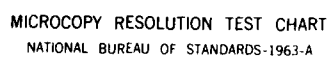


UNCLASSIFIED

FEB 85 ESD-TR-85-129 F33615-81-C-5098

NL

[illegible]



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

(12)



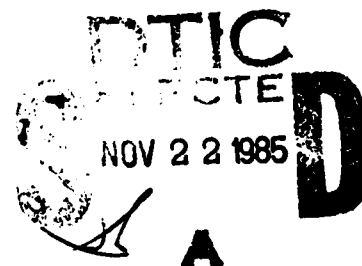
AISIM VAX VERSION TRAINING MANUAL

S. KNEEBURG
V. ALLERTON
T. WOODCOCK

Hughes Aircraft Co.
Ground Systems Group
P.O. Box 3310
Fullerton, CA 92634

February 1985

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED



Prepared for

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

AD-A161 436

DTIC FILE COPY

85 11 18 85

LEGAL NOTICE

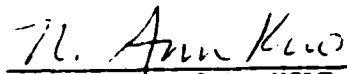
When U.S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in 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.


N. ANN KUO, 2Lt, USAF
Project Manager,
Requirements Analysis


WILLIAM J. LETENDRE
Program Manager,
Computer Resource Management
Technology

FOR THE COMMANDER


ROBERT J. KENT
Director, Computer Systems Engineering
Deputy for Acquisition Logistics
and Technical Operations

DISCLAIMER NOTICE

**THIS DOCUMENT IS BEST QUALITY
PRACTICABLE. THE COPY FURNISHED
TO DTIC CONTAINED A SIGNIFICANT
NUMBER OF PAGES WHICH DO NOT
REPRODUCE LEGIBLY.**

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS NONE													
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.													
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE																
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ESD-TR-85-129			5. MONITORING ORGANIZATION REPORT NUMBER(S)													
6a. NAME OF PERFORMING ORGANIZATION HUGHES AIRCRAFT COMPANY GROUND SYSTEMS GROUP		6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION COMPUTER RESOURCE MANAGEMENT TECHNOLOGY PROGRAM, DEPUTY FOR ACQUISITION (OVER)													
6c. ADDRESS (City, State and ZIP Code) P. O. BOX 3310 FULLERTON, CA 92634		7b. ADDRESS (City, State and ZIP Code) ELECTRONIC SYSTEMS DIVISION (AFSC) HANSCOM AFB, MA 01731														
8a. NAME OF FUNDING/SPONSORING ORGANIZATION COMPUTER RESOURCE (OVER)		8b. OFFICE SYMBOL (If applicable) ESD/ALSE	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-81-C-5098													
8c. ADDRESS (City, State and ZIP Code) ELECTRONIC SYSTEMS DIVISION (AFSC) HANSCOM AFB, MA 01731		10. SOURCE OF FUNDING NOS. <table border="1"><thead><tr><th>PROGRAM ELEMENT NO.</th><th>PROJECT NO.</th><th>TASK NO.</th><th>WORK UNIT NO.</th></tr></thead><tbody><tr><td>64740F</td><td>2522</td><td></td><td></td></tr></tbody></table>			PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.	64740F	2522						
PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.													
64740F	2522															
11. TITLE (Include Security Classification) AISIM VAX VERSION TRAINING MANUAL (U)																
12. PERSONAL AUTHOR(S) S. KNEEBURG, V. ALLERTON, T. WOODCOCK																
13a. TYPE OF REPORT FINAL		13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 1985 FEBRUARY	15. PAGE COUNT 154												
16. SUPPLEMENTARY NOTATION																
17. COSATI CODES <table border="1"><thead><tr><th>FIELD</th><th>GROUP</th><th>SUB. GR.</th></tr></thead><tbody><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></tbody></table>			FIELD	GROUP	SUB. GR.										18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) AISIM DESIGN PROCESSES SIMULATION TRAINING MODELING ARCHITECTURE	
FIELD	GROUP	SUB. GR.														
19. ABSTRACT (Continue on reverse if necessary and identify by block number) THIS DOCUMENT IS THE TRAINING MANUAL FOR THE AUTOMATED INTERACTIVE SIMULATION MODELING SYSTEM (AISIM). THIS MANUAL PROVIDES STEP-BY-STEP INFORMATION NECESSARY TO BEGIN USING AISIM ON A VAX 11/780.																
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☒ DTIC USERS ☐			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED													
22a. NAME OF RESPONSIBLE INDIVIDUAL		22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL													

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

7a. NAME OF MONITORING ORGANIZATION (CONTINUED)
LOGISTICS AND TECHNICAL OPERATIONS

8a. NAME OF FUNDING/SPONSORING ORGANIZATION (CONTINUED)
MANAGEMENT TECHNOLOGY PROGRAM, DEPUTY FOR ACQUISITION LOGISTICS AND
TECHNICAL OPERATIONS

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

ACKNOWLEDGEMENTS

This manual was prepared by Hughes Aircraft Company under contract # F33615-81-C-5098. The work was sponsored by the Computer Engineering Applications Division (ALSE), Deputy for Acquisition Logistics and Technical Operations of the Electronic Systems Division (ESD) of the United States Air Force, Hanscom AFB, MA 01731. Funding for the effort was provided by the Air Force Computer Resource Management Technology Program, Program Element 64740F.

Program Element 64740F is the Air Force engineering development program established to develop and transfer into active use the technology, tools, and techniques needed to cope with the explosive growth in Air Force systems that use computer resources. The goals of the Program are to:

- (a) Provide for the transition of computer system technology developments in laboratories, industry, and academia to Air Force systems;
- (b) Develop and apply software acquisition management techniques to reduce life cycle costs;
- (c) Provide improved software design tools;
- (d) Address the various problems associated with computer security;
- (e) Develop advanced software engineering tools, techniques, and systems;
- (f) Support the implementation of high order languages, e.g. Ada;
- (g) Address human engineering for computer systems; and
- (h) Develop and apply computer simulation techniques for the acquisition process.

Accession For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Organization	
Availability Codes	
A-1 23	

CONTENTS

1. INTRODUCTION.....	1
1.1 MODELING.....	1
1.2 OVERVIEW OF THE AISIM SYSTEM.....	1
1.3 OVERVIEW OF AISIM MODELING CONSTRUCTS.....	2
1.3.1 ENTITIES REPRESENTING ELEMENTS EXTERNAL TO THE MODELED SYSTEM.....	2
1.3.1.1 The Load Entity.....	2
1.3.1.2 The Scenario Entity.....	3
1.3.2 ENTITIES REPRESENTING ELEMENTS INTERNAL TO THE MODELED SYSTEM.....	3
1.3.2.1 The Process Entity.....	3
1.3.2.1.1 The Process Primitives.....	3
1.3.2.2 The Resource Entity.....	5
1.3.2.3 The Action Entity.....	5
1.3.2.4 The Legal Path Table.....	5
1.3.2.5 The Queue Entity.....	5
1.3.3 ENTITIES WHICH SUPPORT MATHEMATICAL OPERATIONS.....	5
1.3.3.1 The Constant Entity.....	5
1.3.3.2 The Variable Entity.....	5
1.3.3.3 The Table Entity.....	5
2. CREATING SYSTEM ARCHITECTURES.....	6
2.1 DRAW AND NODRAW MODES.....	8
2.2 DEFINING ATTRIBUTES FOR SYMBOLS.....	9
2.3 PLACING NODES ON THE GRID.....	10
2.4 CONNECTING NODES.....	12
2.5 CHANGING THE SIZE, TYPE, AND NAME OF NODES AND LINKS.....	15
2.6 DELETING NODES AND LINKS.....	15
2.7 MOVING PREVIOUSLY PLACED NODES.....	16
2.8 RECONNECTING EXISTING LINKS.....	17
2.9 ALTERING ONE'S VIEW OF THE ARCHITECTURE GRID.....	18
2.10 DEFINING LEGAL PATHS.....	20
3. DEFINING PROCESSES IN THE DUI.....	23
3.1 PROCESS EDITOR DRAW/NODRAW MODES.....	25
3.2 ACTION.....	25
3.3 HOLD.....	26
3.4 ENTRY AND LOOP.....	28
3.5 PROB, TEST, COMPARE AND BRANCH.....	29
3.5.1 PROB.....	30
3.5.2 TEST.....	31
3.5.3 COMPARE.....	32
3.6 VARIABLE MANIPULATION.....	34
3.7 ITEM MANIPULATION.....	38
3.8 RELATIONS AMONG PROCESSES.....	41
3.9 RESOURCE ALLOCATION.....	43
3.10 CALL.....	47

4.	REMAINING MODEL ELEMENTS.....	49
4.1	ACTIONS.....	49
4.2	RESOURCES.....	49
4.3	QUEUES.....	51
4.4	CONSTANTS AND VARIABLES.....	51
4.5	LOADS AND SCENARIOS.....	52
5.	A WORKING EXAMPLE.....	55
5.1	DEFINING PROCESSES.....	55
5.2	REMAINING MODEL ELEMENTS.....	59
5.2.1	RESOURCE DEFINITIONS.....	60
5.2.2	QUEUE DEFINITIONS.....	60
5.2.3	ITEM DEFINITION.....	60
5.2.4	VARIABLE DEFINITION.....	60
5.2.5	ACTION DEFINITION.....	56
5.3	LOADS AND SCENARIOS.....	60
6.	SIMULATION EXERCISES OF AISIM MODELS.....	62
6.1	INITIALIZING A MODEL.....	62
6.2	DEFINING PLOTS.....	63
6.3	STARTING THE SIMULATION.....	64
6.4	EDITING VARIABLES BETWEEN SIMULATION STAGES.....	64
7.	A MORE ELABORATE EXAMPLE.....	66
7.1	MESSAGE ROUTING SUBMODEL.....	66
7.2	DEFINING ARCHITECTURAL ELEMENTS.....	66
7.3	DEFINING REMAINING MODEL ELEMENTS.....	80
7.3.1	RESOURCE DEFINITIONS.....	80
7.3.2	FILLING IN THE ACTION DEFINITIONS.....	81
7.3.4	DEFINING LOADS AND SCENARIOS.....	81
7.4	ANALYZING THE MODEL.....	82
	APPENDIX A - TERMINAL PROFILES FOR FORMS.....	83
	APPENDIX B - SIMULATION REPORT FOR WORKING EXAMPLE.....	86
	APPENDIX C - SIMULATION REPORT FOR ELABORATE EXAMPLE.....	99

FIGURES

FIGURE	PAGE
1 Typical Display Upon Entering the AISIM READY Level.....	6
2 Typical Information on Entering the DUI.....	7
3 Grid on Which an Architecture is Designed.....	7
4 Designations of the Fourteen Symbols.....	8
5 Symbol Definition Form.....	9
6 Architectural Grid With a Single Node.....	11
7 Six Nodes on an Architectural Grid.....	12
8 Architecture with One Link Defined.....	13
9 Architecture with a Bent Link.....	13
10 Architecture with Four Links.....	14
11 The Result of Deleting LINK1 and NODE6.....	16
12 The Result of Moving NODE4.....	17
13 Architecture After Reconnecting.....	18
14 Result of WINDOW Command.....	19
15 The Result of Further Use of WINDOW.....	20
16 Augmented Architecture.....	21
17 Typical List of Legal Paths Obtained in ADE.....	22
18 Initial Form for Process.....	24
19 Graphic Display That Follows Entering a Standard Process.....	24
20 Form for the ACTION Primitive.....	25
21 Process with a Single ACTION.....	26
22 Process with Three Identical ACTION Primitives.....	27
23 Process with Triple Repetition of the ACTION Delay.....	28
24 Form for the LOOP Primitive.....	29
25 Form for the ENTRY Primitive.....	29
26 EXAMPLE after Deletion of LOOP Primitive.....	30
27 Form for PROB Primitive.....	30
28 Process with Probabilistic Branch.....	31
29 Form for TEST Primitive.....	31
30 Process with TEST Branching.....	32
31 Form for COMPARE Primitive.....	33
32 Process with COMPARE Primitive.....	34
33 Form for ASSIGN Primitive.....	35
34 Process with Primitives ASSIGN, ACTION, and COMPARE.....	35
35 Form for EVAL Primitive.....	36
36 Process with ASSIGN, ACTION, COMPARE, and EVAL.....	37
37 Process with Comparative Branching.....	38
38 Form for CREATE Primitive.....	39
39 Item Creating Process.....	39
40 Form for FILE Primitive.....	40
41 Process which Creates and Files Message Items.....	40
42 Secondary Form for Process.....	41
43 Item Passing Process.....	42
44 Process with ACTION Primitive.....	42
45 Form for SEND Primitive.....	43
46 Graphic Representation of EXAMP-2.....	43
47 Forms for Primitives ALLOC and DEALLOC.....	44
48 Process which Allocates and Deallocates a Resource.....	45

49	Forms for Primitives LOCK and UNLOCK.....	46
50	Process with Protected Resources.....	46
51	Secondary Form for Parameter-Passing Process.....	47
52	Form for CALL Primitive.....	48
53	Form for Action Entity.....	49
54	Form for Resource Entity.....	50
55	Form for Queue Entity.....	51
56	Forms for Constant and Variable.....	52
57	Form for Load Entity.....	53
58	Form for Scenario Entity.....	54
59	Transmitting Process.....	57
60	Receiving Process.....	59
61	Information Displayed on Entering the AUI.....	62
62	Aspects of Queue Behavior.....	63
63	Options for Statistics.....	63
64	Plot from Simple Example.....	65
65	System Architecture.....	67
66	Defined Resource Entities.....	80
67	Defined Action Entities.....	81
68	Defined Constant Entities.....	81
69	Terminal Profiles for Forms.....	84

1. INTRODUCTION

This training manual presupposes virtually no programming experience and is intended to provide step by step information necessary to begin using AISIM as hosted on a VAX 11/780. It is not intended as a complete account of the system, and many topics covered in the companion AISIM User's Manual are covered here in less detail, or not at all. For further details on the operation of AISIM or on the kind of simulation AISIM is adapted to, the reader is referred to the more detailed AISIM User's Manual.

This manual consists of seven sections. This first section provides a brief overview of AISIM and its main concepts. Sections 2, 3 and 4 concern the Design User Interface (DUI), i.e., that part of AISIM in which models are created. Section 5 describes the complete construction of a simple model. Section 6 turns to the use of the Analysis User Interface--the part of AISIM where simulation and analysis occur--using the model developed in Section 5 as an example. Finally, Chapter 7 will document the creation, simulation and analysis of a more complex system.

1.1 MODELING

A computer model is a description of a system that is developed as a basis for calculations, predictions or further investigation. AISIM is especially designed to model systems that incorporate parallel processing. The purpose of an AISIM model is to give information on the workability of a system design, especially by providing statistics that serve to predict the operation of the modeled system if implemented.

Modeling is accomplished in AISIM by representing the elements of the system being modeled in terms of AISIM "entities." A detailed description of each entity is provided in Section 3 of the AISIM User's Manual. A general introduction to the types of system elements modeled by these AISIM entities is contained in section 1.3 of this manual.

1.2 OVERVIEW OF THE AISIM SYSTEM

AISIM consists of five subsystems, each of which performs a distinct function. These subsystems are: (1) the Design User Interface (DUI); (2) the Analysis User Interface (AUI); (3) the Replot User Interface (RUI); (4) the Hardcopy User Interface (HUI); and (5) the Library User Interface (LUI). Each of these subsystems is briefly described below.

(1) DESIGN USER INTERFACE

The DUI is the facility which enables the user to create or alter models of systems. It contains two sublevels, the Architecture Design Editor (ADE) and the Process Editor Interface (PEI). The ADE is used to model the physical layout of the given system, which is called the architecture. The PEI is used to define the processes or logic that are associated with that architecture. Other model entities are defined at the DUI level.

(2) ANALYSIS USER INTERFACE

With the AUI one subjects the model defined in the DUI function to simulation runs that test the behavior and response of the modeled system to various hypothetical conditions. In this function statistics are gathered on the operation of the system during simulation and, if desired, graphs of selected parameters are generated and are available for plotting.

(3) RELOT USER INTERFACE

The RELOT function enables the user to plot the statistics from various executions of a model, and to combine these plots as desired for future reference.

(4) HCOPI USER INTERFACE

The Hardcopy function provides the connection between the AISIM system and a printing device for automatically producing hardcopies of model logic. Process flowcharts constructed in the DUI can be printed on an HP2631G printer/plotter a TEK 4695 graphics copier, or the internal printer on the HP2623 terminal.

(5) LIBRARY USER INTERFACE

In the LUI the user is able to break apart and recombine parts of AISIM models, and obtain parts of models from a central system library. This feature is provided because some model components are used in other models and it is sometimes useful to store entire models for later reuse.

1.3 OVERVIEW OF AISIM MODELING CONSTRUCTS

This section provides a brief description of AISIM modeling constructs, to be followed by a more precise description of them in subsequent sections.

With some qualifications, AISIM's modeling constructs can be divided into the following four categories: (1) those used to represent the operations, properties, structure and internal relations of the modeled system itself; (2) those used to represent the environmental stimuli to which the system model is exposed; (3) those which represent the physical layout of the system; and (4) those which represent and facilitate mathematical operations.

1.3.1 ENTITIES REPRESENTING ELEMENTS EXTERNAL TO THE MODELED SYSTEM

1.3.1.1 The Load Entity. The Load entity is used to represent aspects of the modeled system that trigger processes within it. The Load entity represents the normal demand which is placed on the modeled system. Loads are defined by specifying the nodes at which certain Processes are to take place within a given period (see Scenario), together with specifications of several parameters which indicate the schedule that the Process triggering follows. The definition of a Load will also assign a priority to each of the Processes being triggered.

1.3.1.2 The Scenario Entity. A Scenario is used to represent the external demand on a system (i.e., Process triggerings from the outside) throughout a simulation exercise. The Scenario divides a simulation run into a number of periods that determine the frequency with which Loads will be initiated. They will also trigger Processes in a way that is not systematically related to the Loads in order to represent abnormal impositions on the system.

1.3.2 ENTITIES REPRESENTING ELEMENTS INTERNAL TO THE MODELED SYSTEM

1.3.2.1 The Process Entity. A Process is used to represent the operations, decisions, actions or activities that can be decomposed and defined in terms of more fundamental AISIM entities, called Primitives. A Process can take place in one or more of the system's nodes (or may execute independent of the nodes) and can make use of one or more Resources.

1.3.2.1.1 The Process Primitives. Primitives, of which there are 25, are the elements of which Processes are composed. A Process may be considered to be a collection of Primitives whose sequential execution describes the logic of the Process.

The 25 Primitives can be arranged into nine categories according to similarity of function. For the present, rather than give the meaning of each Primitive individually, it is sufficient to describe the categories and in a general way characterize the roles that members of each will play in the definition of a Process.

1. INTERNAL PROCESS EXECUTION CONTROL. The Primitives

COMPARE
BRANCH
ENTRY
PROB
LOOP

serve as a "framework" for Processes, enabling the Process to branch (either unconditionally or under certain conditions) to another portion of the Process, or to repeat certain segments of the Process a specified number of times.

2. RESOURCE ALLOCATION. As mentioned earlier, a Process frequently competes with other Processes for Resources. The Primitives

ALLOC
DEALLOC
RESET
TEST
LOCK
UNLOCK

govern the allocation of Resources among the various competing Processes.

3. PROCESS EXECUTION CONTROL. Since a principal feature of AISIM is its capacity to model parallel Processing, i.e., distinct Processes executing at the same time, these Primitives govern the timing of various Processes in the system relative to one another. The Primitives

CALL
SEND
SUSPEND
RESUME
WAIT

will either interrupt the Process in which they stand, or trigger or re-initiate some other Process.

5. QUEUE HANDLING. The Primitives

FILE
FIND
REMOVE

govern the placement and retrieval of Items in Queues that have been defined by the user.

6. ITEM HANDLING. The Primitives

CREATE
DESTROY

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

7. VARIABLE MANIPULATION. The Primitives

ASSIGN
EVAL

assign values to variables (both numerical and non-numerical) and allow for the mathematical manipulation of numerical variables.

8. TIME SEQUENCING. The Primitive

ACTION

which is associated with the Action entity described below, is included in Process definitions to indicate the time a certain Action (or process, decision, etc.) takes up without further describing the Action's nature.

9. DEBUGGING. The Primitive

TRACE

is not used to represent a system's operations, but is rather provided as a debugging facility to aid the user in the task of tracing a history of Process execution during simulations.

1.3.2.2 The Resource Entity. A Resource entity represents a component of the modeled system which may be necessary to the execution of a Process. Typically, a Resource will be required for more than one Process. Where several Processes demand a Resource that can serve only one Process at a time, all but one will stand in a queue until the Resource is available for them. The order in which the Processes will make use of the contended Resource is a function of the priorities associated with the various requests for the Resource.

1.3.2.3 The Action Entity. The Action entity is used in conjunction with the ACTION Primitive to represent any action, activity, decision, etc. that consumes time.

1.3.2.4 The Legal Path Table. The Legal Path Table (LPT) is a set of routes or paths between nodes in the system's architecture. The LPT is selected from all the possible paths between the nodes along the links, so that there is but one permissible routing of communication between the various nodes in the architecture. The LPT is accessed by several other elements of AISIM such as the EVAL Primitive, the keywords \$NODE, \$NXTNODE, and \$LINK, and the Message Routing Submodel Processes.

1.3.2.5 The Queue Entity. A Queue represents any holding area, such as a memory buffer or job queue, for elements waiting to take up their role in the operation of the system. User-defined Queues can be used as a holding area for Items. A user-defined Queue can be manipulated in a number of ways described later and in the AISIM User's Manual section 3.4.

1.3.3 ENTITIES WHICH SUPPORT MATHEMATICAL OPERATIONS

1.3.3.1 The Constant Entity. A Constant is an entity whose value does not change during a simulation run. Constants are specified or altered in the DUI and can be edited before a simulation run in the AUI but cannot be changed (and do not change) once the execution of a model has begun. Several parameters required in the definition of an AISIM model, (e.g., the number of Resource units available, the period length of the simulation and the size for Queues) can only take Constants or simple numerics as values.

1.3.3.2 The Variable Entity. Variables, by contrast, are entities whose values can change during the exercise of a model. The majority of the parameters in the specification of a model can take Variables as values.

1.3.3.3 The Table Entity. Tables are single-value, single-argument functions defined by the user. They may be defined either as discrete, continuous, or alphanumeric and may have from 1 to 15 entries. Tables are accessed by the EVAL Primitive and serve as a supplement to the mathematical operations automatically available as part of the EVAL Primitive.

2. CREATING SYSTEM ARCHITECTURES

With the basic understanding of AISIM modeling concepts presented in the previous section, the reader should now be able to interact with the DUI. The exercises here are intended both to deepen the user's grasp of AISIM modeling constructs and to familiarize him with the prompts encountered while interacting with the DUI. In general, it is not a good idea to begin the design of an AISIM model without having done research and preparation on paper beforehand. However, as a teaching device, we shall develop fragments of an architecture from requirements formulated as we go along.

The method of logging on and invoking AISIM is computer-specific so we shall assume that the user has reached the point at which the computer prompts him with

AISIM READY

The user will have been offered a collection of information that looks something like that depicted in figure 1.

```
=====
This is AISIM Production Version 4.0V which was built from
AISIM Version 3.0V.
2/1/85

***Please report any problems to: Donald Constantine (617) 271-7754***
=====
```

Figure 1. Typical Display Upon Entering the AISIM READY Level.

To enter the DUI, type

design project(test) term(trmtyp)

"test" is the name of the model to be designed. Trmtyp represents the type of terminal being used.

The valid terminal types are the following:

HP - HP2647A, HP2648A
HP23 - HP2623
VT - VT100
TEK - Tektronix 4105 with Selanar graphics

The user will be prompted with information that looks something like that shown in figure 2.

```

AISIM READY
design project(test) term(hp)
CURRENT PARAMETERS IN EFFECT:
VERSION:      PRODUCTION VERSION 4.0
TERMINAL:     HP
PROJECT:      TEST
USER:         [USER]
ENTER YES TO PROCEED, NO TO ABORT...

```

Figure 2. Typical Information on Entering the DUI

By typing

NO

the user will return to the AISIM READY level. Typing

YES

will put the user in the DUI and the screen will display an asterisk to indicate that the user may enter DUI commands.

When the computer displays the prompt "*", enter the Architecture Design Editor (ADE) by typing

ARCH

A grid like that in figure 3 will appear on the screen.

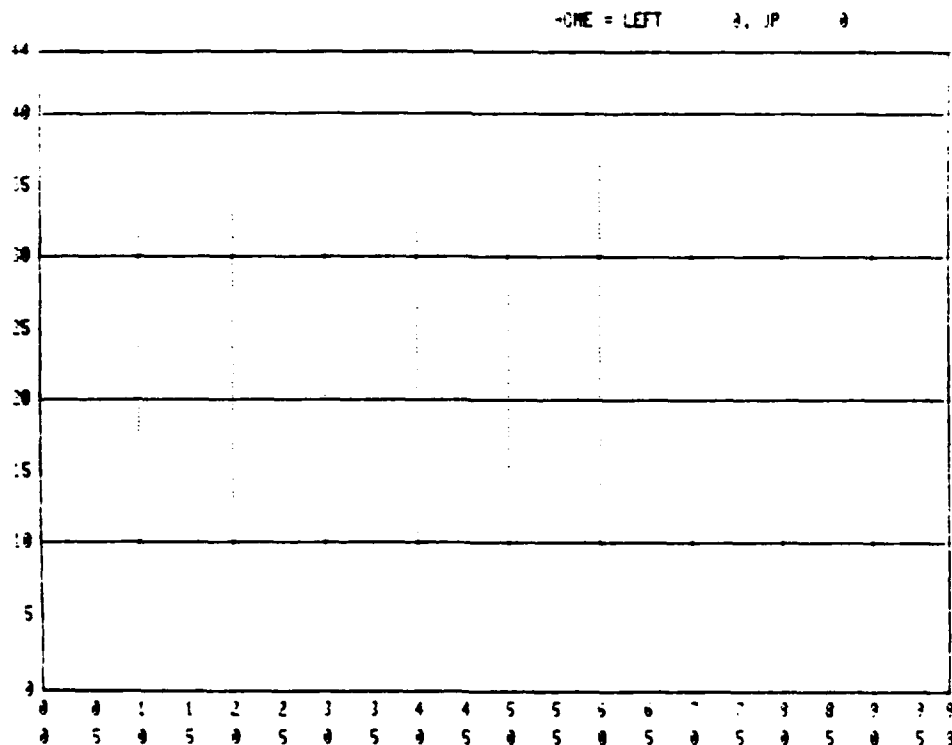


Figure 3. Grid on Which an Architecture is Designed

The AISIM constructs manipulated in the ADE are nodes which represent the hardware elements of a system and links which represent lines of communication between them. The physical layout of the system is represented by placing nodes and links on the grid to represent various hardware elements of a system and their (available) lines of communication. A Resource modeling entity is automatically created for each node or link when it is placed in the architecture.

As a mnemonic aid in distinguishing system elements, AISIM provides fourteen geometrical symbols for nodes. The symbols are called by the three-letter designations given in figure 4.

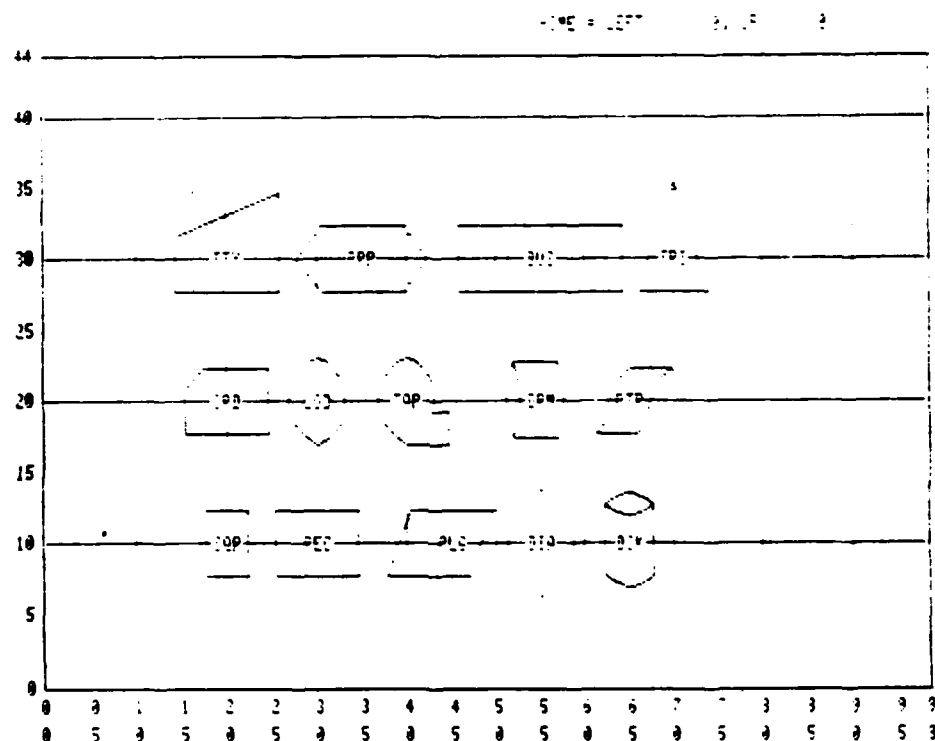


Figure 4. Designations of the Fourteen Symbols

With two exceptions these node symbols differ from one another only in their appearance. The two exceptions are the so-called "leaf-nodes" TTY and LOD. These nodes may be connected to the modeled architecture by only one link. All other nodes may be connected to any number of other nodes through any number of links. The rationale for this restriction is explained in the AISIM User's Manual, section 6.3.3.

2.1 DRAW AND NODRAW MODES

In the ADE there are two modes called DRAW mode and NODRAW mode. When the user is in DRAW mode, all architecture commands which change something in the architecture display cause the display to be updated immediately. If the user is in NODRAW mode, the architecture display is not automatically updated. The user can cause the display to be updated by typing

REDRAW

DRAW mode is the default on HP terminals and TEK4105 terminals. NODRAW mode is the only mode available on VT100 terminals. This restriction is due to a lack of capability in the terminal to make it operate like the other terminals. Therefore, if a user is on a VT100, he must type REDRAW to see the results of any ADE commands. The following discussions assume the user is on a terminal other than a VT100 except where specifically noted. If the user is on a VT100 terminal, he will need to use the REDRAW command to view the results of the ADE commands described in the following sections.

2.2 DEFINING ATTRIBUTES FOR SYMBOLS

As mentioned earlier, when a symbol is placed in the architecture, an AISIM entity called a Resource is created to represent the hardware element depicted by the node or link. Resources can have a number of user-named attributes. The DEFINE SYMBOL command allows the user to associate attributes with each symbol type so that when placed in the architecture, the Resource will be created with all attributes defined. For example, typing:

DEFINE SQR

(SQR could be replaced with any of the 14 symbol mnemonics or the mnemonic CON if a link is being defined.) The user will be prompted by a form as shown in figure 5.

RESOURCE NAME: [REDACTED]

TOTAL NUMBER OF UNITS: [REDACTED]

INITIAL NUMBER OF UNITS: [REDACTED]

DESCRIPTION: [REDACTED]

ATTRIBUTES

[illegible]

Figure 5. Symbol Definition Form

The user can add or modify fields within this form by using the terminal keys as defined in appendix A. Any values in the inverse video fields of the form are default values supplied from the AISIM design data base. The user may change these fields by typing over the existing values. The user may enter up to fifteen attribute names and related values of his choice into the attribute fields. For example, when defining attributes of a symbol type which is to represent a disk in the modeled system, the attribute names may be something like seek time, latency, etc., and the values would be the corresponding values for the particular disk being modeled.

A second use of the DEFINE SYMBOL command is the following:

```
DEFINE SYMBOL,RESOURCE NAME
```

where SYMBOL is one of the symbol mnemonics or CON, and RESOURCE NAME is the name of an existing Resource entity. If the named Resource entity exists, a form similar to the form shown above would be displayed. Instead of the default attributes, the form displayed would have the names and values of any attributes previously defined for the Resource entity referenced in the command.

This command will only be accepted if a Resource entity has been previously defined before entering the ADE. Since the user has not defined any Resource entities in his test data base yet, this command would fail. The user might want to try this command later.

2.3 PLACING NODES ON THE GRID

To place a node at a certain location on the grid--i.e., centered on that location--issue the PLACE command designating (1) the type of node to be placed, (2) a user-given name, and (3) horizontal and vertical position coordinates. One can also opt to indicate the size of the geometrical shape if the default value, equal to the number of characters in the user-given name, is unsuitable. To center a square named NODE1 twenty units from the left-hand side and thirty units from the bottom in figure 4 above, type

```
PLACE SQR,NODE1,20,30
```

Figure 6 shows the screen display that would result from this command.

