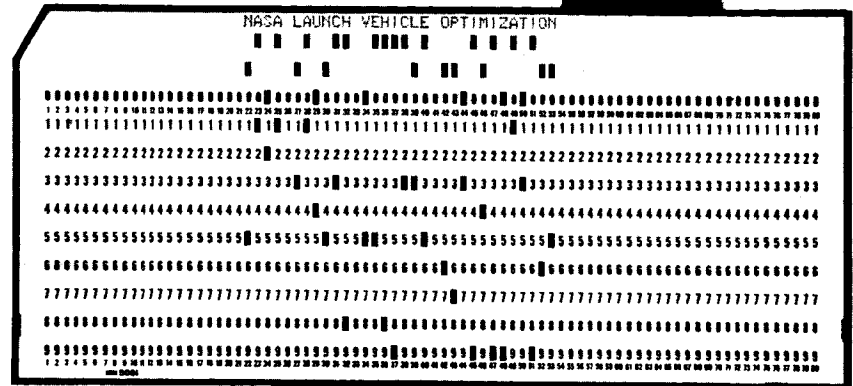


NASA



LAUNCH VEHICLE OPTIMIZATION

FACILITY FOR 4 602	N66-15066	
	(ACCESSION NUMBER)	(THRU)
	<i>294</i>	<i>1</i>
	(PAGES)	(CODE)
	<i>CR 69214</i>	<i>31</i>
	(NASA CR OR TMX OR AD NUMBER)	(CATEGORY)

GPO PRICE \$ _____

CFSTI PRICE(S) \$ _____

Hard copy (HC) *6.00*

Microfiche (MF) *1.50*

653 July 65

PHASE II FINAL REPORT

Volume III
 Techniques Development
 University of Michigan



LEAR SIEGLER PUBLICATION NO. GRR-65-1089-III

to

Office of Launch Vehicle and Propulsion Program

NASA Headquarters

Washington, D.C.

L AUNCH
V EHICLE
O PTIMIZATION

PHASE II FINAL REPORT

September 1965

Prepared by
Instrument Division
Under
Contract NASw-938

LEAR SIEGLER, INC.



INSTRUMENT DIVISION

4141 EASTERN AVENUE, S. E.
GRAND RAPIDS, MICHIGAN 49508

ABSTRACT

VOLUME III

TECHNIQUES DEVELOPMENT - UNIVERSITY OF MICHIGAN

This third part of a three-volume final report contains documentation of a unique approach to the simulation of the behavior of complex physical systems, performed as part of a Launch Vehicle System Optimization (LVO) Program. This approach has been adapted to a digital computer and is known as the Performance Simulator. The methodology utilized within the Simulator in solving the system problem is one which proceeds from "desired results" through an analysis of an "Element Descriptor Library" to form an algorithm which models system performance in terms of known input parameters. The most critical portion of the methodology is the compilation of the "Element Descriptor Library" which contains the physical laws governing the behavior of the constituent elements of the system. An ancillary program, Regression, which may be utilized in developing the Library, is also documented in this volume.

FOREWORD

VOLUME III

TECHNIQUES DEVELOPMENT - UNIVERSITY OF MICHIGAN

A final report in three Volumes is herewith submitted by Lear Siegler, Inc., Instrument Division to the National Aeronautics and Space Administration Headquarters in fulfillment of Contract NASw-938. The study, entitled Launch Vehicle Optimization Study -- Phase II, was pursued under the technical direction of the Launch Vehicle and Propulsion Program Office, Code SV, NASA Headquarters, by the following participant organizations:

- LSI Instrument Division
- LSI Defense Systems Operations
- The University of Michigan

This summary report of the Phase II Launch Vehicle Optimization Study is contained in three separate volumes.

- Volume I
Concept Development
And Application
Volume I contains a general review of the program, an exposition of the PRESTO concept and techniques, a presentation of its application to a Study Launch Vehicle System, and a discussion of special problems and significant achievements.
- Volume II
Techniques
Development -
Lear Siegler, Inc.
Volume II contains a comprehensive review of the PRESTO simulation and optimization techniques, as formulated and applied at LSI, in addition to a description of the Study Launch Vehicle System to be analyzed.
- Volume III
Techniques
Development -
University of Michigan
Volume III contains a report of related efforts compiled by the University of Michigan under the direction of Dr. Frank H. Westervelt. It includes documentation on the Westervelt Performance Simulator and the U of M Regression Routine.

PART A

PERFORMANCE SIMULATOR MANUAL

FOREWORD

The Performance Simulator is a computer program designed specifically for the purpose of constructing a wide variety of special computer programs that have the ability to simulate or represent the behavior of systems. The types of systems that may be so treated are limited only by the availability of descriptors for the components or elements of the system.

The creation of these descriptors is by no means a trivial task. Indeed, the development of a logically complete and consistent set of descriptors is the most difficult task facing the user of the Performance Simulator Program.

However, it should be recognized that two groups of users exist. In many cases, the two merge into a single person but it is important to identify both types:

1. The Systems Analyst whose task it is to procure, encode, and test the descriptors for the system elements.
2. The Systems Synthesist whose task it is to configure collections of elements to form systems and study the behavior of these aggregations.

The following material is designed to serve as a self-contained reference manual for both the Systems Analyst and Systems Synthesist.

TABLE OF CONTENTS

Section		Page No.
1	INTRODUCTION	A-5
2	SOURCE PROGRAM	A-6
2.1	System Description	A-6
2.1.1	Elements	A-6
2.1.1.1	Element Names	A-6
2.1.1.2	Element Identifiers	A-7
2.1.2	Attachments	A-7
2.1.2.1	Attachment Names	A-7
2.1.2.2	Attachment Branch Identifiers	A-8
2.1.3	Connections	A-10
2.1.3.1	CONNECTIONS. Declaration	A-10
2.1.3.2	Connection Statements	A-10
2.2	Parameters	A-11
2.2.1	Input Parameters	A-12
2.2.1.1	INPUT PARAMETERS. Declaration	A-12
2.2.1.2	Input Parameters Statements	A-12
2.2.2	Desired Results	A-13
2.2.2.1	DESIRED RESULTS. Declaration	A-13
2.2.2.2	Desired Result Statements	A-13
2.3	New Element Tape	A-14
2.3.1	Prologue	A-15
2.3.2	Epilogue	A-15
2.3.3	Calculations	A-15
2.3.4	Special Indicators	A-16
2.3.4.1	Meaning and Use of (-) Symbol	A-16
2.3.4.2	Meaning and Use of (+) Symbol	A-17
2.3.4.3	Meaning and Use of (⊕) Symbol	A-17
3	ELEMENT DESCRIPTOR LIBRARY	A-19
3.1	Element Description	A-19
3.1.1	ELEMENT DESCRIPTION. Declaration	A-19
3.1.2	NAME OF ELEMENT Assertion	A-19
3.1.3	ATTACHMENT NAMES Assertion	A-20
3.1.4	BROAD SCOPE PARAMETERS Assertion	A-20
3.1.5	Statement Collection	A-21

TABLE OF CONTENTS (cont)

Section		Page No.
3.1.5.1	STATEMENT COLLECTION. Declaration	A-21
3.1.5.2	Capability Statement	A-21
3.1.5.3	Calculational Statements	A-24
3.1.5.4	End of Statement Collection Declarations	A-29
3.1.6	DESCRIPTION FINISHED. Declaration	A-29
4	UTILITY DECLARATIONS	A-30
4.1	Function Substitutions	A-30
4.1.1	FUNCTION SUBSTITUTIONS. Declaration	A-30
4.1.2	Function Substitution Statements	A-30
4.2	Equivalence	A-32
4.2.1	EQUIVALENCE. Declaration	A-32
4.2.2	Equivalence Statements	A-32
4.3	Diagnostic Aids	A-33
4.3.1	CHECKOUT. and CHECKRUN. Declarations	A-33
5	PERFORMANCE SIMULATOR LOGIC	A-34
5.1	Discussion of Performance Simulator Logic	A-35
6	PERFORMANCE SIMULATOR SYSTEM DIAGRAMMING	A-38
7	COMPLETE EXAMPLE	A-40
7.1	Problem Considered	A-40
7.2	Source Program	A-41
7.3	Element Descriptor Library	A-42
7.4	Generated Program	A-44
7.5	Complete Program	A-45
7.6	Numerical Calculations	A-46
8	PERFORMANCE SIMULATOR STRUCTURAL DETAILS	A-50

1 INTRODUCTION

The Performance Simulator utilizes the capability of modern digital computers to produce a program which will simulate the performance of a system in machine translatable language.

The information that is required by the Performance Simulator consists of the following:

- a. Source Program.
- b. Element Descriptor Library.

The Source Program describes the particular problem under consideration to the Performance Simulator. The Element Descriptor Library describes the physical laws or relationships for each element in the system to the Performance Simulator.

This manual was written to delineate the rules for communicating with the Performance Simulator and to provide a complete discussion of the logic used by the Performance Simulator. The reader is assumed to be generally familiar with Michigan Algorithmic Decoder (MAD) machine language.

The Performance Simulator Manual is intended to give enough detailed information so that a person can use the Performance Simulator effectively to produce a system performance simulation. Some sections of this manual may require careful thought to realize fully the capability offered by the Performance Simulator.

The Performance Simulator is currently available on both the 704 and 7090 IBM computers. It could, however, be adapted to any digital computer. All information will be conveyed to the Performance Simulator on 80-column IBM cards.

2 SOURCE PROGRAM

The Performance Simulator is a digital computer program designed to simulate a physical system in the form of a computer program in machine translatable language. The information required by the Performance Simulator includes the Source Program which consists of the following:

- a. System Description
- b. Input Parameters
- c. Desired Results
- d. New Element Tape

2.1 SYSTEM DESCRIPTION

A physical system consists of a collection of interconnected elements. In order to simulate such a system, it must be described to the Performance Simulator in an unambiguous manner. This requires the communication of the following information:

- a. The elements that comprise the system.
- b. The attachments associated with each of these elements.
- c. The interconnection of these attachments in the system.

2.1.1 Elements

Each element in the system must be uniquely identified to the Performance Simulator. This identification is implemented through use of an Element Name and, if necessary, an Element Identifier.

2.1.1.1 Element Names

Element Names are restricted to six or less alpha-numeric characters. No restrictions are placed on the choice or ordering of these characters, but it is most beneficial to choose the name so that it conveys a close correspondence to the normal name. Once an element has been given a name, all reference to this particular element must agree exactly with the given name.

Examples of Element Names

<u>Element</u>	<u>Element Name</u>
Resistor	RES
Battery	BATT
Amplifier	AMPL

2. 1. 1. 2 Element Identifiers

If a system has more than one element with the same name, Element Identifiers must be used to avoid ambiguity. Element Identifiers are restricted to six or less alpha-numeric characters. No restrictions are placed on the choice or ordering of these characters.

Once an element has been identified in a system, all reference to this particular element must contain both the Element Name and Element Identifier. If a reference is made to an identified element by the Element Name alone, the Performance Simulator interprets this as a reference to all elements in the system with the same given name.

The Element Identifier follows the Element Name and is separated from it by a comma.

Examples of Element Names With Element Identifiers

<u>Element</u>	<u>Element Name</u>	<u>Element Identifier</u>	<u>E.N., E.I.</u>
Resistor	RES	131	RES, 131
Battery	BATT	A1	BATT, A1
Amplifier	AMPL	15	AMPL, 15

2. 1. 2 Attachments

Each attachment associated with a particular element must be uniquely identified to the Performance Simulator. This identification is implemented through the use of an Attachment Name. Should the system interconnections require that a particular attachment branch out to more than one other attachment, each of these branches must also be uniquely identified. This identification is implemented through the use of Attachment Branch Identifiers.

2. 1. 2. 1 Attachment Names

Attachment Names are restricted to six or less alpha-numeric characters. No restrictions are placed on the choice or ordering of these characters, but it is most beneficial to choose the name so that it conveys a close correspondence to the normal name. Once an attachment has been given a name, all reference to this particular attachment must agree exactly with the given name.

The Attachment Name follows the Element Identifier in parentheses. If the Element Identifier can be omitted, the Attachment Name follows the Element Name in parentheses.

Examples of Attachment Names

<u>Element</u>	<u>Element Name</u>	<u>Attachment Name</u>
Resistor	RES	1
Battery	BATT	POS
Amplifier	AMPL	PLATE

2.1.2.2 Attachment Branch Identifiers

If there is more than one branch from an attachment, Attachment Branch Identifiers must be used to avoid ambiguity. Attachment Branch Identifiers are restricted to six or less alpha-numeric characters. No restrictions are placed on the choice or ordering of these characters.

The Attachment Branch Identifier follows the Attachment Name and is separated from it by a comma. The pair is enclosed in parentheses.

Examples of Attachment Names With Attachment Branch Identifiers

<u>Element</u>	<u>Attachment Name</u>	<u>Attachment Branch Identifier</u>	<u>(A. N., A. B. I.)</u>
Resistor	1	Q	(1, Q)
Battery	POS	D7	(POS, D7)
Amplifier	PLATE	63	(PLATE, 63)

There are four cases that occur when defining an unambiguous attachment branch in a system. The statement, which defines an unambiguous attachment branch, is called an Attachment Branch Address.

Case 1. Element Identifiers and Attachment Branch Identifiers Required.

					Attachment Branch Address
<u>Element</u>	<u>E. N.</u>	<u>E. I.</u>	<u>A. N.</u>	<u>A. B. I.</u>	<u>E. N. , E. I. (A. N. , A. B. I.)</u>
Resistor	RES	131	1	Q	RES, 131(1, Q)
Battery	BATT	A1	POS	D7	BATT, A1(POS, D7)
Amplifier	AMPL	15	PLATE	63	AMPL, 15(PLATE, 63)

Case 2. Element Identifiers Not Required and Attachment Branch Identifiers Required.

				Attachment Branch Address
<u>Element</u>	<u>E. N.</u>	<u>A. N.</u>	<u>A. B. I.</u>	<u>E. N. (A. N. , A. B. I.)</u>
Resistor	RES	1	Q	RES(1, Q)
Battery	BATT	POS	D7	BATT(POS, D7)
Amplifier	AMPL	PLATE	63	AMPL(PLATE, 63)

Case 3. Element Identifiers Required and Attachment Branch Identifiers Required.

				Attachment Branch Address
<u>Element</u>	<u>E. N.</u>	<u>E. I.</u>	<u>A. N.</u>	<u>E. N. , E. I. (A. N.)</u>
Resistor	RES	131	1	RES, 131(1)
Battery	BATT	A1	POS	BATT, A1(POS)
Amplifier	AMPL	15	PLATE	AMPL, 15(PLATE)

Case 4. Element Identifiers and Attachment Branch Identifiers Not Required.

