. This is the node where the requested Process will be initiated or the node in which the requesting Process executes (which is currently suspended). The action CS.OH simulates the delay time involved in the context switching. The value for this time is an attribute of the node. Next the Process determines if the message is a request or response message. If it is a response message, this Process resumes the suspended instance of the requesting Process, destroys the message, deallocates the current node, and terminates. If the message is a request message, the name of the requested Process is retrieved from the PTASK attribute of the message and the Process is initiated. CONTROL waits until the requested Process completes. Next CONTROL checks the message attribute RESPONSE to see if the requesting Process is waiting for a response. If a response is not desired, the message is destroyed and CONTROL terminates. If a response is requested, the message type is changed to response, the destination node is changed to the from node and the from node is changed to the current node. Then the Process ROUTER is called to route the message back to its origin. ROUTER is called with a WAIT option. CONTROL then terminates.

E.5 PROCESS: CHLIO

Process CHLIO determines the current node and the destination node for the message which is passed to it. It then accesses the Legal Path Table to determine the next node along the route and

the channel to get there. The channel is allocated to simulate its use, and the routine IHANDLER is called to interrupt the next node.

PROCESS NAME: CHLIO ~ full and half duplex channel logic

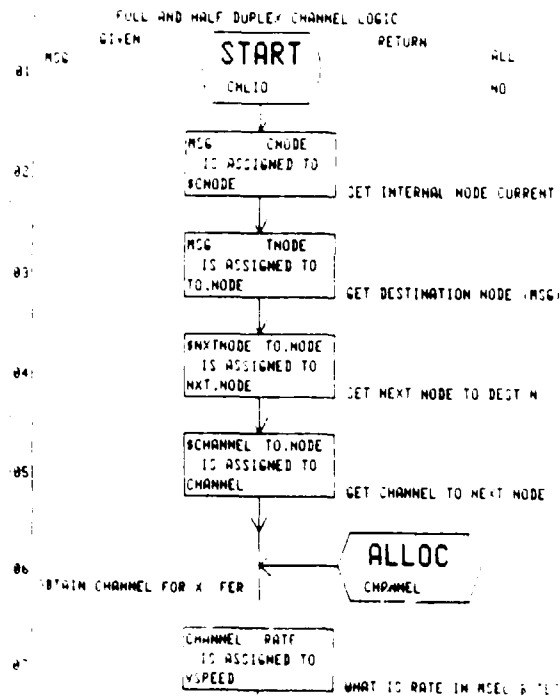
LOCATION: executes in all nodes

GIVEN: MSG (DATA TYPE: ITEM) - This parameter is the communication message created in REQ-I/O which contains the data for the logical process communication.

RETURN: NONE

CALLS: IHANDLER

Following is the graphical representation of the Process CHLIO.

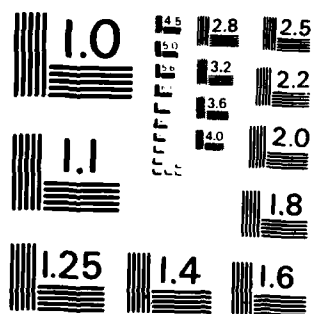


DD-A135 /61 AISIM (AUTOMATED INTERACTIVE SIMULATION MODEL) USER'S
MANUAL (U) HUGHES AIRCRAFT CO FULLERTON CA GROUND
SYSTEMS GROUP W AUSTELL ET AL. 26 FEB 82 ESD-TR-83-218
UNCLASSIFIED F19628-79-C-0153 F/G 9/2

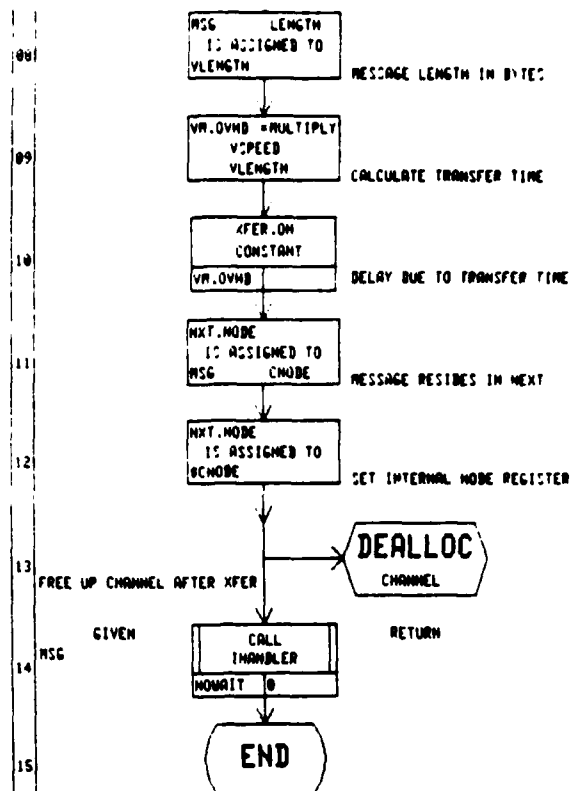
4/4

NL

END
DATA
FILMED
1-84
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A



The first step of this Process is to assign the current node of the message to SCNODE and to get the destination node for the message. Then the next node and link are determined based on SCNODE and the destination node. Then the link is allocated. The transfer time for the message to cross the channel is always a constant rate (the EVAL calculation using the channel rate and the message length is ignored). The Action XFER.OH simulates the time used to traverse the channel. Then the current node attribute of the message is changed to the next node to update the message's position, and the current node (SCNODE) is set to the next node. The link is then deallocated and the routine IHANDLER is called to interrupt the next node Processor.