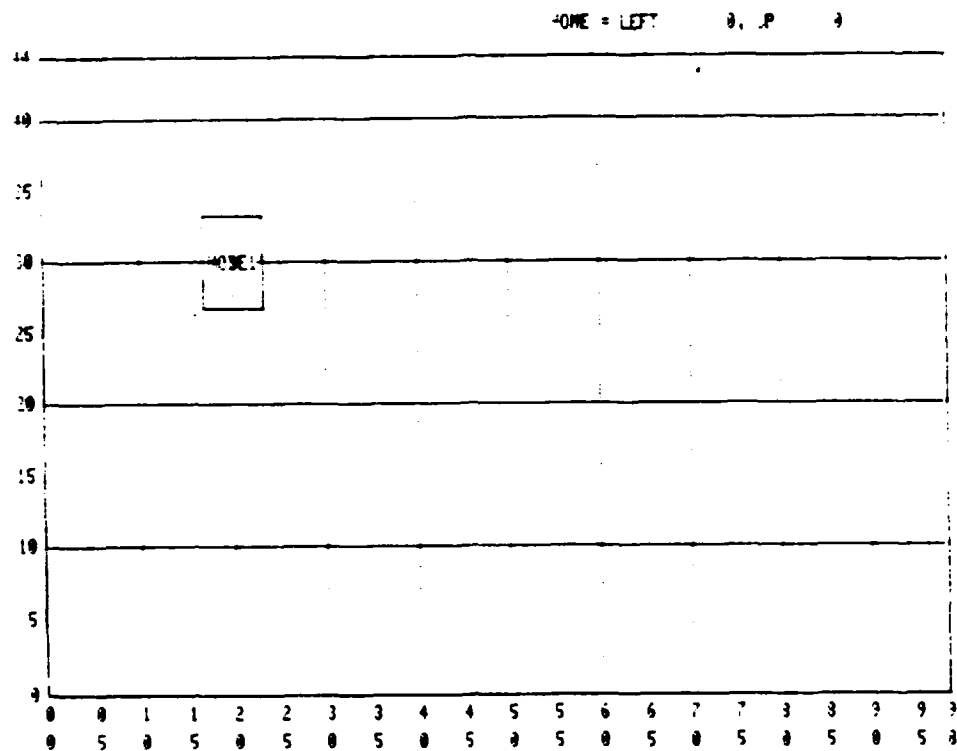


Figure 6. Architectural Grid With a Single Node

All nodes are placed in this way. To place nodes in the positions shown in figure 7, type the following sequence of commands:

P TTY,NODE2,10,10

P PRP,NODE3,40,30

P TRI,NODE4,85,10

P TAP,NODE5,45,15

P CRD,NODE6,80,35

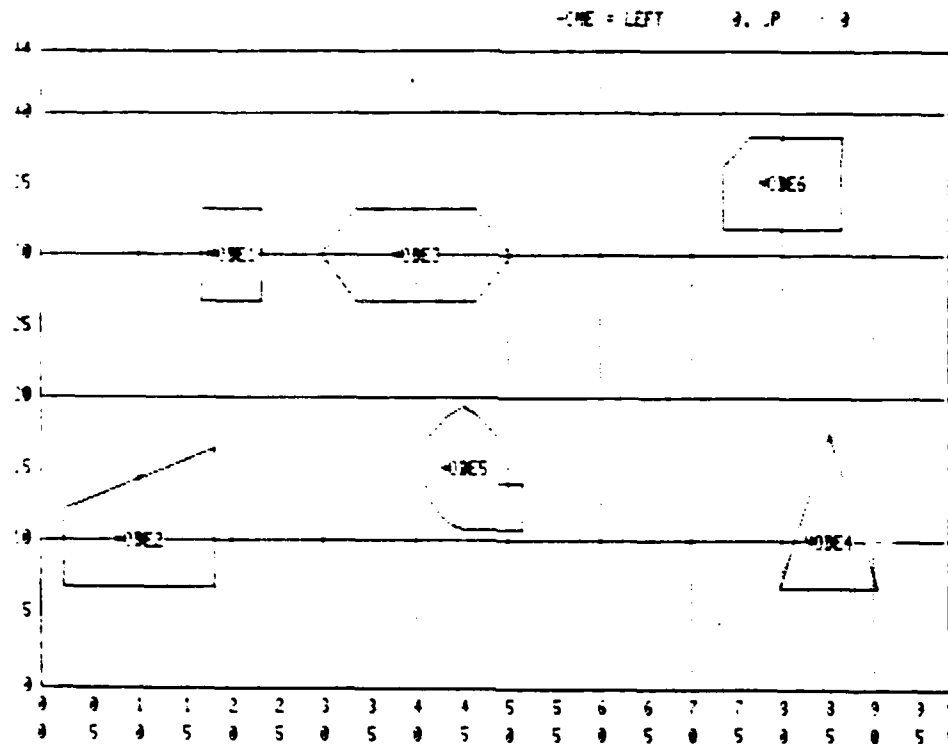


Figure 7. Six Nodes on an Architectural Grid

2.4 CONNECTING NODES

The second step in creating a system architecture is the placement of connections between the nodes. The CONNECT command is used to connect two nodes. Such connections, or "links", are defined by specifying (1) the node from which the link is to run, (2) the node to which the link is to run, and (3) a user-given name of the link. To place a link called "LINK1" from NODE1 to NODE2, type

```
CON NODE1,NODE2,LINK1
```

This command places a cursor at NODE1 if the user is on an HP terminal, or in the lower left corner if the user is on a TEK terminal; typing any alphanumeric character other than a period causes a straight line to be drawn between the centers of the two nodes, thereby drawing the link. The graphic result is shown in figure 8.

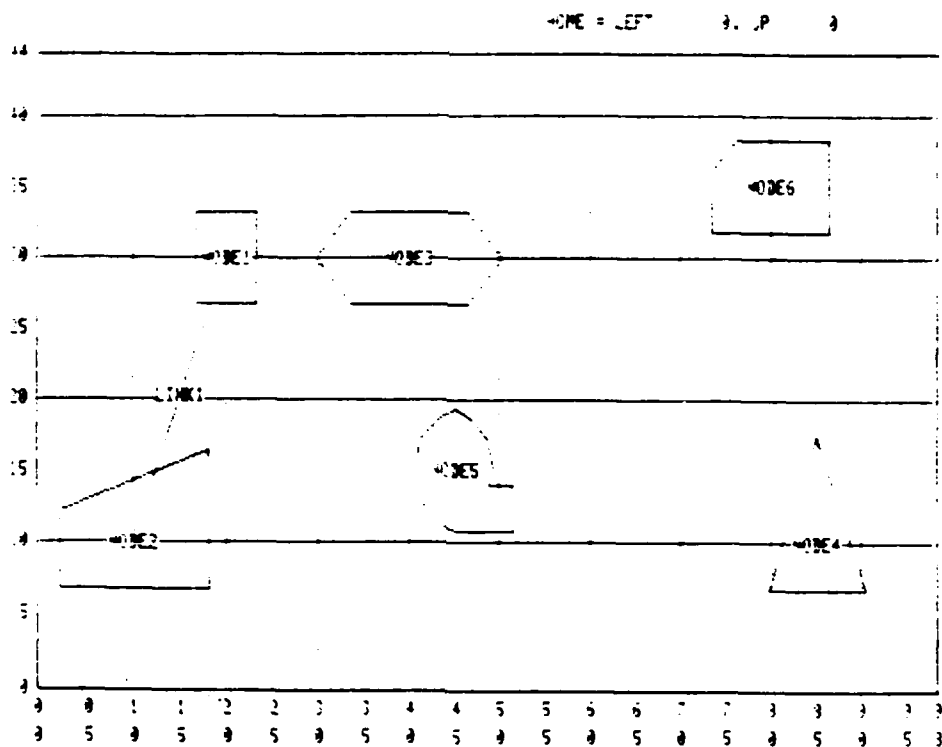


Figure 8. Architecture with One Link Defined

Links need not always appear as straight lines, as is shown in figure 9.

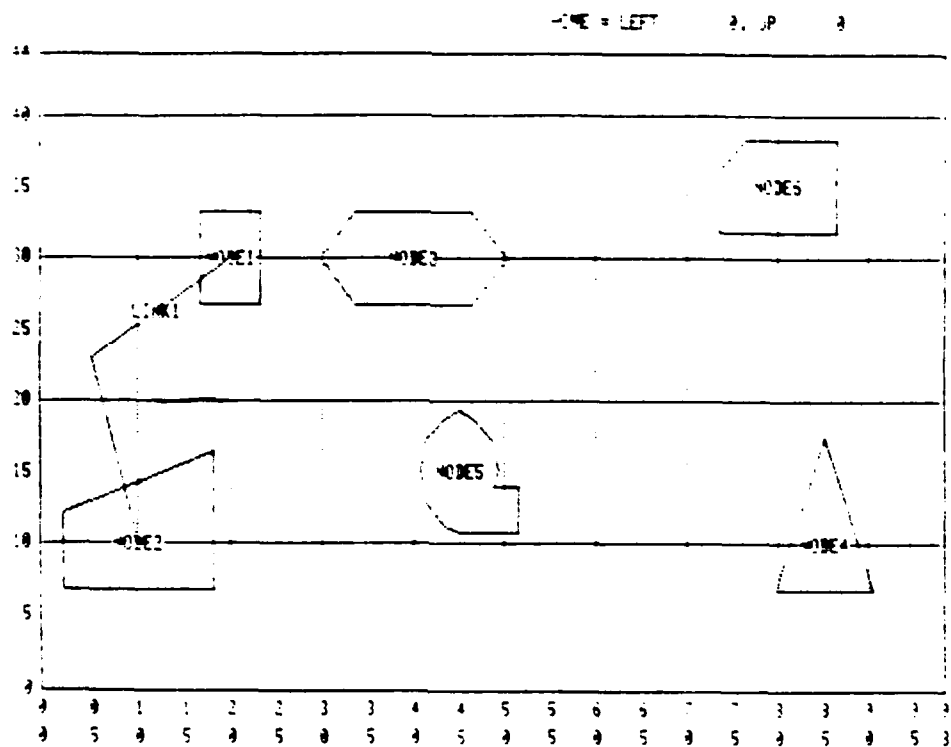


Figure 9. Architecture with a Bent Link

To create links that bend, enter the CON command as above. Then using the graphics cursor controls on the terminal (either the arrow keys on the HP terminals or the joystick button on the TEK terminal), the cursor can be moved to the spot where the link is to bend. When the cursor is at the point of the bend, type in a period (.). If no further bending is desired, typing any other non-period alphanumeric character will complete the connection. The resulting connection will resemble the one depicted above in figure 9.

Links may be given more than one bend by repeating the sequence of moving the cursor and typing a period (.), and then depressing any non-period character only when all the desired bends (up to 5 bends, i.e., six segments) have been created.

To create the links shown in Figure 10, type the following sequence of commands:

```
CON NODE1,NODE4,LINK4
```

```
CON NODE3,NODE6,LINK3
```

```
CON NODE3,NODE5,LINK5
```

NOTE: If the user is logged on through a VT100 terminal, there is no capability to create bent links. All connections are automatically made as straight lines between the two nodes.

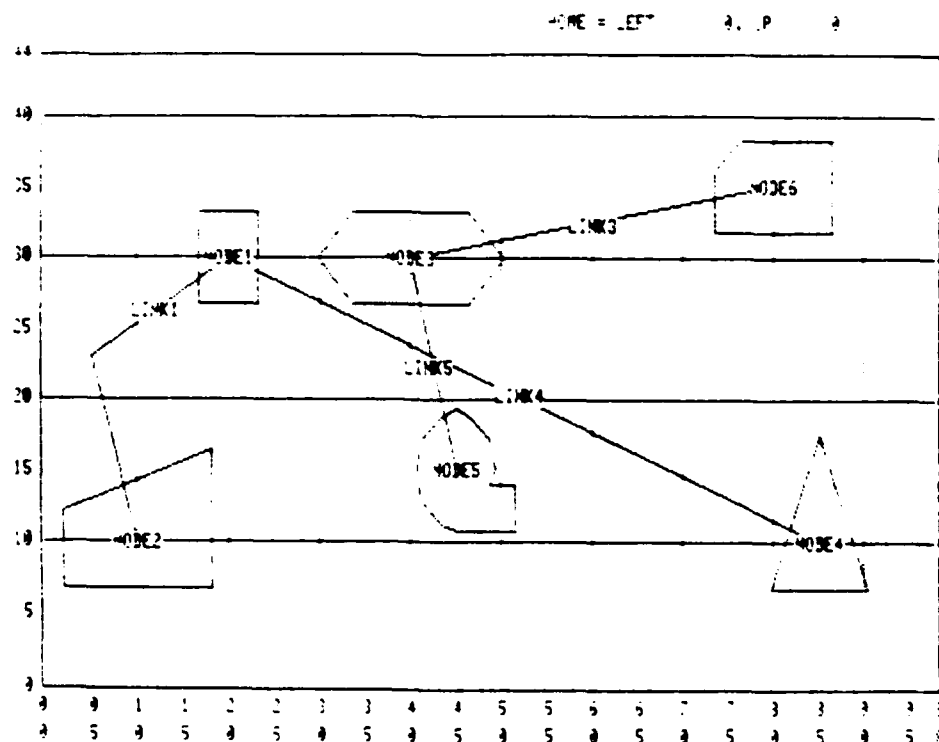


Figure 10. Architecture with Four Links

2.5 CHANGING THE SIZE, TYPE, AND NAME OF NODES AND LINKS

The size, type, or name of nodes and the names of links can be changed using the CHANGE command. By typing the following commands, nodes and links may be altered:

```
CHG NAME,NODE1,NODEX
```

```
CHG TYPE,NODE2,LOD
```

```
CHG SIZE,NODE3,7
```

```
CHG NAME,LINK4,LINKZ
```

The user may note the changes on his screen. By typing the following commands, the architecture is returned to its original configuration:

```
CHG NAME,NODEX,NODE1
```

```
CHG TYPE,NODE2,TTY
```

```
CHG SIZE,NODE3,5
```

```
CHG NAME,LINKZ,LINK4
```

As mentioned earlier, Resource entities are created by the AISIM system to model the architecture elements. When the name or type of a node is changed or the name of a link is changed, the appropriate changes are also made to the associated Resource entities. That is, when a node or link name is changed, the associated Resource name is changed. When the type of a node is changed, new attributes may replace the existing attributes of the Resource since different attributes may be defined for the new symbol type. Refer to Section 2.2 of this manual.

2.6 DELETING NODES AND LINKS

Existing nodes and links may be deleted from a system architecture with the DELETE command. For this example, to eliminate the connection between NODE1 and NODE2 type

```
DELETE LINK1
```

The result on the screen would be that the link named "LINK1" would disappear.

When a node is deleted, all of the links associated with it also disappear. As an example type

```
DELETE NODE6
```

The result of deleting LINK1 and NODE6 is shown in figure 11. Note that LINK3 disappeared also.

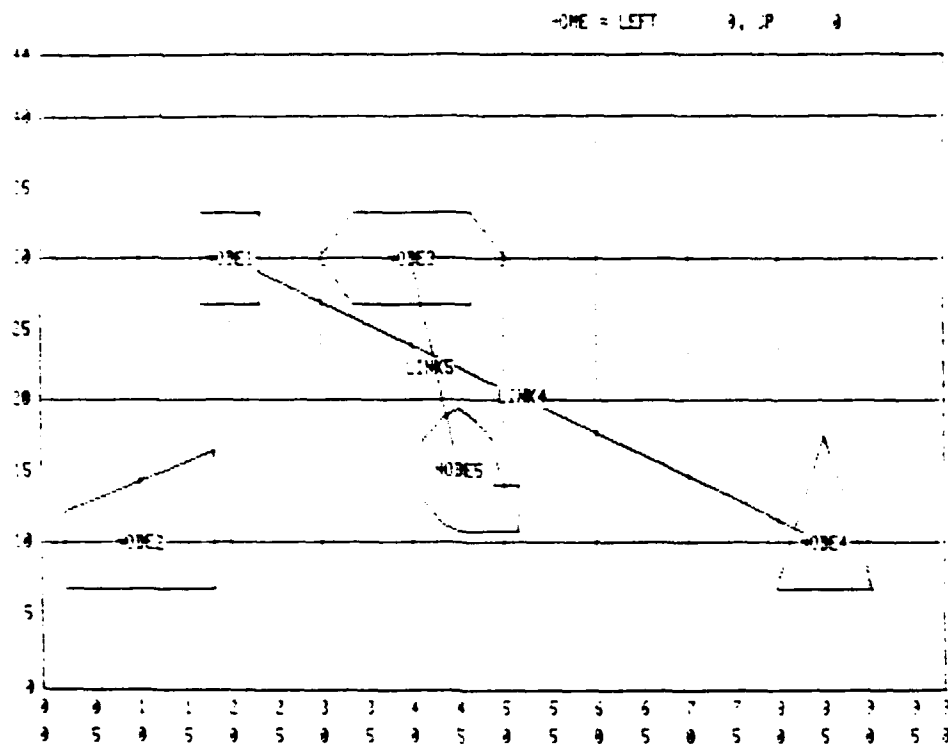


Figure 11. The Result of Deleting LINK1 and NODE6

2.7 MOVING PREVIOUSLY PLACED NODES

The location of a node on the architecture grid may be changed with the MOVE command. For example, to move NODE4 from its current position to the coordinates 55,5 one issues the command:

MOVE NODE4,55,5

The graphic result is shown in figure 12. The symbol is now centered at 55,5 with all of its previously defined connections intact.

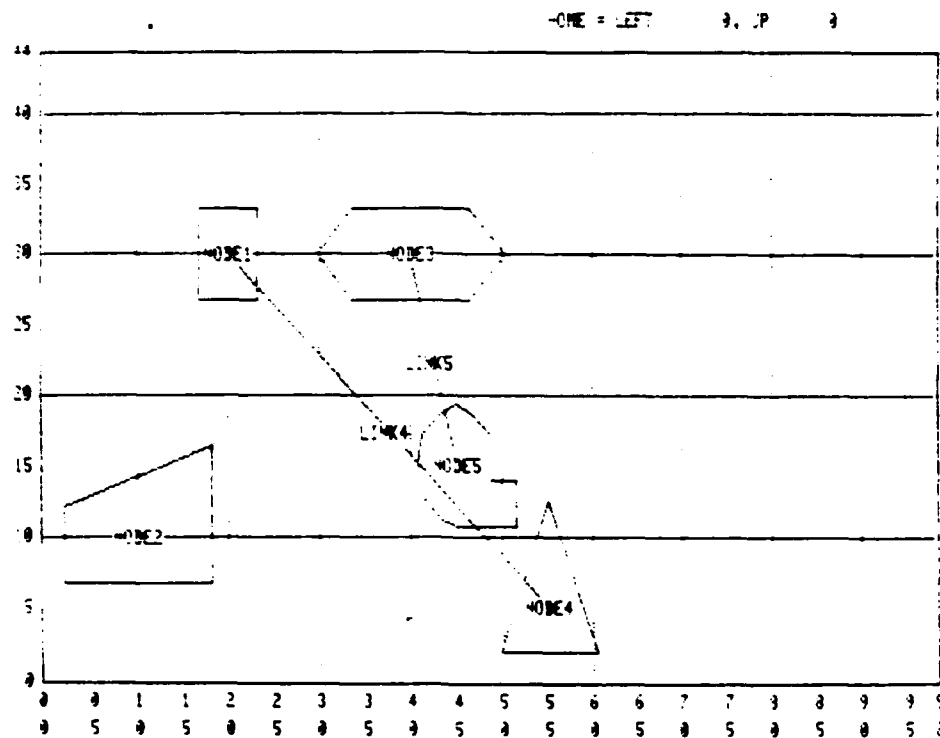


Figure 12. The Result of Moving NODE4

2.8 RECONNECTING EXISTING LINKS

The previous example of moving NODE4 created a problem that can be solved with the command RECON. As figure 12 shows, the link between NODE1 and NODE4 now runs through NODE5. The connection can be made to bend around NODE5 by first typing

RECON LINK4

This command will delete the existing graphics for the link between NODE1 and NODE4 and, as in the original CONNECT command, place the cursor at NODE1. Using the sequence of cursor movements and periods (.) described in section 2.4, up to five bends in the existing connection between NODE1 and NODE4 can be created. To complete the connection, type any non-period character. The graphic result will be something like that shown in figure 13.

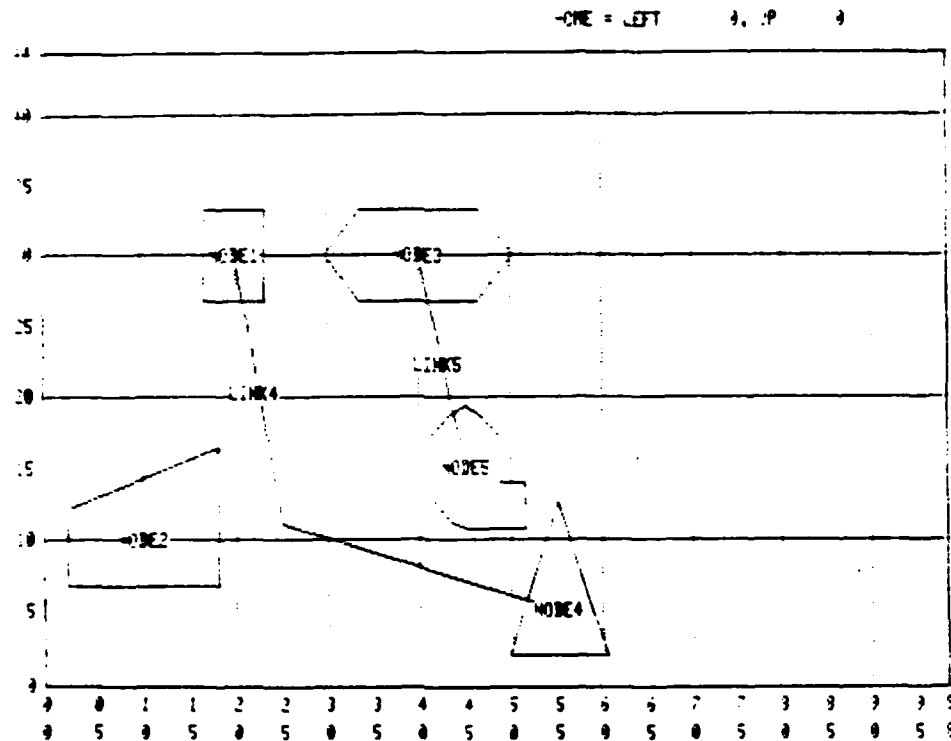


Figure 13. Architecture After Reconnecting

NOTE: The same restriction applies to the RECON command as the CONNECT command when the user is logged on to a VT100 terminal; i.e., only one-segment connections are allowed (see section 2.4).

2.9 ALTERING ONE'S VIEW OF THE ARCHITECTURE GRID

The usable grid space in ADE is actually larger than what can be displayed on the terminal screen at one time. If an architectural design is too large for the screen to accommodate, different parts of the total workspace can be viewed and manipulated through the WINDOW command. The WINDOW command allows the directional change of the user's view of the grid. The command specifies the direction of change--up, down, right or left--as well as the number of grid units the view is to be changed.

For example, to move the view of the screen in figure 13 down 15 units, type

WINDOW D,15

Figure 14 shows the result of this command.

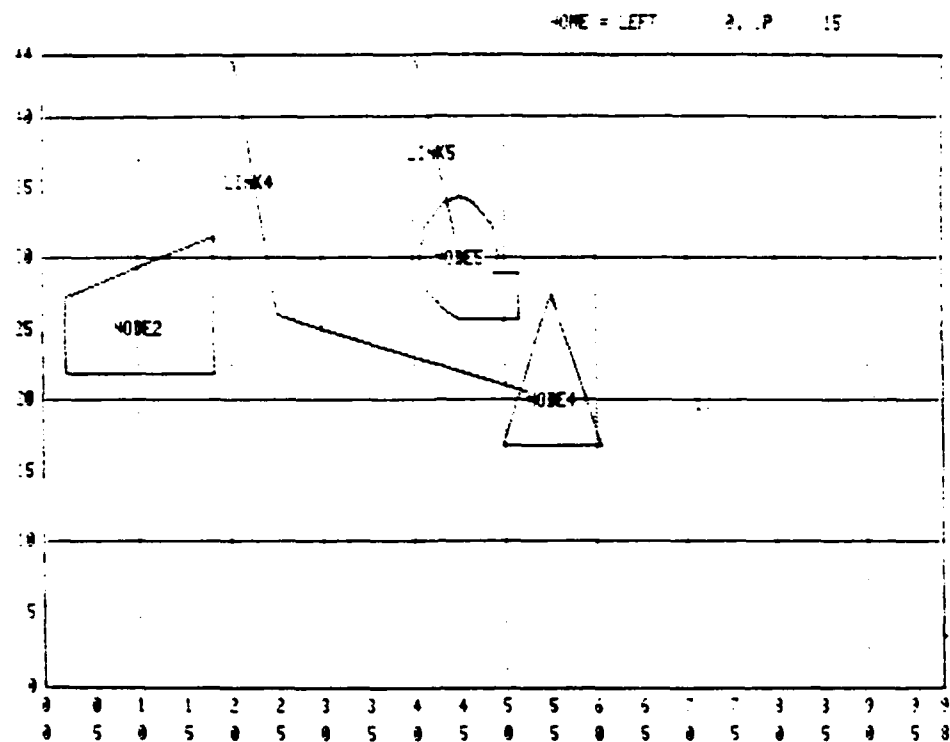


Figure 14. Result of WINDOW Command

The WINDOW command will accomplish both horizontal and vertical movements at the same time. To move our view of the screen further down 15 units and 20 units to the right, type

WINDOW D,15,R,15

Figure 15 shows the result of this command.

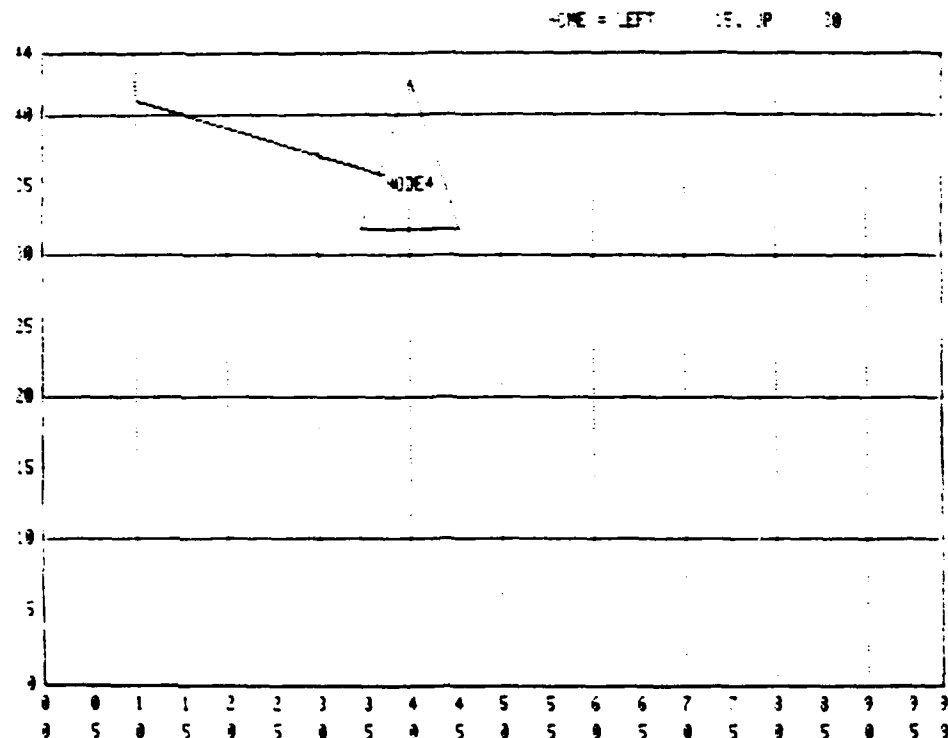


Figure 15. The Result of Further Use of WINDOW

Note that the WINDOW command parameters required to get back to the original (HOME) position are always displayed above the upper right corner of the architecture grid.

2.10 DEFINING LEGAL PATHS

The purpose of the Architecture facility is to specify routes of communication between hardware elements so that Process execution will be realistically related to the physical layout of a system. Such routes are represented by a Legal Path Table which specifies the links and the nodes through which communication from one node to another must take place. There are several methods of defining a Legal Path Table (LPT). Three methods are offered to the user at the end of an ADE session. These methods are predefined algorithms for the definition of an LPT which can be executed optionally at the user's discretion. See the AISIM User's Manual section 6.3.18 for details of how these algorithms function. For many architectures it is more economical to create the Legal Path Table while defining the configuration of hardware elements rather than using the algorithms mentioned above. If an LPT is generated according to the following discussion, the predefined algorithms should be bypassed since they would erase the LPT so defined.

Suppose we augment the architecture developed above with more links so that it resembles that shown in figure 16.

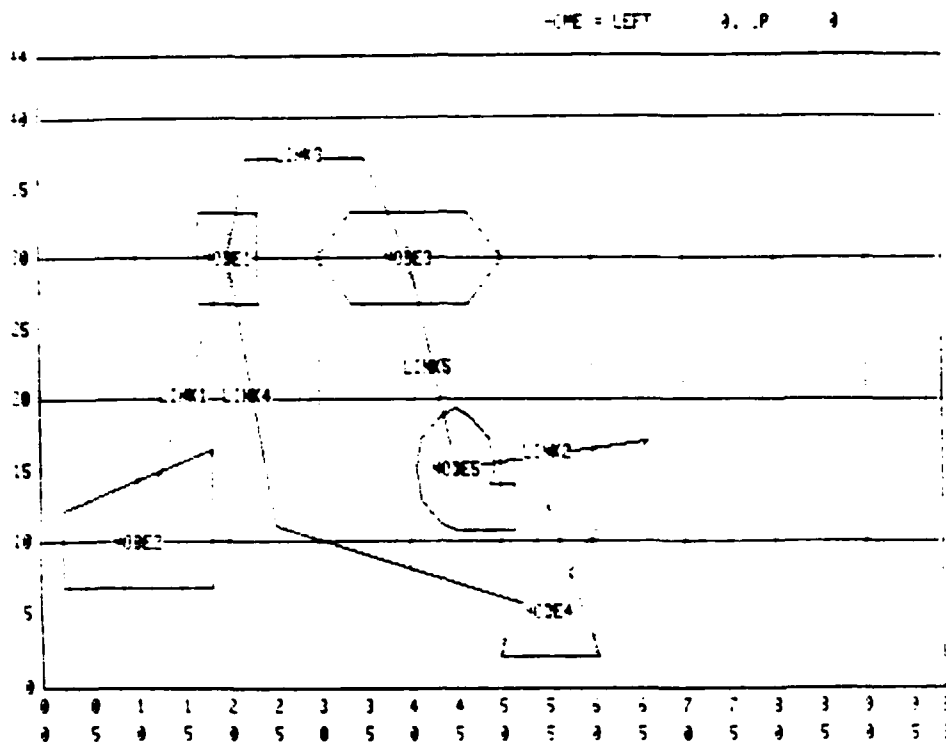


Figure 16. Augmented Architecture

The LPT is defined by means of the command `DEFINE PATH`. If, for example, `NODE1` is to communicate with `NODE4` along the communication lines represented by the links `LINK3`, `LINK5`, and `LINK2`, type

```
DEFINE PATH,NODE1,NODE4,LINK3,LINK5,LINK2
```

No confirmation will be displayed immediately at the screen, but the Legal Path Table will have been augmented to reflect the new paths. However, the command `LIST PATH` enables the user to inspect the Legal Path definitions currently in effect. To obtain a listing at the screen of the legal path from `NODE1` to `NODE4`, type

```
LIST PATH,NODE1,NODE4
```

The resulting list is shown in the upper right-hand corner of figure 17.

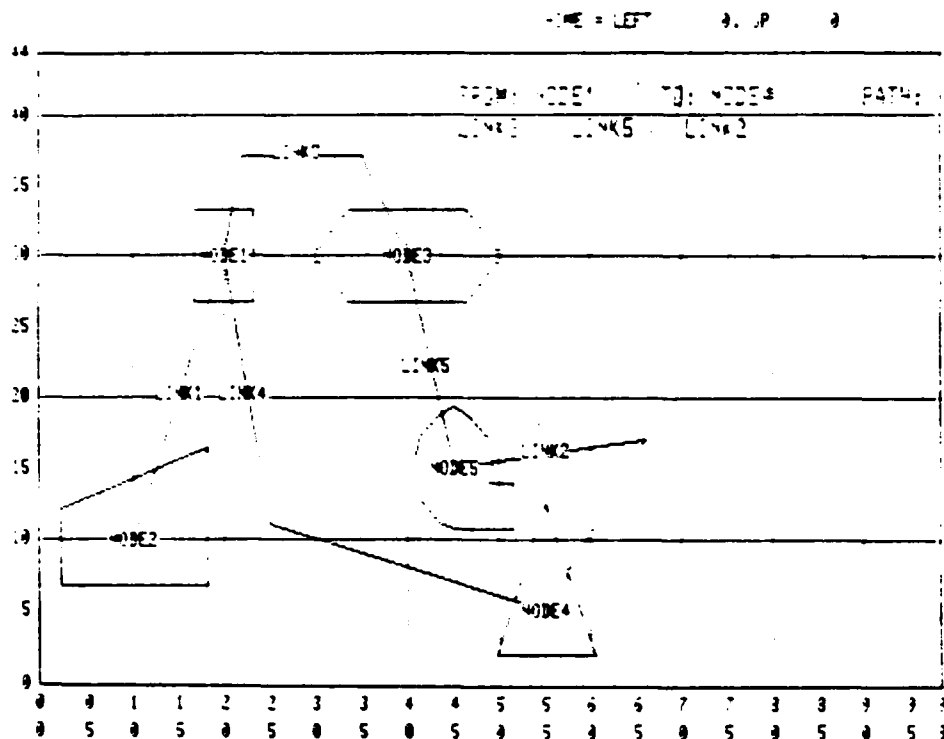


Figure 17. Typical List of Legal Paths Obtained in ADE

Note that paths from NODE3 and NODE5 to NODE4 have been automatically defined by the preceding DEFINE PATH command. The principle is that any time a legal path is defined through a number of nodes, the AISIM system creates legal paths from all nodes through which the path passes to the destination to node. Care should be taken in defining subsequent legal paths according to this method. Any conflicts of path routing in paths defined later would result in the elimination of previously defined paths. Following is an illustration of this operation of the system. Assume that the path has been created as above. If the user should now enter the command:

DEFINE PATH,NODE2,NODE4,LINK1,LINK4

not only would the path from NODE2 to NODE4 be established, but the path from NODE1 to NODE4 would be altered to be the direct path via LINK4. The paths defined automatically from the previous command (i.e., the paths from nodes NODE3 and NODE5 to NODE4) would still exist since there was no conflict with these paths and the newly defined path.

3. DEFINING PROCESSES IN THE DUI

Whereas the Architecture Design Editor (ADE) is used to represent the physical layout of a system, the Process Editor Interface (PEI) is used to represent the logic and data-handling behavior of processes in the system.

This section provides examples to familiarize the user with the commands and prompts used in the PEI. Earlier the user was urged to begin the design of an AISIM model with sufficient research and planning to fully understand the system to be modeled. However, as a teaching device, we shall develop fragments of a Process from requirements formulated as we go along.

The exercises here are intended both to deepen the user's grasp of the Process Primitives and to familiarize him with the prompts encountered while interacting with the PEI.

Assuming the user has just completed the foregoing section, entered an END command to exit the ADE sublevel, and another END command to bypass the LPT generation, he will be at the DUI level of operation. A "*" should be displayed. To invoke the PEI sublevel of the DUI, one enters the EDIT PROCESS command designating the name of the Process to be edited. Once in the PEI, the user can terminate the PEI session by entering an END command.

Consider first the simplest Process that could be of any use to the modeler of a system: the Process starts, a certain amount of time is taken with an Action and then the Process ends. This Process will be represented in AISIM by the START symbol, the ACTION Primitive and the END symbol. To represent such a Process in AISIM, one begins by issuing the command

EDIT PROCESS,EXAMPLE,NEW

which informs the system that one wishes to create a Process named "EXAMPLE". To alter a Process that has been previously defined, one would not enter the "NEW" part of the command. The computer will respond with a form to be filled in at the terminal. This is done by typing into the fields provided in the form. The form is shown in figure 18.

START

PROCESS NAME [REDACTED] NODE [REDACTED]

ATTRIBUTES ATTACHED (YES OR NO) [REDACTED]

PROCESS DESCRIPTION
[REDACTED]

START BLOCK TYPE [REDACTED]

ENTER "PARM" FOR PARAMETER PASSING

ENTER "ITEM" FOR ITEM PASSING

ENTER "STD" FOR STANDARD PROCESS

Figure 18. Initial Form for Process

The NODE field asks for the node in the architecture with which the Process is associated. Since this Process is not yet related to an architecture, the field is left blank. The next field allows the assignment of attributes to the Process. For the present, we shall decline to do so. The field labeled PROCESS DESCRIPTION is for a comment to describe the Process. In this case, type "Example Process".

Depress the key that enters the form (see appendix A for specific key). The screen will go blank for a moment and then display the image depicted in figure 19.