			Attachment Branch Address
<u>Element</u>	<u>E. N.</u>	<u>A. N.</u>	<u>E. N. (A. N.)</u>
Resistor	RES	1	RES(1)
Battery	BATT	POS	BATT(POS)
Amplifier	AMPL	PLATE	AMPL(PLATE)

The four cases given above encompass all possible ways of defining an unambiguous attachment branch in a system.

2.1.3 Connections

The interconnection of the element attachments in the system describes the particular system configuration to the Performance Simulator. These interconnections are conveyed to the Performance Simulator through the use of Connection Statements. These Connection Statements must be preceded by the declaration CONNECTIONS.

2.1.3.1 CONNECTIONS. Declaration (Abbreviation CN's)

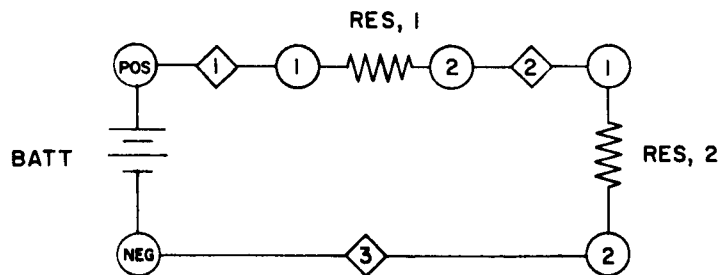
The declaration CONNECTIONS. must precede the Connection Statements. The declaration CONNECTIONS., which is located in columns 2 through 13, prepares the Performance Simulator to receive the Connection Statements.

2.1.3.2 Connection Statements

A Connection Statement conveys to the Performance Simulator an unambiguous interconnection of two attachments in a system. A Connection Statement consists of two uniquely defined Attachment Branch Addresses separated by the word TO which is set off by commas. Connection Statements will be located one per card starting in column 5.

Examples Illustrating Connection Statements

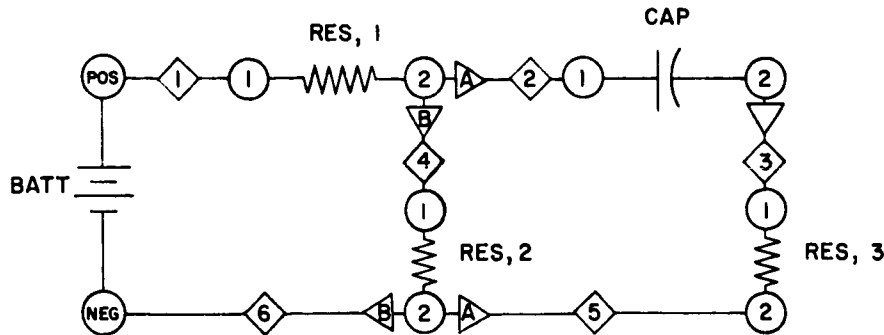
Example 1.



The following Connection Statements convey the interconnections of the above system to the Performance Simulator in an unambiguous manner.

Columns	1	2	5
	<u>CONNECTIONS.</u>		
	BATT(POS), TO, RES, 1(1)		
	RES, 1(2), TO, RES, 2(1)		
	RES, 2(2), TO, BATT(NEG)		

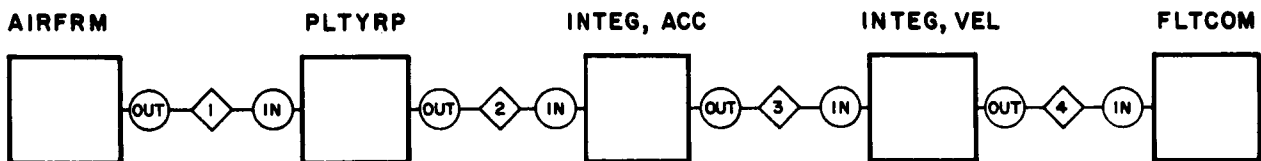
Example 2.



The following Connection Statements convey the interconnections of the above system to the Performance Simulator in an unambiguous manner.

Column	1	2	5
			CONNECTIONS.
			BATT(POS), TO, RES, 1(1)
			RES, 1(2, A), TO, CAP(1)
			CAP(2), TO, RES, 3(1)
			RES, 1(2, B), TO, RES, 2(1)
			RES, 2(2, A), TO, RES, 3(2)
			RES, 2(2, B), TO, BATT(NEG)

Example 3.



The following Connection Statements convey the interconnections of the above system to the Performance Simulator in an unambiguous manner.

Columns	1	2	5
			CONNECTIONS.
			AIRFRM(OUT), TO, PLTYRP(IN)
			PLTYRP(OUT), TO, INTEG, ACC(IN)
			INTEG, ACC(OUT), TO, INTEG, VEL(IN)
			INTEG, VEL(OUT), TO, FLTCOM(IN)

2.2 PARAMETERS

Parameters are used to describe the characteristics or physical relationships of elements that compose the system. Some examples of parameters are: voltage, resistance, current, and power. Each parameter used in the system must be uniquely identified to the Performance

Simulator. This identification is implemented through use of Parameter Names.

Parameter Names are restricted to six or less alpha-numeric characters. The first character of the Parameter Name must be an alphabetic character. No restrictions are placed on the choice or ordering of the remaining characters but it is most beneficial to choose the name so that it conveys a close correspondence to the normal name. Once a parameter has been given a name, all reference to this particular parameter must agree exactly with the given name.

2.2.1 Input Parameters

Input Parameters are the parameters which the user is willing to specify in order to obtain some desired results. Input Parameters are conveyed to the Performance Simulator through Input Parameter Statements, which must be preceded by the declaration INPUT PARAMETERS. .

2.2.1.1 INPUT PARAMETERS. Declaration (Abbreviation IN's)

The declaration INPUT PARAMETERS. must precede the Input Parameter Statements. The declaration INPUT PARAMETERS., which is located in columns 2 through 18, prepares the Performance Simulator to receive the Input Parameter Statements which follow it.

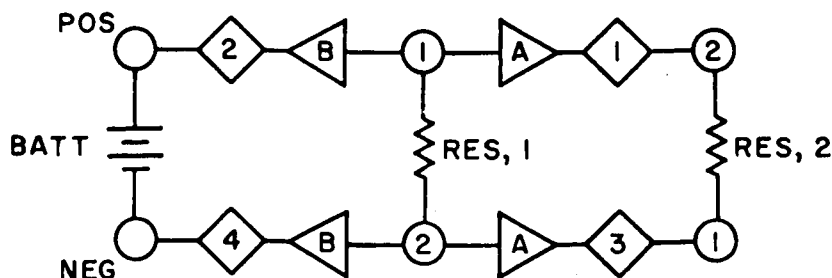
2.2.1.2 Input Parameter Statements

The Input Parameter Statements communicate the Input Parameters to the Performance Simulator. All Input Parameters are conveyed to the Performance Simulator at a uniquely defined attachment branch which may or may not be identified.

Input Parameter Statements consist of a Parameter Name followed by an Attachment Branch Address which is enclosed in parentheses. The Parameter Name must correspond exactly to the name assigned to this particular parameter in the Element Descriptor Library.

The Input Parameter Statements will be located starting in column 5.

Examples Illustrating Input Parameter Statements



Suppose the user wishes to declare the following parameters as Input Parameters:

Voltage at the positive terminal of the battery.

Current in attachment branch labeled A of attachment 1 of resistor 1.

In this case the following statements would be appropriate:

Columns	1	2	5
			INPUT PARAMETERS.
			VOLTS(BATT(POS))
			AMPS(RES, 1(1, A))

2.2.2 Desired Results

Desired Results are the parameters which the user wishes to obtain as a function of the Input Parameters. Desired Results are conveyed to the Performance Simulator through Desired Result Statements, which must be preceded by the declaration DESIRED RESULTS.

2.2.2.1 DESIRED RESULTS. Declaration (Abbreviation DS's)

The declaration DESIRED RESULTS. must precede the Desired Result Statements. The declaration DESIRED RESULTS., which is located in columns 2 through 17, prepares the Performance Simulator to receive the Desired Result Statements which follow it.

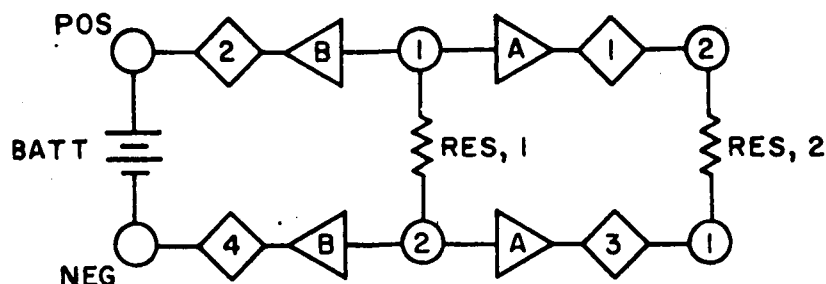
2.2.2.2 Desired Result Statements

The Desired Result Statements communicate the Desired Results to the Performance Simulator. All Desired Results are conveyed to the Performance Simulator at a uniquely defined attachment branch which may or may not be identified.

Desired Result Statements consist of a Parameter Name followed by an Attachment Branch Address which is enclosed in parentheses. The Parameter Name must correspond exactly to the name assigned to this particular parameter in the Element Descriptor Library.

The Desired Result Statements will be located starting in column 5.

Examples Illustrating Desired Result Statements



Suppose the user wishes to declare the following parameters as Desired Results:

Voltage at the negative terminal of the battery.

Current at the attachment branch labeled B of attachment 1 of resistor 1.

In this case the following statements would be appropriate:

Columns	1	2	5
	DESIRED RESULTS.		
	VOLTS(BATT(NEG))		
	AMPS(RES, 1(1, B))		

2.3 NEW ELEMENT TAPE (Abbreviation NW's)

The majority of the work done by the Performance Simulator is in the production of the Generated Program, which is that portion of the Complete Program that will yield the Desired Results in terms of the Input Parameters. The function of the New Element Tape is to organize this Generated Program into a Complete Program in machine translatable language.

The declaration NEW ELEMENT TAPE, must precede the New Element Tape. The declaration NEW ELEMENT TAPE., which is located in columns 2 through 18, prepares the Performance Simulator to receive the New Element Tape. The statements and instructions in the New Element Tape must be written in standard MAD format except for the special indicator symbols. The meaning and use of these special indicator symbols (-), (+), and (⊕) are discussed in Section 2.3.4.

The New Element Tape is divided into the following three sections:

- a. Prologue
- b. Epilogue
- c. Calculations

2.3.1 Prologue

The Prologue section is assigned the task of initiating the Generated Program into an acceptable program in machine translatable language. Included in the Prologue section are such things as specifying the input and output format, reading in and printing out the Input Parameter values, and printing of the Desired Results. The function of the Prologue is to provide the Performance Simulator with the actual computer language statements which perform these functions.

The Prologue section consists of all the statements following the declaration NEW ELEMENT TAPE. up to and including the statement with the third (-) symbol encountered in column one of the New Element Tape.

The Prologue section will precede the Generated Program in the Complete Program which simulates the system.

2.3.2 Epilogue

The Epilogue section is assigned the task of terminating the Generated Program into an acceptable program in machine translatable language. Included in the Epilogue section are such things as testing the computed results to determine if pre-assigned accuracy limits have been obtained, and transferring the program back to the Prologue section or to the Generated Program for further calculations. The function of the Epilogue section is to provide the Performance Simulator with the actual computer language statements which perform these functions.

The Epilogue section consists of all the statements immediately following the statement with the third (-) symbol encountered in column one of the New Element Tape up to and including the statement with the first (+) symbol encountered in column one. The final statement in the Epilogue section must contain a (+) symbol in column one.

The Epilogue section will immediately follow the Generated Program in the Complete Program which simulates the system.

2.3.3 Calculations

The Calculations section gives the user the ability to supplement the capability of the Element Descriptions.

If for any reason a capability is difficult to implement into the Element Descriptor Library, then the Calculations section can be used to supplement the Element Descriptor Library with this capability. For example, suppose the user wishes to calculate the total power dissipated in a system. This would require a summation of the power dissipated in every element in the system. It would be most difficult to implement this capability into the Element Descriptor Library. In this case the Calculations section could be easily used to supplement the Element Descriptor Library with this capability.

The Calculations section must follow the Epilogue section in the New Element Tape. The declaration CALCULATIONS. (Abbreviation CL's) must precede the Calculations section. The declaration CALCULATIONS. , which is located in columns 2 through 14, prepares the Performance Simulator to receive the Calculations section. The Calculations section is divided into two parts. The first part contains all the statements following the declaration CALCULATIONS. up to and including the statement with (-) symbol in column one. The second part contains all the remaining statements in the Calculations section. The last statement in the Calculations section must contain the (+) symbol in column one.

2.3.4 Special Indicators

The statements in the New Element Tape are given in standard MAD format except the symbols (-), (+), and ($\oplus$) which are used as special indicators for the Performance Simulator.

2.3.4.1 The Meaning and Use of the (-) Symbol When Located in Column One of the New Element Tape

There are four (-) symbols located in column one in the New Element Tape. The first (-) symbol encountered in column one of the New Element Tape indicates where the first part of the Calculations section will be located in the Complete Program. The first part of the Calculations section immediately follows the statement which contains the first (-) symbol in column one.

The second (-) symbol encountered in column one indicates where the second part of the Calculations section will be located in the Complete Program. The second part of the Calculations section immediately follows the statement which contains the second (-) symbol in column one.

The third (-) symbol encountered in column one of the New Element Tape indicates where the Generated Program will be located in the Complete Program. The Generated Program immediately follows the statement which contains the third (-) symbol in column one.

The fourth (-) symbol is located in column one of the Calculations section. This symbol identifies the first and second parts of the Calculations section. The first part includes all the statements following the declaration CALCULATIONS, through and including the statement which contains the (-) symbol. The second part includes all the remaining statements in the Calculations section. The final statement in the Calculations section must contain a (+) symbol in column one.

2.3.4.2 The Meaning and Use of the (+) Symbol When Located in Column One Of The New Element Tape

There are two (+) symbols located in column one in the New Element Tape. The first (+) symbol must appear in column one of the last statement in the Epilogue section. The second (+) symbol must appear in column one of the last statement in the Calculations section.

2.3.4.3 Meaning and Use of the (⊕) Symbol When Located in The New Element Tape

When the ⊕ symbol is encountered for the first time in the New Element Tape, a list of MAD names which correspond to the Input parameters is punched on cards and inserted at the place where ⊕ symbol was first encountered.

The second ⊕ symbol encountered causes the same list to be punched out again. This policy was adapted so that the Input Parameters could be printed out immediately after they are read in for verification.

The third and final ⊕ symbol encountered causes a list of the Desired Results to be punched so that the results may be printed through the use of a PRINT FORMAT MAD statement.