E.6 PROCESS: IHANDLER

This Process is similar to the Process ROUTER. IHANDLER is passed the message. The Process then interrupts a processor node by allocating the node. If the message is not at its destination node, then IHANDLER computes the next node in the route to the destination node and calls the routine CHLIO to perform the routing. If the message is at its destination node, then IHANDLER calls CONTROL to initiate the requested Process in the destination node.

PROCESS NAME: IHANDLER - Operating System: Interrupt Handling and Routing

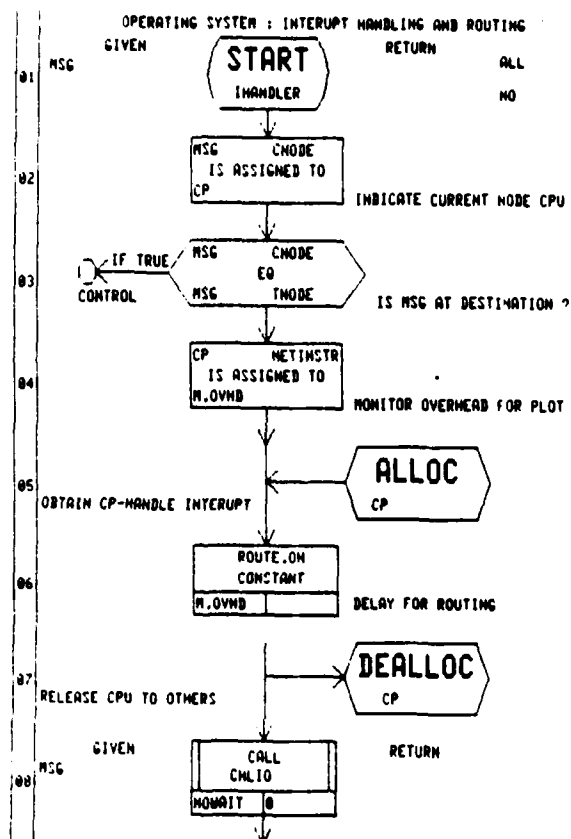
LOCATION: executes in all nodes

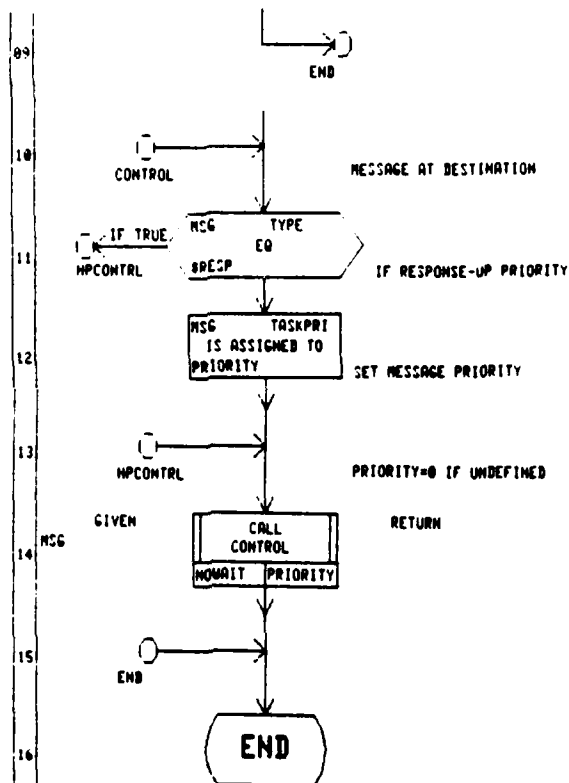
GIVEN: MSG (DATA TYPE: ITEM) - This parameter is the communication message created in REQ-I/O which contains the data for the logical process communication protocol.

RETURN: NONE

CALLS: CHLIO, CONTROL

Following is the graphical representation of Process IHANDLER.





This Process first determines whether the message is at its destination node. If it is, IHANDLER calls routine CONTROL to

initiate the requested Process. If a response message is to be sent, then CONTROL is called with a zero priority; otherwise, it is called with the priority of the requested Process. Either way, CONTROL is called with a NOWAIT option. If the message is not at its destination, the current node is allocated. The Action ROUTE.OH is used to simulate the delay time for Processing the routing. Then the node is deallocated and routine CHLIO is called NOWAIT to perform the routing. IHANDLER then completes.