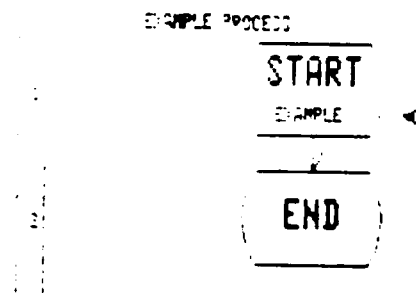


Figure 19. Graphic Display That Follows Entering a Standard Process

Much of the information you entered on the form now appears in this graphic representation of EXAMPLE. The "NO" to the right of the START symbol indicates the the Process has no attached attributes.

3.1 PROCESS EDITOR DRAW/NODRAW MODES

The Process editor maintains DRAW and NODRAW modes similar to the Architecture Design Editor except that both DRAW and NODRAW modes are available for all terminal types in the Process Editor. If the user is in DRAW mode, changes made to any Primitives on the screen will cause the specifically affected Primitives to be redrawn. If the user is in NODRAW mode, then any changes to Primitives will not cause the screen to be updated until a REDRAW command is entered, which will cause the current display to be redrawn. The user can also use the commands TOP, BOTTOM, UP and DOWN to cause the display to be redrawn at a different location.

The default mode for the Process Editor is DRAW mode. The user can switch to NODRAW mode or back to DRAW mode by entering the command NODRAW or DRAW. The following discussions assume the user in DRAW mode.

3.2 ACTION

To place an ACTION Primitive between the Start and the End symbols, enter the command

P ACTION

which tells the computer to place an ACTION Primitive between the last Primitive defined and the END symbol. The computer will now display a new form to be filled in. This is shown in figure 20.

PARAMETERS FOR ACTION

ACTION NAME: METHOD:

MEAN TIME: DELTA-TIME:

COMMENT:

Figure 20. Form for the ACTION Primitive

The field ACTION NAME requests a name for the Primitive in the Process, which should be identical with the name of the associated Action entity (Action entities are described in section 4.1). We shall call it "Delay". The field COMMENT is a place to write a short reminder of what the ACTION Primitive is supposed to represent. The three remaining fields METHOD, MEAN TIME, and DELTA-TIME enable the user to vary the time taken up by the ACTION by invoking various statistical distribution methods (such as exponent, uniform, etc. See section 3.9.1 of the AISIM User's Manual for a description of the valid distributions.) Assume that the ACTION always requires the same amount of time, and hence the MEAN TIME requested will be equal to the duration of the ACTION. Indicate the time with a variable whose value for this example is specified elsewhere, calling it "T1".

The form should then be filled in thus: call the ACTION "Delay", set the method at "Constant", and set the MEAN TIME at "T1". Type the comment field "Action which causes delay". Leave the field labeled DELTA blank. After leaving the form, the screen will display a new version of EXAMPLE, as depicted in figure 21.

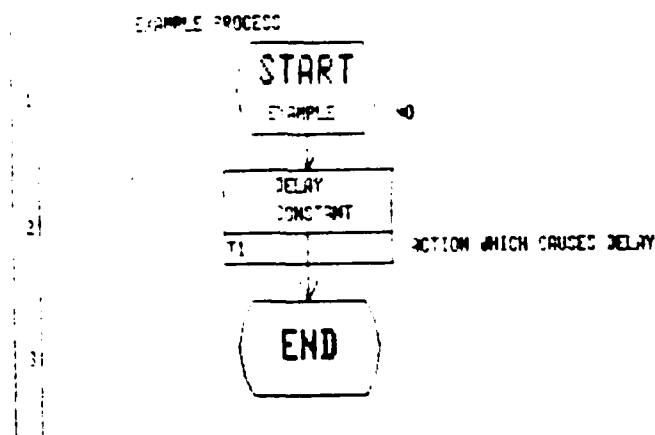


Figure 21. Process with a Single ACTION

3.3 HOLD

EXAMPLE may be augmented in a number of ways. For example, the ACTION Delay may be repeated in succession. There are two ways to repeat this ACTION in a revised version of EXAMPLE. First, one can place more copies of Delay in the Process, one after another, with the command

P ACTION

This command may be repeated as many times as one wishes the Action to be performed in the Process. The second method of creating several instances of an ACTION which is less time-consuming, involves the HOLD storage area. HOLD constitutes a storage area for Primitives that are likely to be used more than once with little or no alteration. To place a previously defined Primitive into HOLD, type

HOLD 2

The number 2 represents the position in the Process of the Primitive to be stored in HOLD. The position is indicated by the numbers in the column on the left. Hereafter, the ACTION Primitive "Delay" may be placed in a Process by typing

P HOLD

Each time this latter command is issued, the user will be presented with the form associated with the Primitive in case any small alterations in its parameters are to be made. Whether or not any alterations are made, leaving the form will result in the placement of the Primitive stored in HOLD in the Process being edited.

Figure 22 shows the display that will appear after several identical ACTION Primitives have been placed in succession or after the HOLD storage area containing the ACTION "Delay" has been placed. The procedure of repetition will occur as many times as the user requests it.

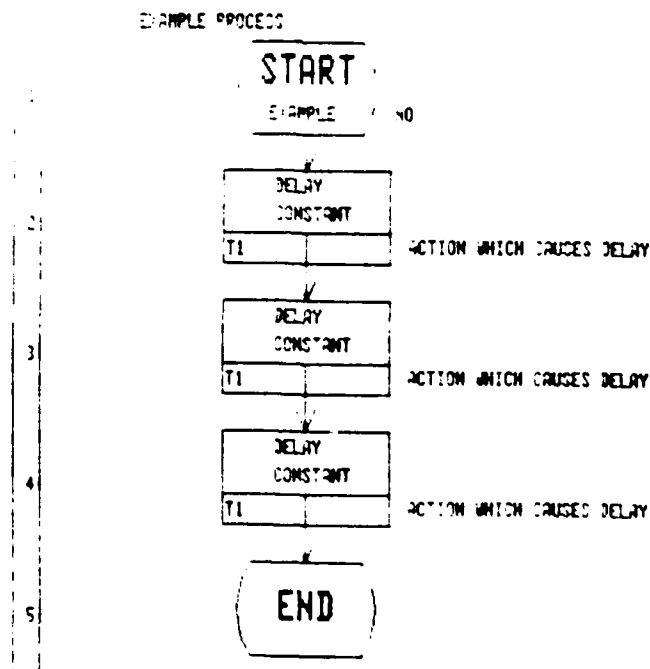


Figure 22. Process with Three Identical ACTION Primitives

3.4 ENTRY AND LOOP

If the ACTION "Delay" is to be performed a certain n number of times, as in the most recent version of EXAMPLE, a much simpler procedure is available than that of placing n instances of the ACTION between the START and END symbols. One can instead indicate more directly that a certain part of a Process is to repeat itself an n number of times. This is accomplished by means of the Primitives LOOP and ENTRY. Figure 23 shows EXAMPLE altered with LOOP and ENTRY Primitives to cause a triple repetition of the ACTION "Delay".

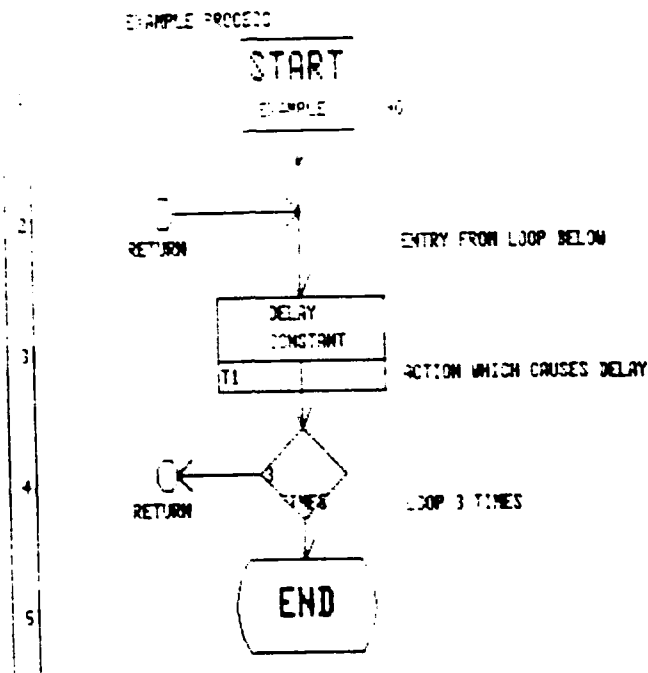


Figure 23. Process with Triple Repetition of the ACTION Delay

The diamond-shaped figure indicates that the line of processing is to be diverted to the point labeled "Return" above it (it could have been given any label whatever up to 8 characters).

To effect the LOOP Primitive as shown in figure 23, we must first get rid of the two extra ACTION Primitives. This is done by typing the following commands:

DELETE 3

DELETE 3

P LOOP

The screen will show the form shown in figure 24.

PARAMETERS FOR LOOP:

LOOP TO LABEL: [REDACTED]

LOOP [REDACTED] TIMES

COMMENT: [REDACTED]

Figure 24. Form for the LOOP Primitive

The first field asks for the name of the entry point to which Process control is to be diverted. The second asks for the number of times the primitives between the ENTRY and the LOOP are to be performed (i.e., control will be diverted to the entry point one less time than the number since the steps will have been executed once already when the LOOP is reached). The remaining field is self-explanatory.

The ENTRY Primitive must now be placed above the ACTION Primitive. Since the PLACE (or "P") command by default inserts the new Primitive just before the END symbol, a modified PLACE command is required to place a Primitive elsewhere in the sequence. To use this command, type

P ENTRY,2

The number "2" indicates where the Primitive is to be placed, in reference to the column of numbers on the left of the Process diagram.

The screen will then display the form shown in figure 25.

PARAMETERS FOR ENTRY:

ENTRY LABEL: [REDACTED]

COMMENT: [REDACTED]

Figure 25. Form for the ENTRY Primitive

The label will, in this case, be determined by the LOOP label previously defined. The ENTRY LABEL field should be entered exactly as in the LOOP LABEL, i.e., as "Return". Type an appropriate comment in the field provided, such as, "Entry From Loop Below". The result should be as in figure 23.

3.5 PROB, TEST, COMPARE AND BRANCH

Four other Primitives, PROB, COMPARE, BRANCH, and TEST are similar to LOOP in that they represent a branching to an ENTRY Primitive. EXAMPLE may be altered in the following four ways.

3.5.1 PROB. The PROB Primitive is used to indicate that the re-execution of the ACTION "Delay" has only a certain degree of probability.

Since AISIM has no command for directly replacing one Primitive with another, the existing Primitive LOOP must first be deleted.

Enter the command

DEL 4

where, as before, "4" indicates the location of the Primitive to be deleted. Figure 26 shows the display produced when the LOOP Primitive is deleted.

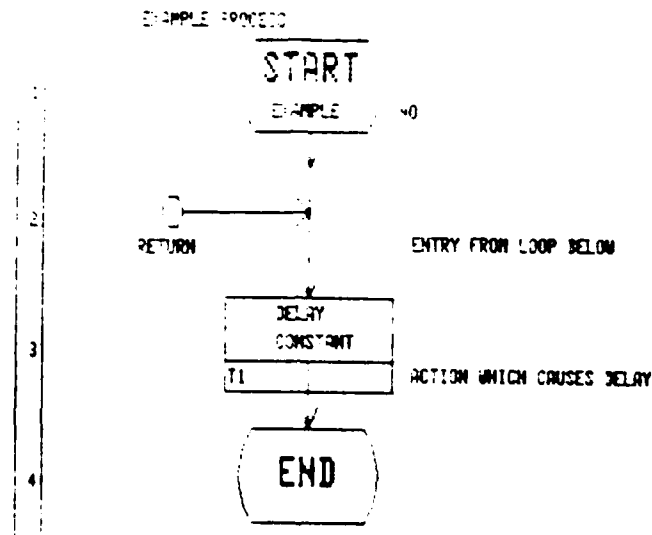


Figure 26. EXAMPLE after Deletion of LOOP Primitive

The PLACE command is used to insert a PROB Primitive between the ACTION "Delay" and the END symbol. Type

P PROB

The screen will offer the form shown in figure 27.

PARAMETERS FOR PROBABILISTIC BRANCH:

BRANCH TO LABEL: [REDACTED]

PROBABILITY OF BRANCH: [REDACTED]

COMMENT: [REDACTED]

Figure 27. Form for PROB Primitive

Complete the first field with "Return". The second field should be filled in with the probability of branching, given as a percentage. Suppose there is a 25% chance of branching. Type the appropriate comment, "25% chance of branching". The resulting display diagram is given in figure 28.

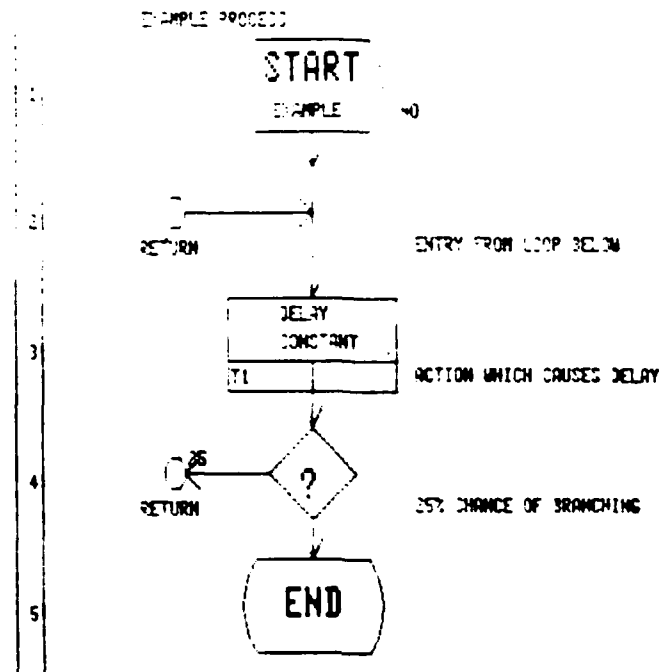


Figure 28. Process with Probabilistic Branch

3.5.2 TEST. Another kind of branching Primitive is TEST. As mentioned earlier, Processes often make use of Resources for which there is competition. The TEST Primitive represents the procedure of ascertaining the availability of a given Resource and branching if that Resource is not available, or continuing if it is. This Primitive does not allocate the Resource. To place the TEST Primitive (after having deleted the PROB Primitive from the latest version of EXAMPLE), type

P TEST

The screen will display the form shown in figure 29.

PARAMETERS FOR TEST:

RESOURCE NAME: [REDACTED]

BRANCH TO LABEL [REDACTED] IF NOT AVAILABLE

COMMENT: [REDACTED]

Figure 29. Form for TEST Primitive

In the RESOURCE NAME field, type the name of the Resource whose status is to be ascertained. The LABEL is the ENTRY Primitive to branch to, COMMENT is self-explanatory. If the PROB Primitive in the previous version of EXAMPLE is replaced by TEST (as in this example), EXAMPLE will now appear as in figure 30.

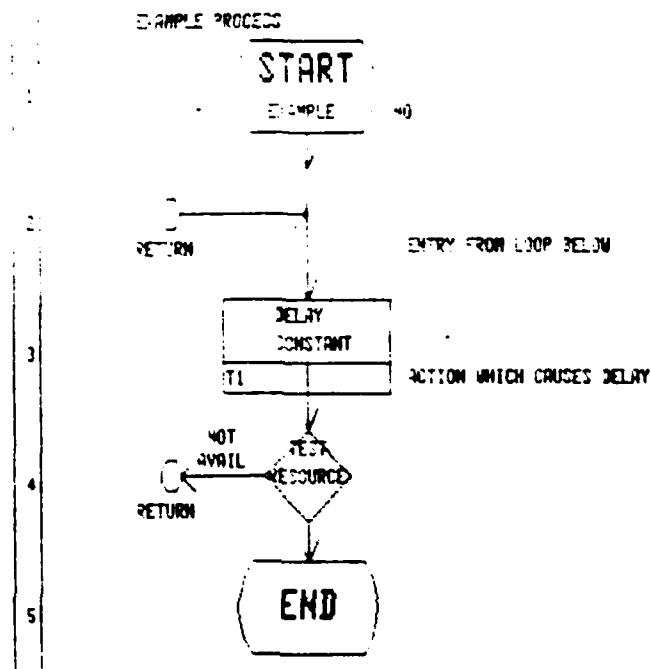


Figure 30. Process with TEST Branching

3.5.3 COMPARE In addition to probabilistic branching, AISIM also allows for conditional branching less specialized than the TEST Primitive. Most of these branchings will require the COMPARE Primitive. The COMPARE Primitive compares two numerical or alphanumeric values with respect to some relation and branches to a named ENTRY Primitive if the relation holds.

To place the COMPARE Primitive, delete the previously defined TEST Primitive and type

P COMPARE

The screen will display the form depicted in figure 31.

PARAMETERS FOR COMPARE

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

RELATION: [REDACTED]

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

BRANCH TO: [REDACTED]

COMMENT: [REDACTED]

Figure 31. Form for COMPARE Primitive

The fields OPERAND 1 and OPERAND 2 hold the parameters whose values are to be compared. The values may be represented by arbitrarily chosen names of variables (such as Var1 and Var2). They are compared with respect to the following six arithmetical relations indicated by the two letter code:

EQ for "equal to"

NE for "not equal to"

GT for "greater than"

LT for "less than"

GE for "greater than or equal to"

LE for "less than or equal to"

The BRANCH TO and COMMENT labels are now self-explanatory. The two QUALIFIER fields serve several purposes, the most important of which is to allow the comparison of attributes of entities as opposed to simple variables or numerics. The user should for the present disregard the complication posed by these fields and leave them empty. Fill in the OPERAND fields with arbitrarily chosen names of variables, "Var1" and "Var2". If the TEST Primitive is replaced by the COMPARE Primitive with the foregoing information entered on its form, the new version of EXAMPLE will be as displayed in figure 32.

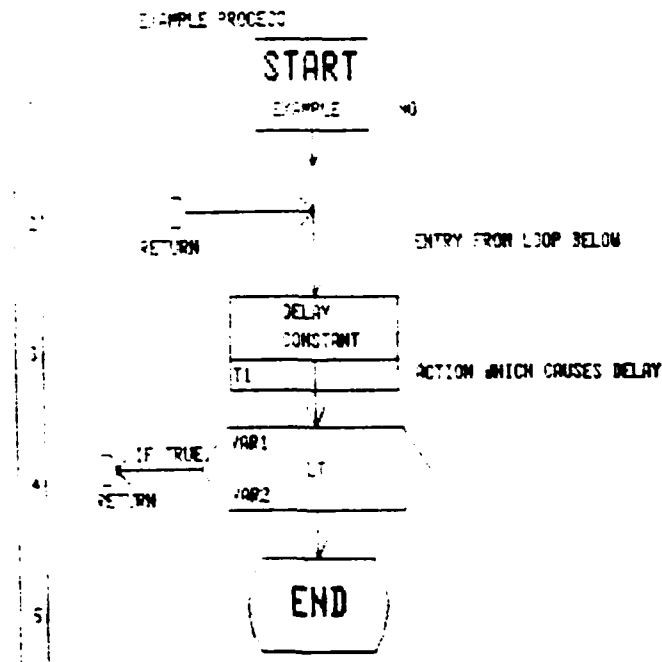


Figure 32. Process with COMPARE Primitive

And thus EXAMPLE is set to return control to the ENTRY Primitive if the value assigned to the variable Var1 is less than the value assigned to the variable Var2. These assignments are presumed to be made elsewhere.

3.6 VARIABLE MANIPULATION

In the previous example of the COMPARE Primitive, note that if the condition solicited is true, i.e., if VAR1 was less than Var2, EXAMPLE would perform the ACTION "Delay" indefinitely. On each occasion in which the comparison is made, the relation will hold and hence the Process will always be instructed to branch to Return. If neither variable changes its value, the Process will continue until it is halted by other causes (such as having a Resource necessary to it allocated elsewhere).

Using two new Primitives, ASSIGN and EVAL, we can alter EXAMPLE so that the ACTION "Delay" does not go on forever but only for a certain maximum time ("Maxtime"). This is accomplished with the ASSIGN Primitive which introduces a new variable for the accumulation of time consumed by the ACTION's execution times and by the EVAL Primitive, which recalculates the value of this accumulated time each time the ACTION is performed.

First, to command AISIM to place an ASSIGN Primitive between the START and the ENTRY, type,

P ASSIGN,2

The screen will now show the form displayed in figure 33.

PARAMETERS FOR ASSIGN

V1: [REDACTED] Q1: [REDACTED]

TO

V2: [REDACTED] Q2: [REDACTED]

COMMENT: [REDACTED]

Figure 33. Form for ASSIGN Primitive

For this example disregard the fields labeled Q1 and Q2; they serve the same purpose as do the QUALIFIER fields in the COMPARE Primitive. The purpose of this exercise is to create a temporarily useful, local variable, which we shall call "Acctime" whose value represents the amount of time that has been consumed in the repeated execution of "Delay". At the beginning of the Process the initial value of the variable will be zero. Hence, complete the V1 field with "Acctime" and the V2 field with "0". When this information is entered, the screen will display the graphic representation shown in figure 34.

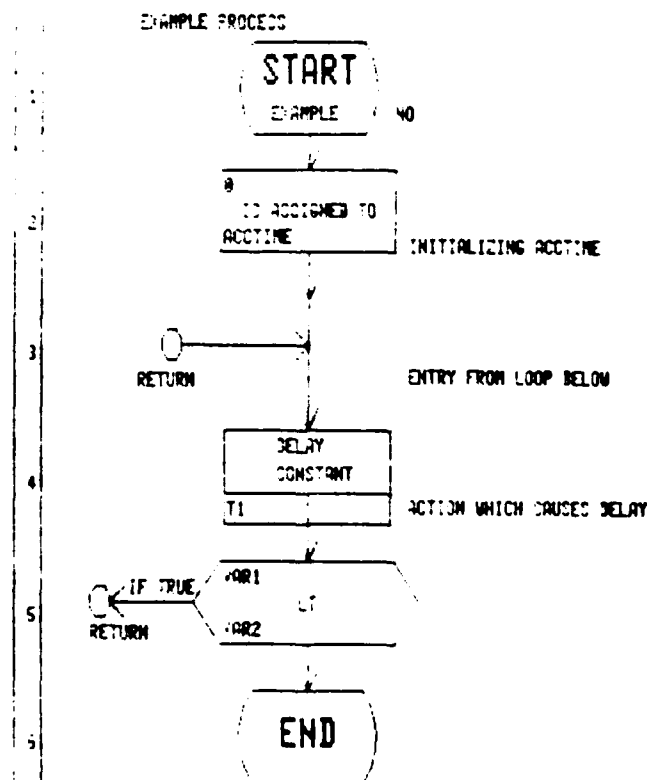


Figure 34. Process with Primitives ASSIGN, ACTION, and COMPARE

To provide an apparatus for updating the variable "Acctime" on each occasion of the ACTION's execution, an EVAL Primitive must be placed

between the ACTION and COMPARE Primitives. To do this, type

P EVAL,5

The screen will display the form shown in figure 35.

PARAMETERS FOR EVALUATE

SET VARIABLE: [REDACTED] FUNCTION: [REDACTED]

OPERAND1 [REDACTED] OPERAND2: [REDACTED]

COMMENT: [REDACTED]

Figure 35. Form for EVAL Primitive

The SET VARIABLE field holds the name of the variable whose value is to be calculated. The FUNCTION field contains the name of the operation to be performed on the two operands contained in the fields OPERAND1 and OPERAND2. A large variety of functional operations are available for this field (see AISIM User's Manual, section 3.9.11 for a list). For this example, the SET VARIABLE field should be "Acctime"; the Function, "Add"; OPERAND1, "T1" and OPERAND2 "Acctime". Type an appropriate comment, such as "Evaluating Acctime". The graphic representation of EXAMPLE will be as shown in figure 36.

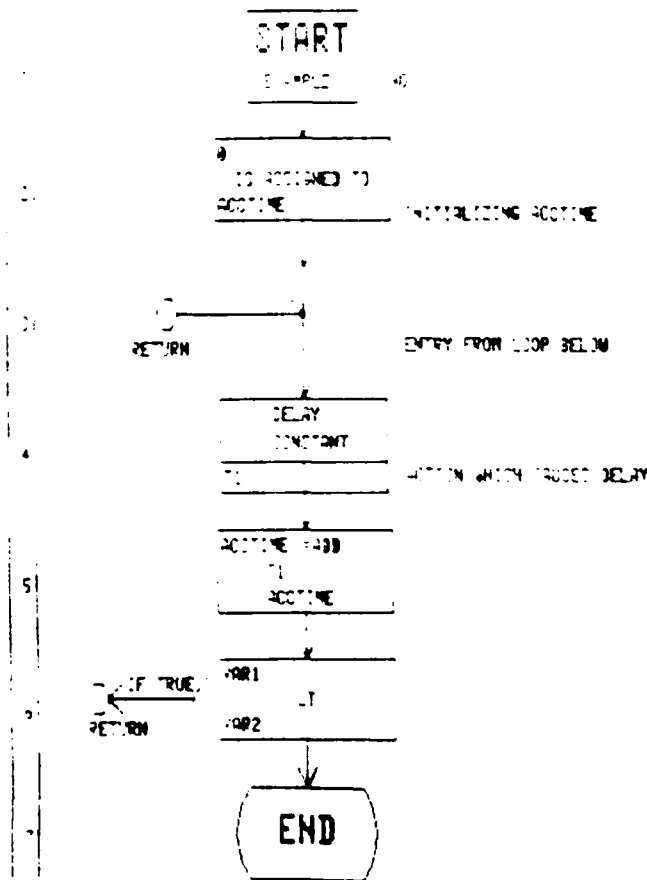


Figure 36. Process with ASSIGN, ACTION, COMPARE, and EVAL

This Primitive adds the current value of the variable "T1" to the value of "Acctime", producing an updated figure for the total time consumed by "Delay". Type an appropriate comment such as "Updating Accumulated Time".

The Process still requires alteration. The variables presently in the COMPARE Primitive must be changed from Var1 and Var2, respectively, to "Acctime" and "Maxtime". To do this, we must edit the COMPARE Primitive by typing

C 6 (cr)

This command tells AISIM that you wish to alter one or more of the previously defined parameters in the Primitive at location 6. The screen will display the form for the Primitive. It can be altered simply by writing over the existing information. When this is done and the form is "entered", EXAMPLE will satisfy the specifications for its alteration. Its graphic representation will be as in figure 37.

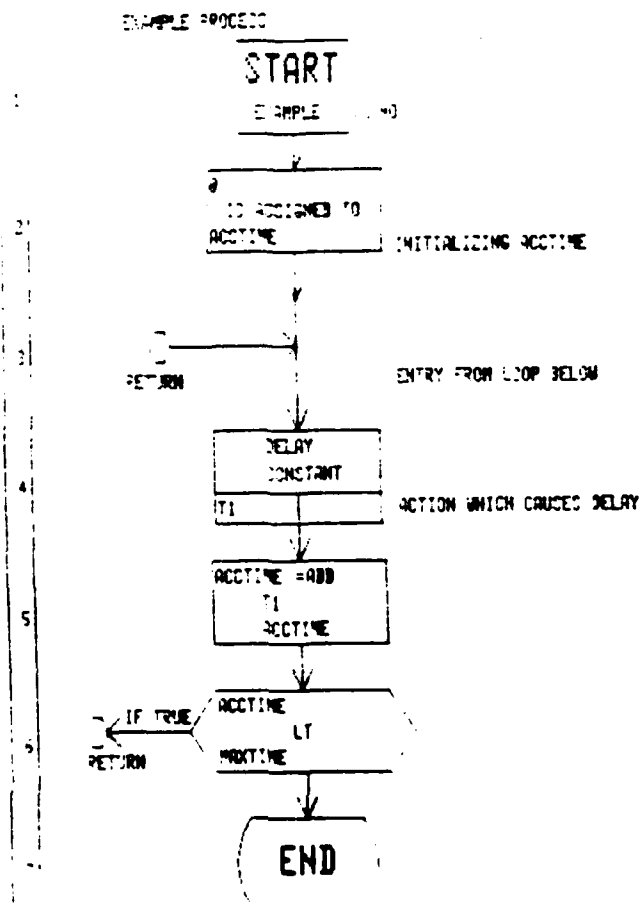


Figure 37. Process with Comparative Branching

3.7 ITEM MANIPULATION

Another group of Primitives is categorized under the headings Queue Handling and Item Handling. These include CREATE, DESTROY, FILE, FIND and REMOVE. The Primitives CREATE and FILE will be used in this example. (Consult the AISIM User's Manual for information on the Primitives DESTROY, FIND and REMOVE.)

Consider the first version of EXAMPLE which consisted of the single ACTION Primitive "Delay". Suppose now that we conceive of EXAMPLE as a Process which gives rise to new data elements--messages, information, potential communications. This function of the Process may be represented by means of the CREATE Primitive, which represents the introduction of Items--the AISIM modeling entity that represents transient data elements--into the modeled system. To place the Primitive CREATE below the ACTION Delay in EXAMPLE type

P CREATE

The form for CREATE is shown in figure 38.

PARAMETERS FOR CREATE

ITEMS TO BE CREATED ARE:

██████████ ██████████ ██████████ ██████████ ██████████ ██████████

COMMENT:

██

Figure 38. Form for CREATE Primitive

Complete this form with the names of the Items to be created. Enter the Item name "Msg" and an appropriate comment, "Transient Data Element".
EXAMPLE will now appear as indicated in figure 39.

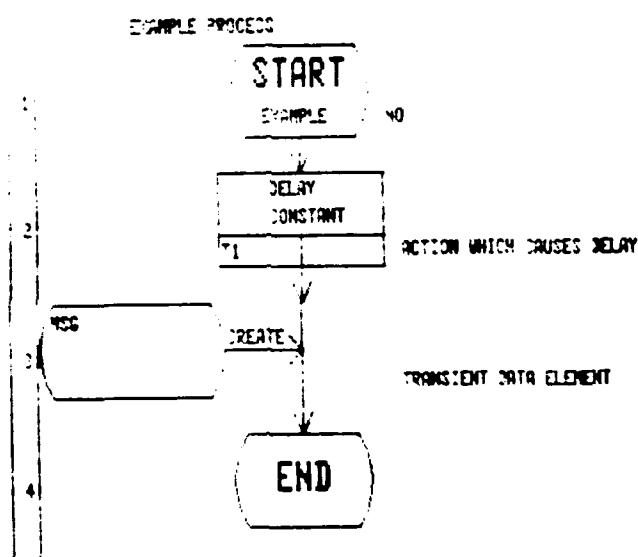


Figure 39. Item Creating Process

Items--transient data elements--can also be filed in holding areas called Queues with the FILE Primitive. To place a FILE Primitive below the CREATE Primitive in EXAMPLE, type

P FILE

The form for FILE is as shown in figure 40.

PARAMETERS FOR FILE:

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

COMMENT: [REDACTED]

Figure 40. Form for FILE Primitive

Complete the field FILE ITEM NAME with "Msg". The OPTION field tells where in the Queue the Item is to be placed. This location is specified relative either to absolute locations on the Queue ("FIRST" and "LAST") or relative to some other Item already on the Queue ("BEFORE" and "NEXT")*. The OPTION field will have as a default parameter LAST. In the ON QUEUE field enter "Msg-que". The graphic representation of this Process is indicated in figure 41.

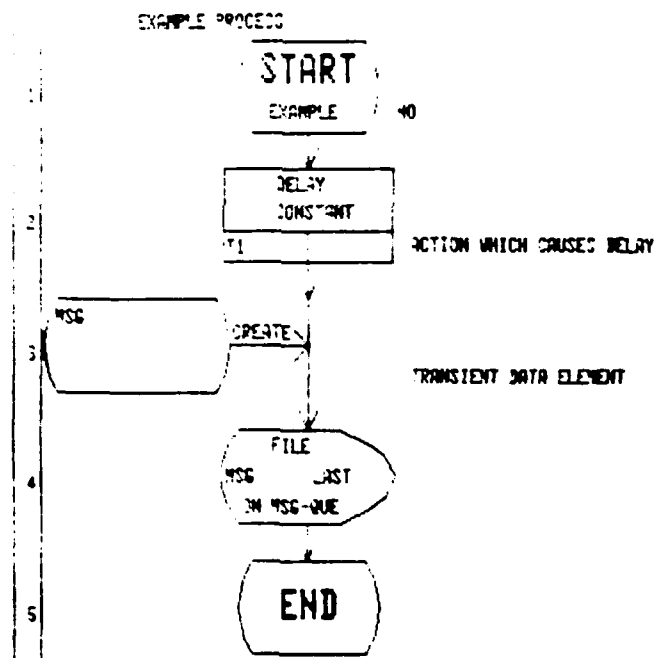


Figure 41. Process which Creates and Files Message Items

*The method by which the system identifies the Item relative to which other Item is to be placed on a Queue (with the OPTION BEFORE or NEXT) need not concern us here. For more on this, see AISIM User's Manual, section 3.9.12.

3.8 RELATIONS AMONG PROCESSES

This section deals with the relationships that the execution of Processes bear to one another in an AISIM Model, and how one Process, and its execution, affects the execution of another. Processes affect one another's execution in four ways:

1. By sending Items that trigger the execution of another Process.
2. By triggering the execution of another Process through a CALL Primitive where the CALL may or may not pass parameters.
3. By competing for and obtaining use of a Resource needed by another Process.
4. By resuming (with the RESUME Primitive) a Process which at some time suspended itself.

To understand how parameter and Item "passing" affect the execution of a Process, consider the form completed in the first version of EXAMPLE. In the form presented as a result of the command to edit a Process (i.e., E PROCESS,EXAMPLE,NEW), in the START field TYPE, the choices included "STD", "PARM" and "ITEM", standing for, respectively, "standard", "parameter-passing" and "Item-passing". These options are distinguished from one another in the following way. A Process can, before it is fully designed, be thought of as a "black box" whose internal workings are unknown. If the Process is conceived to be one that performs its function without having to be given anything in the way of information or data elements it will be a standard Process. If the Process requires certain data elements--discrete, countable entities--in order to execute, then it is an "Item-passing" Process. Finally, if the Process uses values of variables local to another Process, it is a parameter-passing Process.

For the first example, consider Item Passing Processes. In this exercise, delete the FILE and CREATE Primitives from EXAMPLE. To change EXAMPLE from a standard Process, as it now is, to an Item-passing Process, type

C 1

The screen will display the form originally filled out for EXAMPLE. Type "ITEM" over the existing "STD" in START field TYPE. Entering this, the screen will now show this secondary form on which Items needed by this Process are to be typed, as shown in figure 42.

ITEM PASSING START

ITEMS RECEIVED:

--	--	--

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

--

Figure 42. Secondary Form for Process

Type the single Item name "Msg" in the upper left field and type "N" in the match field. Entering this changes the Process representation so that it appears as in figure 43.