Example of New Element Tape

Columns	12	12
		NEW ELEMENT TAPE.
Prologue Section	*	COMPILE MAD, PRINT OBJECT, PUNCH OBJECT DIMENSION VOLTS(50), TVOLTS(50), AMPS(50), TAMP(50), F(13) VECTOR VALUES FORM=\$12C6*\$ BOOLEAN FIRST, REPEAT INTEGER TRYCNT, F, J, START READ FORMAT FORM, F(1), ..., F(12) F(13)=\$*\$ F(0)=\$1H, \$ READ FORMAT F(1), ⊕ - PUNCH FORMAT F, ⊕ FIRST PART OF CALCULATIONS END HERE TRYCNT=1 FIRST=1B REPEAT=0B TRANSFER TO BEGIN BACK CONTINUE WHENEVER TRYCNT, L. 50. AND. REPEAT REPEAT=0B FIRST=0B TRYCNT=TRYCNT+1 TRANSFER TO BEGIN END OF CONDITIONAL - WHENEVER REPEAT, PRINT FORMAT FORM, \$NO CONVERGENCE IN 50 TRYSS RSECOND PART OF CALCULATIONS END HERE PUNCH FORMAT F, ⊕ TRANSFER TO START - BEGIN CONTINUE
		TRANSFER TO BACK
		END OF PROGRAM
CALCULATIONS.		
Calculations Section		THROUGH SETLP, FOR J=0, 1, J. G. 50
	First Part	AMPS(J)=0. TAMP(J)=0. VOLTS(J)=0. TVOLTS(J)=0.
	- SETLP	
	Second Part	THROUGH RESET, FOR J=0, 1, J. G. 50 TAMP(J)=AMPS(J) TVOLTS(J)=VOLTS(J)
	+ RSET	

3 ELEMENT DESCRIPTOR LIBRARY

In order to simulate a physical system, the Performance Simulator must have available information concerning the physical laws or relationships for each element in the system. This information is contained in the Element Descriptor Library. The Element Descriptor Library is composed of Element Descriptions. Each Element Description describes the characteristics of a particular element to the Performance Simulator.

3.1 ELEMENT DESCRIPTION

The information which defines the physical laws or relationships of an element is communicated to the Performance Simulator by an Element Description.

An Element Description consists of the following information which must appear in the given order.

3.1.1 ELEMENT DESCRIPTION. Declaration (Abbreviation EL'N)

The declaration ELEMENT DESCRIPTION. is the first statement in the Element Description. This declaration prepares the Performance Simulator to receive the Element Description.

The declaration ELEMENT DESCRIPTION. will be located in columns 2 through 21.

3.1.2 NAME OF ELEMENT Assertion (Abbreviation NM'T)

The assertion NAME OF ELEMENT = followed by the Element Name is the second statement in the Element Description. Rules governing the choice of Element Names have been given previously.

The assertion NAME OF ELEMENT = should be located starting in column 2.

Examples Illustrating Element Name Assertion

Columns	1	2
		NAME OF ELEMENT = RES
		NAME OF ELEMENT = BATT
		NAME OF ELEMENT = AMPL

3. 1. 3 ATTACHMENT NAMES Assertion (Abbreviation AT's)

The assertion ATTACHMENT NAMES = followed by the Attachment Names, which are separated by commas, is the third statement in the Element Description. Rules governing the choice of the Attachment Names have been given previously.

The assertion ATTACHMENT NAMES = should be located in columns 2 through 72. Continuation to additional cards is allowed. Continuation cards should use columns 21 through 72.

Examples Illustrating Attachment Names Assertions

Columns	1	2	21
		ATTACHMENT NAMES =	GRID, PLATE, CATH
		ATTACHMENT NAMES =	1, 2
		ATTACHMENT NAMES =	INPUT, OUTPUT

3. 1. 4 BROAD SCOPE PARAMETERS Assertion (Abbreviation BR's)

A Broad Scope Parameter is a parameter whose value is the same at all attachment branches of an attachment. Some examples of Broad Scope Parameters are: volts, pressure, and displacement. Once a parameter has been declared Broad Scope in any Element Description, it must be declared Broad Scope in all Element Descriptions where this parameter occurs.

A Narrow Scope Parameter is a parameter whose value is not necessarily the same at all attachment branches of an attachment. Some examples of Narrow Scope Parameters are: current, flow, and force.

The Performance Simulator assumes all parameters to be Narrow Scope unless they are declared Broad Scope.

The assertion BROAD SCOPE PARAMETERS = followed by the Broad Scope Parameters, which are separated by commas, is the fourth statement in the Element Description. Rules governing the choice of the Parameter Names have been given previously.

The assertion BROAD SCOPE PARAMETERS = should be located in columns 2 through 72. Continuation to additional cards is allowed. Continuation cards should use columns 27 through 72.

Examples Illustrating Broad Scope Parameter Assertion

Columns	1	2	27
		BROAD SCOPE PARAMETERS =	VOLTS, OHMS
		BROAD SCOPE PARAMETERS =	PRES

3. 1. 5 Statement Collection

A Statement Collection communicates a specific physical law or relationship of an element to the Performance Simulator. The complete description of these physical laws or relationships for a particular element may contain many Statement Collections. The ordering of these Statement Collections is arbitrary.

The Statement Collection consists of the following information which must appear in the given order.

3. 1. 5. 1 STATEMENT COLLECTION. Declaration (Abbreviation ST'N)

The declaration STATEMENT COLLECTION. is the first statement in the Statement Collection. This declaration prepares the Performance Simulator to receive the Statement Collection.

The declaration STATEMENT COLLECTION. will be located in columns 2 through 22.

3. 1. 5. 2 Capability Statement

The Capability Statement is the second statement in the Statement Collection. The Capability Statement conveys to the Performance Simulator what parameters are necessary in order to calculate other parameters. All Capability Statements will be located starting in column 5.

The Capability Statements add the following special words to the Performance Simulator language:

- a. THEN
- b. ESTIMATE
- c. COMPUTE
- d. WITHOUT

THEN Capability Statement

The word THEN set off by commas separates and identifies the "If Known" Parameters from the Calculable Parameters in the Capability Statement. The "If Known" Parameters are located to the left of the word THEN and

the Calculable Parameters are located to the right. The "If Known" Parameters of the Capability Statement are specified by a series of parameters each followed by an Attachment Name in parentheses. These "If Known" Parameters must be separated by commas. The Calculable Parameters are specified by a series of parameters each followed by an Attachment Name and if necessary an Attachment Branch Identifier in parentheses. These Calculable Parameters must be separated by commas. The Attachment Branch Identifier is used if and only if the Calculable Parameter is Narrow Scope. This Attachment Branch Identifier in the Calculable Parameters implies the attachment branch currently being considered.

Examples Illustrating THEN Capability Statement

Suppose the user wishes to implement into the Element Description for an amplifier the capability of determining the output voltage of an amplifier if the input voltage and amplification factor are known. Also assume VOLTS has been declared Broad Scope. In this case

Columns	5
	VOLTS(INPUT), AMPLF, THEN, VOLTS(OUTPUT)

would be an appropriate Capability Statement, where AMPLF represents amplification factor.

Suppose the user wishes to implement into the Element Description for a resistor, whose Attachment Names are 1 and 2, the capability of determining the current at attachment 1 when the voltages at attachments 1 and 2 and the resistance, which is specified at attachment 1, are known. In this case

Columns	5
	OHMS(1), VOLTS(1), VOLTS(2), THEN, AMPS(1, A)

would be the appropriate Capability Statement. Since AMPS is Narrow Scope, the Attachment Branch Identifier A is used to imply the attachment branch presently being considered.

ESTIMATE Capability Statement (Abbreviation ES'E)

Often physical systems are simulated most conveniently through iterative techniques. That is, the program is so constructed that an initial estimate is improved by repeated calculations. The ESTIMATE capability is useful

only if there exists another independent method, in the Element Descriptor Library, to calculate the desired parameter. The word ESTIMATE followed by a comma and the parameter to be estimated conveys to the Performance Simulator that the method of solution is an iteration technique.

Again an example would serve to illustrate the ESTIMATE capability. Suppose the user wishes to convey to the Performance Simulator the Capability of the iterative approach to the solution for the output voltage of the amplifier. In this case

Columns	5
ESTIMATE, VOLTS(OUTPUT)	

would be an appropriate Capability Statement.

COMPUTE Capability Statement (Abbreviation CM'E)

Often it is useful to have a Capability Statement that requires no "If Known" Parameters to yield the Computational Parameters. The word COMPUTE followed by a comma which is followed by the Calculable Parameters gives this capability to the user. For example, suppose the user wishes to convey to the Performance Simulator the capability for determining the weight of a resistor. Suppose the weight is associated with the attachment labeled 1. In this case

Columns	5
COMPUTE, WEIGHT(1)	

would be the appropriate Capability Statement. Following this Capability Statement the user must either supply a method to calculate the weight or transfer to an appropriate subroutine where this information is given.

WITHOUT Capability Statement (Abbreviation WT'T)

In order to allow a generic element name to be applied to a fairly wide class of elements -- some of which may not have all of the attachments possessed by others -- the WITHOUT statement is used. In this way, only those statement collections which fit a given element will be used in the simulation.

For example, suppose that the user wants to use the name HMOTOR for hydraulic motors that have an attachment SHAFT. Some HMOTOR elements may also have a power take-off attachment (PTO). If it were not possible to indicate those descriptions which are valid on HMOTOR elements that do not have PTO attachments, two separate elements would be required. To indicate this situation one writes in this case

Columns	5
	PRESS (INLET), WITHOUT (PTO), THEN, TORQUE (SHAFT)

3.1.5.3 Calculational Statements

Immediately following each Capability Statement the user must supply the program statements which will yield the Calculable Parameters given the "If Known" Parameters listed in the Capability Statement. These statements must be written in Michigan Algorithmic Decoder language (MAD language), except as modified by the $\oplus$ and $\ominus$ symbols which convey special meaning to the Performance Simulator.

Meaning and Use of the Symbol $\oplus$

The $\oplus$ symbol is used in general to delimit the parameter, Attachment Name, and/or Attachment Branch Identifier for which the Performance Simulator must generate a unique system variable name of six or less alphanumeric characters.

The $\oplus$ symbol implies different procedures depending upon the preceding Capability Statement. There are three special cases.

Case 1.

If a particular Attachment Name is located only on the left side of the word THEN in the Capability Statement, the statement $\oplus$ Para. (A.N., I) $\oplus$ in the calculational statements implies a summation of the parameter values over all branches of the attachment. The Attachment Branch Identifier (I) following the Attachment Name has no significance in itself. Other letters may be used if desired.

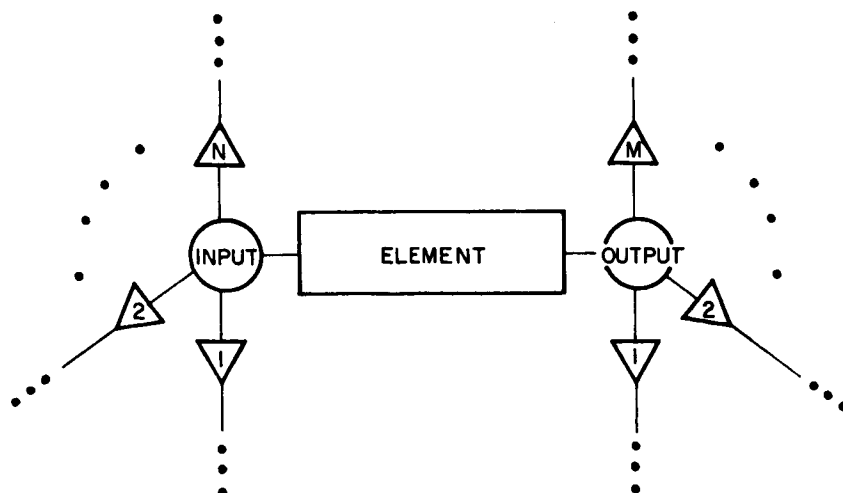
Case 2.

The statement $\oplus$ Para. (A.N.) $\oplus$ in the calculational statements implies the substitution of the parameter value in the attachment branch presently being considered.

Case 3.

If a particular Attachment Name is located on both sides of the word THEN in the Capability Statement, the statement $\oplus$ Para. (A. N., I) $\oplus$ implies a summation of the parameter values over all branches of the attachment except the attachment branch presently being considered.

General Example



Suppose the user wishes to indicate the capability and calculational procedure to compute the AMPS in the attachment branch presently being considered at the INPUT attachment if the AMPS of all remaining attachment branches at both the INPUT and OUTPUT attachments are known.

In this case the following Statement Collection would be appropriate.

Columns	2	5	12
	STATEMENT		COLLECTION.
Capability Statement	AMPS(INPUT), AMPS(OUTPUT), THEN, AMPS(INPUT, A)		
Calculational Statements			T = 0.
			T = T + $\oplus$ AMPS(OUTPUT, A) $\oplus$
			S = 0.
			S = S + $\oplus$ AMPS(INPUT, I) $\oplus$
			$\oplus$ AMPS(INPUT) $\oplus$ = T - S

Since the attachment OUTPUT is located only on the left side of the word THEN in the Capability Statement, the statement $\oplus$ AMPS(OUTPUT, A) $\oplus$ implies a summation of AMPS in each attachment branch of the OUTPUT attachment. Therefore the following statements:

$$T = 0.$$

$$T = T + \oplus \text{AMPS(OUTPUT, A)} \oplus$$

sum the AMPS for all of the OUTPUT attachment branches. The letter A carries no meaning in itself. Any six or less alphanumeric character word could have been used.

Since the attachment INPUT is located on both sides of the word THEN in the Capability Statement, $\oplus$ AMPS(INPUT, I) $\oplus$ implies a summation of AMPS in each attachment branch of the INPUT attachment except the attachment branch presently being considered. Therefore the following statements:

$$S = 0.$$

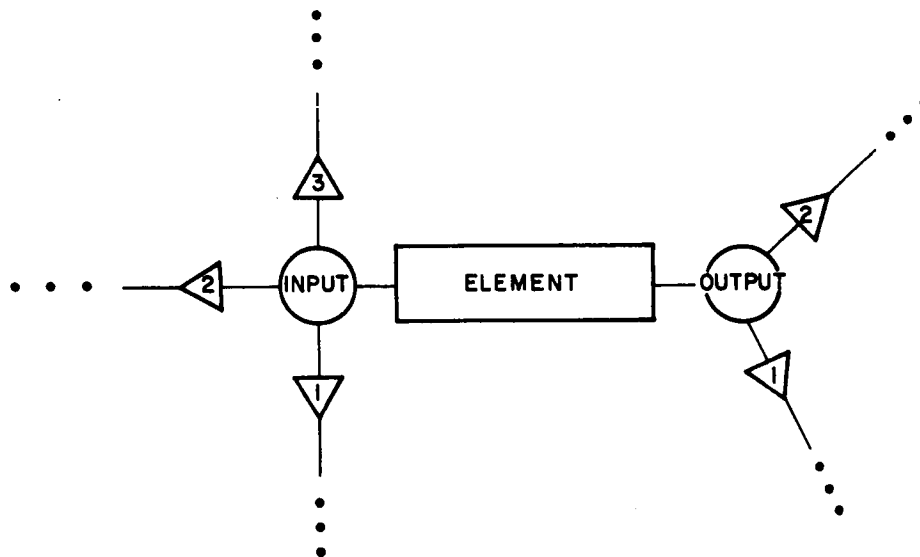
$$S = S + \oplus \text{AMPS(INPUT, I)} \oplus$$

sum the AMPS at all the INPUT attachments except for the AMPS in the attachment branch presently being considered. Again the letter I carries no meaning in itself.