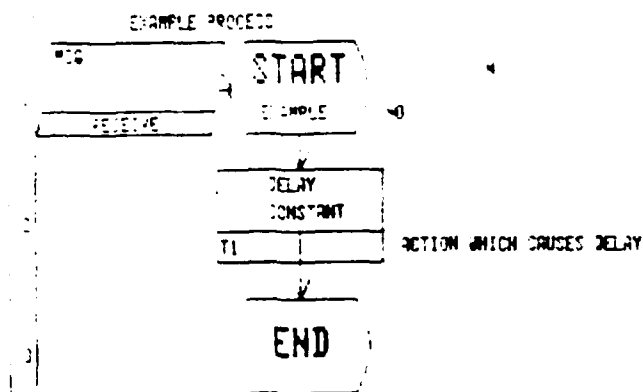


Figure 43. Item Passing Process

The figure to the left of the START figure indicates that the Process starts when, and only when, the Item MSG is delivered to it from some other Process.

None of the Primitives in the categories Item Handling and Queue Manipulation represents the delivery of Items to a Process. This delivery function is accomplished by the SEND. To exemplify SEND, a new Process, called "EXAMP-2", must be created. EXAMP-2 triggers the execution of EXAMPLE by delivering Items to it. For this example consider a Process identical except in name to the original EXAMPLE with the single ACTION Delay as depicted in figure 44.

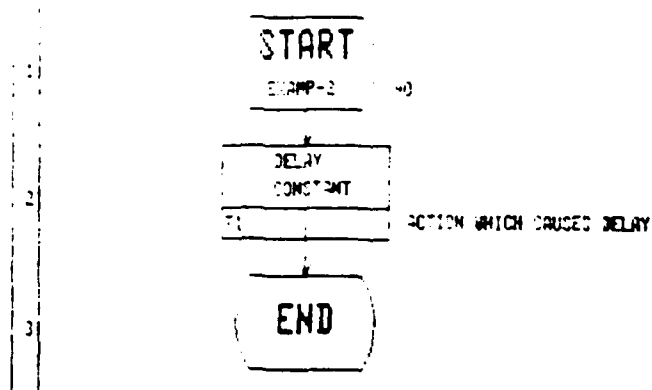


Figure 44. Process with ACTION Primitive

Type the command

P SEND

The screen will display the form shown in figure 45.

PARAMETERS FOR SEND

SEND ITEMS TO [REDACTED]

ITEMS TO BE SENT ARE:

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

COMMENT: [REDACTED]

Figure 45. Form for SEND Primitive

Complete the SEND ITEMS TO field with "Example". In the first field of ITEMS TO BE SENT, type "Msg". Enter the comment "Sending Messge Item". Figure 46 shows the graphic representation of the Process that will appear on the screen.

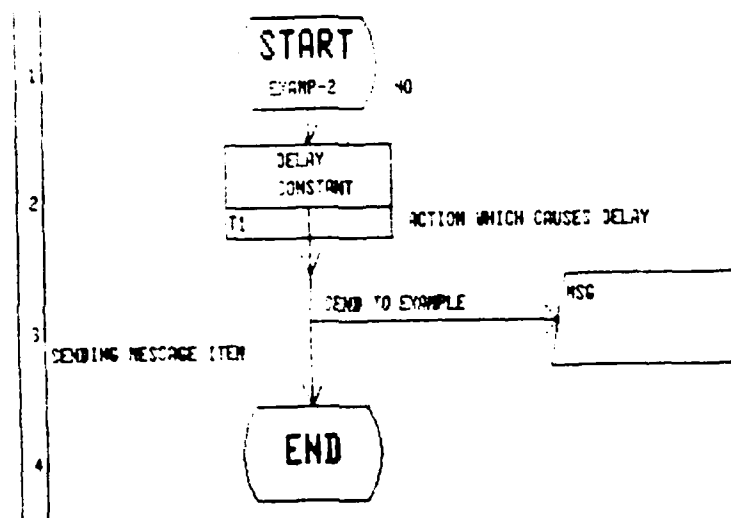


Figure 46. Graphic Representation of EXAMP-2

EXAMP-2 now triggers EXAMPLE by delivering to it Items required for its execution. The Item is automatically created by the SEND Primitive. An Item-passing Process may only be initiated through the SEND Primitive in some other Process, although the Items needed, and hence the Items sent, may be distributed among several Processes or several stages of a single Process.

3.9 RESOURCE ALLOCATION

As mentioned earlier, Processes in an AISIM model frequently make use of Resources. A Resource has a finite capacity which will limit the number of Processes it can accommodate at the same time. The five Primitives which relate to the allocation of such Resources are ALLOC, DEALLOC, RESET, LOCK and UNLOCK.

ALLOC and DEALLOC signal the allocation and deallocation of a Resource by the Process in which they appear. To place the ALLOC Primitive above the ACTION Primitive in EXAMPLE, type

P ALLOC,2

To place a DEALLOC Primitive just above the SEND symbol in EXAMP-2, type

P DEALLOC,4

The forms for these two Primitives are shown below in figure 47.

PARAMETERS FOR ALLOCATE:

ALLOCATE RESOURCE NAME:

NUMBER OF UNITS REQUESTED:

PARTIAL/ALL ALLOCATION:

ALLOCATION PRIORITY:

COMMENT:

PARAMETERS FOR DEALLOCATE:

DEALLOCATE RESOURCE NAME:

NUMBER OF UNITS DEALLOCATED:

COMMENT:

Figure 47. Forms for Primitives ALLOC and DEALLOC

In each case, enter the name of the Resource to be allocated or deallocated, such as "Cpu", in the field provided. The field NUMBER OF UNITS REQUESTED holds the number of units of this Resource to be allocated or released. The PARTIAL/ALL field specifies the type of allocation scheme. Partial allocation will allocate Resources as they become available. All allocation means all of the units must be available at once. The ALLOCATION PRIORITY field specifies the priority at which Resources are allocated. Enter "1" for number of units and "\$Priority" for priority. "\$Priority" means to use the priority the Process was invoked with. Enter the appropriate comment, "Obtaining Cpu" or "Releasing Cpu" in the COMMENT field.

Placing these Primitives in EXAMP-2 (one above the ACTION and one below), produces a graphic representation like that shown in figure 48.

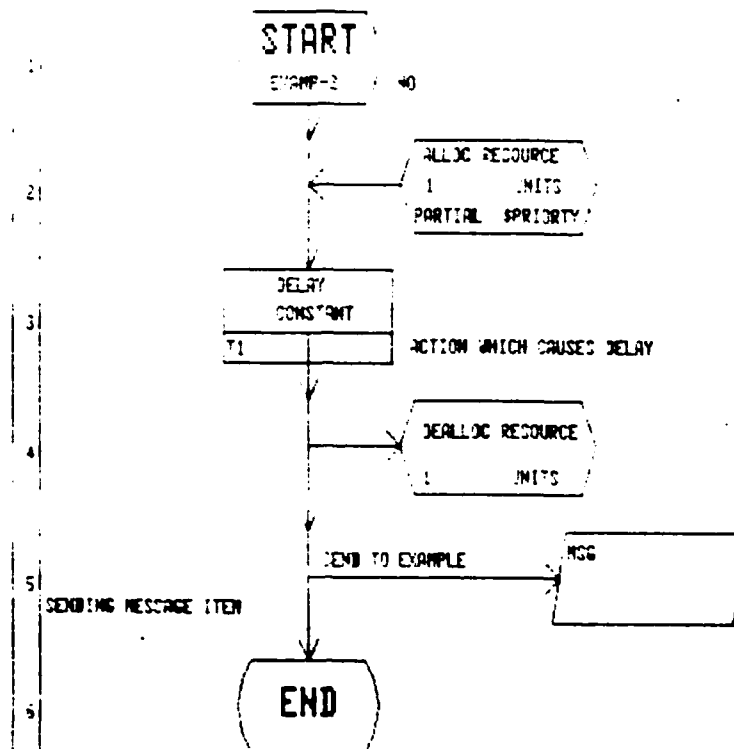


Figure 48. Process which Allocates and Deallocates a Resource

Allocating a Resource does not normally insure the uninterrupted availability of that Resource to a Process. Any Resources may be usurped by an ALLOC request with a higher priority. If the Process being modeled is one which, once begun, cannot be interrupted, the Primitives LOCK and UNLOCK must be used.

To obtain the forms for these Primitives,

P LOCK,*n*

or

P UNLOCK,*n*

where *n* is the position in the Process where the Primitive is to be placed. The forms for these Primitives are shown in figure 49.

PARAMETERS FOR LOCK:

COMMENT: [REDACTED]

PARAMETERS FOR UNLOCK:

COMMENT: [REDACTED]

Figure 49. Forms for Primitives LOCK and UNLOCK

These Primitives, if placed above the ALLOC Primitive and below the DEALLOC Primitive in EXAMP-2 would give a graphic representation like that shown in figure 50.

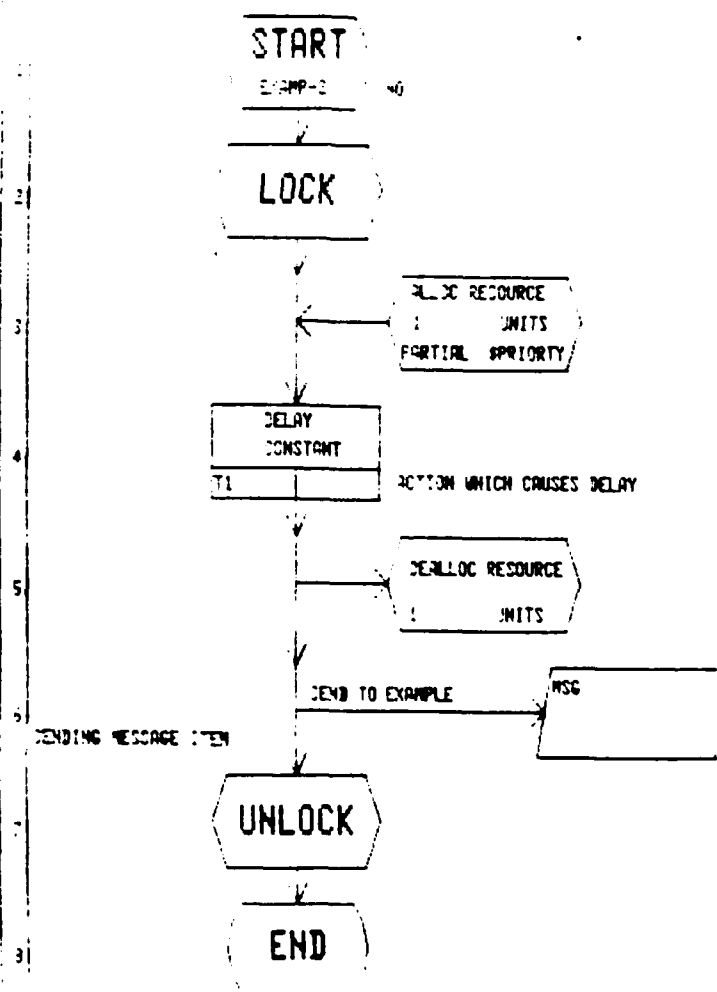


Figure 50. Process with Protected Resources

One final way to affect a Resource is through the RESET Primitive. It is used to reset the capacity of a Resource, where "capacity" is a measure of the number of Processes it will accommodate (support) at one time. For details on its use, see the AISIM User's Manual, section 3.9.18.

3.10 CALL

The function of the CALL Primitive is similar to that of the SEND Primitive, but whereas the SEND Primitive triggers Item-passing Processes, the CALL Primitive triggers both standard Processes and parameter-passing Processes. Thus, to understand how CALL works requires a brief discussion of parameter-passing Processes.

A parameter-passing Process is one that is "given" values for input variables and "returns" values for output variables. To create a parameter-passing Process, one would type "PARM" in the field START TYPE in the original form for Process. Entering this information on the Process form yields the secondary form shown in figure 51.

PARAMETERS FOR PASSING START

GIVEN:

--	--	--

RETURN:

--	--	--

Figure 51. Secondary Form for Parameter-Passing Process

On the form in figure 51, the user types the variables whose values are passed to the Process and the variables whose values are passed back.

The CALL Primitive values, i.e., parameters, are passed (given) to a called Process and returned to the calling Process. Parameter passing can occur only through the use of a CALL Primitive. A CALL Primitive is placed in a Process by typing

P CALL

The form for CALL is shown in figure 52.

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]

Figure 52. Form for CALL Primitive

The field CALLED-PROCESS NAME asks for the name of the Process to be triggered. The field PRIORITY determines the priority associated with the called Process which may be used in cases of Resource contention. (It can be overridden by the priority specified in an ALLOC Primitive.) The GIVEN and RETURNS fields hold the local variables whose values are passed to and from the called Process. A CALL Primitive may trigger a standard Process and hence these fields may be empty. The COMMENT field is self-explanatory. The field labeled WAIT/NOWAIT/BLOCK determines whether the calling Process will wait for the called Process before continuing execution or will continue to execute independently of it. The reader is referred to the AISIM User's Manual for details on their use.

4. REMAINING MODEL ELEMENTS

Although the Processes and the Architecture are core modeling elements, their specification does not complete the task of model construction. They must be supplemented by definitions of other elements. These elements are grouped into two categories. The first category consists of those entities explicitly referred to in Processes, namely, Actions, Constants, (global) Variables, Tables, Queues and Resources. The second category consists of the two entities that are used to represent the impact of the environment on the modeled system. All these remaining entities are defined at the DUI level of AISIM operation.

The following two sections briefly describe the parameters, significance and principle commands associated with these remaining entities.

4.1 ACTIONS

Any ACTION Primitive placed in a Process must have a corresponding Action entity defined outside the Process. Such a definition is created by typing

E ACTION,ACTION NAME,NEW

The form for the Action entity is shown in figure 53.

ACTION: [REDACTED]

CLASS: [REDACTED]

DESCRIPTION: [REDACTED]

Figure 53. Form For Action Entity

The ACTION field should hold the name of an Action referenced in some ACTION Primitive. The CLASS is an optional parameter for the user to provide a categorization--man, machine, etc.--of the sort of activity the Action represents. It functions as a second comment field. This field does not affect AISIM's operation and may be left blank. The field DESCRIPTION is for any convenient reminder of what the Action represents. It can be the same as the description of the corresponding ACTION Primitive.

4.2 RESOURCES

As mentioned earlier, any Resource mentioned in a Process--through the ALLOC, DEALLOC, or RESET Primitives--must be defined separately in the DUI. To create a new Resource, type

E RESOURCE,NAME,NEW

The screen will display the form shown in figure 54.

RESOURCE NAME: [REDACTED]

TOTAL NUMBER OF UNITS: [REDACTED]

INITIAL NUMBER OF UNITS: [REDACTED]

DESCRIPTION: [REDACTED]

ATTRIBUTES

NAME	VALUE	NAME	VALUE
[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]

Figure 54. Form For Resource Entity

Complete the first field, RESOURCE NAME, with the name by which it is referred in any Process. The fields TOTAL NUMBER OF UNITS and INITIAL NUMBER OF UNITS indicate, respectively, the maximum number of Processes the Resource can accommodate at any one time and the number of Processes it can accommodate at the beginning of a simulation run (i.e., before being increased or decreased by the RESET Primitive). Enter the appropriate numbers. DESCRIPTION has its usual function. Type an appropriate description in the field provided. The fields under ATTRIBUTES indicate whether the Resource has associated with it attributes, including default attribute COST.

Up to fifteen attribute names may be entered with their initial values. The attribute COST functions as any other Resource attribute.

Though all Resources referred to require separate definitions, some Resources are defined automatically. For each node or link created in a model's network architecture, a Resource definition of the same name with default parameters is automatically written into the database. In other words, all nodes and links are identified with Resources. Thus, after an architecture has been created, the command

```
nodename
E RESOURCE,
linkname
```

can be issued without having to indicate that the Resource entity is new (with "NEW"). Typically, however, not all of a system's Resources will

be represented in the architecture and not all of the Resources automatically created in ADE will have any positive role in the operation of the model. That is, such automatically defined Resources need not be invoked in the Process Primitives. Importantly, if an operative Resource is to be identified with an architectural element, it should be defined first in ADE and thereafter edited to provide it with suitable parameters (on the assumption that the default parameters are incorrect). ADE will not allow the definition of a node or link whose name is identical with that of a Resource already in existence.

4.3 QUEUES

Not all the Queues functioning in a system model need be defined by the user, since many are implicit in the operation of the system. The general rule is that any Queue manipulated by the FILE, FIND or REMOVE Primitives must be given a separate definition in the DUI, with the exception of a cross-reference set.

The cross-reference sets are explained in the AISIM User's Manual, section 3.5.2.

To define a new Queue, type

E QUEUE,NAME,NEW

The form for this entity is shown in figure 55.

QUEUE: SIZE:
DESCR:

Figure 55. Form for Queue Entity

The three fields should be filled in with, respectively, (a) the name of the Queue as found in the FILE, FIND, or REMOVE Primitive which uses the Queue, (b) the maximum number of Items that can be placed in it (the default value for which is "infinite") and (c) any useful reminder of the Queue's role in the modeled system.

4.4 CONSTANTS AND VARIABLES

Constants differ from global Variables only in that they do not change their values during the simulation exercise of a model. Once a value has been assigned to a Constant and a simulation is begun, its value is unchanging. Accidental attempts to alter the value of a Constant through the EVAL or ASSIGN Primitives will yeild an execution error

message. The forms for Constants and Variables are quite similar and are called up by issuing the command EDIT CONSTANT or EDIT VARIABLE and then giving the name of the Constant or Variable.

The forms for Constants and Variables are shown in figure 56.

```
CONSTANT: [REDACTED]
VALUE: [REDACTED]
DESCRIPTION: [REDACTED]

VARIABLE: [REDACTED]
VALUE: [REDACTED]
DESCRIPTION: [REDACTED]
```

Figure 56. Forms for Constant And Variable

The fields CONSTANT and VARIABLE call for the entities' names. The VALUE fields call for numerical values for Constants and alpha-numerical values for Variables, and the DESCRIPTION fields call for any description.

4.5 LOADS AND SCENARIOS

The effect of the environment on a model is represented collectively by Loads and Scenarios. The relationship between Loads and Scenarios is this: Loads specify a number of Process triggerings to take place sometime during the simulation exercise of a model. Loads do not specify when the Process triggerings are to take place. They specify the distribution of the time passing between triggerings of the Process. Scenarios specify a collection of Loads and/or individual Processes together with a schedule indicating when the specified Loads or Processes are to be initiated.

To define a Load, type

```
E LOAD,NAME,NEW
```

The form for the Load Entity is shown in figure 57.

LOAD: [REDACTED]					
NODE1	NODE2	NODE3	NODE4		
[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]		
NODE5	NODE6	NODE7	NODE8		
[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]		
DESCR: [REDACTED]					
PROCESS	MAX #	SCHMDT	MEAN	DELTA	PRIORITY
[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]

Figure 57. Form for Load Entity

The LOAD field holds the name of the Load. The fields labeled NODE1 through NODE8 indicate the architectural nodes in which the Processes named in the Load take place. The field DESCR is for any helpful description.

The field labeled PROCESS holds up to five names of Processes. The fields SCHMDT, MEAN and DELTA together define the statistical method of distribution to be used in scheduling the Process triggerings. SCHMDT holds the name of the distribution method, MEAN holds the average time between Process initiations (in terms of the simulation clock) and DELTA is a second numerical parameter used to specify variation about the mean, if applicable. The field MAX # indicates the maximum number of Process instances to be initiated by this Load.

The Scenario entity defines the impact of the environment on the system for the entire simulation exercise of a model. In it the user specifies a number of "periods" into which a simulation exercise is to be divided, together with a uniform length each period is to have. The user then specifies a collection of Loads or Processes to be initiated at a specified time during the simulation. A priority is also given for each Process.

To define a Scenario, type

E SCENARIO,NAME,NEW

The form for the Scenario entity is shown in figure 58.

5. A WORKING EXAMPLE

This section documents the construction of an AISIM model that can be run through simulation tests and analyzed in the subsequent chapter. The model will be a representation of the transmitter/receiver relationship, an element of any communication system.

The transmitter/receiver relation modeled here is of the "polling" or "mailbox" type, as opposed to the "interrupt" type. In it, one transmitting Process generates messages and delivers them to a buffer. There the messages await treatment from a receiving Process. The transmitting and receiving Processes are not in direct communication with one another. Rather, the transmitter broadcasts messages according to need, and the receiving Process reads them from the buffer at intervals in accordance with expected need. In the system envisioned, transmission is randomized in two respects, (1) in the lengths of transmitted messages and (2) in the intervals between transmission. Reception is undertaken at regular intervals and the time consumed in processing a message is a linear function of its length.

The origination of a message in the transmitting Process will be represented by the creation of an Item (through the CREATE Primitive). The Item will have a variable attribute which will represent its length. Since the length will be randomized over a range of approximately 700 bytes, some mechanism must be incorporated for altering the variable attribute of each data Item (i.e., message). This is accomplished by (1) generating a random number in the range $[0,1]$ subsequent to the creation of each Item, (2) multiplying the random number by twice the average message length and (3) assigning the number so obtained to the message length. This figure will then be used to calculate the time taken to send the message to the buffer (where it will be available to the receiving Process). Through an ACTION Primitive, the clock is then updated by the amount calculated.

In this system the buffer will not be manipulated by both the receiving and the transmitting Processes at the same time, so the buffer is considered a Resource and its allocation and deallocation by the ALLOC and DEALLOC Primitives will prevent it from being accessed simultaneously by both Processes.

5.1 DEFINING PROCESSES

This description of the transmitting Process gives the steps of its execution. The transmitting Process:

- (1) Starts
- (2) Allocates one unit of a Resource BUF1 representing the buffer
- (3) Creates a message, represented as an Item called "Msg"
- (4) Generates a random number between 0 and 1

- (5) Multiplies the random number generated by twice the average message length
- (6) Assigns the number obtained in the previous step to the Item attribute representing the message length
- (7) Calculates the delay time which is proportional to the message length (i.e., an amount equal to the message length divided by the transmission rate in seconds per byte)
- (8) Delay for the calculated amount of time
- (9) Delivers the message Item to the Queue called Buffer through the FILE Primitive
- (10) Releases the Resource BUF1 representing the buffer

Figure 59 shows the Process flowchart derived from this description.

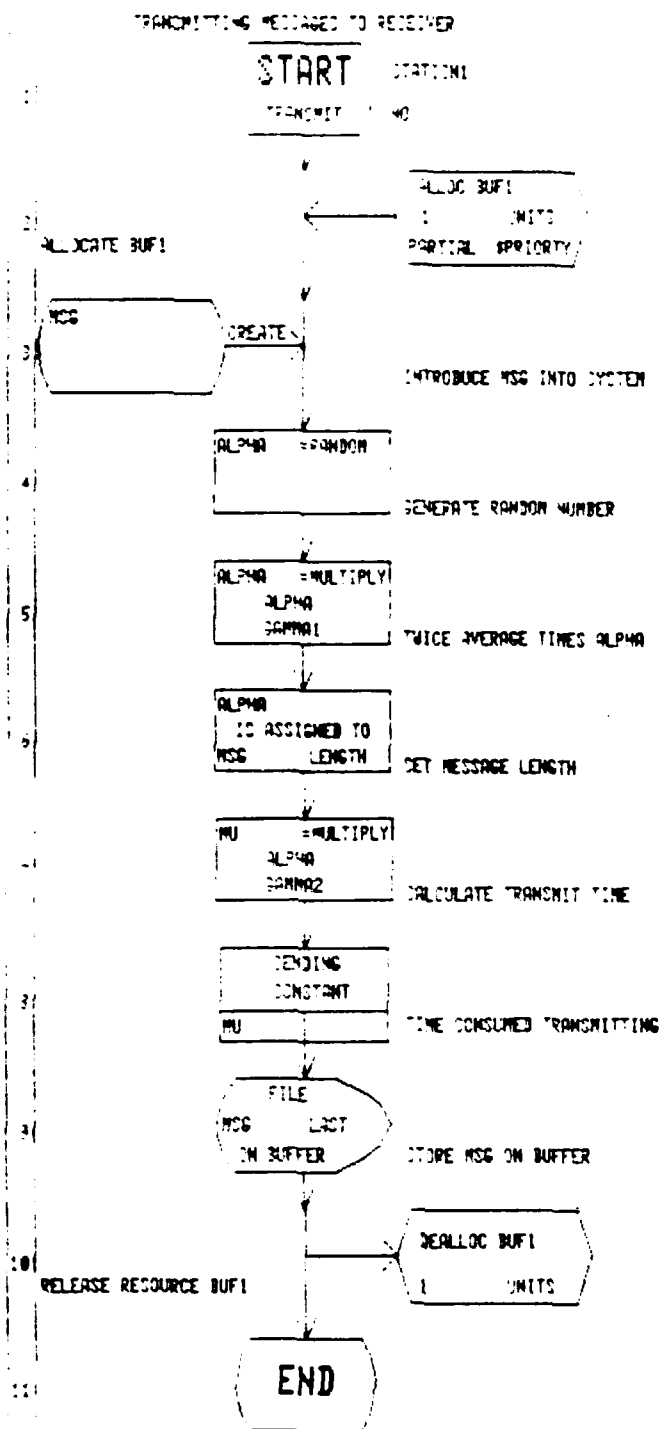


Figure 59. Transmitting Process

The receiving Process will first determine whether or not the buffer is being manipulated by the other Process by testing for utilization of the Resource call BUF1. If the Resource is in use, the Process will abort by branching to the END symbol. If BUF1 is free, the Process will read

the next message from the buffer, and calculate a receiving time in roughly the same way that the transmitting time for that same Item was determined in the transmitting Process. The clock is then updated by the amount of time calculated.

This description can be expanded into more specific design requirements. The receiving Process

- (1) Starts
- (2) Tests for the availability of the buffer by determining whether or not the Resource is in use through the TEST Primitive. If so, the Processes execution will branch to the END symbol.
- (3) The next message Item on the Queue called Buffer is read via the REMOVE Primitive.
- (4) If there is nothing on the buffer, Process execution, as in step (2), branches to the END. This step is represented by a COMPARE Primitive.
- (5) The message length is assigned to a local variable through the ASSIGN Primitive.
- (6) A receiving time is calculated to be proportional to the message length (i.e., equal to the message length times some reception speed in seconds per byte).
- (7) The clock is updated through the ACTION Primitive in the amount required to receive the message.
- (8) The message Item, having been read, is eliminated from the system through the DESTROY Primitive.
- (9) An ENTRY Primitive is inserted just before the END symbol of the Process to indicate where execution is to resume from the branchings in steps (2) and (4).

Figure 60 shows the flowchart representation of the Process derived from these requirements.

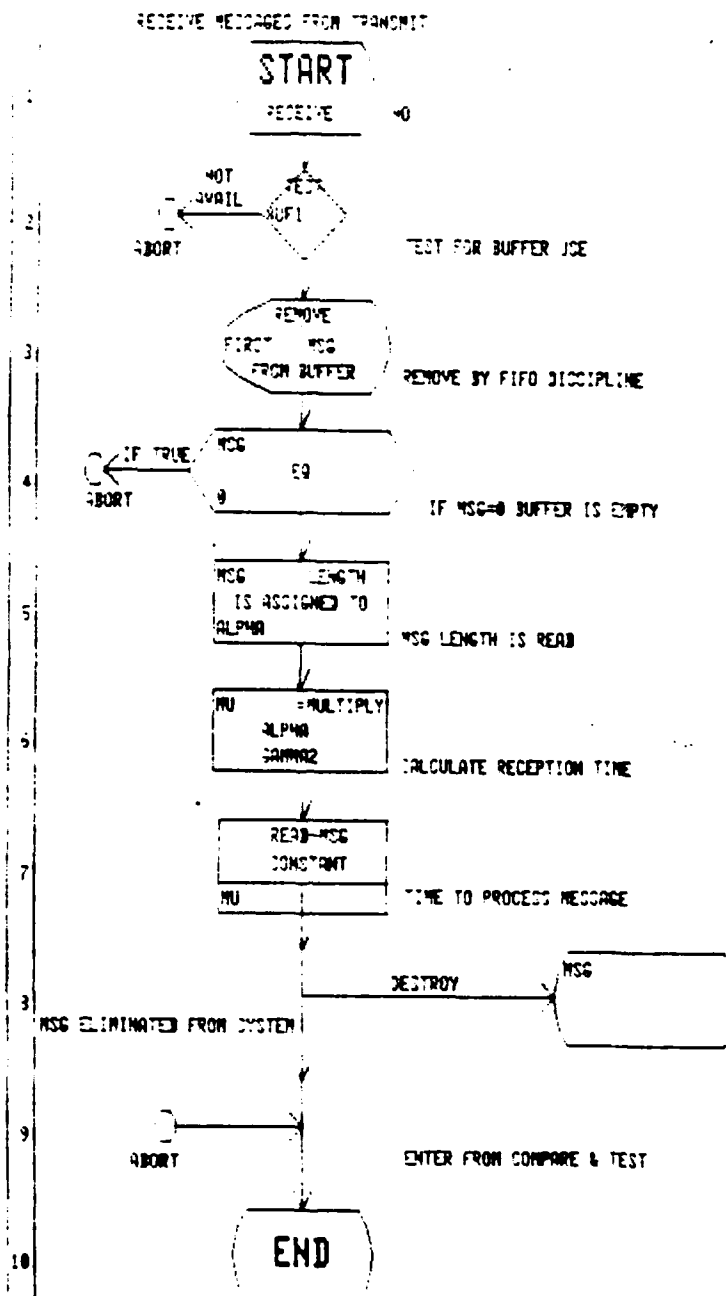


Figure 60. Receiving Process

5.2 REMAINING MODEL ELEMENTS

The remaining model entities must now be defined. These include all the entities mentioned in any Process Primitive. These are the following:

- 1) The Queue named "Buffer" onto which messages are filed;
- 2) The Resource Buf1 which represents a device to protect the buffer against manipulation by two Processes at once;
- 3) The Item Msg, each instance of which is to represent a message transmitted onto the Queue;

- 4) The global Variables named Gammal, which is twice the average message length; and Gamma2, which is the transmission rate.
- 5) The Action entities named Read-msg, which charges time for reading a message, and Sending, which charges time for sending a message.

5.2.1 RESOURCE DEFINITIONS

The Resource BUF1 is given proper parameters. It will have only one initial unit and will have a maximum of one. It will retain the default of no attributes and a cost of zero. An appropriate description is: "Resource Associated With Buffer".

5.2.2 QUEUE DEFINITIONS

The Queue called "Buffer", which is accessed by the FILE and REMOVE Primitives, will retain its default size of "Infinite". A helpful description is: "Buffer On Which Messages Are Stored".

5.2.3 ITEM DEFINITION

The Item Msg which represents messages transmitted and received will have one attribute called "Length". Its initial value will be the literal "\$Length", since the value of this attribute will always be assigned within the Process that transmits it to the buffer.

5.2.4 VARIABLE DEFINITION

The Variables "Gammal" and "Gamma2" are defined with initial values of .700 and .002 respectively. These values are used in calculating the transmission and reception time. Gammal is twice the average message length, and Gamma2 is the transmission rate.

5.2.5 ACTION DEFINITION

Action entities "Sending" and "Read-Msg" must be defined in order to satisfy the references in the ACTION Primitives in the Processes Transmit and Receive. The class and description fields can be filled in as desired by the user. These fields have no effect on the simulation.

5.3 LOADS AND SCENARIOS

Finally, we must define the hypothetical conditions to which the modeled system will be exposed. Six Loads are defined for this model. L1, L2 and L3 each trigger the transmitting Process. L11, L22 and L33 each trigger the receiving Process in a schedule of expected need associated with L1, L2 and L3. The triggerings of the transmitting Process are randomized, whereas the triggerings of the receiving Process are scheduled at regular intervals. The complete Load definitions are found on pages 3 through 5 of the Model Verification Report which appears in Appendix B.

The Scenario for this model consists of six periods. The Loads are distributed throughout the simulation period as follows: each pair of Loads is triggered at intervals of 200 units on the simulation clock. The complete definition is found on page 5 of the Model Verification Report in Appendix B.

6. SIMULATION EXERCISES OF AISIM MODELS

The model is now ready to be run through a simulation exercise to determine its behavior under the defined environmental conditions. To begin this exercise, enter the Analysis User Interface (AUI) from the AISIM READY level by typing

A P(projectname)

Projectname is the name of the model we wish to expose to a simulation exercise. The user will be prompted with information that will look something like that shown in figure 61.

```
CURRENT PARAMETERS IN EFFECT:
VERSION:      PRODUCTION VERSION 4.0
TERMINAL:     HP
PROJECT:      TEST
USER:         [USER]
ENTER YES TO PROCEED, NO TO ABORT...
```

Figure 61. Information Displayed on Entering The AUI

After declining the abort prompt by typing

YES

and following the translation of the model, the user is in a position to issue commands before the execution of the simulation.

6.1 INITIALIZING A MODEL

If more than one Scenario has been defined, the system will ask

WHICH SCENARIO DO YOU WISH TO TRANSLATE?

Type the name of the Scenario that defines the environmental conditions to which the model is to be subjected. For this model, we have defined only one Scenario so the program will perform model initialization. If no errors are detected at this stage the computer will prompt with,

NO ERRORS DETECTED DURING MODEL TRANSLATION
YOU MAY NOW ENTER COMMANDS

If an error had been made, AISIM would have prompted with

ERRORS DETECTED IN MODEL TRANSLATION

This prompt indicates that some aspect of the model definition is in error. If such is the case, determine where the errors are, see Appendix B of the AISIM User's Manual for a description of the error messages, and return to the DUI to correct them. The matter of getting to the DUI has already been covered in previous chapters. For this example, assume that the AISIM model is properly defined.

6.2 DEFINING PLOTS

Two choices are available at this point: Proceed to the simulation exercise of the model or request that graphs of some of the activities monitored during the simulation be defined so that they can later be inspected at the terminal.

For example, in the model under consideration, one of our main concerns is to determine whether the buffer onto which the transmitting Process places messages (and from which the receiving Process retrieves the messages) reaches some maximum burden or whether it shows a tendency to infinite queueing. To produce a graph of the behavior of the buffer we type

```
DEF QUEUE,BUFFER
```

The screen will display a selection of the aspects of the behavior of a Queue for which a graph can be defined. These are shown in figure 62.

ATTRIBUTES (PLACE AN X NEXT TO ONLY ONE)

```
X NUMBER IN QUEUE
NUMBER BLOCKED
TIME IN QUEUE
TIME BLOCKED
```

Figure 62. Aspects of Queue Behavior

To define a graph showing the number of Items in the Buffer, we would enter an "x" for "NUMBER IN QUEUE". The screen would then display the options for defining the type statistic on the number of Items in the Queue. These options are shown in figure 63.

STATISTICS (PLACE AN X NEXT TO ONLY ONE)

```
X CURRENT
CUMULATIVE MEAN
CUM STANDARD DEV
CUMULATIVE MIN
CUMULATIVE MAX
PERIOD MEAN
PER STANDARD DEV
PERIOD MIN
PERIOD MAX
```

Figure 63. Options for Statistics

To calculate the current number of message Items in the Queue called "Buffer" at any given time, enter an "X" next to "CURRENT". The entities with respect to which graphs can be defined are Resources, Queues, Processes, Items and Variables. Up to ten such graphs may be defined per analysis session.

6.3 STARTING THE SIMULATION

Once the model is initialized and graphs are defined, the model may be executed through a simulation run. The execution of the model may be triggered either for the entire Scenario or for a specified number of periods, so that global Variables can be given new values different from those previously defined in the DUI.

The values of Constants and the initial values of global Variables may also be changed before a simulation exercise begins. The latter option will be chosen in this example to investigate the effect of altering the time required to transmit or to process message Items. To begin the simulation, type

GO 1

This command indicates that the simulation is to be run for 1 of the simulation periods defined when the model was created in the DUI. When this first stage of the simulation is completed the screen will offer the following message:

END OF PERIOD
YOU MAY NOW ENTER COMMANDS

6.4 EDITING VARIABLES BETWEEN SIMULATION STAGES

To change the value of a variable, issue the appropriate command with information as to (a) the type of entity to be edited, (b) the name of the entity whose value is to be changed and (c) the new numerical value of the entity. The Variable Gamma2 formerly had the value of .002. To change it to .001, type

E V,gamma2,.001

The simulation may be continued for two more periods by typing

GO 2

When this stage of the simulation is completed, the value of Variables may be changed back to .002 by typing

E V,gamma2,.002

To command that the remainder of the Scenario be run through without further interruption, type the GO command without a numeric parameter, thus:

GO

If no mistakes were made in constructing the model that cause the simulation to abort, the computer will prompt, after some time, that the simulation is completed.

The plot produced by this run is shown in figure 64, and the output report appears in Appendix B.

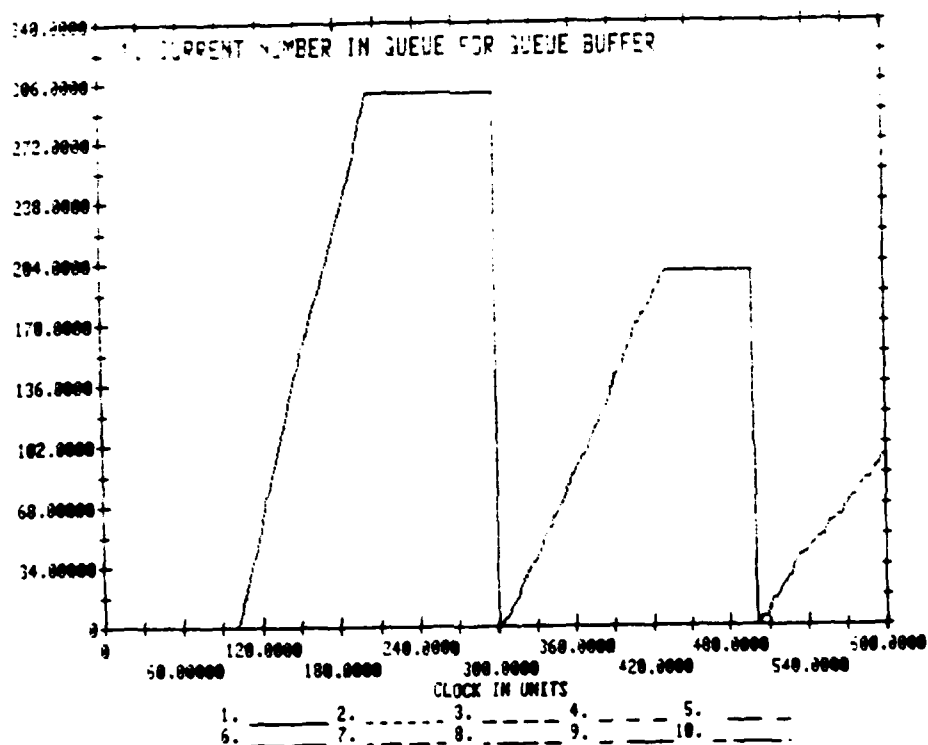


Figure 64. Plot from Simple Example

7. A MORE ELABORATE EXAMPLE

In this chapter a communication system slightly more complicated than that designed in Chapter 5 and analyzed in Chapter 6 is constructed. To do this, however, requires that we introduce one further AISIM feature.

7.1 MESSAGE ROUTING SUBMODEL

When one Process is triggered by another through a CALL primitive, the called Process will execute in the same architectural node as the one that triggered it, i.e., utilize the same Resource, even if the two Processes are normally associated with different nodes. This is inconvenient in the representation of communication systems in which an activity in one hardware element causes activity in another one. AISIM therefore embodies a submodel to represent the situation in which a Process in one node triggers a Process in another node by communicating through the network architecture. This submodel consists of a collection of Processes and one Item.

The Processes that accomplish this must be placed in a project database with the commands available in the Library User Interface. The entities of this submodel need not be defined anew. For information on the use of this facility, see the AISIM User's Manual, Section 10.

7.2 DEFINING ARCHITECTURAL ELEMENTS

Consider modeling a communication system between two airbases, a headquarters and a command headquarters that communicates directly with a computer disk. Between these end-points are switches that govern the routing of messages through the system. The physical layout of this system is shown in figure 65.

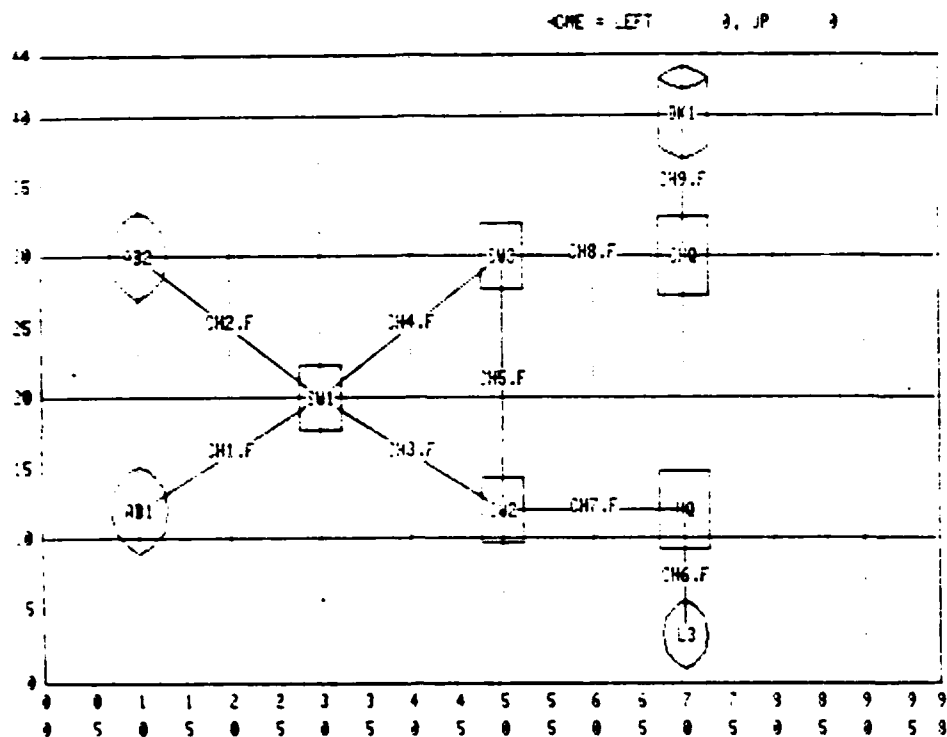


Figure 65. System Architecture

For this example, the shortest paths between the nodes will be used. Therefore, subsequent to defining the architecture, method B is used to create the Legal Path Table. The resulting table is depicted in the analysis report given in Appendix C.

The operations associated with this architecture are as follows. Both airbases periodically broadcast messages to the other nodes in the system and request plans from the command headquarters.

The effect of each broadcast is to (1) stimulate processing in the HQ and CHQ and to cause the updating of information in all other nodes. Periodically an applications program in L3 requests plans from the CHQ, as do AB1 and AB2. The effect of any such request is to engage the operation of the disk that communicates directly (and only) with the CHQ.

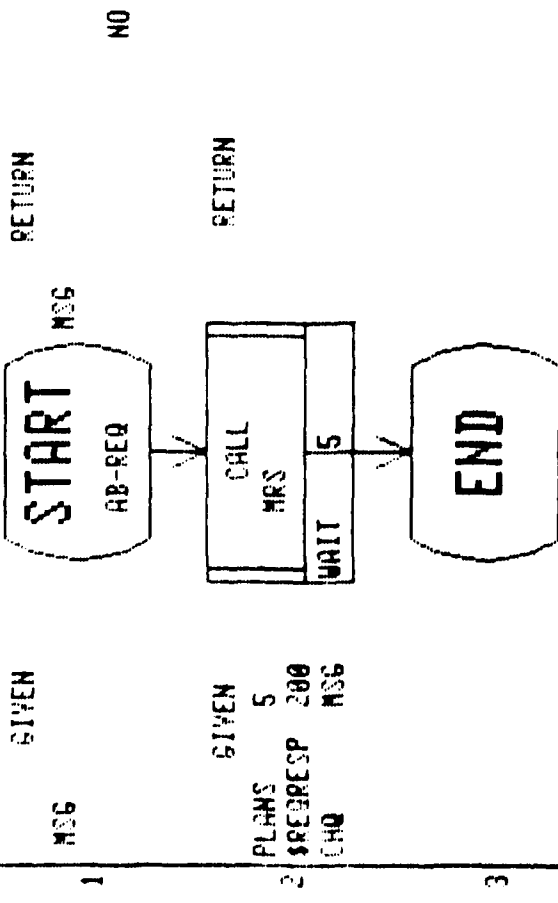
This description of the main operations of the system implies the following more rigorous listing of the Processes that need to be defined to represent such a system. The Processes required will be:

- A Process to represent the request from the HQ to the CHQ for plans. It will execute in the HQ node and will trigger a Process in the CHQ node.

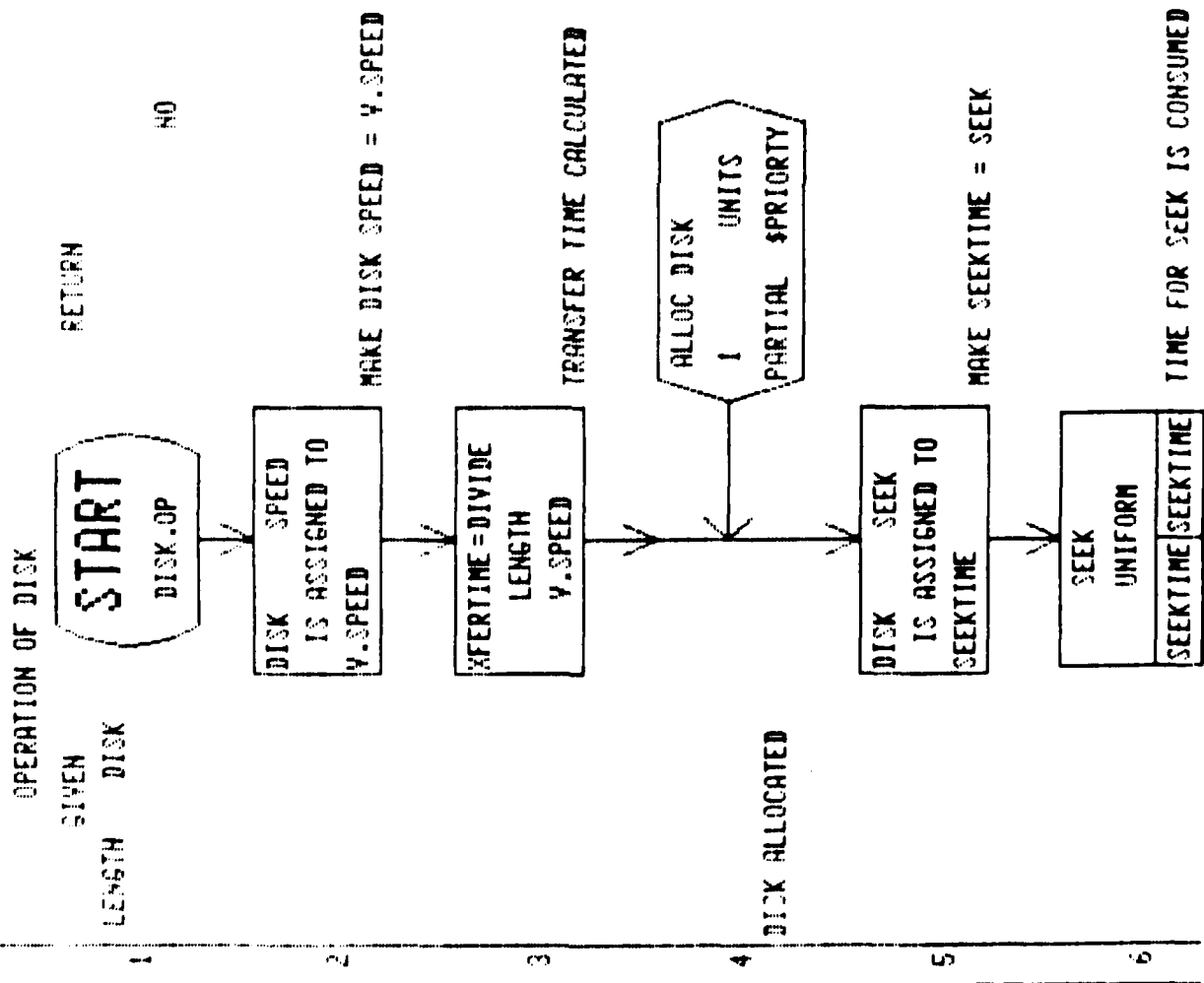
- A Process to represent the broadcast of data from AB1 and AB2 to all other nodes. This Process will execute in the nodes AB1 and AB2 and will trigger (a) an updating Processes in (a) the HQ node, (b) the CHQ node and (c) each other, i.e., a broadcast in one airbase will update information in the other.
- A Process to represent the updating activity that occurs in the CHQ, triggered by broadcasts from the airbases.
- A Process to represent the updating activity that occurs in the HQ that is triggered by broadcasts from the airbases.
- A Process to represent the updating activity in the airbases which is triggered by broadcasts from one another.
- A Process to represent the formulation of plans at the CHQ, which is triggered by requests from AB1, AB2 and HQ. This Process executes in the CHQ node and triggers another Process representing disk operation in the disk node.
- A Process to represent the operation of the disk that communicates with the CHQ node.

These descriptions can be used to generate the AISIM Process definitions found on the following pages. The Process flowcharts for each are displayed, together with annotations to clarify the rationale for the steps that might otherwise be obscure.

ALDBASE REQUEST FOR PLANS REPORT FROM CHQ



The Process is given the value of the variable MSG which happens to be an Item.



The Process is given the variables LENGTH and DISK which resolve, respectively, to a numeric and a Resource.

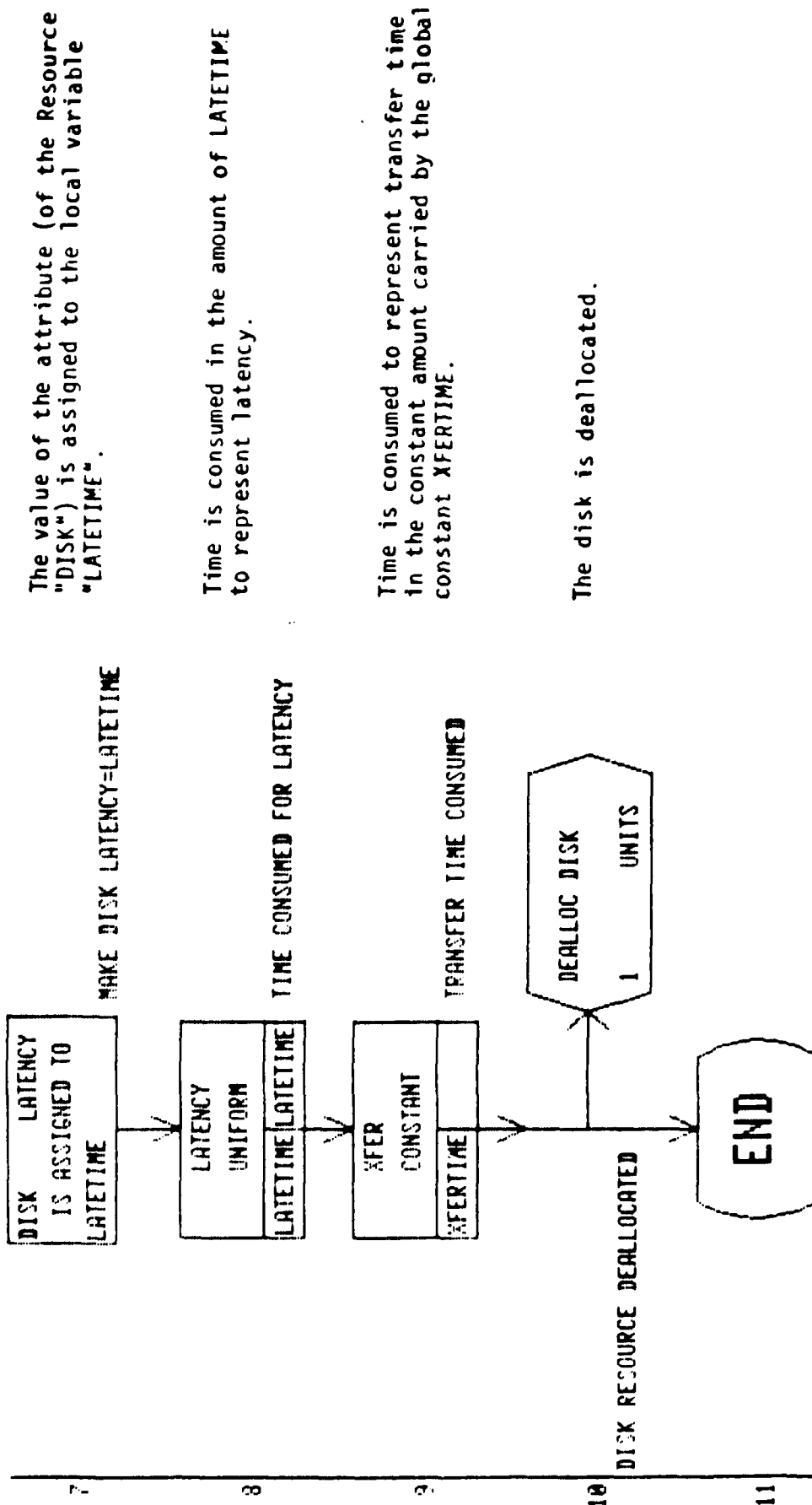
The attribute of the (Resource) DISK called SPEED is assigned to the local variable V.SPEED.

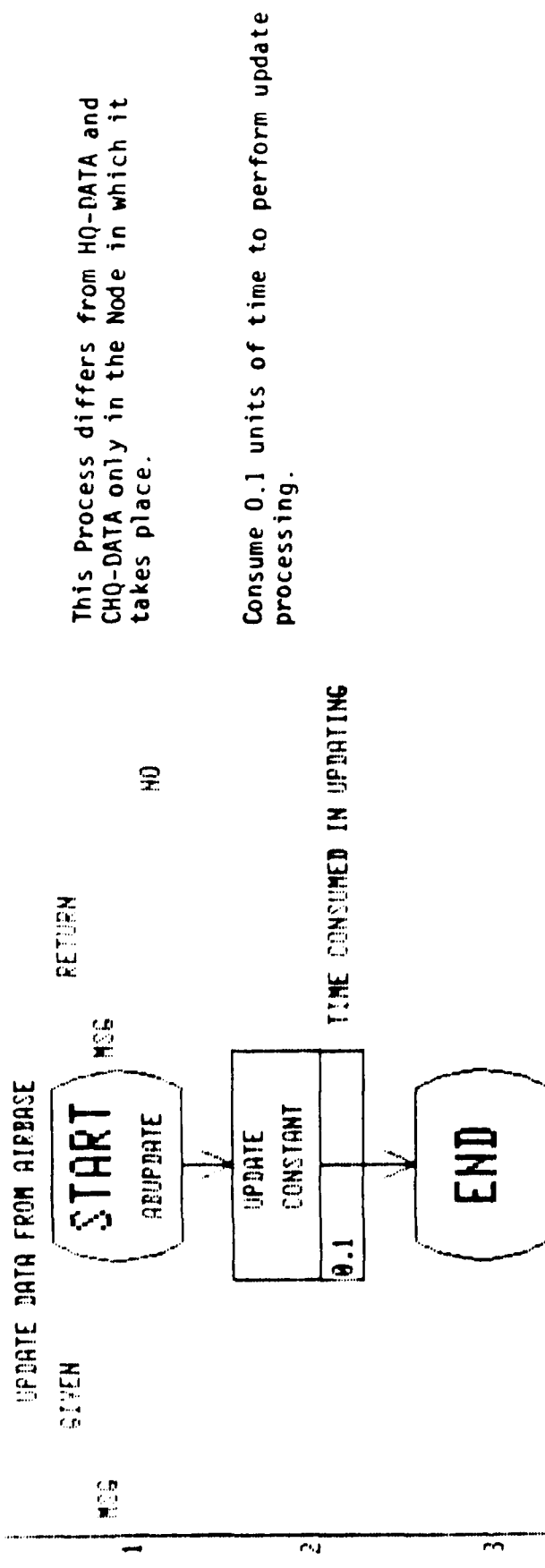
The transfer time is set equal to the length of the message divided by V.SPEED.

The Resource DISK is allocated.

The value of the disk attribute SEEK is assigned to the local variable SEEKTIME.

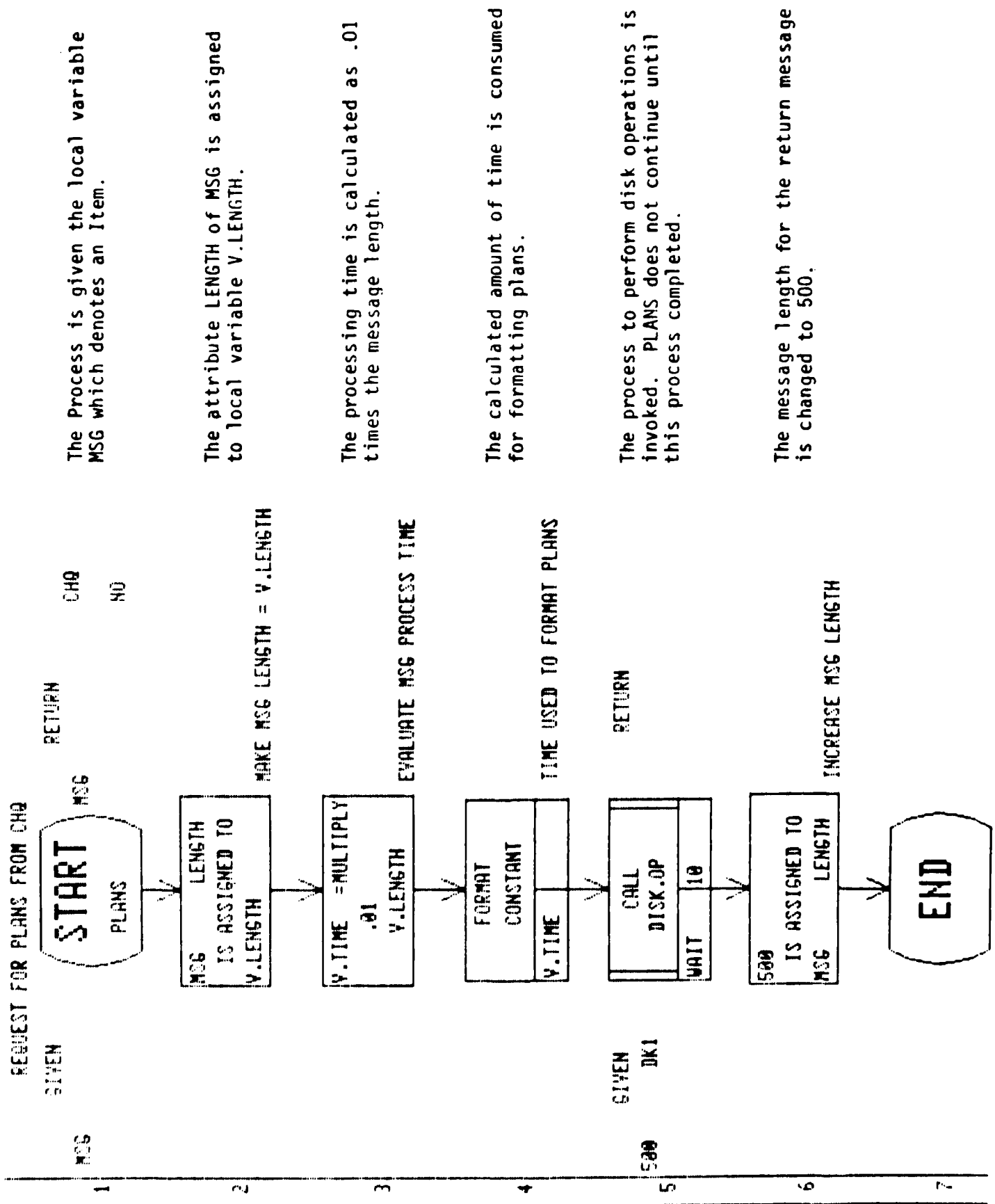
Time is taken up finding data on the disk in the amount of SEEKTIME.

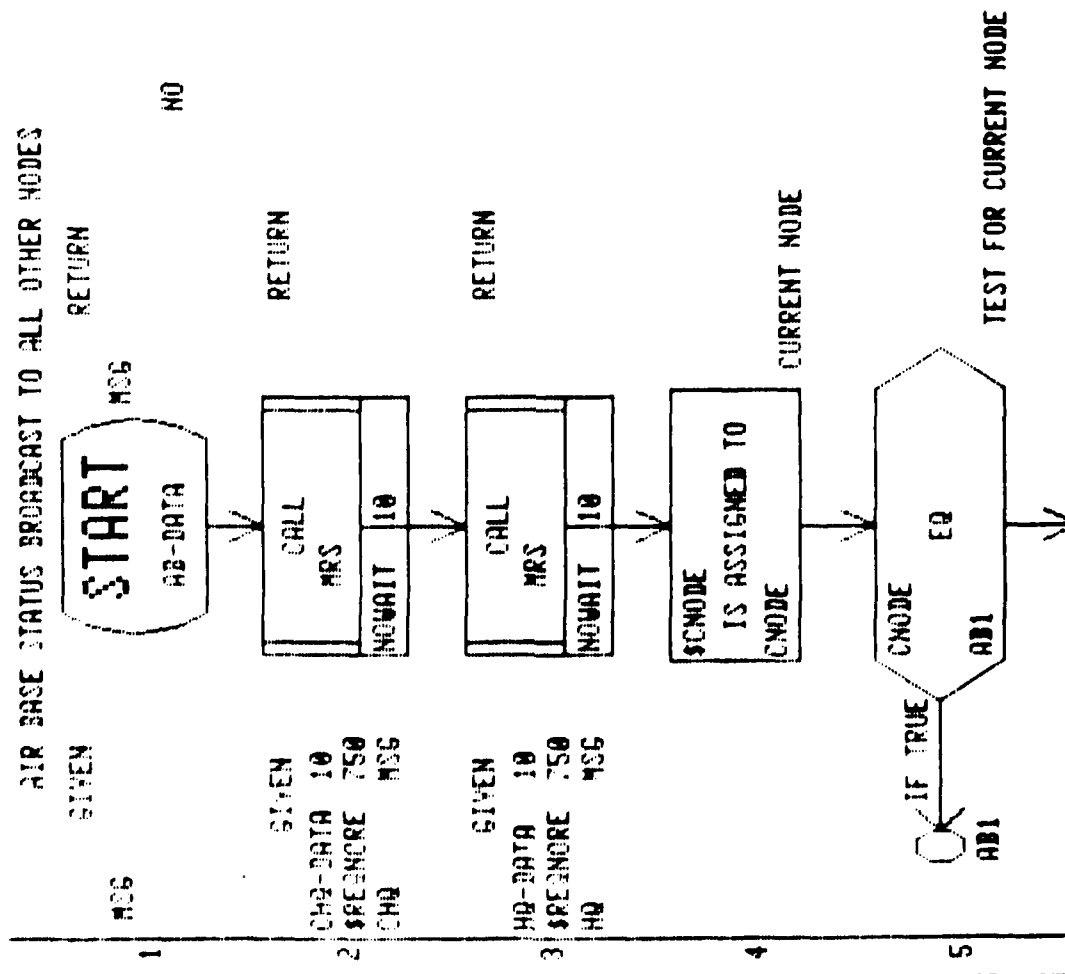




This Process differs from HQ-DATA and CHQ-DATA only in the Node in which it takes place.

Consume 0.1 units of time to perform update processing.





The process is given the value of a local variable "MSG" which in this case resolves to an Item representing a transient data element.

This primitive triggers a data update in the CHQ.

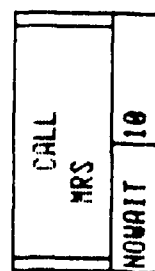
This primitive triggers a data update in the HQ.

The literal "\$CNODE" is assigned to the keyword "CNODE" which denotes the Node in which the Process is currently triggering. "AB-DATA" takes place in two different Nodes, AB1 and AB2.

This primitive determines whether the Node in which the Process is presently executing is AB1 or AB2. If it is in AB1 it branches to the Entry primitive labeled "AB1" below.

AIR BASE STATUS BROADCAST TO ALL OTHER NODES

RETURN



GIVEN
ABUPDATE 10
\$REGNORE 750
AB1 MSG

6

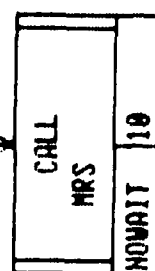
BRANCH TO THE END

END

ENTRY FROM COMPARE NODE



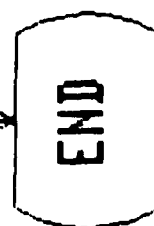
RETURN



GIVEN
ABUPDATE 10
\$REGNORE 750
AB2 MSG

9

ENTRY FROM REQUEST TO AB1



END

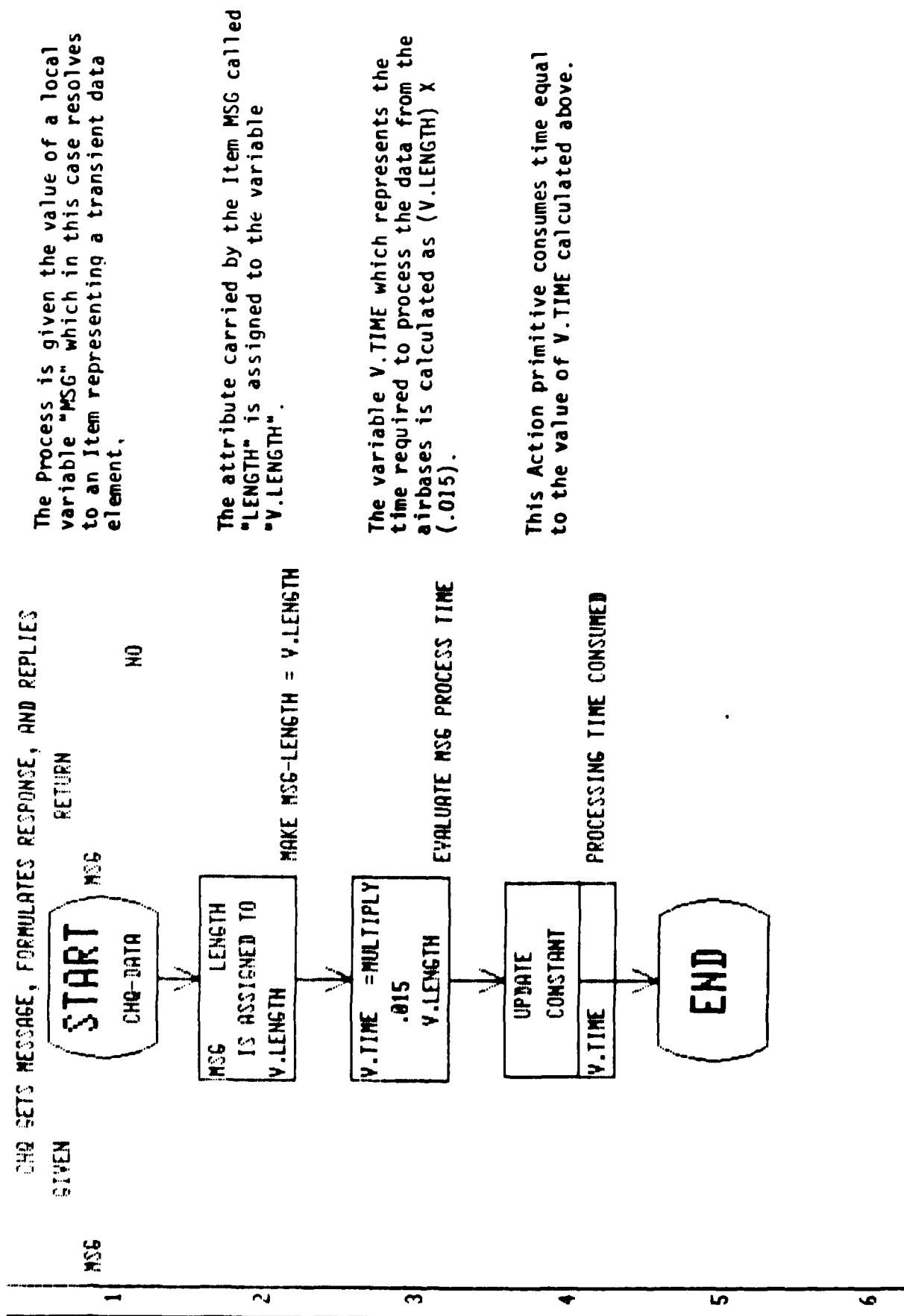
This Entry primitive is the one branched to subsequent to the calling of an updating Process in AB2.

This Call primitive triggers a data update in the AB1 Node.

This primitive branches to the Entry primitive labeled END below.

The Process branches to this Entry primitive if the Process is executing in the AB1 Node.

This Call primitive triggers data updating in the AB2 Node.