The statement $\oplus \text{AMPS}(\text{INPUT}) \oplus$ implies the value of AMPS in the attachment branch presently being considered.

The final statement in the Statement Collection indicates that the value of AMPS in the attachment branch presently being considered at the INPUT attachment equals the difference between the summation of the AMPS over all attachment branches at the OUTPUT attachment and the summation of the AMPS over all attachment branches at the INPUT attachment except the attachment branch presently being considered.

Particular Example



Consider how the following Statement Collection, which was used in the general example above, would be used to compute AMPS at Attachment Branch Identifier 2 of the INPUT attachment.

Columns	2	5	12
	STATEMENT		COLLECTION.
Capability } Statement }	-----		AMPS(INPUT), AMPS(OUTPUT), THEN, AMPS(INPUT, A)
	-----		T = 0.
	-----		T = T + ⊕ AMPS(OUTPUT, A) ⊕
Calculational } Statements }	-----		S = 0.
	-----		S = S + ⊕ AMPS(INPUT, I) ⊕
	-----		⊕ AMPS(INPUT) ⊕ = T - S

The form of the calculational statements that would be generated by the Performance Simulator for the particular problem under consideration is as follows:

$$T = 0.$$

$$T = T + \text{AMPS(OUTPUT, 1)}$$

$$T = T + \text{AMPS(OUTPUT, 2)}$$

$$S = 0.$$

$$S = S + \text{AMPS(INPUT, 1)}$$

$$S = S + \text{AMPS(INPUT, 3)}$$

$$\text{AMPS(INPUT, 2)} = T - S$$

The final statement indicates that

$$\text{AMPS(INPUT, 2)} = \text{AMPS(OUTPUT, 1)} + \text{AMPS(OUTPUT, 2)} - (\text{AMPS(INPUT, 1)} - \text{AMPS(INPUT, 3)})$$

Meaning and Use Of The ⊕ Symbol

The ⊕ symbol is used to delimit Function Substitutions and Floating Statement Labels.

The ⊖ symbol is used to delimit the statements which are to be modified by Function Substitution Statements. The statement which is enclosed by ⊖ symbols will be modified when the Generated Program is formulated. The reader is referred to Section 4.1 for complete discussion of the ⊖ symbol in conjunction with Function Substitution Statements.

The Floating Statement Label allows the use of Statement Labels in Statement Collections. Since an element may appear any number of times in a system, the same Statement Collections might therefore occur any number of times in a program, which simulates the system. To avoid ambiguity, the user must not use fixed Statement Labels within a Statement Collection.

Floating Statement Labels allow the Performance Simulator to generate unique Statement Labels and thus eliminate any ambiguity in labels.

The user wishing to use a Floating Statement Label writes:

⊖ ** X X X X X X ⊖

where the X's represents any six or less alphanumeric character Statement Label which the user may care to use in his Statement Collection. The floating statement label is initiated by the ⊖ symbol followed by two asterisks (*) and is closed by the ⊖. The Performance Simulator will generate a unique Statement Label and replace the entire Floating Statement Label by this unique label. Should the Statement Collection appear again in the program, another unique Statement Label would be generated and substituted for the Floating Statement Label.

3.1.5.4 End of Statement Collection Declarations

Each Statement Collection is terminated by another STATEMENT COLLECTION declaration or by the DESCRIPTION FINISHED declaration. These declarations convey the end of the Statement Collection to the Performance Simulator.

3.1.6 DESCRIPTION FINISHED Declaration (Abbreviation DS'D)

The final statement in an Element Description must be the declaration DESCRIPTION FINISHED. The declaration DESCRIPTION FINISHED, which is located in columns 2 through 22, conveys the end of the Element Description to the Performance Simulator.

4 UTILITY DECLARATIONS

Utility declarations are additional features which are not necessarily essential to the Performance Simulator. These features allow the Performance Simulator to produce better programs in some cases and to produce programs more easily in others.

4.1 FUNCTION SUBSTITUTIONS (Abbreviation FN's)

Function Substitutions allows modification of any six or less alphanumeric word which is enclosed by the $\ominus$ symbols at the time the Generated Program is formulated. This modification is implemented by Function Substitution Statements which must be preceded by the declaration FUNCTION SUBSTITUTIONS.

4.1.1 FUNCTION SUBSTITUTIONS. Declaration

The declaration FUNCTION SUBSTITUTIONS. must precede the Function Substitution Statements. The declaration FUNCTION SUBSTITUTIONS., which is located in columns 2 through 23, prepares the Performance Simulator to receive the Function Substitution Statements.

4.1.2 Function Substitution Statements

The Function Substitution Statements convey the modifications that are to be made and the scope of these modifications to the Performance Simulator. Any six or less alpha-numeric character word may be substituted for any other word which is enclosed in $\ominus$ symbols.

The form of the Function Substitution Statements is defined as follows:

Let OLDWRD represent any general six or less alphanumeric word which is enclosed by the $\ominus$ symbols.

Let NUWORD represent any general six or less alpha-numeric word which is to be substituted where the $\ominus$ symbols occur.

The allowable Function Substitution Statements are:

NUWORD, OLDWRD(E. N. , E. I)

NUWORD, OLDWRD(E. N.)

NUWORD, OLDWRD

The effect of these statements is to replace OLDWRD, when set off by the $\ominus$ symbols, by NUWORD.

The scope of the substitution is controlled as follows:

- a. The Function Substitution Statement NUWORD, OLDWRD(E. N. , E. I.) implies that a substitution will take place only for the element with the particular Element Name and Identifier.
- b. The Function Substitution Statement NUWORD, OLDWRD(E. N.) implies that a substitution will take place for all the elements with the given Element Name.
- c. The Function Substitution Statement NUWORD, OLDWRD implies a substitution will occur for any occurrence of OLDWRD which is set off by $\ominus$ symbols.

If the Function Substitution Statement is not given, the original OLDWRD enclosed in $\ominus$ symbols will be used.

The best way to illustrate the use of $\ominus$ symbol in conjunction with Function Substitution is by way of an example. Suppose the user wishes to represent the resistance of a resistor as a function of temperature. Also, assume each resistor has a different functional relationship with respect to temperature and, for example purposes assume there are three resistors in the system.

Consider a particular Statement Collection in the Element Description of a resistor, and see how this capability could be introduced.

Columns	2	5	12
	STATEMENT		COLLECTION.
	OHMS(1),		VOLTS(1), AMPS(2), THEN, VOLTS(2)
			T = 0.
			T = T + $\oplus$ AMPS(2, A) $\oplus$
			$\oplus$ OHMS(1) $\oplus$ = $\ominus$ OHMS(1) $\ominus$. (TEMP)
			$\oplus$ VOLTS(2) $\oplus$ = $\oplus$ VOLTS(1) $\oplus$ - (T + $\oplus$ AMPS(2) $\oplus$) * $\oplus$ OHMS(1) $\oplus$

The statement $\oplus$ OHMS(1) $\oplus$ = $\ominus$ OHMS(1) $\ominus$. (TEMP) instructs the Performance Simulator to go to the Function Substitution Statements and find the name of the subroutine that will be used to calculate the value of the resistance which is a function of temperature.

Consider the Functional Substitution Statements needed to complete this capability.

Columns	1	2	5
			FUNCTIONS SUBSTITUTION.
			SUBRT1, OHMS(RES, 1)
			SUBRT2, OHMS(RES, 2)
			SUBRT3, OHMS(RES, 3)

SUBRT1, SUBRT2, and SUBRT3 are the names of the subroutines which must be given to express resistance as a function of temperature for each particular resistor in the system.

4.2 EQUIVALENCE (Abbreviation EQ'E)

Often it is desirable to know the exact name that will be assigned by the Performance Simulator to a parameter at a specific point in a system. For example, suppose the user wishes to instruct the Performance Simulator to print out the Desired Results for verification. In this case all the Desired Results would be made equivalent to known names. The user would then instruct the Performance Simulator to print out the known names. This Equivalence capability is implemented by Equivalence Statements which are preceded by the declaration EQUIVALENCE. .

4.2.1 EQUIVALENCE. Declaration

The declaration EQUIVALENCE. must precede the Equivalence Statements. The declaration EQUIVALENCE., which is located in columns 2 through 13, prepares the Performance Simulator to receive the Equivalence Statements.

4.2.2 Equivalence Statements

Equivalence Statements conveys the specific equivalence to the Performance Simulator.

A general Equivalence Statement has the following form:

Equivalence Name = Parameter(Attachment Branch Address)

The Equivalence Name must be six or less alpha-numeric characters. No restrictions are placed on the choice or ordering of these characters.

A specific example might be:

Columns	1	2	5
		EQUIVALENCE.	
		PWR = POWER (RES, 1 (1, A))	

The above statement equates PWR to the name given by the Performance Simulator to the parameter POWER at the particular attachment branch in the system given by the Attachment Branch Address RES, 1 (1, A).

4.3 DIAGNOSTIC AIDS

To assist the user in obtaining successful simulator runs, several diagnostic aids are available.

4.3.1 CHECKOUT. (Abbreviation CH'T) and CHECKRUN. (Abbreviation CH'N). Declarations

The declarations CHECKOUT and CHECKRUN are equivalent in that each activates the same set of diagnostic printing during a performance simulator run. The two forms are available in order to allow the user to employ whichever form occurs more naturally to him.

5 PERFORMANCE SIMULATOR LOGIC

The logic used by the Performance Simulator in the generation of a program, which will simulate the system, is the reverse of the usual logic used by humans. The usual human approach is a search that proceeds from the given information to the requested results. This approach could be used by the Performance Simulator but because of storage limitations is impractical. The storage difficulty arises because the Performance Simulator has no way of knowing when a method is necessary to the program which will yield the requested results. The Performance Simulator would be forced to enumerate every method available from the given information to the requested results. If the problem is well posed, this logic will eventually yield a program that will simulate the system. However, this logic constitutes an exhaustive search with only a small fraction of the enumerated methods used in the program, which will yield the requested results in terms of the given information.

In order to eliminate this exhaustive search, the logic used by the Performance Simulator proceeds from the requested results to the given information. In this way every step generated is necessarily essential to the program except when the Performance Simulator gets to a point where the program cannot be extended to yield the requested results. When this occurs, the Performance Simulator will regress to a point where a choice was made in the method used and will make a different choice and again try to produce a program that will yield the requested results. This logic minimizes the number of unnecessary steps.

The program generated by this logic is backward in that the first Statement Collection specified is the last one needed and so on throughout all the Statement Collections.

5.1 DISCUSSION OF PERFORMANCE SIMULATOR LOGIC

The following is a step by step detailed discussion of the Performance Simulator Logic.

- Step 1. The Performance Simulator will first search each attachment branch in the system for parameters which have been requested by the Desired Result Statements. (If no requested Desired Results can be found, the problem is trivial.)
- Step 2. Whenever an attachment branch is found where there are requested parameters, steps must be taken to satisfy these requests.
 - 2A. The requested parameters may all be Input Parameters. (If this is true, the program is complete.)
- Step 3. Whenever a requested parameter is found which is not an Input Parameter, steps must be taken to satisfy these requests.
 - 3A. The Performance Simulator will collect all the Statement Collections that are pertinent to obtain the requested parameter in the Element Descriptor Library for the two elements that occur at the attachment branch where the results are requested.
 - 3B. The Performance Simulator will check each pertinent Statement Collection to determine its effectiveness. The effectiveness factor is based upon the number of requested parameters which are not Input Parameters.
- Step 4. A requested parameter will be eliminated if there exist a statement Collection, which will yield the requested parameter, in the Element Descriptor Library.
 - 4A. If one or more Statement Collections are available, the Performance Simulator will choose one based upon its effectiveness and will indicate the number of choices available. The chosen Statement Collection will generally make a request for new requested parameters. The Performance Simulator logic will then transfer back to Step 2 and continue.
 - 4B. If the Element Descriptor Library does not contain a Statement Collection which will yield the requested parameter, the program is "non-extendable". When this occurs, the Performance Simulator will check to determine if any previous choices were made.
 - 4C. There may have been no previous choices. (In this case the problem is not well posed.)

- 4D. There may have been one or more previous choices. In this case the Performance Simulator will regress to the first previous step where a choice was made, and will choose a different Statement Collection. This Statement Collection will generally make a request for new requested parameters. The Performance Simulator Logic will then transfer back to Step 2 and continue.
- Step 5. The above search is repeated until Step 2A is satisfied, or until the problem is shown to be not well posed, or until the requested parameters have been requested twice. If the requested results have been requested twice and if the program generated is not trivial, the Performance Simulator will choose the Statement Collections with the estimate capability to produce an iterative program which will simulate the system.

The following Performance Simulator Logic Diagram (Figure A-1) illustrates the logic which is used by the Performance Simulator in formulating the Generated Program.

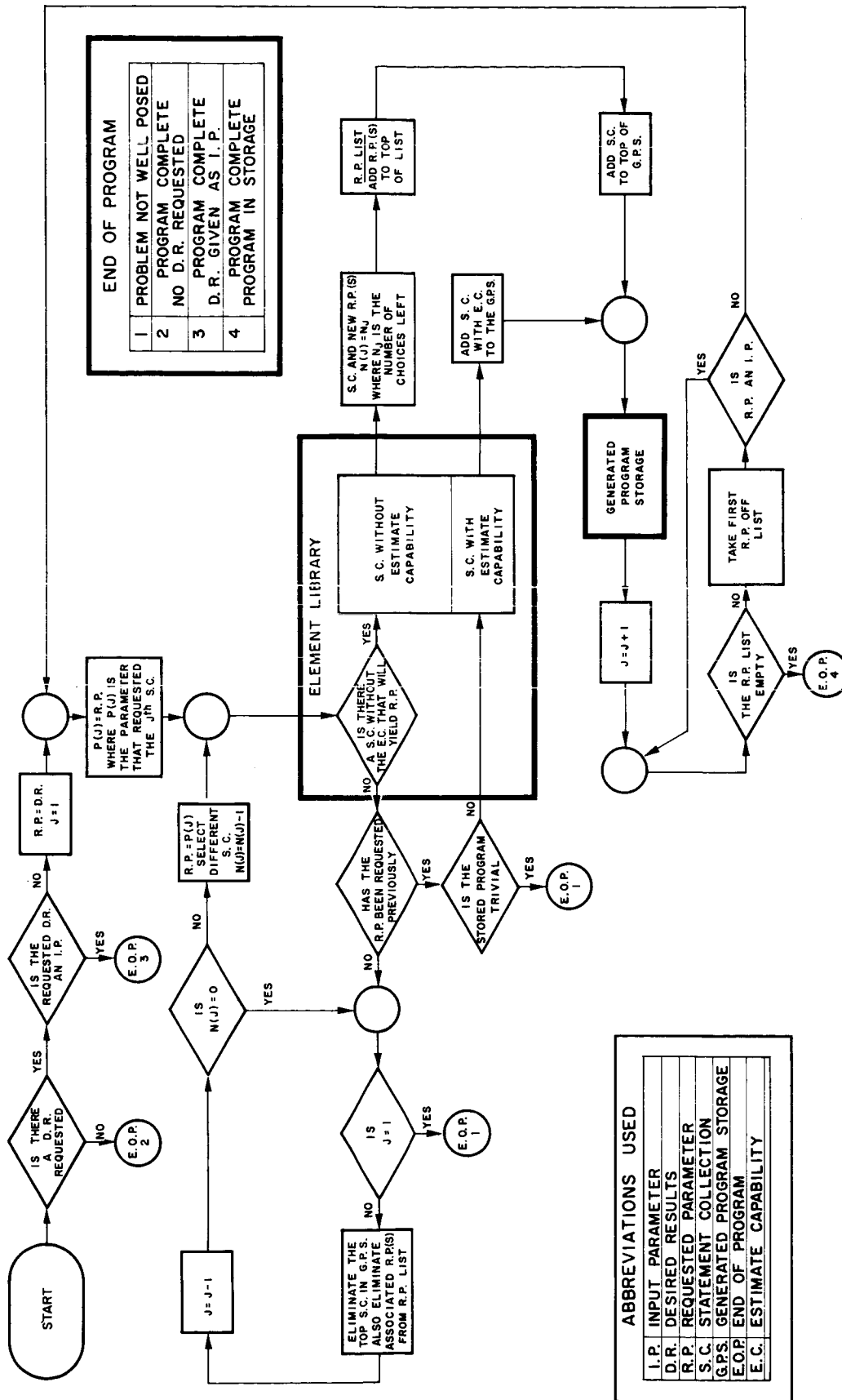


Figure A-1. Performance Simulator Logic Diagram

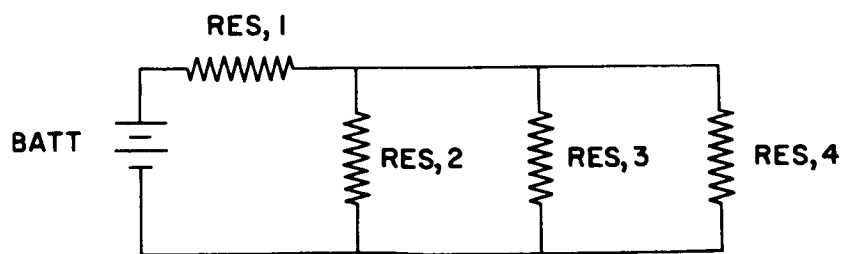
6 PERFORMANCE SIMULATOR SYSTEM DIAGRAMMING

In order to diagram a system in a manner which is easily associated with the logic of the simulator, the following conventions will be used:

- a. An Attachment is any unique, specified point of an element where Input Parameters, Desired Results, and/or Connections are specified. When forming the system diagram, attachments are denoted by the symbol $\bigcirc$.
- b. When an attachment has more than one attachment branch, Attachment Branch Identifiers must be used. When forming the system diagram, Attachment Branch Identifiers are enclosed by the symbol $\triangleright$ where the apex is oriented away from the identified attachment branch.
- c. The symbol $\diamond$ is used to show the connection of one attachment to another. The symbol $\diamond$ is associated with storage location in the computer. This storage location contains all the parameters of the two attachments which are connected by the $\diamond$ symbol. To avoid ambiguity, no two attachments, which specify the same parameters, can be connected.

Examples Illustrating the Above Symbols

Suppose one wants to form the System Diagram for the following network using the above symbols.



In this case the following System Diagrams would be appropriate.

DIAGRAM 1

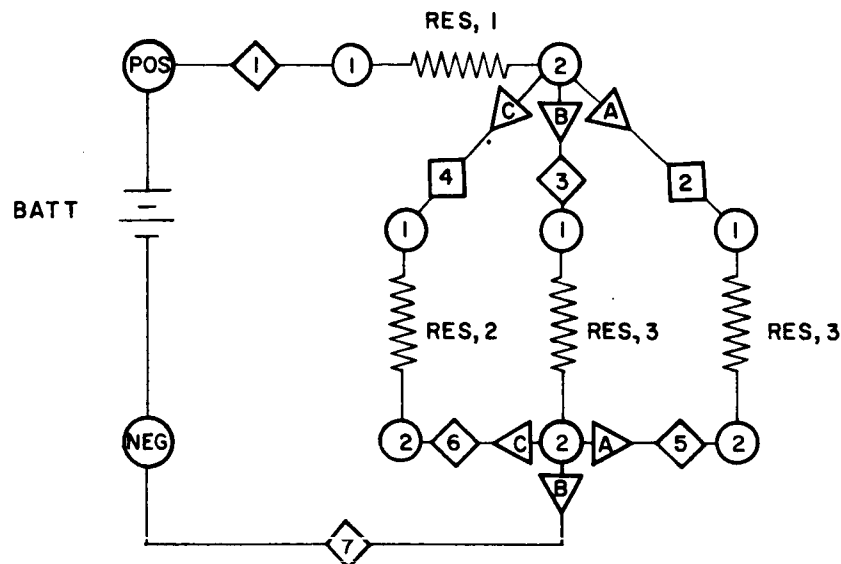
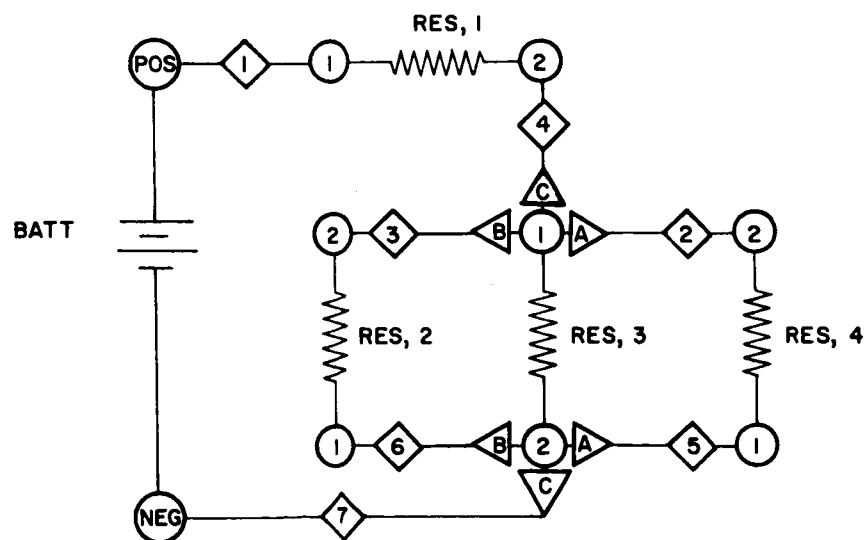


DIAGRAM 2



NOTE: There usually exists more than one satisfactory system diagram for any given system.

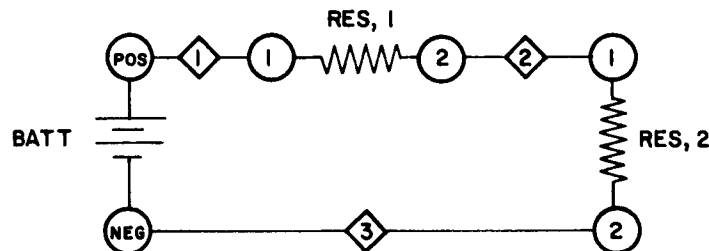
7 COMPLETE EXAMPLE

To illustrate the technique used by the Performance Simulator a simple example is completely solved using the Performance Simulator logic. The example is intended to illustrate the following:

- a. What information is required by the Performance Simulator.
- b. How this information is presented to the Performance Simulator.
- c. The logic used by the Performance Simulator in generating the Complete Program that can be used to simulate the system.

The example, which in no way measures the capability offered by the Performance Simulator, was chosen to be simple so that the rules and logic associated with the Performance Simulator are not hidden by complexity. The reader is assumed to be a person who is generally familiar with the Performance Simulator rules and logic which are given previously.

7.1 PROBLEM CONSIDERED



Suppose the user wishes to use the Performance Simulator to produce a computer program for determining the current at attachment 1 of resistor 2 in the above circuit when given the following Input Parameters:

VOLTS(BATT(NEG))	The specified reference voltage at the negative terminal of the battery
OCV(BATT(NEG))	The open circuit voltage of the battery
OHMS(BATT(NEG))	The internal resistance of the battery
OHMS(RES, 1(1))	The resistance of resistor 1
OHMS(RES, 2(1))	The resistance of resistor 2

7.2 SOURCE PROGRAM

The user's first task is to supply a Source Program to the Performance Simulator. The Source Program consists of the following information:

- a. System Description (Connection Statements)
- b. Input Parameters
- c. Desired Results
- d. New Element Tape

The rules for presenting the above information have been given previously.

```
● CONNECTIONS.
  BATT(POS),TO,RES,1(1)
  RES,1(2),TO,RES,2(1)
  RES,2(2),TO,BATT(NEG)

● INPUT PARAMETERS.
  VOLTS(BATT(NEG))
  OCV(BATT(NEG))
  OHMS(BATT(NEG))
  OHMS(RES,1(1))
  OHMS(RES,2(1))

● DESIRED RESULTS.
  AMPS(RES,2(1))

● NEW ELEMENT TAPE.
*  COMPILER MAD,PRINT OBJECT,PUNCH OBJECT
  DIMENSION VOLTS(50),TVOLTS(50),AMPS(50),TAMPS(50),F(13)
  VECTOR VALUES FORM=$12C6*$
  BOOLEAN FIRST,REPEAT
  INTEGER TRYCNT,F,J
  START  READ FORMAT FORM,F(1),....,F(12)
        F(13)=$*$
        F(0)=$1H,$
        READ FORMAT F(1),0
        PUNCH FORMAT F,0
  -      RFIRST PART OF CALCULATIONS END HERE
        TRYCNT=1
        FIRST=1B
        REPEAT=0B
        TRANSFER TO BEGIN
  BACK   CONTINUE
        WHENEVER TRYCNT.L.50,AND,REPEAT
        REPEAT=0B
        FIRST=0B
        TRYCNT=TRYCNT+1
        TRANSFER TO BEGIN
        END OF CONDITIONAL
  -      WHENEVER REPEAT,PRINT FORMAT FORM,$NO CONVERGENCE IN 50 TRYSS
        RSECOND PART OF CALCULATIONS END HERE
        PUNCH FORMAT F,0
        TRANSFER TO START
  -BEGIN CONTINUE
        TRANSFER TO BACK
  +      END OF PROGRAM
```

```

● CALCULATIONS.
    THROUGH SETLP, FOR J=0.1,J.G.50
    AMPS(J)=0.
    TAMPS(J)=0.
    VOLTS(J)=0.
● -SETLP
    TVOLTS(J)=0.
    THROUGH RESET, FOR J=0.1,J.G.50
    TAMPS(J)=AMPS(J)
● +RESET
    TVOLTS(J)=VOLTS(J)
●

```

7.3 ELEMENT DESCRIPTOR LIBRARY

The Element Descriptor Library communicates the information concerning the physical laws or relationships for each element in the system to the Performance Simulator. The rules for presenting the above information have been given previously.

ELEMENT DESCRIPTION.	0001D
NAME OF ELEMENT = RES	0001N
ATTACHMENT NAMES = 1,2	0001A
BROAD SCOPE PARAMETERS = VOLTS,OHMS	0001B
STATEMENT COLLECTION.	0001C001
AMPS(2),AMPS(1),THEN,AMPS(1,A)	0001C001
T=0.	0001C001
T=T+@AMPS(1,A)@	0001C001
S=0.	0001C001
S=S+@AMPS(2,A)@	0001C001
@AMPS(1)@=S-T	0001C001
STATEMENT COLLECTION.	0001C002
AMPS(1),AMPS(2),THEN,AMPS(2,A)	0001C002
T=0.	0001C002
T=T+@AMPS(2,A)@	0001C002
S=0.	0001C002
S=S+@AMPS(1,A)@	0001C002
@AMPS(2)@=S-T	0001C002
STATEMENT COLLECTION.	0001C003
OHMS(1),AMPS(1),VOLTS(1),VOLTS(2),THEN,AMPS(1,A)	0001C003
T=0.	0001C003
T=T+@AMPS(1,A)@	0001C003
@AMPS(1)@=(@VOLTS(1)@-@VOLTS(2)@)/@OHMS(1)@-T	0001C003
STATEMENT COLLECTION.	0001C004
OHMS(1),AMPS(2),VOLTS(2),VOLTS(1),THEN,AMPS(2,A)	0001C004
T=0.	0001C004
T=T+@AMPS(2,A)@	0001C004
@AMPS(2)@=(@VOLTS(1)@-@VOLTS(2)@)/@OHMS(1)@-T	0001C004
STATEMENT COLLECTION.	0001C005
OHMS(1),AMPS(1),VOLTS(1),THEN,VOLTS(2)	0001C005
T=0.	0001C005
T=T+@AMPS(1,A)@	0001C005
@VOLTS(2)@=@VOLTS(1)@-T*@OHMS(1)@	0001C005
STATEMENT COLLECTION.	0001C006
OHMS(1),AMPS(2),VOLTS(2),THEN,VOLTS(1)	0001C006
T=0.	0001C006
T=T+@AMPS(2,A)@	0001C006
@VOLTS(1)@=@VOLTS(2)@-T*@OHMS(1)@	0001C006
STATEMENT COLLECTION.	0001C007
OHMS(1),VOLTS(1),AMPS(2),THEN,VOLTS(2)	0001C007
T=0.	0001C007
T=T+@AMPS(2,A)@	0001C007
@VOLTS(2)@=@VOLTS(1)@-(T+@AMPS(2)@)*@OHMS(1)@	0001C007
STATEMENT COLLECTION.	0001C008
OHMS(1),VOLTS(2),AMPS(1),THEN,VOLTS(1)	0001C008
T=0.	0001C008
T=T+@AMPS(1,A)@	0001C008
@VOLTS(1)@=@VOLTS(2)@-(T+@AMPS(1)@)*@OHMS(1)@	0001C008