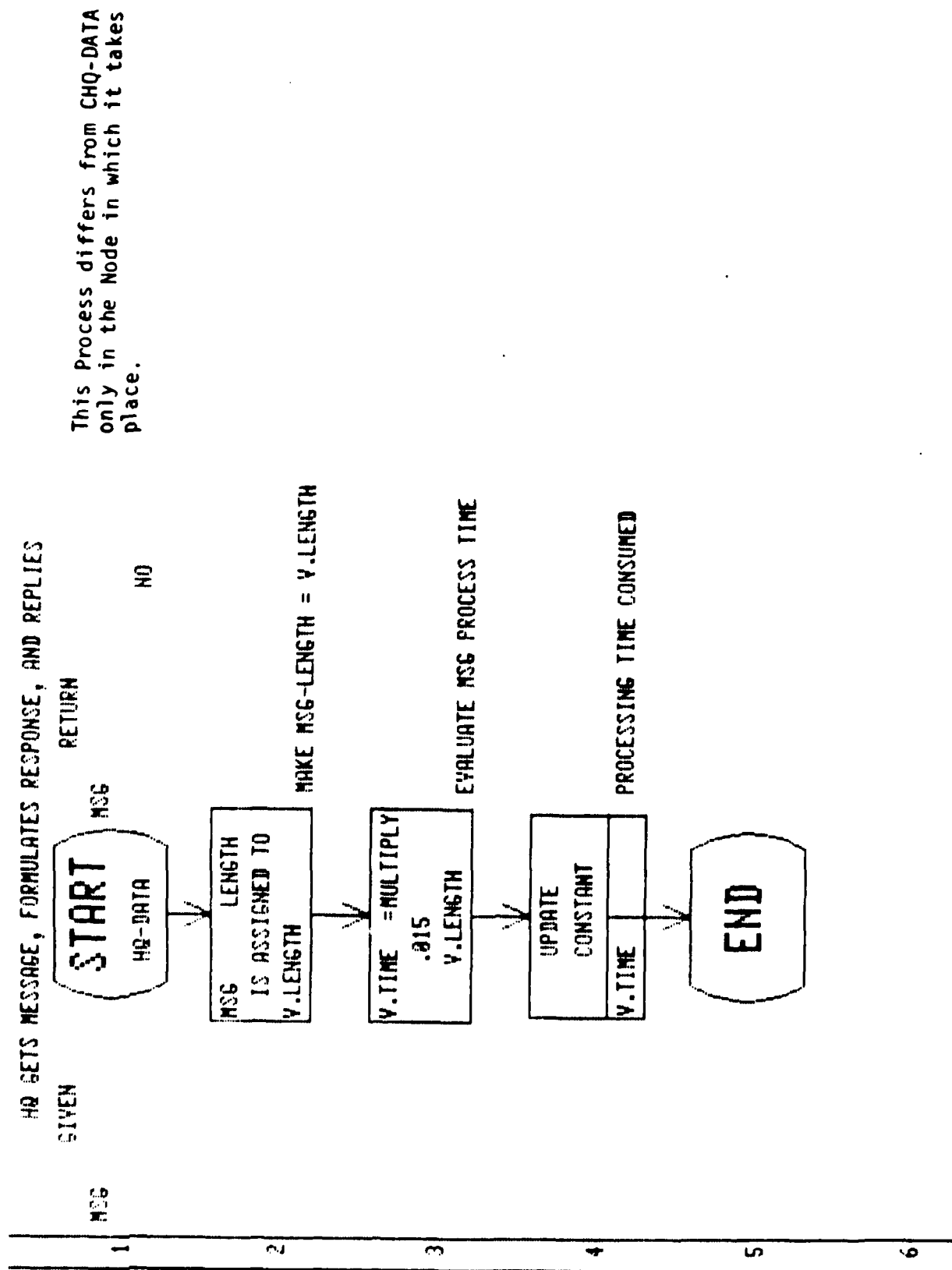


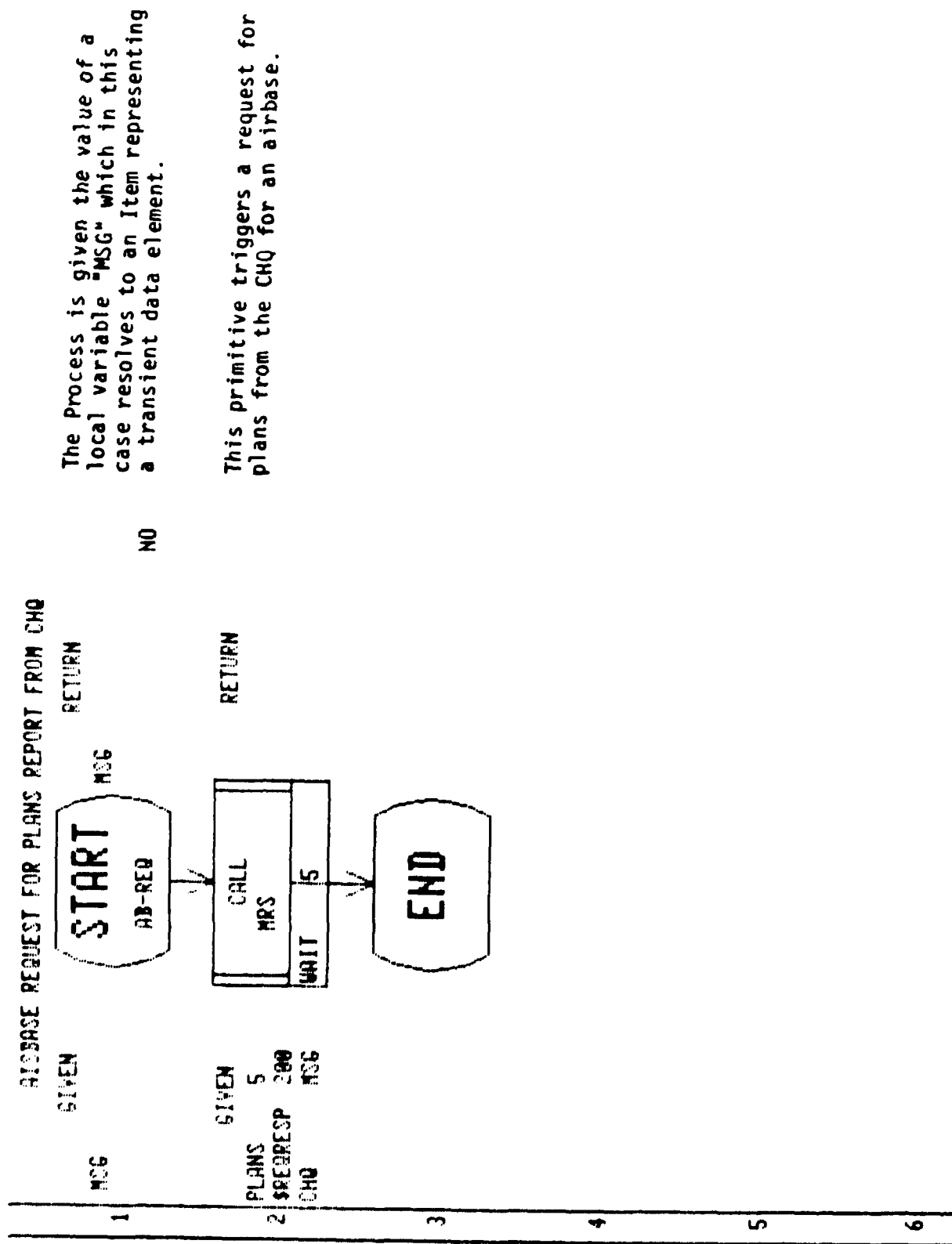
The Process is given the value of a local variable "MSG" which in this case resolves to an Item representing a transient data element.

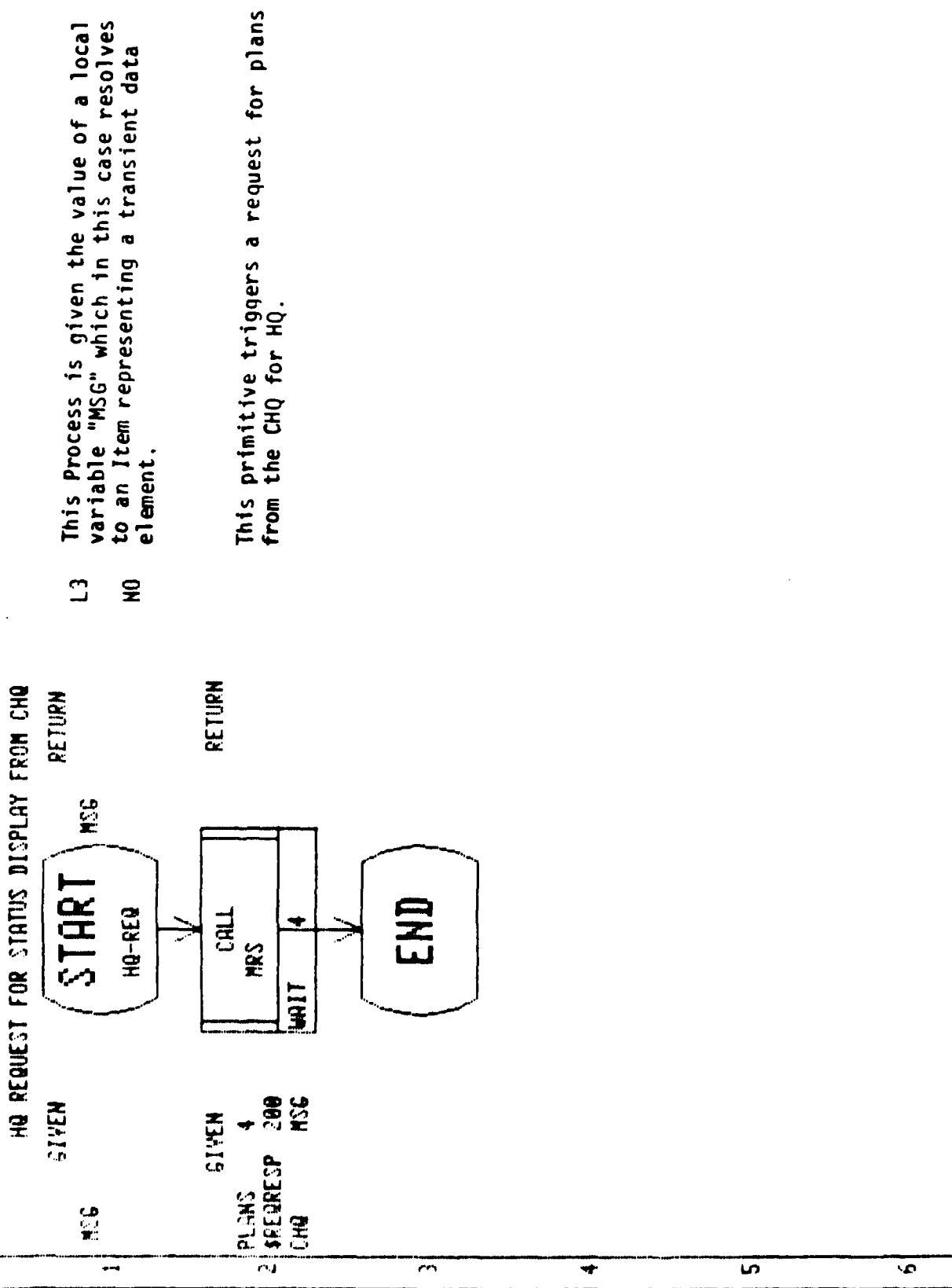
The attribute carried by the Item MSG called "LENGTH" is assigned to the variable "V.LENGTH".

The variable V.TIME which represents the time required to process the data from the airbases is calculated as (V.LENGTH) X (.015).

This Action primitive consumes time equal to the value of V.TIME calculated above.







L3 This Process is given the value of a local variable "MSG" which in this case resolves to an Item representing a transient data element.

This primitive triggers a request for plans from the CHQ for HQ.

7.3 DEFINING REMAINING MODEL ELEMENTS

The remaining components of the model must be defined. These include (1) all Resources accessed in the Processes (2) all Actions which appear as Primitives; (3) all Constants and global Variables used in the Processes; (4) Loads; and (5) a Scenario.

7.3.1 RESOURCE DEFINITIONS

All the Resources necessary to this model will have been defined automatically with default values while representing the physical layout represented in the Architecture Design Editor. Thus, if the nodes and links have the same names as in figure 65 above, the following list of Resources will already exist in the model database.

AB1	RESOURCE FOR NODE
AB2	RESOURCE FOR NODE
CH1.A	RESOURCE FOR CHANNEL CONNECTOR
CH1.B	RESOURCE FOR CHANNEL CONNECTOR
CH2.A	RESOURCE FOR CHANNEL CONNECTOR
CH2.B	RESOURCE FOR CHANNEL CONNECTOR
CH3.A	RESOURCE FOR CHANNEL CONNECTOR
CH3.B	RESOURCE FOR CHANNEL CONNECTOR
CH4.A	RESOURCE FOR CHANNEL CONNECTOR
CH4.B	RESOURCE FOR CHANNEL CONNECTOR
CH5.A	RESOURCE FOR CHANNEL CONNECTOR
CH5.B	RESOURCE FOR CHANNEL CONNECTOR
CH6.A	RESOURCE FOR CHANNEL CONNECTOR
CH6.B	RESOURCE FOR CHANNEL CONNECTOR
CH7.A	RESOURCE FOR CHANNEL CONNECTOR
CH7.B	RESOURCE FOR CHANNEL CONNECTOR
CH8.A	RESOURCE FOR CHANNEL CONNECTOR
CH8.B	RESOURCE FOR CHANNEL CONNECTOR
CH9.A	RESOURCE FOR CHANNEL CONNECTOR
CH9.B	RESOURCE FOR CHANNEL CONNECTOR
HQ	COMMAND HEAD-QUARTERS
DK1	DISK FOR COMMAND HEAD-QUARTERS
HQ	HEAD-QUARTERS
L3	RESOURCE FOR NODE
SW1	SWITCH BETWEEN AIRBASES AND OTHER TWO SWITCHES ('12)
SW2	SWITCH BETWEEN SWITCH 1 & 3 AND HQ
SW3	SWITCH BETWEEN SWITCH 1 & 2 AND CHQ

Figure 66. Defined Resource Entities

Since these Resources are created with default values (an initial unit of 1, a maximum unit of 1, and no description), they must be edited to provide helpful descriptions and to give them attributes since attributes of these Resources are accessed in several places in the Message Routing Submodel. Descriptions and attributes can be edited before generating the architecture using the ADE's DEFINE command. See Section 2.2 of this manual.

7.3.2 FILLING IN THE ACTION DEFINITIONS. The Action Primitives in the Processes must have corresponding Action entity definitions outside the Process. The Actions used are these:

FORMAT	TIME USED TO FORMAT PLANS FROM CHQ
LATENCY	LATENCY PAUSE SUBSEQUENT TO SEEK
ROUTE.CH	PROCESSING DELAY TO ROUTE A MESSAGE
SEEK	SEEKING INFORMATION ON DISK
UPDATE	UPDATING INFO SINCE PREVIOUS BROADCAST TO OTHER NODES
XFER	TRANSFER INFORMATION BOUGHT ON DISK
XFER.CH	PROCESSING DELAY TO ROUTE A MESSAGE OVER A CHANNEL

Figure 67. Defined Action Entities

7.3.3 CONSTANTS AND GLOBAL VARIABLES

This model contains five global Variables (ABDRATE, ABRATE, HORATE, TIME1 and VRATE) and one Constant (V.TRACE). Their defined values and descriptions (which explain their role in the model) are as shown in figure 68.

ABDRATE	INTERVAL RATE BETWEEN SIGNALS
ABRATE	INTERVAL RATE BETWEEN SIGNALS
HORATE	INTERVAL BETWEEN SIGNALS
TIME1	AVERAGE SEEK TIME FOR DISK IN MILLISECONDS
VRATE	SWITCH-OTHER NODE CHANNEL SPEED IN MS/BYTE
V.TRACE	DEFAULT IS NO TRACE ON

Figure 68. Defined Constant Entities

7.3.4 DEFINING LOADS AND SCENARIOS. In this model we wish to represent the several Process triggerings that are due to causes outside the system. First, the AB1 and AB2 will broadcast communications to the other nodes (which trigger updating Processes in them) every minute, by an interval scheduling method. In addition, AB1 and AB2 will issue requests for plans from the CHQ sixty times in one hour by an exponential scheduling method. We define a second Load to represent requests from the leaf-node, L3, also for plans from the CHQ. This Process will also be undertaken sixty times per hour, exponentially distributed. The Load definitions implied by these requirements are printed in appendix C.

The length of the entire Scenario is 360,000 milliseconds (one hour), which is divided into ten periods of six minutes each. To simulate the operation of the system with the worst case, we stipulate that both of the functional Loads are triggered simultaneously, at the beginning of the Scenario. In addition, as a monitoring device, we initiate the Trace Process at the beginning of the simulation run. The parameters for the Scenario implied by these requirements are printed page 16 of the analysis report in appendix C.

7.4 ANALYZING THE MODEL

To run the model through a simulation test, invoke the Analysis User Interface from the AISIM READY level. For this example, the simulation will not be interrupted at the ends of periods, nor will graphs be defined.

The analysis report obtained from a simulation run of this model appears in appendix C.

APPENDIX A

TERMINAL PROFILES FOR FORMS

	UP	DOWN	LEFT	RIGHT	ENTER	+FIELD	-FIELD
HP2647A	F1	F2	F3	F4	F5	<cr>	F6
HP2648A	F1	F2	F3	F4	F5	<cr>	F6
HP2623	F1	F2	F3	F4	F5	<cr>	F6
TEK4105	F1	F2	F3	F4	F5	<cr>	F6
VT100	↑	↓	<--	-->	PF1	<cr>	PF2

Figure 69. Terminal Profiles for Forms

Figure 69 describes the function keys which are used to move through the forms on each terminal. Following is a description of the ways in which a user can move through a form. These movements correspond to the column headings in the figure.

UP - If the cursor is in a block of fields, such as Resource attributes, the cursor will move up to the field above it. If the cursor is in a single field or at the top of a block, the cursor will move to the end of the next field above it. If there are no fields above it, the cursor will wrap to the end of the last field in the form.

DOWN - If the cursor is in a block of fields, such as Resource attributes, the cursor will move down to the field below it. If the cursor is in a single field or at the bottom of a block, the cursor will move to the beginning of the next field below it. If there are no fields below it, the cursor will wrap to the beginning of the first field in the form.

LEFT - The cursor will move one position to the left in the current field. If the cursor is at the beginning of a field, it will move to the end of the previous field. If the cursor is at the top of the form, it will wrap to the end of the last field in the form.

RIGHT - The cursor will move one position to the right in the current field. If the cursor is at the end of a field, it will move to the beginning of the next field. If the cursor is at the end of the form, it will wrap to the beginning of the first field in the form.

ENTER - Cut the form and send the data in the form to be processed by the ATISM system.

+FIELD - Move the cursor to the beginning of the next field in the form. If the cursor is at the end of the form, it will wrap to the top of the form.

-FIELD - Move the cursor to the end of the previous field in the form. If the cursor is at the top of the form, it will wrap to the end of the last field in the form.

APPENDIX B

SIMULATION REPORT FOR WORKING EXAMPLE

AD-A161 436

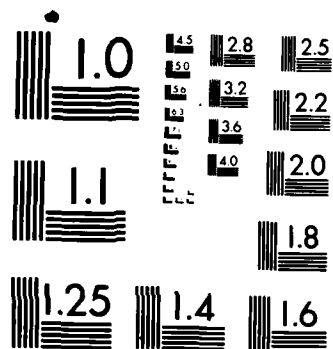
AISIM (AUTOMATED INTERACTIVE SIMULATION MODELING
SYSTEM) VAX VERSION TRAI (U) HUGHES AIRCRAFT CO
FULLERTON CA GROUND SYSTEMS GROUP S KNEEBURG ET AL
FEB 85 ESD-TR-85-129 F33615-81-C-5098 F/G 9/2

2/2

UNCLASSIFIED

NL

					END								
					FORMED								
					DTG								



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

```

#####
$ SIMULATION REPORT $
$ AISIM VERSION 4.0 $
$ HUGHES AIRCRAFT COMPANY $
$ 02/06/86 $
#####
GLOBAL CONSTANT DEFINITION....

```

```

CONSTANT INITIAL
MNEMONIC VALUE COMMENT
=====

```

```

TABLE DEFINITION....

```

```

GLOBAL VARIABLE DEFINITION....

```

```

VARIABLE INITIAL
MNEMONIC VALUE COMMENT
=====
GAMMA1 700
GAMMA2 .002

```

```

ITEM DEFINITION.....

```

```

ITEM DESCRIPTION
=====
MSG MSG TO BE TRANSMITTED
=====
ATTR. INITIAL
NAME VALUE
=====
LENGTH $LENGTH
=====

```

```

QUEUE DEFINITION....

```

```

QUEUE MAXIMUM
MNEMONIC SIZE COMMENT
=====
BUFFER INFINITE BUFFER ON WHICH MESSAGES ARE STORED
=====

```

```

RESOURCE DEFINITION....

```

```

RESOURCE TOTAL INITIAL
MNEMONIC # UNITS # UNITS DESCRIPTION
=====
BUF1 1 1 RESOURCE ASSOCIATED WITH BUFFER
=====

```

PAGE 2

ATTR. INITIAL
NAME VALUE
=====

COST 0

ARCHITECTURE LEGAL PATH DEFINITION

FROM TO NEXT VIA
DEVICE DEVICE LINK
=====

ACTION DEFINITION.....

ACTION ACTION COMMENT
=====

READ-MSG MACHINE READ A MESSAGE
SENDING MACHINE TRANSMIT A MESSAGE

PROCESS DEFINITION.....

PROCESS MNEMONIC DESCRIPTION
=====

RECEIVE =====
RECEIVE MESSAGES FROM TRANSMIT

ENTRY	OPCODE	PARM	PARM	PARM	COMMENT
START				NO	
TEST	BUF1	ABORT			TEST FOR BUFFER USE
REMOVE	FIRST	MSG			REMOVE BY FIFO DISCIPLINE
COMPARE	MSG		EQ		WHEN MSG=0 BUFFER IS EMPTY
ASSIGN	MSG	LENGTH		ABORT	MSG LENGTH IS READ
EVAL	MU	ALPHA			CALCULATE RECEPTION TIME
READ-MSG	CONSTANT	MU			TIME TO PROCESS MESSAGE
DESTROY	MSG				MSG ELIMINATED FROM SYSTEM
ENTRY					ENTER FROM COMPARE & TEST
END					

LOCAL VARIABLES OF PROCESS RECEIVE

1 BUF1 (R) 2 MSG (I) 3 BUFFER (Q) 4 ALPHA (A)

PROCESS MNEMONIC DESCRIPTION

PAGE 3
 =====
 TRANSMIT
 =====
 TRANSMITTING MESSAGES TO RECEIVER
 =====

ENTRY	OPCODE	PARAM	PARAM	PARAM	COMMENT
START	ALLOC	BUF1	NO	ALL	ALLOCATE BUF1
CREATE	MSG	PRIORITY			INTRODUCE MSG INTO SYSTEM
EVAL	ALPHA	RANDOM			GENERATE RANDOM NUMBER
EVAL	ALPHA	MULTIPLY			TWICE AVERAGE TIMES ALPHA
ASSIGN	ALPHA	GAMMA1			SET MESSAGE LENGTH
EVAL	MSG	LENGTH			CALCULATE TRANSMIT TIME
EVAL	MU	MULTIPLY			TIME CONSUMED TRANSMITTING
SENDING	CONSTANT	MU			STORE MSG ON BUFFER
FILE	MSG	LAST			RELEASE RESOURCE BUF1
DEALLOC	BUF1	1			
END					

LOCAL VARIABLES OF PROCESS TRANSMIT
 =====
 1 BUF1 (R) 2 MSG (I) 3 ALPHA 4 MU
 5 SENDING (A) 6 BUFFER (Q)
 =====

LOAD DEFINITION.....

LOAD	MNEMONIC	DESCRIPTION
L1		
LOAD	NODES	
=====	=====	=====

PROCESS	MNEMONIC	MAX #	SCHEDULE	METHOD	MEAN	DELTA	PRIORITY
TRANSMIT	300		POISSON				0
LOAD	MNEMONIC	DESCRIPTION					
L11							
LOAD	NODES						
=====	=====	=====	=====	=====	=====	=====	=====

```

LOAD
=====
MNEMONIC
=====
L2
=====
PROCESS      SCHEDULE
MNEMONIC    MAX #  METHOD  MEAN  DELTA  PRIORITY
=====
RECEIVE     600    INTERVAL 0.000001
=====
DESCRIPTION
=====
LOAD
=====
NODES
=====
=====

```

```

LOAD
=====
MNEMONIC
=====
L22
=====
PROCESS      SCHEDULE
MNEMONIC    MAX #  METHOD  MEAN  DELTA  PRIORITY
=====
TRANSMIT    200    POISSON
=====
DESCRIPTION
=====
LOAD
=====
NODES
=====
=====

```

```

LOAD
=====
MNEMONIC
=====
L3
=====
PROCESS      SCHEDULE
MNEMONIC    MAX #  METHOD  MEAN  DELTA  PRIORITY
=====
RECEIVE     400    INTERVAL 0.000001
=====
DESCRIPTION
=====
LOAD
=====
NODES
=====
=====

```

```

LOAD
=====
MNEMONIC
=====
PROCESS      SCHEDULE
MNEMONIC    MAX #  METHOD  MEAN  DELTA  PRIORITY
=====
TRANSMIT    100    POISSON
=====
DESCRIPTION
=====

```

PAGE 5
L33

LOAD NODES
=====

PROCESS SCHEDULE
MNEMONIC MAX # METHOD MEAN DELTA PRIORITY
=====

RECEIVE 200 INTERVAL 0.00001 0

SCENARIO DEFINITION....

SCENARIO DESCRIPTION
=====

SCEN

PERIOD
LENGTH

=====

PERIOD	PERIOD	PERIOD	PERIOD	PERIOD	PERIOD	PERIOD	PERIOD
MNEMONIC	MNEMONIC	MNEMONIC	MNEMONIC	MNEMONIC	MNEMONIC	MNEMONIC	MNEMONIC
1	2	3	4	5	6		

TRIGGER	TIME	TO SCHEDULE	TRIGGER	TIME	TO SCHEDULE
MNEMONIC	SCHEDULE	PRIORITY	MNEMONIC	SCHEDULE	PRIORITY
L1	100	0	L11	100	0
L2	300	0	L22	300	0
L3	500	0	L33	500	0

0 ERRORS WERE DETECTED DURING MODEL INITIALIZATION

PAGE 6

SIMULATION TIME = 600.00000 UNITS

VARIABLE REPORT

NUMERIC VARIABLES...

TOTAL		VALUE				
VARIABLE	SAMPLES	CURRENT	MEAN	STD DEV	MINIMUM	MAXIMUM
GAMMA1	1	700.000	700.000	0.	700.000	700.000
GAMMA2	3	0.002	0.002	0.000	0.001	0.002

NON-NUMERIC VARIABLES...

VARIABLE	CURRENT
TYPE	VALUE

PAGE 7

SIMULATION TIME = 600.00000 UNITS

ITEM REPORT

ITEM NAME	NUMBER CREATED	NUMBER DESTR'D	MINIMUM	MAXIMUM	AVERAGE	STD DEV
MSG	590	500	70.60	200.88	142.90	33.70

PAGE 8

SIMULATION TIME = 600.00000 UNITS

QUEUE REPORT

QUEUE	TOTAL NUMBER	CURRENT...	MEAN.....	STD DEV...	MINIMUM...	MAXIMUM...
=====	=====	=====	=====	=====	=====	=====
BUFFER						
FILED ON	590					
REMOVED FROM	500					
# IN QUEUE		96.000	126.480	107.531	0.	300.000
TIME IN QUEUE			141.755	33.713	67.932	199.654
TASKS BLOCKED	0					
TASKS RESUMED	0					
# BEING BLOCKED		0.	0.	0.	0.	0.
TIME BLOCKED			0.	0.	0.	0.

PAGE 9

SIMULATION TIME = 800.00000 UNITS

RESOURCE REPORT

RESOURCE	TOTAL NUMBER	CURRENT	MEAN	STD DEV	MINIMUM	MAXIMUM
=====	=====	=====	=====	=====	=====	=====
BUF1						
# IDLE		1.000	0.513	0.500	0.	1.000
REQUEST TIME			7.171	6.273	0.	24.357
596						
INTO BUSY	596					
OUT OF BUSY	596					
# BUSY		0.	0.487	0.500	0.	1.000
BUSY TIME			0.490	0.348	0.000	1.398
# INACTIVE		0.	0.	0.	0.	0.
596						
INTO WAIT	596					
OUT OF WAIT	596					
# WAITING		0.	7.123	10.523	0.	36.000
WAIT TIME			7.171	6.273	0.	24.357

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

PAGE 10

SIMULATION TIME = 600.00000 UNITS

ACTION REPORT

ACTION	TOTAL	MEAN	STD DEV	MINIMUM	MAXIMUM	% TIME
	SAMPLES					OF TOTAL
=====	=====	=====	=====	=====	=====	=====
READ-MSG						
USEFUL TIME	500	0.675	0.410	0.001	1.398	56.244
DELAY TIME	500	0.	0.	0.	0.	

ACTION	TOTAL	MEAN	STD DEV	MINIMUM	MAXIMUM	% TIME
	SAMPLES					OF TOTAL
=====	=====	=====	=====	=====	=====	=====
SENDING						
USEFUL TIME	596	0.490	0.348	0.000	1.398	48.722
DELAY TIME	596	0.	0.	0.	0.	

PAGE 11

SIMULATION TIME = 600.00000 UNITS

PROCESS REPORT

PROCESS	TOTAL SAMPLES	SUM	MEAN	STD DEV	MINIMUM	MAXIMUM
RECEIVE	1200	337.465	0.281	0.425	0.	1.398
PROCESS WAIT	0	0.	0.	0.	0.	0.
RESOURCE WAIT	0	0.	0.	0.	0.	0.

TOTAL #	# AUTO	# CALL	# OF	# NOT	# TIMES
SCHEDULE	1200	0	0	0	0
SCHEDULE	1200	0	1200	0	0

ITEM	CREATED	RECEIVED	SENT	DESTR'D
MSG	0	0	0	500

ITEM	# SMPLS	MEAN	MINIMUM	MAXIMUM	STD DEV
MSG	500	0.67	0.00	1.40	0.41

PROCESS	DESCRIPTION
RECEIVE	RECEIVE MESSAGES FROM TRANSMIT
COUNT ENTRY	OPCODE PARM PARM PARM COMMENT
1200	START NO
1200	TEST ABORT
1200	REMOVE FIRST MSG
1200	COMPARE MSG
1200	ASSIGN 0
500	ALPHA MU
500	EVAL MU
500	READ-MSG CONSTANT MU
500	DESTROY MSG
1200	ABORT
1200	END

PROCESS	TOTAL SAMPLES	SUM	MEAN	STD DEV	MINIMUM	MAXIMUM
RECEIVE	1200	337.465	0.281	0.425	0.	1.398

PAGE 12

```

=====
TRANSMIT
TOTAL          596  4566.214  7.661  6.335  0.008  25.420
PROCESS WAIT   0      0      0      0      0      0
RESOURCE WAIT  596  4273.884  7.171  6.273  0      24.357
=====

```

TOTAL # AUTO # CALL # OF # NOT # TIMES
 SCHEDULE SCHEDULE SCHEDULE COMPLETE COMPLETE SUSPEND.

```

=====
596  0  0  596  0  0
=====

```

ITEM CREATED RECEIVED SENT DESTR'D

```

=====
MSG 596 0 0 0
=====

```

PROCESS HOLDING TIME
 # SMPLS MEAN..... MINIMUM.. MAXIMUM... STD DEV...
 MSG 596 0.49 0.00 1.40 0.35

PROCESS DESCRIPTION

```

=====
TRANSMIT
TRANSMITTING MESSAGES TO RECEIVER
=====

```

COUNT	ENTRY	OPCODE	PARAM	PARAM	COMMENT
596	START				
596	ALLOC	BUF1	1	ALL	ALLOCATE BUF1
596	CREATE	\$PRIORITY			INTRODUCE MSG INTO SYSTEM
596	EVAL	ALPHA	RANDOM		GENERATE RANDOM NUMBER
596	EVAL	ALPHA	MULTIPLY		TWICE AVERAGE TIMES ALPHA
596	ASSIGN	ALPHA	GAMMA1		SET MESSAGE LENGTH
596	EVAL	MU	MULTIPLY		CALCULATE TRANSMIT TIME
596	SENDING	CONSTANT	MU		TIME CONSUMED TRANSMITTING
596	FILE	MSG	LAST	BUFFER	STORE MSG ON BUFFER
596	DEALLOC	BUF1	1		RELEASE RESOURCE BUF1
596	END				

APPENDIX C

SIMULATION REPORT FOR ELABORATE EXAMPLE

```

#####
$ SIMULATION REPORT $
$ AISIM VERSION 4.0 $
$ HUGHES AIRCRAFT COMPANY $
$ 02/05/85 $
#####
GLOBAL CONSTANT DEFINITION.....

```

```

CONSTANT INITIAL
MNEMONIC VALUE COMMENT
=====
V.TRACE 0 DEFAULT IS NO TRACE ON
=====

```

TABLE DEFINITION....

GLOBAL VARIABLE DEFINITION.....

```

VARIABLE INITIAL
MNEMONIC VALUE COMMENT
=====
ABDRATE 60000 INTERVAL RATE BETWEEN SIGNALS
ABRRATE 36000 INTERVAL RATE BETWEEN SIGNALS
HQRATE 72000 INTERVAL BETWEEN SIGNALS
TIME1 30 AVERAGE SEEK TIME FOR DISK IN MILLISECONDS
VRATE 1.6276 SWITCH-OTHER NODE CHANNEL SPEED IN MS/BYTE
=====

```

ITEM DEFINITION.....

```

ITEM DESCRIPTION
=====
MSG MESSAGE FOR INTERNODE COMMUNICATION
=====

```

```

ATTR. INITIAL
NAME VALUE
=====
CNODE $CNODE
FNODE $CNODE
LENGTH 99999999
RPROC $ERROR
RPROC PRI 99999999
TNODE $CNODE
TYPE $REQNORE

```

QUEUE DEFINITION.....

```

QUEUE MAXIMUM

```

PAGE	2	SIZE	COMMENT
MNEMONIC			
=====			
RESOURCE DEFINITION.....			
=====			
RESOURCE TOTAL	INITIAL		
MNEMONIC # UNITS	# UNITS		DESCRIPTION
=====			
AB1	1	1	RESOURCE FOR NODE
ATTR.	INITIAL		
NAME	VALUE		
COST			
M.ROUTE	8		
=====			
AB2	1	1	RESOURCE FOR NODE
ATTR.	INITIAL		
NAME	VALUE		
COST			
M.ROUTE	8		
=====			
CH1.A	1	1	RESOURCE FOR CHANNEL CONNECTOR
ATTR.	INITIAL		
NAME	VALUE		
COST			
RATE	VRATE		
=====			
CH1.B	1	1	RESOURCE FOR CHANNEL CONNECTOR
ATTR.	INITIAL		
NAME	VALUE		
COST			
RATE	VRATE		
=====			
CH2.A	1	1	RESOURCE FOR CHANNEL CONNECTOR
ATTR.	INITIAL		
NAME	VALUE		
COST			
RATE	VRATE		
=====			
CH2.B	1	1	RESOURCE FOR CHANNEL CONNECTOR
ATTR.	INITIAL		
NAME	VALUE		
COST			
RATE	VRATE		
=====			

PAGE 3

CH3.A RESOURCE FOR CHANNEL CONNECTOR

1	ATTR.	1	INITIAL
	NAME		VALUE
=====			
COST		0	
RATE		0.4069	

CH3.B RESOURCE FOR CHANNEL CONNECTOR

1	ATTR.	1	INITIAL
	NAME		VALUE
=====			
COST		0	
RATE		0.4069	

CH4.A RESOURCE FOR CHANNEL CONNECTOR

1	ATTR.	1	INITIAL
	NAME		VALUE
=====			
COST		0	
RATE		0.4069	

CH4.B RESOURCE FOR CHANNEL CONNECTOR

1	ATTR.	1	INITIAL
	NAME		VALUE
=====			
COST		0	
RATE		0.4069	

CH5.A RESOURCE FOR CHANNEL CONNECTOR

1	ATTR.	1	INITIAL
	NAME		VALUE
=====			
COST		0	
RATE		0.4069	

CH5.B RESOURCE FOR CHANNEL CONNECTOR

1	ATTR.	1	INITIAL
	NAME		VALUE
=====			
COST		0	
RATE		0.4069	

CH6.A RESOURCE FOR CHANNEL CONNECTOR

1	ATTR.	1	INITIAL
	NAME		VALUE
=====			
COST		0	
RATE		0.4069	

PAGE 4

CH6.B RESOURCE FOR CHANNEL CONNECTOR

1	ATTR.	1	INITIAL
	NAME		VALUE
=====		=====	
COST		0	
RATE		VRATE	

CH7.A RESOURCE FOR CHANNEL CONNECTOR

1	ATTR.	1	INITIAL
	NAME		VALUE
=====		=====	
COST		0	
RATE		VRATE	

CH7.B RESOURCE FOR CHANNEL CONNECTOR

1	ATTR.	1	INITIAL
	NAME		VALUE
=====		=====	
COST		0	
RATE		VRATE	

CH8.A RESOURCE FOR CHANNEL CONNECTOR

1	ATTR.	1	INITIAL
	NAME		VALUE
=====		=====	
COST		0	
RATE		VRATE	

CH8.B RESOURCE FOR CHANNEL CONNECTOR

1	ATTR.	1	INITIAL
	NAME		VALUE
=====		=====	
COST		0	
RATE		VRATE	

CH9.A RESOURCE FOR CHANNEL CONNECTOR

1	ATTR.	1	INITIAL
	NAME		VALUE
=====		=====	
COST		0	
RATE		VRATE	

CH9.B RESOURCE FOR CHANNEL CONNECTOR

1	ATTR.	1	INITIAL
	NAME		VALUE
=====		=====	
COST		0	
RATE		VRATE	

PAGE 5

COMMAND HEAD-QUARTERS

1 ATTR. INITIAL
NAME VALUE
=====

COST 0
M.ROUTE 8

DISK FOR COMMAND HEAD-QUARTERS

1 ATTR. INITIAL
NAME VALUE
=====

COST 0
LATDELTA 16
LATENCY 16
M.ROUTE 0
SEEK TIME1
SPEED 20000

HEAD-QUARTERS

1 ATTR. INITIAL
NAME VALUE
=====

COST 0
M.ROUTE 8

RESOURCE FOR NODE

1 ATTR. INITIAL
NAME VALUE
=====

COST 0
M.ROUTE 8

SWITCH BETWEEN AIRBASES AND OTHER TWO SWITCHES (1&2)

1 ATTR. INITIAL
NAME VALUE
=====

COST 0
M.ROUTE 8

SWITCH BETWEEN SWITCH 1 & 3 AND HQ

1 ATTR. INITIAL
NAME VALUE
=====

COST 0
M.ROUTE 8

SWITCH BETWEEN SWITCH 1 & 2 AND CHQ

1 ATTR. INITIAL

PAGE 8 NAME VALUE
=====

COST 0
M.ROUTE 8

ARCHITECTURE LEGAL PATH DEFINITION

FROM DEVICE	TO DEVICE	NEXT DEVICE	VIA LINK
AB1	AB2	SW1	CH1.A
AB1	CHQ	SW1	CH1.A
AB1	DK1	SW1	CH1.A
AB1	HQ	SW1	CH1.A
AB1	L3	SW1	CH1.A
AB1	SW1	SW1	CH1.A
AB1	SW2	SW1	CH1.A
AB1	SW3	SW1	CH1.A
AB2	AB1	SW1	CH2.A
AB2	CHQ	SW1	CH2.A
AB2	DK1	SW1	CH2.A
AB2	HQ	SW1	CH2.A
AB2	L3	SW1	CH2.A
AB2	SW1	SW1	CH2.A
AB2	SW2	SW1	CH2.A
AB2	SW3	SW1	CH2.A
CHQ	AB1	SW3	CH8.A
CHQ	AB2	SW3	CH8.A
CHQ	DK1	DK1	CH9.A
CHQ	HQ	SW3	CH8.A
CHQ	L3	SW3	CH8.A
CHQ	SW1	SW3	CH8.A
CHQ	SW2	SW3	CH8.A
CHQ	SW3	SW3	CH8.A
DK1	AB1	CHQ	CH9.B
DK1	AB2	CHQ	CH9.B
DK1	CHQ	CHQ	CH9.B
DK1	HQ	CHQ	CH9.B
DK1	L3	CHQ	CH9.B
DK1	SW1	CHQ	CH9.B
DK1	SW2	CHQ	CH9.B
DK1	SW3	CHQ	CH9.B
HQ	AB1	SW2	CH7.A
HQ	AB2	SW2	CH7.A
HQ	CHQ	SW2	CH7.A
HQ	DK1	SW2	CH7.A
HQ	L3	L3	CH6.A
HQ	SW1	SW2	CH7.A
HQ	SW2	SW2	CH7.A

PAGE	7	SW3	CH7.A
HQ	AB1	HQ	CH6.B
L3	AB2	HQ	CH6.B
L3	CHQ	HQ	CH6.B
L3	DK1	HQ	CH6.B
L3	HQ	HQ	CH6.B
L3	SW1	HQ	CH6.B
L3	SW2	HQ	CH6.B
L3	SW3	HQ	CH6.B
SW1	AB1	AB1	CH1.B
SW1	AB2	AB2	CH2.B
SW1	CHQ	SW3	CH4.A
SW1	DK1	SW3	CH4.A
SW1	HQ	SW2	CH3.A
SW1	L3	SW2	CH3.A
SW1	SW2	SW2	CH3.A
SW1	SW3	SW3	CH4.A
SW2	AB1	SW1	CH3.B
SW2	AB2	SW1	CH3.B
SW2	CHQ	SW3	CH5.A
SW2	DK1	SW3	CH5.A
SW2	HQ	HQ	CH7.B
SW2	L3	HQ	CH7.B
SW2	SW1	SW1	CH3.B
SW2	SW3	SW3	CH5.A
SW3	AB1	SW1	CH4.B
SW3	AB2	SW1	CH4.B
SW3	CHQ	CHQ	CH8.B
SW3	DK1	CHQ	CH8.B
SW3	HQ	SW2	CH5.B
SW3	L3	SW2	CH5.B
SW3	SW1	SW1	CH4.B
SW3	SW2	SW2	CH5.B

ACTION DEFINITION.....

ACTION	ACTION	CLASS	COMMENT
=====	=====	=====	=====
FORMAT	MACHINE		TIME USED TO FORMAT PLANS FROM CHQ
LATENCY	MACHINE		LATENCY PAUSE SUBSEQUENT TO SEEK
ROUTE.OH	CPU		PROCESSING DELAY TO ROUTE A MESSAGE
SEEK	MACHINE		SEEKING INFORMATION ON DISK
UPDATE	AIRBASE		UPDATING INFO SINCE PREVIOUS BROADCAST TO OTHER NODES
XFER	MACHINE		TRANSFER INFORMATION SOUGHT ON DISK
XFER.OH	CHANNEL		PROCESSING DELAY TO ROUTE A MESSAGE OVER A CHANNEL
PROCESS DEFINITION.....			

PAGE 8
PROCESS
WNEUMONIC

===== DESCRIPTION =====
===== AIR BASE STATUS BROADCAST TO ALL OTHER NODES =====

AB-DATA

ENTRY	OPCODE	PARAM	PARAM	PARAM	COMMENT
=====	=====	=====	=====	=====	=====
START				NO	
GIVEN	MSG				
RETURN	MSG				
CALL	MRS		NOWAIT		PROCESS REQUEST TO CHQ
GIVEN	CHQ-DATA	10	\$REQNORE		
	760	CHQ	MSG		
CALL	MRS		NOWAIT		PROCESS REQUEST TO HQ
GIVEN	HQ-DATA	10	\$REQNORE		
	760	HQ	MSG		
ASSIGN	\$CNODE				CURRENT NODE
	CNODE				
COMPARE	CNODE				TEST FOR CURRENT NODE
	AB1				
CALL	MRS		NOWAIT		PROCESS REQUEST TO AB1
GIVEN	ABUPDATE	10	\$REQNORE		
	760	AB1	MSG		
BRANCH	END	100			BRANCH TO THE END
ENTRY					ENTRY FROM COMPARE NODE
CALL	MRS		NOWAIT		PROCESS REQUEST TO AB2
GIVEN	ABUPDATE	10	\$REQNORE		
	760	AB2	MSG		
ENTRY					ENTRY FROM REQUEST TO AB1
END					

LOCAL VARIABLES OF PROCESS AB-DATA

=====	=====	=====	=====	=====	=====
1 MSG	(I)	2 MRS	(P)	3 CHQ-DATA	(P) 4 CHQ
6 HQ-DATA	(P)	6 HQ	(R)	7 CNODE	8 AB1
9 ABUPDATE	(P)	10 AB2	(R)		
					(R)

PROCESS
WNEUMONIC
AB-REQ

===== DESCRIPTION =====
===== AISBASE REQUEST FOR PLANS REPORT FROM CHQ =====

ENTRY	OPCODE	PARAM	PARAM	PARAM	COMMENT
=====	=====	=====	=====	=====	=====
START				NO	
GIVEN	MSG				
RETURN	MSG				
CALL	MRS		WAIT	5	PROCESS REQUEST TO CHQ

PAGE 9 GIVEN PLANS 5 \$REQRESP
200 CHQ MSG
END
LOCAL VARIABLES OF PROCESS AB-REQ
===== (I) 2 MRS (P) 3 PLANS (P) 4 CHQ (R)
1 MSG

PROCESS
MNEMONIC DESCRIPTION
===== UPDATE DATA FROM AIRBASE
ABUPDATE

ENTRY	OPCODE	PARM	PARM	PARM	COMMENT
=====	=====	=====	=====	=====	=====
START			NO		
GIVEN		MSG			
RETURN		MSG			
UPDATE		CONSTANT 0.1			TIME CONSUMED IN UPDATING
END					

LOCAL VARIABLES OF PROCESS ABUPDATE
===== (I) 2 UPDATE (A)
1 MSG
PROCESS
MNEMONIC DESCRIPTION
===== FULL AND HALF DUPLEX CHANNEL LOGIC
CHANPROC

ENTRY	OPCODE	PARM	PARM	PARM	COMMENT
=====	=====	=====	=====	=====	=====
START		ALL	NO		
GIVEN		MSG			SET INTERNAL NODE CURRENT
ASSIGN		\$CNODE	CNODE		GET DESTINATION NODE
ASSIGN		MSG	TNODE		SET NEXT NODE TO DESTN
ASSIGN		\$NXTNODE	TO.NODE		GET CHANNEL TO NEXT NODE
ASSIGN		\$CHANNEL	TO.NODE		OBTAIN CHANNEL FOR XFER
ALLOC		CHANNEL	1	ALL	WHAT IS CHANNEL RATE?
ASSIGN		\$PRIORITY	RATE		MESSAGE LENGTH
ASSIGN		\$VSPEED	LENGTH		
ASSIGN		MSG	VLENGTH		

PAGE 10

```

EVAL          VM.OVHD  MULTIPLY
VSPEED        VLENGTH
XFER.OH       CONSTANT VM.OVHD
ASSIGN        NXT.NODE  CNODE
MSG           NXT.NODE
ASSIGN        $CNODE
DEALLOC       CHANNEL 1
CALL          NODEPROC WAIT 0
GIVEN         MSG
END

```

```

CALCULATE TRANSFER TIME
DELAY DUE TO TRANSFER TIME
MSG RESIDES IN NEXT NODE
SET INTERNAL NODE REGISTER
FREE UP CHANNEL AFTER XFER
ROUTE MESSAGE TO NEXT NODE

```

LOCAL VARIABLES OF PROCESS CHANPROC

```

=====
1 MSG (I) 2 TO.NODE 3 NXT.NODE 4 CHANNEL
5 VSPEED 6 VLENGTH 7 VM.OVHD 8 XFER.OH
9 NODEPROC (P)
=====

```

(A)

PROCESS
MNEMONIC

DESCRIPTION

```

=====
CHQ-DATA CHQ GETS MESSAGE, FORMULATES RESPONSE, AND REPLIES
=====

```

```

ENTRY  OPCODE  PARM  PARM  PARM  COMMENT
=====
START
GIVEN  MSG
RETURN MSG
ASSIGN MSG
LENGTH
VLENGTH
V.TIME
VLENGTH
.016
UPDATE CONSTANT V.TIME
END

```

```

MAKE MSG-LENGTH = V.LENGTH
EVALUATE MSG PROCESS TIME
PROCESSING TIME CONSUMED

```

LOCAL VARIABLES OF PROCESS CHQ-DATA

```

=====
1 MSG (I) 2 V.LENGTH 3 V.TIME 4 UPDATE
=====

```

(A)

PROCESS
MNEMONIC

DESCRIPTION

```

=====
DESTPROC PROCESSING AT DESTINATION OF MESSAGE
=====

```

```

ENTRY  OPCODE  PARM  PARM  PARM  COMMENT
=====
START  ALL
GIVEN  MSG

```

PAGE 11

```

ASSIGN MSG C.NODE C.NODE CURRENT NODE
COMPARE MSG TYPE EQ IF RESPONSE, DESTROY
$RESP C.NODE 1 DESTROY
$PRIORITY RPROC ALL ALLOCATE CURRENT NODE
MSG PROCESS EXECUTE THE CALLED PROCESS
MSG RPROC PRI SET PRIORITY FOR REQ PROC
PRIORITY WAIT PRIORITY WAIT UNTIL COMPLETE
CALL MSG
GIVEN MSG
RETURN MSG
DEALLOC C.NODE 1 DEALLOCATE CURRENT NODE
COMPARE MSG EQ NO RESPONSE REQ -> DESTROY
$REQNORE $RESP CHANGE MSG RESPONSE TYPE
MSG MSG TYPE SWITCH FROM AND TO NODES
MSG MSG TNODE CURRENT NODE IS FROM NODE
MSG MSG CNODE RETURN MESSAGE TO ORIGIN
CALL CHANPROC WAIT 0
GIVEN MSG
BRANCH MSG 100
ENTRY DESTROY
ENTRY MSG
END ENTRY
END

```

DESTROY ENTRY
END ENTRY
END

LOCAL VARIABLES OF PROCESS DESTPROC

```

=====
1 MSG (I) 2 C.NODE 3 PROCESS (X) 4 PRIORITY
5 CHANPROC (P)
=====

```

PROCESS
MNEMONIC

DISK.OP

```

=====
ENTRY OPCODE PARM PARM PARM COMMENT
=====
START NO
GIVEN LENGTH DISK
ASSIGN DISK SPEED MAKE DISK SPEED = V.SPEED
EVAL XFERTIME DIVIDE TRANSFER TIME CALCULATED
LENGTH V.SPEED
ALLOC DISK 1 PARTIAL DISK ALLOCATED
=====

```

```

$PRIORITY      $PRIORITY      $PRIORITY      $PRIORITY      $PRIORITY
DISK            DISK            DISK            DISK            DISK
SEEK            SEEK            SEEK            SEEK            SEEK
UNIFORM        UNIFORM        UNIFORM        UNIFORM        UNIFORM
DISK            DISK            DISK            DISK            DISK
LATENCY        LATENCY        LATENCY        LATENCY        LATENCY
CONSTANT       CONSTANT       CONSTANT       CONSTANT       CONSTANT
XFER           XFER           XFER           XFER           XFER
DEALLOC        DEALLOC        DEALLOC        DEALLOC        DEALLOC
END            END            END            END            END

```

LOCAL VARIABLES OF PROCESS DISK.OP

```

=====
1 LENGTH      2 DISK      3 V.SPEED      4 XFERTIME
5 SEEKTIME    6 SEEK      7 LATETIME      8 LATENCY
9 XFER        (A)
=====

```

PROCESS

MNEMONIC

```

=====
DESCRIPTION
=====
HQ GETS MESSAGE, FORMULATES RESPONSE, AND REPLIES
=====

```

HQ-DATA

```

=====
ENTRY  OPCODE  PARM  PARM  PARM  PARM  COMMENT
=====
START
GIVEN  MSG
RETURN MSG
ASSIGN LENGTH
EVAL   V.LENGTH  MULTIPLY
UPDATE V.TIME    V.LENGTH
END     CONSTANT V.TIME

```

LOCAL VARIABLES OF PROCESS HQ-DATA

```

=====
1 MSG      (I)  2 V.LENGTH      3 V.TIME      4 UPDATE
=====

```

PROCESS

MNEMONIC

```

=====
DESCRIPTION
=====
HQ REQUEST FOR STATUS DISPLAY FROM CHQ
=====

```

HQ-REQ

```

=====
ENTRY  OPCODE  PARM  PARM  PARM  PARM  COMMENT
=====
START
GIVEN  L3
RETURN MSG

```

PAGE 13

CALL	MRS	WAIT	4	MAKES I/O REQUEST TO CHQ
GIVEN	PLANS	4	\$REQRESP	
	200	CHQ	MSG	
END				

LOCAL VARIABLES OF PROCESS HQ-REQ

1 MSG	(I)	2 MRS	(P)	3 PLANS	(P)	4 CHQ	(R)
-------	-----	-------	-----	---------	-----	-------	-----

PROCESS
 MNEMONIC
 MRS

DESCRIPTION
 GENERATE A PROCESS REQUEST MESSAGE AND INITIATE I/O

ENTRY	OPCODE	PARM	PARM	PARM	COMMENT
START	ALL	NO			
GIVEN	PROCESS	PRIORITY	MSG.TYPE		
	MSG	MSG.LNTH	TO.NODE	MSG	
CREATE	MSG				CREATE MESSAGE
ASSIGN	MSG	MSG.LNTH	LENGTH		SET MESSAGE LENGTH
ASSIGN	PROCESS				SET PROCESS
ASSIGN	MSG	PRIORITY	RPROC		SET PRIORITY
ASSIGN	MSG	TO.NODE	RPROC PRI		SET DESTINATION
ASSIGN	MSG	MSG.TYPE	TNODE		SET MESSAGE TYPE
CALL	MSG	NODEPROC	WAIT	0	EXECUTIVE SERVICING OF MSG
GIVEN					
END					

LOCAL VARIABLES OF PROCESS MRS

1 PROCESS	(X)	2 PRIORITY	3 MSG.TYPE	4 MSG.LNTH
5 TO.NODE		6 MSG	(I)	7 NODEPROC (P)

PROCESS
 MNEMONIC
 NODEPROC

DESCRIPTION
 NODAL PROCESSING AND ROUTING

ENTRY	OPCODE	PARM	PARM	PARM	COMMENT
START	ALL	NO			
GIVEN					

```

MSG          C.NODE      C.NODE      INDICATE CURRENT NODE
C.NODE      C.NODE      C.NODE      PROCESSING RATE OF NODE
RT.OVHD     M.ROUTE     RT.OVHD     GET MESSAGE LENGTH
MSG         LENGTH      MSG         COMPUTE PROCESSING DELAY
MSG.LNTH    MULTIPLY    MSG.LNTH    ALLOCATE CURRENT NODE
MSG.LNTH    RT.OVHD     MSG.LNTH    DELAY FOR ROUTING
C.NODE      1          C.NODE      1  RELEASE C.NODE TO OTHERS
$PRIORITY   ALL        C.NODE      EQ  IS MSG AT DESTINATION?
CONSTANT    OVERHEAD   C.NODE      EQ  CONTROL
DEALLOC     C.NODE      TNODE      0  FORWARD MSG TO CHANNEL
COMPARE      MSG        MSG         MESSAGE AT DESTINATION
CALL        CHANPROC   WAIT        CONTEXT SWITCH MESSAGE
GIVEN       MSG        END         100
BRANCH      DESTPROC   WAIT        0
ENTRY       MSG        MSG
CALL        MSG
GIVEN       MSG
ENTRY       MSG
END
CONTROL
END

```

LOCAL VARIABLES OF PROCESS NODEPROC

```

=====
1 MSG      (I)      2 C.NODE      3 RT.OVHD      4 MSG.LNTH
5 OVERHEAD      6 ROUTE.OH      (A)      7 CHANPROC      (P)      8 DESTPROC      (P)
=====

```

PROCESS
MNEMONIC
PLANS

DESCRIPTION

REQUEST FOR PLANS FROM CHQ

ENTRY	OPCODE	PARAM	PARAM	PARAM	COMMENT
START	CHQ			NO	
GIVEN	MSG				
RETURN	MSG				
ASSIGN	MSG				
EVAL	V.TIME	LENGTH			MAKE MSG LENGTH = V. LENGTH
	.01	MULTIPLY			EVALUATE MSG PROCESS TIME
FORMAT	CONSTANT	V.LENGTH			TIME USED TO FORMAT PLANS
CALL	DISK.OP	WAIT	10		CALLING PROCESS DISK.OP
GIVEN	500	DK1			
ASSIGN	500	MSG			INCREASE MSG LENGTH
END					

PAGE 15

LOCAL VARIABLES OF PROCESS PLANS

===== (A)
1 MSG (I) 2 V. LENGTH 3 V. TIME 4 FORMAT
5 DISK. OP (P) 6 DK1 (R)

PROCESS
MNEMONIC DESCRIPTION
===== TURN ON TRACE OUTPUT
TRACE

ENTRY	OPCODE	PARM	PARM	PARM	COMMENT
=====	=====	=====	=====	=====	=====
START	ALL	NO			
COMPARE	V. TRACE				
	0				
TRACE	ON				
ENTRY					
END					

EQ TEST IF FLAG SET FOR TRACE
NOTRACE

LOAD DEFINITION.....

LOAD
MNEMONIC DESCRIPTION
===== COMMUNICATIONS FROM AIRBASES
ABLOAD
LOAD
NODES
===== AB1 AB2

PROCESS	MNEMONIC	MAX #	SCHEDULE	METHOD	MEAN	DELTA	PRIORITY
=====	=====	=====	=====	=====	=====	=====	=====
AB-DATA	60		INTERVAL	ABDRATE			10
AB-REQ	60		EXPONENT	ABRRATE			6

LOAD
MNEMONIC DESCRIPTION
===== REQUEST DATA FROM CHQ
HQLoad
NODES
===== L3

PROCESS	MNEMONIC	MAX #	SCHEDULE	METHOD	MEAN	DELTA	PRIORITY
=====	=====	=====	=====	=====	=====	=====	=====
HQ-REQ	60		EXPONENT	HQRRATE			4

PAGE 18

SCENARIO DEFINITION....

SCENARIO DESCRIPTION
=====

TEST01 SCENARIO FOR MINI MITRE 1

PERIOD

LENGTH

=====

360000

PERIOD	PERIOD	PERIOD	PERIOD	PERIOD	PERIOD	PERIOD	PERIOD
MNEMONIC	MNEMONIC	MNEMONIC	MNEMONIC	MNEMONIC	MNEMONIC	MNEMONIC	MNEMONIC
1	2	3	4	5	6	7	
PERIOD	PERIOD	PERIOD	PERIOD	PERIOD	PERIOD	PERIOD	PERIOD
MNEMONIC	MNEMONIC	MNEMONIC	MNEMONIC	MNEMONIC	MNEMONIC	MNEMONIC	MNEMONIC
8	9	10					

TRIGGER	TIME	TO	SCHEDULE	TRIGGER	TIME	TO	SCHEDULE
MNEMONIC	SCHEDULE	PRIORITY	MNEMONIC	SCHEDULE	PRIORITY	MNEMONIC	SCHEDULE
ABLOAD	0	0	0	HQLOAD	0	0	0
TRACE	0	0	0				

0 ERRORS WERE DETECTED DURING MODEL INITIALIZATION

PAGE 17

SIMULATION TIME = 3600000.00000 UNITS

CONSTANT REPORT

CURRENT

CONSTANT VALUE...

=====

V. TRACE 0.

PAGE 18

SIMULATION TIME = 360000.0000 UNITS

VARIABLE REPORT

NUMERIC VARIABLES...

TOTAL		VALUE				
VARIABLE	SAMPLES	CURRENT	MEAN	STD DEV	MINIMUM	MAXIMUM
ABDRATE	1	60000.000	60000.000	0.	60000.000	60000.000
ABRRATE	1	36000.000	36000.000	0.	36000.000	36000.000
HQRRATE	1	72000.000	72000.000	0.	72000.000	72000.000
TIME1	1	30.000	30.000	0.	30.000	30.000
VRATE	1	1.628	1.628	0.	1.628	1.628

NON-NUMERIC VARIABLES...

VARIABLE	CURRENT
TYPE	VALUE

PAGE 19

SIMULATION TIME = 3600000.00000 UNITS

ITEM REPORT

ITEM NAME	NUMBER CREATED	NUMBER DESTR'D	TIME IN SYSTEM		
			MINIMUM...	AVERAGE...	STD DEV...
MSG	522	522	22805.69	194855.44	71890.94
					38971.35

PAGE 20

SIMULATION TIME = 3600000.00000 UNITS

RESOURCE REPORT

RESOURCE	TOTAL NUMBER	CURRENT	MEAN	STD DEV	MINIMUM	MAXIMUM
AB1						
# IDLE		1.000	0.513	0.500	0.	1.000
REQUEST TIME			4694.304	6063.327	0.	22000.000
INTO BUSY	416					
OUT OF BUSY	416	0.	0.487	0.500	0.	1.000
# BUSY						
BUSY TIME			4221.696	2309.316	0.	6000.000
# INACTIVE		0.	0.	0.	0.	0.
INTO WAIT	416					
OUT OF WAIT	416	0.	0.541	0.973	0.	5.000
# WAITING						
WAIT TIME			4694.304	6063.327	0.	22000.000

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT	MEAN	STD DEV	MINIMUM	MAXIMUM
AB2						
# IDLE		1.000	0.513	0.500	0.	1.000
REQUEST TIME			4329.880	5316.789	0.	22000.000
INTO BUSY	416					
OUT OF BUSY	416	0.	0.487	0.500	0.	1.000
# BUSY						
BUSY TIME			4221.696	2309.317	0.	6000.000
# INACTIVE		0.	0.	0.	0.	0.
INTO WAIT	416					
OUT OF WAIT	416	0.	0.499	0.911	0.	5.000
# WAITING						
WAIT TIME			4329.880	5316.789	0.	22000.000

PAGE 21

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT...	MEAN.....	STD DEV...	MINIMUM...	MAXIMUM...
CH1.A						
# IDLE		1.000	0.935	0.247	0.	1.000
REQUEST TIME						
INTO BUSY	237					
OUT OF BUSY	237	0.	0.065	0.247	0.	1.000
# BUSY			994.094	389.263	325.500	1220.750
BUSY TIME						
# INACTIVE		0.	0.	0.	0.	0.
INTO WAIT	237					
OUT OF WAIT	237	0.	0.	0.	0.	1.000
# WAITING			0.	0.	0.	0.
WAIT TIME						

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT...	MEAN.....	STD DEV...	MINIMUM...	MAXIMUM...
CH1.B						
# IDLE		1.000	0.966	0.180	0.	1.000
REQUEST TIME						
INTO BUSY	119					
OUT OF BUSY	119	0.	0.034	0.180	0.	1.000
# BUSY			1015.532	203.448	813.750	1220.750
BUSY TIME						
# INACTIVE		0.	0.	0.	0.	0.
INTO WAIT	119					
OUT OF WAIT	119	0.	0.	0.	0.	1.000
# WAITING			0.	0.	0.	0.
WAIT TIME						

CURRENTLY ALLOCATED
TO PROCESSES: NONEPROCESSES CURRENTLY
WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT	MEAN	STD DEV	MINIMUM	MAXIMUM
CH2.A						
# IDLE		1.000	0.936	0.247	0.	1.000
REQUEST TIME			0.	0.	0.	0.
INTO BUSY	237					
OUT OF BUSY	237					
# BUSY		0.	0.066	0.247	0.	1.000
BUSY TIME			994.092	389.262	325.500	1220.750
# INACTIVE		0.	0.	0.	0.	0.
INTO WAIT	237					
OUT OF WAIT	237					
# WAITING		0.	0.	0.	0.	1.000
WAIT TIME			0.	0.	0.	0.

CURRENTLY ALLOCATED
TO PROCESSES: NONEPROCESSES CURRENTLY
WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT	MEAN	STD DEV	MINIMUM	MAXIMUM
CH2.B						
# IDLE		1.000	0.966	0.180	0.	1.000
REQUEST TIME			0.	0.	0.	0.
INTO BUSY	119					
OUT OF BUSY	119					
# BUSY		0.	0.034	0.180	0.	1.000
BUSY TIME			1015.533	203.447	813.750	1220.750
# INACTIVE		0.	0.	0.	0.	0.
INTO WAIT	119					
OUT OF WAIT	119					
# WAITING		0.	0.	0.	0.	1.000

PAGE 23

WAIT TIME

0. 0. 0. 0.

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT...	MEAN.....	STD DEV...	MINIMUM...	MAXIMUM...
CH3.A						
# IDLE		1.000	0.990	0.100	0.	1.000
REQUEST TIME			0.	0.	0.	0.
INTO BUSY	118					
OUT OF BUSY	118					
# BUSY		0.	0.010	0.100	0.	1.000
BUSY TIME			305.088	0.079	305.000	305.188
# INACTIVE		0.	0.	0.	0.	0.
INTO WAIT	118					
OUT OF WAIT	118					
# WAITING		0.	0.	0.	0.	1.000
WAIT TIME			0.	0.	0.	0.

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT...	MEAN.....	STD DEV...	MINIMUM...	MAXIMUM...
CH3.B						
# IDLE		1.000	1.000	0.	1.000	1.000
REQUEST TIME			0.	0.	0.	0.
INTO BUSY	0					
OUT OF BUSY	0					
# BUSY		0.	0.	0.	0.	0.
BUSY TIME			0.	0.	0.	0.
# INACTIVE		0.	0.	0.	0.	0.
INTO WAIT	0					
OUT OF WAIT	0					

PAGE 24

WAITING 0. 0. 0. 0. 0. 0.
WAIT TIME

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT...	MEAN.....	STD DEV...	MINIMUM...	MAXIMUM...
CH4.A						
# IDLE		1.000	0.987	0.112	0.	1.000
REQUEST TIME			0.	0.	0.	0.
INTO BUSY	238					
OUT OF BUSY	238					
# BUSY		0.	0.013	0.112	0.	1.000
BUSY TIME			192.294	111.850	81.375	306.188
# INACTIVE		0.	0.	0.	0.	0.
INTO WAIT	238					
OUT OF WAIT	238					
# WAITING		0.	0.	0.	0.	1.000
WAIT TIME			0.	0.	0.	0.

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT...	MEAN.....	STD DEV...	MINIMUM...	MAXIMUM...
CH4.B						
# IDLE		1.000	0.993	0.082	0.	1.000
REQUEST TIME			0.	0.	0.	0.
INTO BUSY	120					
OUT OF BUSY	120					
# BUSY		0.	0.007	0.082	0.	1.000
BUSY TIME			203.448	0.037	203.375	203.500
# INACTIVE		0.	0.	0.	0.	0.
INTO WAIT	120					

PAGE 25

OUT OF WAIT 120 0. 0. 0. 1.000
 # WAITING 0. 0. 0. 0.
 WAIT TIME

CURRENTLY ALLOCATED
 TO PROCESSES: NONE

PROCESSES CURRENTLY
 WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT...	MEAN.....	STD DEV...	MINIMUM...	MAXIMUM...
CH5.A						
# IDLE		1.000	0.999	0.033	0.	1.000
REQUEST TIME			0.	0.	0.	0.
INTO BUSY	48					
OUT OF BUSY	48					
# BUSY		0.	0.001	0.033	0.	1.000
BUSY TIME			81.408	0.078	81.250	81.500
# INACTIVE		0.	0.	0.	0.	0.
INTO WAIT	48					
OUT OF WAIT	48					
# WAITING		0.	0.	0.	0.	1.000
WAIT TIME			0.	0.	0.	0.

CURRENTLY ALLOCATED
 TO PROCESSES: NONE

PROCESSES CURRENTLY
 WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT...	MEAN.....	STD DEV...	MINIMUM...	MAXIMUM...
CH5.B						
# IDLE		1.000	0.997	0.052	0.	1.000
REQUEST TIME			0.	0.	0.	0.
INTO BUSY	48					
OUT OF BUSY	48					
# BUSY		0.	0.003	0.052	0.	1.000
BUSY TIME			203.447	0.083	203.250	203.500
# INACTIVE		0.	0.	0.	0.	0.

PAGE 26

INTO WAIT	48								
OUT OF WAIT	48								
# WAITING		0.	0.	0.	0.	0.	0.	0.	1.000
WAIT TIME			0.	0.	0.	0.	0.	0.	0.

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT	MEAN	STD DEV	MINIMUM	MAXIMUM
CH6.A						
# IDLE		1.000	0.989	0.104	0.	1.000
REQUEST TIME			0.	0.	0.	0.
INTO BUSY	48					
OUT OF BUSY	48					
# BUSY		0.	0.011	0.104	0.	1.000
BUSY TIME			813.798	0.075	813.750	814.000
# INACTIVE		0.	0.	0.	0.	0.
INTO WAIT	48					
OUT OF WAIT	48					
# WAITING		0.	0.	0.	0.	1.000
WAIT TIME			0.	0.	0.	0.

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT	MEAN	STD DEV	MINIMUM	MAXIMUM
CH6.B						
# IDLE		1.000	0.996	0.066	0.	1.000
REQUEST TIME			0.	0.	0.	0.
INTO BUSY	48					
OUT OF BUSY	48					
# BUSY		0.	0.004	0.066	0.	1.000
BUSY TIME			325.524	0.084	325.500	325.750
# INACTIVE		0.	0.	0.	0.	0.

INTO WAIT 48
 OUT OF WAIT 48
 # WAITING 0.
 WAIT TIME 0. 0. 0. 0. 1.000 0.

CURRENTLY ALLOCATED
 TO PROCESSES: NONE

PROCESSES CURRENTLY
 WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT...	MEAN.....	STD DEV...	MINIMUM...	MAXIMUM...
CH7.A						

IDLE
 REQUEST TIME 1.000 0.996 0.088 0. 1.000 0.

INTO BUSY 48
 OUT OF BUSY 48
 # BUSY 0.
 BUSY TIME 0.004 325.615 0.066 0. 1.000 325.750

INACTIVE 0. 0. 0. 0. 0. 0.

INTO WAIT 48
 OUT OF WAIT 48
 # WAITING 0.
 WAIT TIME 0. 0. 0. 0. 1.000 0.

CURRENTLY ALLOCATED
 TO PROCESSES: NONE

PROCESSES CURRENTLY
 WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT...	MEAN.....	STD DEV...	MINIMUM...	MAXIMUM...
CH7.B						

IDLE
 REQUEST TIME 1.000 0.949 0.220 0. 1.000 0.

INTO BUSY 168
 OUT OF BUSY 168
 # BUSY 0.
 BUSY TIME 0.051 1103.069 0.220 0. 1.000 1220.750

PAGE 28

INACTIVE

168 INTO WAIT
168 OUT OF WAIT
WAITING
WAIT TIME

0. 0. 0. 0. 0.
0. 0. 0. 0. 1.000
0. 0. 0. 0. 0.