STATEMENT COLLECTION.	0001C009
ESTIMATE AMPS(1)	0001C009
WHENEVER FIRST	0001C009
REPEAT=1B	0001C009
AMPS(1)=TAMPS(1)	0001C009
OTHERWISE	0001C009
AMPS(1)=ITTR.(AMPS(1),GAMPS(1),REPEAT)	0001C009
END OF CONDITIONAL	0001C009
GAMPS(1)=AMPS(1)	0001C009
STATEMENT COLLECTION.	0001C010
ESTIMATE AMPS(2)	0001C010
WHENEVER FIRST	0001C010
REPEAT=1B	0001C010
AMPS(2)=TAMPS(2)	0001C010
OTHERWISE	0001C010
AMPS(2)=ITTR.(AMPS(2),GAMPS(2),REPEAT)	0001C010
END OF CONDITIONAL	0001C010
GAMPS(2)=AMPS(2)	0001C010
STATEMENT COLLECTION.	0001C011
ESTIMATE VOLTS(2)	0001C011
WHENEVER FIRST	0001C011
REPEAT=1B	0001C011
VOLTS(2)=TVOLTS(2)	0001C011
OTHERWISE	0001C011
VOLTS(2)=ITTR.(VOLTS(2),GVOLTS(2),REPEAT)	0001C011
END OF CONDITIONAL	0001C011
GVOLTS(2)=VOLTS(2)	0001C011
STATEMENT COLLECTION.	0001C012
ESTIMATE VOLTS(1)	0001C012
WHENEVER FIRST	0001C012
REPEAT=1B	0001C012
VOLTS(1)=TVOLTS(1)	0001C012
OTHERWISE	0001C012
VOLTS(1)=ITTR.(VOLTS(1),GVOLTS(1),REPEAT)	0001C012
END OF CONDITIONAL	0001C012
GVOLTS(1)=VOLTS(1)	0001C012
DESCRIPTION FINISHED.	0001F
ELEMENT DESCRIPTION.	0002D
NAME OF ELEMENT = BATT	0002N
ATTACHMENT NAMES = NEG,POS	0002A
BROAD SCOPE PARAMETERS = VOLTS,OHMS	0002B
STATEMENT COLLECTION.	0002C001
AMPS(NEG),THEN,AMPS(POS)	0002C001
T=0.	0002C001
T=T+AMPS(NEG,A)	0002C001
AMPS(POS)=T	0002C001
STATEMENT COLLECTION.	0002C002
AMPS(POS),THEN,AMPS(NEG)	0002C002
T=0.	0002C002
T=T+AMPS(POS,A)	0002C002
AMPS(NEG)=T	0002C002
STATEMENT COLLECTION.	0002C003
AMPS(NEG),VOLTS(NEG),OCV(NEG),OHMS(NEG),THEN,VOLTS(POS)	0002C003
VOLTS(POS)=VOLTS(NEG)+OCV(NEG)-AMPS(NEG)*OHMS(NEG)	0002C003
STATEMENT COLLECTION.	0002C004
AMPS(NEG),VOLTS(POS),OCV(NEG),OHMS(NEG),THEN,VOLTS(NEG)	0002C004
VOLTS(NEG)=VOLTS(POS)-OCV(NEG)+AMPS(NEG)*OHMS(NEG)	0002C004
STATEMENT COLLECTION.	0002C005
AMPS(NEG),VOLTS(NEG),VOLTS(POS),THEN,POWER(NEG)	0002C005
POWER(NEG)=(VOLTS(NEG)-VOLTS(POS))*AMPS(NEG)	0002C005
DESCRIPTION FINISHED.	0002F

7.4 GENERATED PROGRAM

The information given in the Source Program and Element Descriptor Library is sufficient for the Performance Simulator to formulate the Generated Program. By using the logic, which was discussed previously, the Performance Simulator will produce the following Generated Program:

This Generated Program is backward, in that, the first Statement Collection specified is the last one needed and so on throughout all Statement Collections. The Performance Simulator reverses this Generated Program when the Complete Program is formed.

The $\diamond$ symbol is used to represent a connection in a system. The number which is located inside the $\diamond$ symbol is associated with a particular connection in the system.

The numbers which are located at the right hand side of each statement in the Generated Program indicate the particular Statement Collection which was chosen by the Performance Simulator.

Generated Program

•		
•	T=0.	0001C003
•	AMPS $\diamond 2$ = (VOLTS $\diamond 2$ - VOLTS $\diamond 3$) / OHMS $\diamond 2$ - T	0001C003
•	T=0.	0001C007
•	VOLTS $\diamond 2$ = VOLTS $\diamond 1$ - (T + AMPS $\diamond 2$) * OHMS $\diamond 1$	0001C007
•	VOLTS $\diamond 1$ = VOLTS $\diamond 3$ + OCV $\diamond 3$ - AMPS $\diamond 3$ * OHMS $\diamond 3$	0002C003
•	T=0.	0002C002
•	T = T + AMPS $\diamond 1$	0002C002
•	AMPS $\diamond 3$ = T	0002C002
•	T=0.	0001C001
•	S=0.	0001C001
•	S = S + AMPS $\diamond 2$	0001C001
•	AMPS $\diamond 1$ = S - T	0001C001
•	WHENEVER FIRST	0001C010
•	REPEAT=18	0001C010
•	AMPS $\diamond 2$ = T AMPS $\diamond 2$	0001C010
•	OTHERWISE	0001C010
•	AMPS $\diamond 2$ = ITTR * (AMPS $\diamond 2$ + GAMPS $\diamond 2$ + REPEAT)	0001C010
•	END OF CONDITIONAL	0001C010
•	GAMPS $\diamond 2$ = AMPS $\diamond 2$	0001C010
•		
•		

7.5 COMPLETE PROGRAM

The Complete Program is a computer program in machine translatable language that can be used to give numerical values of the Desired Results when the values of the Input Parameters are given. The Complete Program consists of the Generated Program and the New Element Tape. The Performance Simulator will reverse the Generated Program and insert it between the Prologue and Epilogue sections of the New Element Tape.

Complete Program

```

●      COMPILER MAD,PRINT OBJECT,PUNCH OBJECT
●      DIMENSION VOLTS(50),TVOLTS(50),AMPS(50),TAMPS(50),F(13)
●      VECTOR VALUES FORM=$12C6*$
●      BOOLEAN FIRST,REPEAT
●      INTEGER TRYCNT,F,J
START  READ FORMAT FORM,F(1),...,F(12)
●      F(13)=$*$
●      F(0)=$1H,$
●      READ FORMAT F(1),0
●      PUNCH FORMAT F,0
●      THROUGH SETLP,FOR J=0,1,J,G,50
●      AMPS(J)=0.
●      TAMPS(J)=0.
●      VOLTS(J)=0.
SETLP  TVOLTS(J)=0.
●      RFIRST PART OF CALCULATIONS END HERE
●      TRYCNT=1
●      FIRST=1B
●      REPEAT=0B
●      TRANSFER TO BEGIN
BACK   CONTINUE
●      WHENEVER TRYCNT,L,50,AND,REPEAT
●      REPEAT=0B
●      FIRST=0B
●      TRYCNT=TRYCNT+1
●      TRANSFER TO BEGIN
●      END OF CONDITIONAL
●      WHENEVER REPEAT,PRINT FORMAT FORM,$NO CONVERGENCE IN 50 TRY$
●      THROUGH RESET,FOR J=0,1,J,G,50
●      TAMPS(J)=AMPS(J)
RSET   TVOLTS(J)=VOLTS(J)
●      RSECOND PART OF CALCULATIONS END HERE
●      PUNCH FORMAT F,0
●      TRANSFER TO START
BEGIN  CONTINUE
●      WHENEVER FIRST
●      REPEAT=1B
●      AMPS(2)=TAMPS(2)
●      OTHERWISE
●      AMPS(2)=ITTR.(AMPS(2),GAMPS(2),REPEAT)
●      END OF CONDITIONAL
●      GAMPS(2)=AMPS(2)

```

```

T=0.
S=0.
S=S+AMPS ②
AMPS ① =S-T
T=0.
T=T+AMPS ①
AMPS ③ =AMPS ①
VOLTS ① =VOLTS ③ +OCV ③ -AMPS ③ *OHMS ③
T=0.
VOLTS ② =VOLTS ① -(T+AMPS ②)*OHMS ①
T=0.
AMPS ② =(VOLTS ② -VOLTS ③)/OHMS ② -T
TRANSFER TO BACK
END OF PROGRAM

```

```

$ COMPILE MAD
EXTERNAL FUNCTION (ARG1,ARG2,NC)
ENTRY TO ITTR.
WHENEVER .ABS.(ARG1-ARG2).G..01*(.ABS.ARG1),NC=1
ARG1=(ARG1+ARG2)/2.
END OF FUNCTION
FUNCTION RETURN ARG1

```

ITTR

7.6 NUMERICAL CALCULATIONS

The following is a numerical calculation of the Desired Results using the Complete Program. These calculations were carried out to show that the Complete Program does indeed yield the Desired Results to within a preselected accuracy limit. The accuracy limit, which was preselected in the iteration subroutine, is that the difference between the guessed and calculated values be less than 1% of the calculated value.

The values chosen for the Input Parameters are:

```

VOLTS (BATT(NEG)) = 0.
OCV (BATT(NEG)) = 10.
OHMS (BATT(NEG)) = 1.
OHMS (RES, 1(1)) = 10.
OHMS (RES, 2(1)) = 5.

```

Since the problem considered is simple, the closed expression for AMPS(RES, 2(1)) is given by the following expression:

$$\text{AMPS}(\text{RES}, 2(1)) = \frac{\text{OCV}(\text{BATT}(\text{NEG}))}{\text{OHMS}(\text{BATT}(\text{NEG})) + \text{OHMS}(\text{RES}, 1(1)) + \text{OHMS}(\text{RES}, 2(1))}$$

By making the appropriate substitution in the above equation, the actual value of AMPS(RES, 2(1)) is 0.625 amps. By comparing the actual value of AMPS(RES, 2(1)) to the value calculated by the Complete Program which was generated by the Performance Simulator, we see that the program did indeed give the Desired Results to the preselected accuracy limit.

Numerical Calculations

NOTE: The following calculations have been simplified by deleting statements which are not pertinent in illustrating the calculation procedure but are pertinent to the actual machine calculations.

ALSO NOTE: REPEAT = 1B signifies the desired accuracy has not been obtained.
 REPEAT = 0B signifies the desired accuracy has been obtained.
 GAMPS represents Guessed Amps.

First Iteration	TRYCNT	= 1
	REPEAT	= 1B
Initial Guess	AMPS (2)	= 0
	GAMPS (2)	= 0
	AMPS (1)	= 0
	AMPS (3)	= 0
	VOLTS (1)	= 10.
Calc	VOLTS (2)	= 10.
Second Iteration	TRYCNT	= 2
	REPEAT	= 1B
Ittr	AMPS (2)	= 1
	GAMPS (2)	= 1
	AMPS (1)	= 1
	AMPS (3)	= 1
	VOLTS (1)	= 9
	VOLTS (2)	= -1
Calc	AMPS (2)	= -.2
Third Iteration	TRYCNT	= 3
	REPEAT	= 1B
Ittr	AMPS (2)	= .4
	GAMPS (2)	= .4
	AMPS (1)	= .4
	AMPS (3)	= .4
	VOLTS (1)	= 9.6
	VOLTS (2)	= 5.6
Calc	AMPS (2)	= 1.12
Fourth Iteration	TRYCNT	= 4
	REPEAT	= 1B
Ittr	AMPS (2)	= .76
	GAMPS (2)	= .76
	AMPS (1)	= .76
	AMPS (3)	= .76
	VOLTS (1)	= 9.24
	VOLTS (2)	= 1.64
Calc	AMPS (2)	= .328

Fifth Iteration

Ittr

Calc

Sixth Iteration

Ittr

Calc

Seventh Iteration

Ittr

Calc

Eighth Iteration

Ittr

Calc

Ninth Iteration

Ittr

Calc

Tenth Iteration

Ittr

TRYCNT	= 5
REPEAT	= 1B
AMPS ②	= .544
GAMPS ②	= .544
AMPS ①	= .544
AMPS ③	= .544
VOLTS ①	= 9.456
VOLTS ②	= 4.016
AMPS ②	= .8032
TRYCNT	= 6
REPEAT	= 1B
AMPS ②	= .6736
GAMPS ②	= .6736
AMPS ①	= .6736
AMPS ③	= .6736
VOLTS ①	= 9.3264
VOLTS ②	= 2.5904
AMPS ②	= .51808
TRYCNT	= 7
REPEAT	= 1B
AMPS ②	= .59584
GAMPS ②	= .59584
AMPS ①	= .59584
AMPS ③	= .59584
VOLTS ①	= 9.40416
VOLTS ②	= 3.44576
AMPS ②	= .689152
TRYCNT	= 8
REPEAT	= 1B
AMPS ②	= .642496
GAMPS ②	= .642496
AMPS ①	= .642496
AMPS ③	= .642496
VOLTS ①	= 9.357504
VOLTS ②	= 2.932544
AMPS ②	= .5865088
TRYCNT	= 9
REPEAT	= 1B
AMPS ②	= .6145024
GAMPS ②	= .6145024
AMPS ①	= .6145024
AMPS ③	= .6145024
VOLTS ①	= 9.3854976
VOLTS ②	= 3.2404736
AMPS ②	= .64809472
TRYCNT	= 10
REPEAT	= 1B
AMPS ②	= .63129856
GAMPS ②	= .63129856
AMPS ①	= .63129856
AMPS ③	= .63129856
VOLTS ①	= 9.3687014
VOLTS ②	= 3.0557158

Calc
Eleventh Iteration

Ittr

Calc
Twelveth Iteration

Calc
Thirteenth Iteration

Ittr

Calc
Final Iteration

Ittr

Final Calc

```

AMPS(2) = .61114316
TRYCNT = 11
REPEAT = 1B
AMPS(2) = .62122086
GAMPS(2) = .62122086
AMPS(1) = .62122086
AMPS(3) = .62122086
VOLTS(1) = 9.3787791
VOLTS(2) = 3.1665704
AMPS(2) = .6331409
TRYCNT = 12
REPEAT = 1B
AMPS(2) = .62726748
GAMPS(2) = .62726748
AMPS(1) = .62726748
AMPS(3) = .62726748
VOLTS(1) = 9.3727325
VOLTS(2) = 3.1000577
AMPS(2) = .62001154
TRYCNT = 13
REPEAT = 1B
AMPS(2) = .62363951
GAMPS(2) = .62363951
AMPS(1) = .62363951
AMPS(3) = .62363951
VOLTS(1) = 9.3763604
VOLTS(2) = 3.1399653
AMPS(2) = .62799307
TRYCNT = 14
REPEAT = 0B
AMPS(2) = .62581629
GAMPS(2) = .62581629
AMPS(1) = .62581629
AMPS(3) = .62581629
VOLTS(1) = 9.3741827
VOLTS(2) = 3.1160197
AMPS(2) = .62320395

```

AMPS(RES,2(1)) = .62320395

NOTE: The program generated by the Performance Simulator went through one additional iteration after the preselected accuracy limit had been attained.

After a language for communication of information concerning a system to be simulated is established, the job of the simulator program remains. That job is the translation of the various statements allowed by the language into an algorithm or solution procedure for the system simulation requested.

This is accomplished by several sections of program. The sections and their function are:

- 1) Preprocessing
- 2) Desired Result Reduction
- 3) Program Generation

The preprocessing phase consists of decomposing, analyzing, and re-generating the information from the source program statements in a form more easily handled by the machine. Input Parameters and Desired Results are saved in a very condensed form. Since each attachment point may have up to 70 parameters and these may fall into two groups (input parameters and desired results), each point must retain information on 140 items. Thus 200 attachments require 28000 items to be stored. These items are, fortunately, Boolean constants. In particular, the Boolean constant for an Input Parameter is 1 (True) if the parameter has been stated to be given. The constant is 0 (False) otherwise. For desired results, the constant is 1 if the parameter is required as an output and 0 otherwise. Since the computer is a binary machine it is possible to identify each of the 36 binary digits in the computer word with a specific parameter and thus save the status of 36 parameters in a single storage location. The entire parameter status is compressed into four words for each attachment by the Preprocessing section.

A second task of the Preprocessing section is to generate an image of the system to be simulated within the machine. This is done by forming a connection matrix. This array retains the nature of each attachment point. Each attachment is entered as the joint between two elements. Four items are required to specify uniquely an attachment on an element and thus eight locations specify a connection. The result is an $8 \times n$ matrix, where n is the number of connections in the system. The matrix entries are the true names of the elements, attachments, and identifiers either as supplied directly by the connection statements or as replaced by synonyms.

The connection matrix thus generated may be quite disordered so far as efficient processing is concerned. After the matrix is completely entered in the machine a sorting is done using an indirect list address array to arrange the matrix in the order of occurrence of the Element Descriptions on magnetic tape. The indirect address lists allow the matrix to remain stationary in memory while the effective order is completely changed. Since the finding of information on magnetic tape is the most time consuming of all the operations every effort is used to save tape movement. The ordering also provides for an effective method for locating all occurrences of an element in the connecting array without searching the entire array. This is done as follows:

Let the connection array be denoted:

$$\begin{array}{cccccccc} \text{EL}_{1L} & \text{EID}_{1L} & \text{AT}_{1L} & \text{AID}_{1L} & \text{EL}_{1R} & \text{EID}_{1R} & \text{AT}_{1R} & \text{AID}_{1R} \\ \text{EL}_{2L} & \text{EID}_{2L} & \text{AT}_{2L} & \text{AID}_{2L} & \text{EL}_{2R} & \text{EID}_{2R} & \text{AT}_{2R} & \text{AID}_{2R} \\ \text{EL}_{nL} & \text{EID}_{nL} & \text{AT}_{nL} & \text{AID}_{nL} & \text{EL}_{nR} & \text{EID}_{nR} & \text{AT}_{nR} & \text{AID}_{nR} \end{array}$$

Where

EL = any true element name
 EID = any true element identifier
 AT = any true element attachment name
 AID = any true attachment identification

and the subscripts iL and iR denote the attachment point occurring to the "left" and to the "right" and the i -th such attachment point.

The ordering procedure is then:

- 1) Order the matrix by the EL_{iL} (and group each EL and EID) according to the arrangement of descriptions on the magnetic tape. Call this ordering vector "L".
- 2) Order the matrix by the EL_{iR} in the same way and call this ordering vector "R".

Let i be incremented from 1 through the number of connections, say n . Let L_i (R_i) be the value of the i -th location in the L (R) vector. Then the i -th member of the EL_L (EL_R) is found in EL_{L_i} (EL_{R_i}). This type of list addressing is known as "indirect" addressing.

- 3) Construct a vector for the vector L whose values are the locations of the first occurrence of each element addressed through L in the vector R . If no such occurrence exists in R , then insert the negative of the first occurrence of the element in L itself. Call this vector " L TO R ".
- 4) Construct a similar vector for the vector R relating the occurrences in R to L . Call this vector " R TO L ".

With these vectors the task is simplified for finding the entire set of occurrences of any identified element. The location of all occurrences is accomplished in the following way: (The method is given for L but is equally valid, with appropriate changes, for R)

- 1) If L TO R at a point is negative, the element does not occur in R . The first occurrence in L is found by taking the absolute value of L TO R .
- 2) Each entry in L agrees with the first as long as the corresponding L TO R corresponds to the first L TO R .
- 3) If L TO R is positive, the first L occurrence is found by going first to the first occurrence of the element in R and then back to L by using the associated R TO L value. The same test as in 2 applies to equivalent identified elements.
- 4) If L TO R is positive, the value gives the first R occurrence. The succeeding values in R are for the same identified element so long as R TO L for each succeeding value agrees with the first R TO L value.

The remaining task of Preprocessing is to save all input-output requirements and functions substitutions for the Program Generation. In addition, should the library complement be incomplete, the Preprocessing must construct the library entries. Each Element Description is saved as two files on the tape known as ELTAPE. The first saves the contents of each collection capability in 80 word blocks. This allows two words for input parameters and two words for desired results at each of the 20 allowable attachments. The words are in the

Boolean form previously described. Since two words allow for 72 parameters and only 70 parameters are allowed, the remaining bits are available for special use. In particular, the last bit in the desired result word is used to signal that this capability must apply to elements without this attachment.

The second file contains the MAD statements to generate the capability contained in the first file. The first capability group in the first file corresponds to the first collection of MAD statements in the second file and so on.

Upon completion of the Preprocessing, the status of the storage is as follows:

- 1) The connection matrix is in and contains only true names. The matrix is ordered and the input-output requirements have been packed in Boolean parameter words.
- 2) The Element Descriptions are processed and saved in groups of two files per description on tape ELTAPE, with all permanent description first.
- 3) The function substitutions and input-output parameters in complete notation are saved for the program generation phase on an erasable tape.

At this point, control is passed to the Desired Result Reduction section. This section is charged with the actual generation of the algorithm for simulating the system. The procedure for accomplishing this task is almost the reverse of the usual procedure used by humans in attempting the same task. The human approach, largely because of the extremely large storage capacity of the human brain, is a search that proceeds from the known parameters and is directed toward the desired results. This approach could be implemented in the machine but because of storage limitations may become quite unworkable. The difficulty is that the machine program cannot reject a method until it can be shown to be unnecessary in the program to obtain the desired results. Thus the program would be forced to enumerate all the possible methods available from the input parameters plus the results of the first set and so on. The number of methods available grows rapidly and if the problem is well-posed the desired results will eventually be encompassed. However, this constitutes an exhaustive search with only a small fraction of the methods actually of use.

Therefore a different approach is used. Essentially, the algorithm is produced in reverse by working from the desired results toward the input parameters. In this way every step generated is necessarily of use in the program. The program is, of course, backward, in that the first statement collection specified is the last one needed and so on but this is easily taken care of by the Program Generation section. The method of production of the algorithm is the following:

- Step 1. Inspect the "Desired Result" Boolean words for each connection point in the matrix. Whenever no Desired Result bits can be found in the entire matrix the algorithm is completed.
- Step 2. Whenever a connection is found for which results are desired, steps must be taken to satisfy the request for results.
 - 2A. The requested results may be input parameters. If this is so, remove the corresponding desired result bits.
 - 2B. The requested results may occur at any identified attachment and be of broad scope. If this is so and the result (as an input parameter) can be found at any of the identified attachments, remove the corresponding desired result bit.
 - 2C. If requested results still remain after Steps 2A and 2B, then some additional program must be added to obtain the results.
 - 2C1. Find all of the statement collections for both elements that occur at this attachment point that are useful in obtaining the requested results. That is, ignore any collections that are "without" attachments specified for this element or collections that do not happen to produce any of the desired results.
 - 2C2. Check each useful collection to determine its effectiveness. The effectiveness is the ratio of the number of requested results the collection produces to the number of new requests for results the collection will produce. A new request for results will occur if any of the parameters required by the collection is not an input parameter or already requested by previous statements.

If the number of new requests for results is zero, the collection is always inserted in the algorithm. This collection produces results without requiring any new information.

(The only exception to this rule occurs with the iterative ESTIMATE collection. In this case, no new information request is apparent, however, the ESTIMATE collection is restrained from inclusion in the program until the parameter in question is found by at least one independent calculation method.) Otherwise, retain the effectiveness ratio as the weight of the collection.

- 2C3. When all of the collections have been examined for effectiveness and if desired results still remain, select a set of the collections that will produce the requested results.

At this point, the methods in which a parameter could be found using a method only once at a given attachment point are checked and discarded if already used. Otherwise, these methods are simply placed in competition with any other techniques available.

- 2C3A. First check to be certain that every requested result can be found in at least one way. If any result cannot be so found, the problem may not be well posed. The problem is not well posed if no "branches" have occurred previously in the generation. A "branch" occurs when a choice is made between more than one method of determining a requested result.
- 2C3B. Whenever there is exactly one method for producing a result, this method must be included at this point in the algorithm. The method is inserted, the results produced by the method have the corresponding bits removed, and any new requests occurring anywhere in the matrix have the corresponding bits inserted.
- 2C3C. After all single method results are taken care of there remain only results for which there are several used for each result the selection will constitute a "branch" in the algorithm generation, if the method selected is not always forced to be the same one.

If one considers the available methods, each with its associated weight, the simulator should tend to choose the method of greatest weight. However, the simulator should be allowed to select the method on the basis of the probability of selection being proportional to the weight. If this is not done, one may anticipate that in some case the method of greatest weight may contain a parameter that is incapable of calculation (consider-

ing the input parameter) and therefore the program could not be generated. If, however, the simulator makes the selection probabilistically, the method of greatest weight is most likely to be selected although other methods may be selected in its place. The probabilistic selection is automatically made and the "branch" Boolean constant is set to one. In this way, if later there should arise a case in which no method is available, the simulator may make another trial and possibly work out a satisfactory algorithm by having the chance to choose another method at this point. This is a situation in which the locally "best" method is not always the globally "best" but tends to be so.

Step 3. Steps 1 and 2 are repeated over and over. Each time the requested results are satisfied, a new set of requests are generated except when the request matches an input parameter. If the problem is well posed then a sequence of methods may be found such that all desired results are satisfied, through the sequence, by input parameters. When this has been done, an algorithm for the simulation of the system has been produced.

The algorithm produced tends to be optimal since at each state the method of greatest weight was most likely to be employed but the simulator cannot, with limited storage, view the generation of the algorithm beyond a single step. Thus occasionally the simulator may produce several steps that might be condensed if more information were available. In particular, it may happen that identical set of statements may be produced in the algorithm at different stages of the generation. This redundancy is easily detected and the final algorithm will contain only the first occurrence of the set.

The method of probabilistic selection is also used to discard the least likely method should there be found too many methods to apply at a given point.

The method of probabilistic selection for picking an item from a group on n weighted items is the following.

Let $W > 0$ be the weight of the i -th item from a group of n total items.

Let $W = \sum_{i=1}^n W_i$ be the total weight of the group.

Let N_0^W be a random number selected from a uniformly distributed set of random numbers on the interval 0, W.

Then the K-th member of the group if n items will be selected for the smallest K such that

$$\sum_{i=1}^K W_i \geq N_0^W.$$

N_0^W is most likely to fall in the subinterval such that W_i is maximum but may fall anywhere in the interval. A modification of this method to pick the least likely item (for discard, etc.) consists of defining a new set of weight $\rho = 1/W_i$ and making the selection using ρ in place of W_i . It should be noted particularly that equally probable alternatives receive equal chances and every alternative, no matter how small its weight may be, receives some consideration and may be chosen at any time. This method should find many applications in future programs.

Since there is no way to predict either the number of parameters that may be needed at a point or the number of methods available for any parameter it is necessary to allow an extremely flexible storage assignment so that the storage may be completely used. This is done by means of an "associative memory" list for the storage region. This list functions as follows:

- 1) Associated with each parameter at the attachment point is a storage location whose value is:
 - 1A) Zero if the parameter is not required.
 - 1B) Minus one if the parameter is not required and no method has yet been found to yield the parameter.
 - 1C) Otherwise, the value is a positive integer giving the location of the beginning of the list of methods for this parameter in the "associative memory".

- 2) Each entry in the associative memory list gives the location of the next (associated) entry. The final entry is denoted by a minus sign.
- 3) New entries are made by consulting the associative memory list beginning at the 0th location. The value of this location is the next available location in the memory. An addition to the end of any list is made in the available location, the value that was in this location is stored in the 0th location and the former list end is changed to refer to the new list end.
- 4) Whenever a result collection is selected, the storage space is reassigned as available storage by placing the starting location for the list in the 0th location, and the value formerly in the 0th location at the end of the list being removed. In this way the entire list is made available with only two storage reassignments.
- 5) If the capacity of the storage is exceeded before all the parameters have been treated space can be created by selecting the parameter with the greatest number of methods and picking the method least likely using the probabilistic section technique. The location thus chosen is made available by giving its address to the 0th location and reassigning the preceeding list location to skip this location and refer to the next item in the list.

Upon completion of the removal of all the desired results and those created during the removal of others, the algorithm is completed and written in reverse order on magnetic tape. This type of storage is used because it is not possible to predict the storage required for a program in advance. This is somewhat unfortunate since the program is generated in the form of a "push down" list. A push down list is a list such that each entry occurs at the beginning (rather than the end) of the list and thus moves the former first item to second place, the former second to third and so on. Thus the items are "pushed down" on the list. Removal of items from the list occurs from the beginning of the list with the last item entered. Thus the effective order of the list is reversed. This is precisely the action that must occur in the program generated since the last statement collection found must be the first used in the program and the first collection found is the last one used in the program. The difficulty is resolved by moving the tape backward two records and forward one working from the last record written toward the first.