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT...	MEAN.....	STD DEV....	MINIMUM...	MAXIMUM...
CH8.A						

# IDLE	1.000	0.962	0.191	0.	1.000	4582.375
REQUEST TIME		768.898	1047.385	0.		

168 INTO BUSY
168 OUT OF BUSY
BUSY
BUSY TIME

0. 0.038 0.191 0. 1.000
813.802 0.054 813.750 814.000

INACTIVE

168 INTO WAIT
168 OUT OF WAIT
WAITING
WAIT TIME

0. 0.036 0.312 0. 0.000
768.898 1047.385 0. 4582.375

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT...	MEAN.....	STD DEV....	MINIMUM...	MAXIMUM...
CH8.B						

# IDLE	1.000	0.945	0.228	0.	1.000	0.
REQUEST TIME		0.	0.	0.		

286 INTO BUSY
286 OUT OF BUSY
BUSY
BUSY TIME

0. 0.055 0.228 0. 1.000
694.874 440.708 325.500 1220.750

```

# INACTIVE          0.      0.      0.      0.      0.
INTO WAIT           288
OUT OF WAIT         288
# WAITING           0.      0.      0.      0.      1.000
WAIT TIME           0.      0.      0.      0.      0.

```

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

```

RESOURCE          TOTAL    CURRENT... MEAN... STD DEV... MINIMUM... MAXIMUM...
=====
CH9.A             NUMBER    =====

```

```

# IDLE             1.000    1.000    0.      0.      1.000    1.000
REQUEST TIME       0.      0.      0.      0.      0.

```

```

INTO BUSY          0
OUT OF BUSY        0
# BUSY             0.      0.      0.      0.      0.
BUSY TIME          0.      0.      0.      0.      0.

```

```

# INACTIVE         0.      0.      0.      0.      0.
INTO WAIT          0
OUT OF WAIT        0
# WAITING          0.      0.      0.      0.      0.
WAIT TIME          0.      0.      0.      0.      0.

```

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

```

RESOURCE          TOTAL    CURRENT... MEAN... STD DEV... MINIMUM... MAXIMUM...
=====
CH9.B             NUMBER    =====

```

```

# IDLE             1.000    1.000    0.      0.      1.000    1.000
REQUEST TIME       0.      0.      0.      0.      0.

```

```

INTO BUSY          0
OUT OF BUSY        0
# BUSY             0.      0.      0.      0.      0.

```

PAGE 30

BUSY TIME

INACTIVE

INTO WAIT

OUT OF WAIT

WAITING

WAIT TIME

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT	MEAN	STD DEV	MINIMUM	MAXIMUM
CHQ						

IDLE
REQUEST TIME

INTO BUSY

OUT OF BUSY

BUSY

BUSY TIME

INACTIVE

INTO WAIT

OUT OF WAIT

WAITING

WAIT TIME

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT	MEAN	STD DEV	MINIMUM	MAXIMUM
DK1						

IDLE
REQUEST TIME

INTO BUSY

OUT OF BUSY

PAGE 31

BUSY
BUSY TIME 0. 0.002 0.045 0. 3.658 1.000
44.071 18.937 85.500

INACTIVE 0. 0. 0. 0. 0. 0.

INTO WAIT 168
OUT OF WAIT 168
WAITING 0. 0. 0. 0. 1.000
WAIT TIME 0. 0. 0. 0. 0.

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

RESOURCE TOTAL
NUMBER CURRENT... MEAN... STD DEV... MINIMUM... MAXIMUM...
=====

HQ
IDLE 1.000 0.728 0.445 0. 1.000
REQUEST TIME 1576.327 2742.297 0. 15811.250

INTO BUSY 410
OUT OF BUSY 410
BUSY 0. 0.272 0.445 0. 1.000
BUSY TIME 2385.677 2615.805 0. 6000.063

INACTIVE 0. 0. 0. 0. 0.

INTO WAIT 410
OUT OF WAIT 410
WAITING 0. 0.122 0.380 0. 3.000
WAIT TIME 1073.988 2372.956 0. 15811.250

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

RESOURCE TOTAL
NUMBER CURRENT... MEAN... STD DEV... MINIMUM... MAXIMUM...
=====

L3
IDLE 1.000 0.925 0.263 0. 1.000
REQUEST TIME 69.511 341.462 0. 2224.750

INTO BUSY 96

PAGE 32

OUT OF BUSY	98						
# BUSY	0.	0.075	0.283	0.	0.	1.000	
BUSY TIME		2800.000	1200.000	1800.000	4000.000		
# INACTIVE	0.	0.	0.	0.	0.		
INTO WAIT	98						
OUT OF WAIT	98						
# WAITING	0.	0.002	0.043	0.	0.	1.000	
WAIT TIME		69.511	341.462	0.	2224.750		

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT	MEAN	STD DEV	MINIMUM	MAXIMUM
=====	=====	=====	=====	=====	=====	=====
SW1						
# IDLE	1.000	0.223	0.416	0.	0.	1.000
REQUEST TIME		30346.677	21264.285	0.	84387.313	
INTO BUSY	594					
OUT OF BUSY	594					
# BUSY	0.	0.777	0.416	0.	0.	1.000
BUSY TIME		4707.071	1745.716	1800.000	8000.000	
# INACTIVE	0.	0.	0.	0.	0.	
INTO WAIT	594					
OUT OF WAIT	594					
# WAITING	0.	5.007	5.432	0.	0.	22.000
WAIT TIME		30346.677	21264.285	0.	84387.313	

CURRENTLY ALLOCATED
TO PROCESSES: NONE

PROCESSES CURRENTLY
WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT	MEAN	STD DEV	MINIMUM	MAXIMUM
=====	=====	=====	=====	=====	=====	=====
SW2						
# IDLE	1.000	0.729	0.445	0.	0.	1.000
REQUEST TIME		373.714	1174.642	0.	7902.750	

PAGE 33

INTO BUSY	214						
OUT OF BUSY	214						
# BUSY		0.	0.271	0.445	0.	0.	1.000
BUSY TIME		4684.486	1782.954	1800.000	6000.000		
# INACTIVE		0.	0.	0.	0.	0.	
INTO WAIT	214						
OUT OF WAIT	214						
# WAITING		0.	0.022	0.160	0.	0.	2.000
WAIT TIME		373.714	1174.642	0.	7902.750		

CURRENTLY ALLOCATED TO PROCESSES: NONE

PROCESSES CURRENTLY WAITING: NONE

RESOURCE	TOTAL NUMBER	CURRENT	MEAN	STD DEV	MINIMUM	MAXIMUM
=====	=====	=====	=====	=====	=====	=====
SW3						
# IDLE		1.000	0.542	0.498	0.	1.000
REQUEST TIME			3857.756	5603.183	0.	24434.500
INTO BUSY	454					
OUT OF BUSY	454					
# BUSY		0.	0.458	0.498	0.	1.000
BUSY TIME		3631.718	1742.260	1599.992	6000.000	
# INACTIVE		0.	0.	0.	0.	0.
INTO WAIT	454					
OUT OF WAIT	454					
# WAITING		0.	0.487	1.150	0.	7.000
WAIT TIME		3857.756	5603.183	0.	24434.500	

CURRENTLY ALLOCATED TO PROCESSES: NONE

PROCESSES CURRENTLY WAITING: NONE

SIMULATION TIME = 300000.00000 UNITS

ACTION REPORT

ACTION	TOTAL SAMPLES	MEAN.....	STD DEV...	MINIMUM..	MAXIMUM...	% TIME OF TOTAL.
FORMAT						
USEFUL TIME	188	2.000	0.	2.000	2.000	0.009
DELAY TIME	188	1523.546	2289.177	0.	9663.500	

ACTION	TOTAL SAMPLES	MEAN.....	STD DEV...	MINIMUM..	MAXIMUM...	% TIME OF TOTAL.
LATENCY						
USEFUL TIME	188	14.678	8.537	0.063	29.938	0.068
DELAY TIME	188	0.	0.	0.	0.	

ACTION	TOTAL SAMPLES	MEAN.....	STD DEV...	MINIMUM..	MAXIMUM...	% TIME OF TOTAL.
ROUTE OH						
USEFUL TIME	2570	4337.743	1878.129	1599.992	6000.063	309.667
DELAY TIME	2570	0.	0.	0.	0.	

ACTION	TOTAL SAMPLES	MEAN.....	STD DEV...	MINIMUM..	MAXIMUM...	% TIME OF TOTAL.
SEEK						
USEFUL TIME	188	29.374	17.214	0.188	59.859	0.137
DELAY TIME	188	0.	0.	0.	0.	

ACTION	TOTAL SAMPLES	MEAN.....	STD DEV...	MINIMUM..	MAXIMUM...	% TIME OF TOTAL.
UPDATE						
USEFUL TIME	354	7.522	5.272	0.	11.250	0.074
DELAY TIME	354	1774.778	3092.568	0.	15611.250	

ACTION	TOTAL SAMPLES	MEAN.....	STD DEV...	MINIMUM..	MAXIMUM...	% TIME OF TOTAL.
XFER						
USEFUL TIME	188	0.019	0.037	0.	0.250	0.000
DELAY TIME	188	0.	0.	0.	0.	

ACTION	TOTAL SAMPLES	MEAN.....	STD DEV...	MINIMUM..	MAXIMUM...	% TIME OF TOTAL.

Page 134

PROCESS REPORT

PROCESS	TOTAL SAMPLES	SUM	MEAN	STD DEV	MINIMUM	MAXIMUM
AB-DATA						
TOTAL	118	0.	0.	0.	0.	0.
PROCESS WAIT	0	0.	0.	0.	0.	0.
RESOURCE WAIT	0	0.	0.	0.	0.	0.

	TOTAL #	# AUTO	# CALL	# OF	# NOT	# TIMES
	SCHEDULE	SCHEDULE	SCHEDULE	COMPLETE	COMPLETE	SUSPEND.
1	1	1	1	1	1	1
2	1	1	1	1	1	1
3	1	1	1	1	1	1
4	1	1	1	1	1	1
5	1	1	1	1	1	1
6	1	1	1	1	1	1
7	1	1	1	1	1	1
8	1	1	1	1	1	1
9	1	1	1	1	1	1
10	1	1	1	1	1	1
11	1	1	1	1	1	1
12	1	1	1	1	1	1
13	1	1	1	1	1	1
14	1	1	1	1	1	1
15	1	1	1	1	1	1
16	1	1	1	1	1	1
17	1	1	1	1	1	1
18	1	1	1	1	1	1
19	1	1	1	1	1	1
20	1	1	1	1	1	1
21	1	1	1	1	1	1
22	1	1	1	1	1	1
23	1	1	1	1	1	1
24	1	1	1	1	1	1
25	1	1	1	1	1	1
26	1	1	1	1	1	1
27	1	1	1	1	1	1
28	1	1	1	1	1	1
29	1	1	1	1	1	1
30	1	1	1	1	1	1
31	1	1	1	1	1	1
32	1	1	1	1	1	1
33	1	1	1	1	1	1
34	1	1	1	1	1	1
35	1	1	1	1	1	1
36	1	1	1	1	1	1
37	1	1	1	1	1	1
38	1	1	1	1	1	1
39	1	1	1	1	1	1
40	1	1	1	1	1	1
41	1	1	1	1	1	1
42	1	1	1	1	1	1
43	1	1	1	1	1	1
44	1	1	1	1	1	1
45	1	1	1	1	1	1
46	1	1	1	1	1	1
47	1	1	1	1	1	1
48	1	1	1	1	1	1
49	1	1	1	1	1	1
50	1	1	1	1	1	1
51	1	1	1	1	1	1
52	1	1	1	1	1	1
53	1	1	1	1	1	1
54	1	1	1	1	1	1
55	1	1	1	1	1	1
56	1	1	1	1	1	1
57	1	1	1	1	1	1
58	1	1	1	1	1	1
59	1	1	1	1	1	1
60	1	1	1	1	1	1
61	1	1	1	1	1	1
62	1	1	1	1	1	1
63	1	1	1	1	1	1
64	1	1	1	1	1	1
65	1	1	1	1	1	1
66	1	1	1	1	1	1
67	1	1	1	1	1	1
68	1	1	1	1	1	1
69	1	1	1	1	1	1
70	1	1	1	1	1	

118	118	0	118	0
-----	-----	---	-----	---

PROCESS	DESCRIPTION	OPCODE	PARM	PARM	COMMENT
AB-DATA	AIR BASE STATUS BROADCAST TO ALL OTHER NODES				

NO	START	MSG	NO	PROCESS REQUEST TO CHQ
118	GIVEN	MRS	NOWAIT	\$REQNORE
118	RETURN	MRS	CHQ	MSG
118	CALL	CHQ-DATA	10	
118	GIVEN	750	NOWAIT	
118	CALL	MRS	10	
118	GIVEN	HQ-DATA	\$REQNORE	PROCESS REQUEST TO HQ
118		750	HQ	MSG
118	ASSIGN	\$CNODE		CURRENT NODE
118		CNODE		
118	COMPARE	CNODE		TEST FOR CURRENT NODE
118		AB1	EQ	
59	CALL	MRS	NOWAIT	AB1
59	GIVEN	ABUPDATE	10	PROCESS REQUEST TO AB1
59		750	AB1	\$REQNORE
59		END	100	MSG
59	BRANCH			BRANCH TO THE END
59	ENTRY			ENTRY FROM COMPARE NODE
59	CALL	MRS	NOWAIT	PROCESS REQUEST TO AB2
59	GIVEN	ABUPDATE	10	
59		750	AB2	\$REQNORE
59				MSG
118	ENTRY			ENTRY FROM REQUEST TO AB1
118	END			

TOTAL

PAGE 37
 PROCESS
 AB-REQ
 TOTAL
 PROCESS WAIT
 RESOURCE WAIT

SAMPLES	SUM	MEAN	STD DEV	MINIMUM	MAXIMUM
120	+1.505E+07	126424.735	36615.237	22805.588	194855.438
120	+1.505E+07	126424.735	36615.237	22805.588	194855.438
120	0	0	0	0	0

TOTAL # AUTO # CALL # OF # NOT # TIMES
 SCHEDULE SCHEDULE SCHEDULE COMPLETE COMPLETE SUSPEND

SAMPLES	SUM	MEAN	STD DEV	MINIMUM	MAXIMUM
120	0	0	0	0	0

PROCESS
 AB-REQ
 COUNT ENTRY
 OPCODE
 PARM
 PARM
 COMMENT

OPCODE	PARM	PARM	COMMENT
START			NO
GIVEN	MSG		
RETURN	MSG		
CALL	MRS		
GIVEN	PLANS	5	PROCESS REQUEST TO CHQ
	200	CHQ	
END			

PROCESS
 AB-REQ
 COUNT ENTRY
 OPCODE
 PARM
 PARM
 COMMENT

OPCODE	PARM	PARM	COMMENT
START			NO
GIVEN	MSG		
RETURN	MSG		
CALL	MRS		
GIVEN	PLANS	5	PROCESS REQUEST TO CHQ
	200	CHQ	
END			

PROCESS
 AB-REQ
 COUNT ENTRY
 OPCODE
 PARM
 PARM
 COMMENT

OPCODE	PARM	PARM	COMMENT
START			NO
GIVEN	MSG		
RETURN	MSG		
CALL	MRS		
GIVEN	PLANS	5	PROCESS REQUEST TO CHQ
	200	CHQ	
END			

TOTAL # AUTO # CALL # OF # NOT # TIMES
 SCHEDULE SCHEDULE SCHEDULE COMPLETE COMPLETE SUSPEND

SAMPLES	SUM	MEAN	STD DEV	MINIMUM	MAXIMUM
118	7.945	0.067	0.059	0	0.125
118	0	0	0	0	0
118	0	0	0	0	0

ITEM
 CREATED
 RECEIVED
 SENT
 DESTR'D

ITEM	CREATED	RECEIVED	SENT	DESTR'D
MSG	0	0	0	0

PROCESS
 AB-REQ
 COUNT ENTRY
 OPCODE
 PARM
 PARM
 COMMENT

OPCODE	PARM	PARM	COMMENT
START			NO
GIVEN	MSG		
RETURN	MSG		
CALL	MRS		
GIVEN	PLANS	5	PROCESS REQUEST TO CHQ
	200	CHQ	
END			

PROCESS
 AB-REQ
 COUNT ENTRY
 OPCODE
 PARM
 PARM
 COMMENT

OPCODE	PARM	PARM	COMMENT
START			NO
GIVEN	MSG		
RETURN	MSG		
CALL	MRS		
GIVEN	PLANS	5	PROCESS REQUEST TO CHQ
	200	CHQ	
END			

===== UPDATE DATA FROM AIRBASE =====

COUNT ENTRY	OPCODE	PARAM	PARAM	PARAM	COMMENT
118	START			NO	
118	GIVEN	MSG			
118	RETURN	MSG			
118	UPDATE	CONSTANT 0.1			TIME CONSUMED IN UPDATING
118	END				

PROCESS TOTAL
 SAMPLES. SUM. MEAN. STD DEV. ... MINIMUM. ... MAXIMUM. ...

CHANPROC
 TOTAL 2048 +7.948E+07 38798.344 33437.938 4813.750 183450.813
 PROCESS WAIT 2048 +7.791E+07 38041.118 33523.332 4000.000 183125.313
 RESOURCE WAIT 2048 129174.794 63.074 366.749 0. 4582.375

TOTAL # # AUTO # CALL # OF # NOT # TIMES
 SCHEDULE SCHEDULE SCHEDULE COMPLETE COMPLETE COMPLETE SUSPEND.
 2048 0 2048 2048 0 0 0

ITEM CREATED RECEIVED SENT DESTR'D
 MSG 0 0 0 0

PROCESS HOLDING TIME
 # SMPLS MEAN. MINIMUM. ... MAXIMUM. ... STD DEV. ...
 MSG 2048 757.23 81.25 5396.19 584.48

PROCESS DESCRIPTION

===== FULL AND HALF DUPLEX CHANNEL LOGIC =====

COUNT ENTRY	OPCODE	PARAM	PARAM	PARAM	COMMENT
2048	START	ALL		NO	
2048	GIVEN	MSG			
2048	ASSIGN	MSG	CNODE		SET INTERNAL NODE CURRENT
2048	ASSIGN	\$CNODE			
2048	ASSIGN	MSG	TNODE		GET DESTINATION NODE
2048	ASSIGN	\$NXTNODE	TO.NODE		SET NEXT NODE TO DESTN
2048	ASSIGN	NXT.NODE			
2048	ASSIGN	\$CHANNEL	TO.NODE		GET CHANNEL TO NEXT NODE
2048		CHANNEL			

PAGE 39

2048	ALLOC	CHANNEL 1	ALL	OBTAIN CHANNEL FOR XFER
2048	PRIORITY			
2048	ASSIGN	CHANNEL RATE		WHAT IS CHANNEL RATE?
2048	VSPEED			
2048	ASSIGN	MSG LENGTH		MESSAGE LENGTH
2048	EVAL	VM.OVHD		
2048	VSPEED	VLENGTH		CALCULATE TRANSFER TIME
2048	XFER.OH	CONSTANT VM.OVHD		DELAY DUE TO TRANSFER TIME
2048	ASSIGN	NXT.NODE		MSG RESIDES IN NEXT NODE
2048	MSG	CNODE		
2048	ASSIGN	NXT.NODE		SET INTERNAL NODE REGISTER
2048	SCNODE			
2048	DEALLOC	CHANNEL 1		FREE UP CHANNEL AFTER XFER
2048	CALL	NODEPROC WAIT	0	ROUTE MESSAGE TO NEXT NODE
2048	GIVEN	MSG		
2048	END			

PROCESS	TOTAL	SAMPLES	SUM	MEAN	STD DEV	MINIMUM	MAXIMUM
CHQ-DATA							
TOTAL	118	33577.578	2843.878	3200.056	11.250	12581.000	
PROCESS WAIT	0	0	0	0	0	0	
RESOURCE WAIT	103	334250.078	3245.146	3284.194	11.250	12569.750	

TOTAL # # AUTO # CALL # OF # NOT # TIMES
SCHEDULE SCHEDULE SCHEDULE COMPLETE COMPLETE COMPLETE SUSPEND.

118 0 118 118 0 103

ITEM CREATED RECEIVED SENT DESTR'D

MSG 0 0 0 0

PROCESS	PROCESS HOLDING TIME
ITEM	# SMPLS MEAN... MINIMUM... MAXIMUM... STD DEV...
MSG	118 2843.88 11.26 12581.00 3200.06

PROCESS DESCRIPTION
CHQ GETS MESSAGE, FORMULATES RESPONSE, AND REPLIES

COUNT ENTRY	OPCODE	PARAM	PARAM	PARAM	COMMENT
118	START				
118	GIVEN	MSG			
118	RETURN	MSG			

PAGE 40

```

118 ASSIGN MSG LENGTH MAKE MSG-LENGTH = V. LENGTH
118 V. LENGTH V. LENGTH
118 EVAL V. TIME MULTIPLY EVALUATE MSG PROCESS TIME
118 .015 V. LENGTH
206 UPDATE CONSTANT V. TIME PROCESSING TIME CONSUMED
118 END

```

```

TOTAL
PROCESS SAMPLES SUM MEAN STD DEV MINIMUM MAXIMUM
DESTRPROC
TOTAL 690 +1.014E+07 14697.168 26948.783 0. 108088.126
PROCESS WAIT 690 +9.726E+06 14094.377 26102.148 0. 106124.188
RESOURCE WAIT 522 416926.656 796.792 2890.029 0. 22000.008

```

```

TOTAL # AUTO # CALL # OF # NOT # TIMES
SCHEDULE SCHEDULE SCHEDULE COMPLETE COMPLETE SUSPEND.
=====
690 0 690 690 0 0

```

```

ITEM CREATED RECEIVED SENT DESTR'D
=====
MSG 0 0 0 522

```

```

PROCESS HOLDING TIME
# SMPLS MEAN MINIMUM MAXIMUM STD DEV
=====
MSG 1212 343.17 0. 22000.01 1937.26

```

```

PROCESS DESCRIPTION
=====
DESTRPROC PROCESSING AT DESTINATION OF MESSAGE
=====

```

```

COUNT ENTRY OPCODE PARM PARM PARM COMMENT
=====
690 START ALL NO
690 GIVEN MSG
690 ASSIGN MSG CNODE CURRENT NODE
690 COMPARE MSG TYPE EQ IF RESPONSE, DESTROY
690 $RESP $RESP DESTROY
522 ALLOC C.NODE 1 ALL ALLOCATE CURRENT NODE
522 $PRIORITY RPROC EXECUTE THE CALLED PROCESS
522 ASSIGN MSG RPROC SET PRIORITY FOR REQ PROC
522 ASSIGN MSG RPROC PRI
522 CALL PRIORITY WAIT PRIORITY WAIT UNTIL COMPLETE
522 GIVEN MSG

```

PAGE 41

```

522 RETURN MSG
522 DEALLOC C.NODE 1 TYPE
522 COMPARE MSG $REQNORE
522 ASSIGN $RESP EQ DESTROY
168 MSG TYPE
168 ASSIGN MSG FNODE
168 MSG TNODE
168 ASSIGN MSG CNODE
168 MSG FNODE
168 CALL CHANPROC WAIT 0
168 GIVEN MSG
168 BRANCH END 100
522 DESTROY ENTRY MSG
622 DESTROY ENTRY MSG
690 END
690 END

```

```

DEALLOCATE CURRENT NODE
NO RESPONSE REQ -> DESTROY
CHANGE MSG RESPONSE TYPE
SWITCH FROM AND TO NODES
CURRENT NODE IS FROM NODE
RETURN MESSAGE TO ORIGIN
TERMINATE MESSAGE AT DEST
TERMINATE MSG

```

```

PROCESS TOTAL
SAMPLES. SUM. .... MEAN. .... STD DEV. ... MINIMUM. ... MAXIMUM. ...
=====
DISK.OP 168 7403.990 44.071 18.937 3.858 85.500
PROCESS WAIT 0 0. 0. 0. 0. 0.
RESOURCE WAIT 168 0. 0. 0. 0. 0.

```

```

TOTAL # # AUTO # CALL # OF # NOT # TIMES
SCHEDULE SCHEDULE SCHEDULE COMPLETE COMPLETE SUSPEND.
=====
168 0 168 168 0 0

```

```

PROCESS DESCRIPTION
=====
DISK.OP OPERATION OF DISK
=====

```

```

COUNT ENTRY OPCODE PARM PARM PARM PARM COMMENT
=====
168 START NO
168 GIVEN LENGTH DISK
168 ASSIGN DISK SPEED MAKE DISK SPEED = V.SPEED
168 EVAL V.SPEED
168 XFERTIME DIVIDE TRANSFER TIME CALCULATED
168 LENGTH V.SPEED
168 ALLOC DISK 1 PARTIAL DISK ALLOCATED
168 $PRIORITY
168 ASSIGN DISK SEEK MAKE SEEKTIME = SEEK
168 SEEKTIME
168 SEEK UNIFORM SEEKTIME SEEKTIME TIME FOR SEEK IS CONSUMED
168 ASSIGN DISK LATENCY MAKE DISK LATENCY=LATETIME

```

LATENCY LATETIME LATETIME LATETIME TIME CONSUMED FOR LATENCY
 XFER UNIFORM CONSTANT XFTIME TRANSFER TIME CONSUMED
 DEALLOC DISK 1 DISK RESOURCE DEALLOCATED
 END

PROCESS TOTAL
 HQ-DATA SAMPLES. SUM. MEAN. STD DEV. MINIMUM. MAXIMUM.
 118 318233.750 2690.896 3636.755 11.250 15822.500
 PROCESS WAIT 0 0. 0. 0. 0.
 RESOURCE WAIT 78 318906.250 4082.901 3788.903 11.250 15811.250

TOTAL # # AUTO # CALL # OF # NOT # TIMES
 SCHEDULE SCHEDULE SCHEDULE COMPLETE COMPLETE SUSPEND.
 118 0 118 118 0 78

ITEM CREATED RECEIVED SENT DESTR'D
 MSG 0 0 0 0

PROCESS HOLDING TIME
 # SMPLS MEAN. MINIMUM. MAXIMUM. STD DEV.
 118 2696.90 11.25 15822.50 3636.78

PROCESS DESCRIPTION
 HQ-DATA HQ GETS MESSAGE, FORMULATES RESPONSE, AND REPLIES

COUNT	ENTRY	OPCODE	PARM	PARM	COMMENT
118	START				
118	GIVEN	MSG			
118	RETURN	MSG			
118	ASSIGN	MSG			
118		V.LENGTH			MAKE MSG-LENGTH = V.LENGTH
118	EVAL	V.TIME			EVALUATE MSG PROCESS TIME
118		.015			
196	UPDATE	CONSTANT	V.TIME		PROCESSING TIME CONSUMED
118	END				

PROCESS TOTAL
 HQ-REQ SAMPLES. SUM. MEAN. STD DEV. MINIMUM. MAXIMUM.
 48 +1.852E+06 38586.016 9495.574 27754.250 63392.188

PAGE 43
 PROCESS WAIT 48 +1.852E+06 38585.016 9495.574 27754.260 63392.188
 RESOURCE WAIT 0 0. 0. 0. 0.

TOTAL # # AUTO # CALL # OF # NOT # TIMES
 SCHEDULE SCHEDULE SCHEDULE COMPLETE COMPLETE SUSPEND.
 =====
 48 48 0 48 0 0

PROCESS DESCRIPTION
 =====
 HQ-REQ HQ REQUEST FOR STATUS DISPLAY FROM CHQ

COUNT ENTRY	OPCODE	PARM	PARM	PARM	COMMENT
48	START	L3	NO		
48	GIVEN	MSG			
48	RETURN	MSG			
48	CALL	MRS	WAIT	4	MAKES I/O REQUEST TO CHQ
48	GIVEN	PLANS	4	\$REQRESP	
48		200	CHQ	MSG	
48	END				

TOTAL
 PROCESS SAMPLES. SUM. MEAN. STD DEV. ... MINIMUM. ... MAXIMUM. ...
 =====
 MRS
 TOTAL 522 +3.753E+07 71890.936 38971.351 22005.588 194855.438
 PROCESS WAIT 522 +3.753E+07 71890.936 38971.351 22005.588 194855.438
 RESOURCE WAIT 0 0. 0. 0.

TOTAL # # AUTO # CALL # OF # NOT # TIMES
 SCHEDULE SCHEDULE SCHEDULE COMPLETE COMPLETE SUSPEND.
 =====
 522 0 522 522 0 0

ITEM CREATED RECEIVED SENT DESTR'D
 =====
 MSG 522 0 0 0

PROCESS HOLDING TIME
 ITEM # SMPLS MEAN. MINIMUM. ... MAXIMUM. ... STD DEV. ...
 =====
 MSG 522 0. 0. 0. 0.

PROCESS DESCRIPTION
 =====
 MRS GENERATE A PROCESS REQUEST MESSAGE AND INITIATE I/O

COUNT ENTRY	OPCODE	PARM	PARM	PARM	COMMENT
48	START	L3	NO		
48	GIVEN	MSG			
48	RETURN	MSG			
48	CALL	MRS	WAIT	4	MAKES I/O REQUEST TO CHQ
48	GIVEN	PLANS	4	\$REQRESP	
48		200	CHQ	MSG	
48	END				

[illegible]

PROCESS	TOTAL SAMPLES.	SUM	MEAN	STD DEV	MINIMUM	MAXIMUM
=====						
PROCESS	2570	+1.154E+08	44916.450	37275.894	4000.000	194855.438
PROCESS WAIT	2570	+8.080E+07	31439.400	33478.749	0.	183450.813
RESOURCE WAIT	2570	+2.349E+07	9139.307	16030.260	0.	84387.313
=====						
NODEPROC						

TOTAL #	# AUTO	# CALL	# OF	# NOT	# TIMES
SCHEDULE	SCHEDULE	SCHEDULE	COMPLETE	COMPLETE	SUSPEND.
2670	0	2570	2670	0	0

ITEM	CREATED	RECEIVED	SENT	DESTR'D
MSG	=====	=====	=====	=====
		0	0	0

ITEM	#	SMPLS	MEAN		MINIMUM	..	MAXIMUM	...	STD DEV	...
MSG	2570		13477.05		1599.99		85987.31		16095.00	

PROCESS	COUNT	ENTRY	OPCODE	PARM	PARM	PARM	COMMENT
NODEPROC	2670		START	ALL	NO		
	2570		GIVEN	MSG			

PAGE 45

2670	ASSIGN	MSG	CNODE	INDICATE CURRENT NODE
2670	ASSIGN	C.NODE	M.ROUTE	PROCESSING RATE OF NODE
2670	ASSIGN	RT.OVHD	LENGTH	GET MESSAGE LENGTH
2670	ASSIGN	MSG	LENGTH	COMPUTE PROCESSING DELAY
2670	EVAL	MSG.LNTH	MULTIPLY	ALLOCATE CURRENT NODE
2670	EVAL	MSG.LNTH	RT.OVHD	DELAY FOR ROUTING
2670	ALLOC	C.NODE	1	RELEASE C.NODE TO OTHERS
2670	ALLOC	SPRIORITY	ALL	IS MSG AT DESTINATION?
2670	ROUTE.OH	CONSTANT	OVERHEAD	FORWARD MSG TO CHANNEL
2670	DEALLOC	C.NODE	1	MESSAGE AT DESTINATION
2670	COMPARE	MSG	CNODE	CONTEXT SWITCH MESSAGE
2670	CALL	MSG	TNODE	
1880	GIVEN	CHANPROC	WAIT	
1880	BRANCH	END	100	
690	ENTRY	DESTPROC	WAIT	
690	CALL	MSG		
2670	GIVEN			
2670	ENTRY			
2670	END			

PROCESS	TOTAL	SAMPLES	SUM	MEAN	STD DEV	MINIMUM	MAXIMUM
PLANS	168	270504.811	1610.148	2264.755	9.000	9723.125	
PROCESS WAIT	168	7483.990	44.071	18.937	3.658	85.500	
RESOURCE WAIT	162	262764.820	1728.716	1645.363	11.250	8180.375	
TOTAL #	168	0	168	168	0	162	
SCHEDULE	168	0	168	168	0	162	
AUTO	168	0	168	168	0	162	
CALL	168	0	168	168	0	162	
# OF	168	0	168	168	0	162	
COMPLETE	168	0	168	168	0	162	
SUSPEND	168	0	168	168	0	162	

ITEM	CREATED	RECEIVED	SENT	DESTR'D
MSG	0	0	0	0
PROCESS	168	1610.15	9.00	9723.13
HOLDING	168	1610.15	9.00	9723.13
TIME	168	1610.15	9.00	9723.13
SMPLS	168	1610.15	9.00	9723.13
MEAN	168	1610.15	9.00	9723.13
MINIMUM	168	1610.15	9.00	9723.13
MAXIMUM	168	1610.15	9.00	9723.13
STD DEV	168	1610.15	9.00	9723.13

PROCESS	DESCRIPTION
PLANS	REQUEST FOR PLANS FROM CHQ

PAGE 46
COUNT ENTRY
=====

OPCODE	PARAM	PARAM	PARAM	COMMENT
START	CHQ	NO		
GIVEN	MSG			
RETURN	MSG			
ASSIGN	MSG			
EVAL	V.TIME	LENGTH		MAKE MSG LENGTH = V.LENGTH
FORMAT	CONSTANT	V.LENGTH		EVALUATE MSG PROCESS TIME
CALL	DISK.OP	WAIT	10	TIME USED TO FORMAT PLANS
GIVEN	500	DK1		CALLING PROCESS DISK.OP
ASSIGN	500	LENGTH		INCREASE MSG LENGTH
END				

PROCESS TOTAL
SAMPLES. SUM. MEAN. STD DEV. MINIMUM. MAXIMUM.
=====

PROCESS	TOTAL	SAMPLES	SUM	MEAN	STD DEV	MINIMUM	MAXIMUM
PROCESS WAIT	1	0.	0.	0.	0.	0.	0.
RESOURCE WAIT	0	0.	0.	0.	0.	0.	0.
	0	0.	0.	0.	0.	0.	0.

TRACE
=====

TOTAL # # AUTO # CALL # OF # NOT # TIMES
SCHEDULE SCHEDULE SCHEDULE COMPLETE COMPLETE COMPLETE SUSPEND.
=====

PROCESS	TOTAL	SAMPLES	SUM	MEAN	STD DEV	MINIMUM	MAXIMUM
PROCESS WAIT	1	0.	0.	0.	0.	0.	0.
RESOURCE WAIT	0	0.	0.	0.	0.	0.	0.
	0	0.	0.	0.	0.	0.	0.

PROCESS DESCRIPTION
=====

TURN ON TRACE OUTPUT
=====

COUNT ENTRY	OPCODE	PARAM	PARAM	PARAM	COMMENT
1	START	ALL	NO		
1	COMPARE	V.TRACE			
0	TRACE	ON			
1	ENTRY				
1	END				

EQ TEST IF FLAG SET FOR TRACE
NOTRACE

END

FILMED

1-86

DTIC