As this process is begun after the completion of the algorithm, the simulator is at this point generating the simulation program using the Program Generator. The first output of the Program Generator is the Prologue and its associated input-output statements. The Prologue is followed by the program. The algorithm is stored in a short code giving the connection matrix row number and index in the L vector so that the unique connection could be located by the program generator, the element description name given by position in the element name vector and tagged with a plus (+) sign if the element involved was the left-most and a minus (-) sign if the right-most, and finally the statement collection number. The program generator section moves to the second file in the desired element description and next to the appropriate statement collection. Finally the statement collection is processed and produced both on cards and in print to form the desired simulation program. The rules by which the processing takes place have been stated in the section describing Element Descriptions. Briefly, the MAD statements are written using floating statement labels, and special codes for function substitutions and for parameter and attachment codes. The Program Generator assigns unique fixed statement labels for the floating statement labels. Any function substitution is checked for possible modification. If a substitution has been requested, the substitution is made, otherwise the original text is retained. The parameter and attachment codes are reduced to a six character variable name code for each parameter-attachment combination occurring. Some of the six characters may be blanks. Non-identified attachment parameters are immediately coded and inserted in the output statement. Identified attachments cause multiple copies of the statement to be generated. One copy is made for each different identifier. To avoid possible ambiguity, only one attachment may be identified in each statement. However, this attachment may occur any number of times with any number of parameters within one statement. In this way the effect of a special junction element is produced without specifically requiring such an element.

As noted in the Collection Capability section, there is one exception to this rule. Namely, when an attachment is identified and the current point in the connection matrix agrees exactly with this attachment name, a copy of the statement is produced for all occurrences except the current one. Finally, if an attachment is not identified, the current attachment point will be selected if the name agrees with the name occurring in the collection statement. Otherwise, the first occurrence of the name is used in the code.

Upon completion of the program generation the control is returned to the Preprocessing Section to process any other system simulation problems that may be waiting. Since the output of this program is a program in MAD code and on punched cards, the simulation program may be used as it is produced or modified easily before using it to simulate the system. To use the program as it is generated, the user need only supply the data and special subroutines needed and the special cards needed by the executive system for the data processing system. The program will be translated into machine code and executed using the data supplied.

SCOMPIL MAD,EXECUTE,DUMP,PUNCH OBJECT,PRINT OBJECT

CORE100

STCC(0)	R'N	
	CONTINUE	
	EXECUTE PLACE.(FPAR(0),69,0)	
	EXECUTE PLACE.(ZPAR(0),69,0)	
STCC(1)	CONTINUE	
	EXECUTE PLACE.(A(0),15599,\$ \$)	
	A(15853)=2	
	A(15854)=401	
	A(15855)=400	
	ABRSW=0B	
	ATT CNT = 0	
	CAT = 0B	0320
	CCNT = 0	0340
	CEID = 0B	0310
	CEL = 0B	0300
	CKRN=0B	
	CLC=0B	0720
	CSIDE = 0	0330
	CSUB = 0	0350
	CTO = 0B	0290
	DCNT = 0	
	DONE = 0B	
	E2NDPN = 0B	0500
	EAT = 0B	0480
	ECNT = 0	0510
	EEID = 0B	0470
	EEL = 0B	0460
	EOFB = 0B	
	EPAR = 0B	0450
	ESUB = 0	0520
	FIVE = 9	
	FOUR=4	
	FSCNT = 0	0680
	FSEL = 0B	0670
	FSNEW = 0B	0650
	FSOLD = 0B	0660
	FSSUB = 0	0690
	I = 71	
	ID1=\$ \$	40
	ID2=\$ \$	50
	IFERR1=0B	0560
	IFERR2=0B	0570
	K1=0	0550
	LASTCH=\$.\$	60
	NTAPE=0B	0710
	NUELTP = 0B	721
	OUT=0B	0640
	PARCNT = 0	
	PAT=0B	0610
	PCNT=0	0620
	PCNT(1)=0	0630
	PEID=0B	
	PEL=0B	0600
	PELCNT = 0	
	PEQL=0B	
	PLP1=0B	0580
	PLP2=0B	0590
	PNUM = 0	

	PNUM(1) = 0	
	PPER=0B	
	S1STEQ = 0B	0410
	S1STPN = 0B	
	S2NDPN = 0B	0400
	SAT = 0B	0390
	SBIT(0) = 0	
	SBIT(1) = 0	
	SCNT = 0	0420
	SCOPB1 = 0	
	SCOPB2 = 0	
	SEID = 0B	0380
	SEL = 0B	0370
	SPAR = 0B	0360
	SSIDE = 0	0440
	SSUB = 0	0430
	TC = THREE	
	TELCNT = 0	
	THREE=3	
	TIME = 60	
	EXECUTE TODAY.(Z1,Z2)	
	Z3=DAYTIM.(0)	
	ZH=Z3/216000	
	ZM=(Z3-ZH*216000)/3600	
	ZS=(Z3-ZH*216000-ZM*3600)/60	
	W'R ZH .L. 12	
	Z4=\$AMS	
	O'R ZH .E. 12	
	Z4=\$PMS	
	O'E	
	Z4=\$PMS	
	ZH=ZH-12	
	E'L	
	P'T \$H*1DATA AS GIVEN TO SYSTEM SIMULATOR*,S60,2C6,S5,I2,I3,	
	1I3,S1,C6*\$,Z1,Z2,ZH,ZM,ZS,Z4	
	READ AND PRINT DATA (IDENT,TIME) (TIME IS OPTIONAL)	
	EXECUTE START.(IDENT)	
	EXECUTE SETEOF.(EOF1)	
	TRANSFER TO RESET1	0750
READ1	EXECUTE INPUT.(CH...END1,WORD)	
	EQUIVALENCE(EOF1,END1)	
	WHENEVER NTAPE.OR.CLC	0770
WTAPE	CONTINUE	
	WRITE BINARY TAPE TC, WORD(0),..., WORD(11)	
	WHENEVER CH.NE.\$+\$, TRANSFER TO READ1	0800
	WHENEVER CLC	0810
	CLC=0B	0820
	PRINT COMMENT \$OCALCSS	
	END OF FILE TAPE THREE	
	CLDONE = 1B	
	R CALCULATIONS ARE NOW ON TAPE THREE	
	TRANSFER TO CLBCK	0830
	END OF CONDITIONAL	0840
	PRINT COMMENT \$(EOF4)\$	
	END OF FILE TAPE FOUR	
	R PROLOGUE IS NOW ON TAPE FOUR	
	NTAPE = 0B	0860
CLBCK	I = 71	
	DCNT=0	0870

	LASTCH=\$.\$	
	TRANSFER TO RESET1	0900
	END OF CONDITIONAL	0910
	WHENEVER CH.E. \$/\$, TRANSFER TO READ1	0920
	WHENEVER CH .E. \$*\$	0930
	ID1=WORD(1)	0940
	ID2=WORD(2)	0950
	TRANSFER TO READ1	0960
	END OF CONDITIONAL	0970
	WHENEVER LASTCH .NE. \$.\$.AND. CH .NE. -\$=\$	0980
	LASTCH=\$.\$	0990
	I=-1	
	TRANSFER TO S(K1,1)	1020
	END OF CONDITIONAL	1030
	THROUGH LOOP, FOR I = 0 , 1, I .G. 71	1040
	NUCHAR = CH(I)	1050
	WHENEVER NUCCHAR.E.\$ \$, TRANSFER TO LOOP	1060
	LASTCH=NUCHAR	1070
ERRTST	THROUGH ERRTST, FOR K=0,1,NUCHAR.E.ILCHAR (K).OR.K.G.5	1080
	VECTOR VALUES ILCHAR=\$+\$,\$-\$,\$/\$,\$*\$,\$0\$,\$0\$	1090
	WHENEVER K.LE.5	1100
	P'T ERR1,I+1	
	V'S ERR1=\$H*0ILLEGAL CHARACTER IN THE ABOVE LINE, CC *I2,/*\$	
	IFERR1=1B	1150
	I = 71	
RESET1	HOLCNT=1	1180
	NUWORD=\$ \$	1190
	TRANSFER TO LOOP	1200
	END OF CONDITIONAL	1210
	R	
	R PUNCTUATION CHECK	
PUNTST	THROUGH PUNTST, FOR K=0, 1, NUCCHAR.E.PUNCT (K). OR. K.G.4	1220
	VECTOR VALUES PUNCT=\$,\$,\$,\$,\$(\$,\$)\$,\$,\$=\$	1230
	W'R K .LE. 4 .OR. HOLCNT .G. 6 .OR. ABRSW	
	TRANSFER TO S(K1,K)	
	END OF COIDITIONAL	
	NUWORD=INSRTC.(NUWORD,HOLCNT,NUCHAR)	1250
	HOLCNT=HOLCNT+1	1260
	W'R NUCCHAR .E. \$'\$', ABRSW=1B	
LOOP	CONTINUE	1270
	R READ IN NEXT CARD	
	TRANSFER TO READ1	1280
	R END OF FILE ON INPUT	
	R (**PROGRAM FINISHED. NO MORE DATA**)	
EOF1	EOFB=1B	1300
	EXECUTE END.(0)	
	R (** NEXT SET OF DATA IS READY, COMPLETE THIS PROGRAM FIRST	
	R NEXT SET OF DATA	
NXTSET	CUT=1B	1360
	EXECUTE END.(1)	
	R FINISH PROCESSING	
	REWIND TAPE FOUR	
	WRITE BINARY TAPE THREE,\$FSUB,\$,FSCNT,FOMAT...FOMAT(5)	
	V'S FOMAT=\$PADPADPADPADPADPADPADPADPADPAD\$	
	T'H TLOOP, FOR I=0,1, I .E. FSCNT	
TLOOP	WRITE BINARY TAPE THREE,A(0,I),A(1,I),A(2,I),A(3,I)	
	END OF FILE TAPE THREE	
	WRITE BINARY TAPE THREE,\$INPUT\$,PCNT(0),FOMAT...FOMAT(5)	
	T'H ILOOP, FOR I=0,1, I .E. PCNT(0)	

ILOOP	WRITE BINARY TAPE THREE,A(4,I),A(5,I),A(6,I),A(7,I),A(8,I)	
	END OF FILE TAPE THREE	
	WRITE BINARY TAPE THREE,\$DESRUSS,PCNT(1),FOMAT...FOMAT(5)	
	T'H DLOOP, FOR I=0,1, I .E. PCNT(1)	
DLOOP	WRITE BINARY TAPE THREE,A(9,I),A(10,I),A(11,I),A(12,I),A(13,I	
	1)	
	END OF FILE TAPE THREE	
	REWIND TAPE THREE	
	W'R NUELTF	
	R FILE 1 MUST BE REWRITTEN FOR NEW TAPES.	
	WRITE BINARY TAPE FOUR,\$ELTAPES,IDENT,FOMAT...FOMAT(5)	
	WRITE BINARY TAPE FOUR,PELCNT,TELCNT,PARCNT,FOMAT...FOMAT(5)	
	WRITE BINARY TAPE FOUR,A(15604)...A(15853)	
	REWIND TAPE FOUR	
	E'L	
	R (** PROCESS DATA IN A MATRIX WITH SIMULATOR SUBROUTINES.**\$.	
	EXECUTE END.(2)	
	R DECLARATION SECTION K1=0*****	1470
S(35)	CONTINUE	1480
S(29)	CONTINUE	1490
S(23)	CONTINUE	1500
S(17)	CONTINUE	1510
S(11)	CONTINUE	1520
	K1=0	1530
S(5)	W'R ABRSW	
ABRINS	ABRSW=0B	
	NUWORD=INSRTC.(NUWORD,HOLCNT,NUCHAR)	
	E'L	
	DWORD(DCNT)=NUWORD	
	R	
	DCNT=DCNT+1	1550
	WHENEVER DCNT.G.4, TRANSFER TO S(4)	1560
	TRANSFER TO RESET1	1570
S(17)	T'H S(17), FOR N=0,1, ABR1(N) .E. DWORD .OR. N .G. MAXCNT	
	W'R N .G. MAXCNT, T'O S(4)	
	T'O S(1(N)	
S(0)	CONTINUE	1580
S(2)	CONTINUE	1590
S(3)	CONTINUE	1600
S(18)	CONTINUE	
S(19)	CONTINUE	
S(20)	CONTINUE	
S(4)	PRINT COMMENT \$0***** DECLARATION ERROR IN THE LINE ABOVE\$	
	P'IT DWORDA,DWORD...DWORD(DCNT-1)	
	V'S DWORDA=\$1H0,22C6*\$	
	IFERR2=1B	1630
INS(1)	DCNT=0	1640
	I = 71	
	TRANSFER TO RESET1	1670
S(1)	DWORD(DCNT)=NUWORD	1680
	DCNT=0	1690
DCL	THROUGH DCL, FOR N=0, 1, DKIND1(N).E.DWORD (0).AND.DKIND2(N)	1700
	1.E. DWORD(1).OR.N.G. MAXCNT	
	VECTOR VALUES MAXCNT= 16	
	TRANSFER TO S(1(N)	1720
	R ELEMENT DESCRIPTION SECTION K1 = 6	
S(0)	K1 = 6	
	WHENEVER .NOT. ELTPOK.AND.PELCNT.G.0.AND..NOT.NUELTP,	
	1EXECUTE SKFILE.(FOUR,2*(PELCNT-1))	

	NOATS=0	
	ELTPOK=1B	
	TRANSFER TO RESET1	
	R	
	R	INPUT PARAMETERS
S1(1)	K1=3	1740
	K2 = 4	
	K3=0	1760
	PNUM(K3)=0	
	TRANSFER TO RESET1	1770
	R	SYNONYM (S)
S1(12)	CONTINUE	1780
S1(2)	K1=2	1790
	TRANSFER TO RESET1	1800
	R	CONNECTION (S)
S1(11)	CONTINUE	1810
S1(3)	K1=1	1820
	TRANSFER TO RESET1	1830
	R	DESIRED RESULTS
S1(4)	K1=3	1840
	K2 = 9	
	K3=1	1860
	PNUM(K3)=0	
	TRANSFER TO RESET1	1870
	R	FUNCTION SUBSTITUTIONS
S1(16)	CONTINUE	
S1(5)	K1 = 4	1880
	TRANSFER TO RESET1	1890
	R	NEW ELEMENT TAPE.
S1(6)	CONTINUE	
	NTAPE=1B	1940
	R	(SYSTEM INPUT ON TAPE TWO.)
	NUELTP = 1B	
	TC = FOUR	
	WHENEVER I.LE. 0, TRANSFER TO WTAPE	
	TRANSFER TO READ1	1960
	R	CALCULATION (S)
S1(13)	CONTINUE	1970
S1(7)	CLC=1B	1980
	CLDONE = 0B	
	TC = THREE	
	END OF FILE TAPE THREE	
	WRITE BINARY TAPE THREE, \$ CALC CALC CALC CALC CALC \$	
	WHENEVER I.LE. 0, TRANSFER TO WTAPE	
	TRANSFER TO READ1	2000
	R	NEXT SET OF DATA
S1(8)	TRANSFER TO NXTSET	2010
	R	CHECK RUN
S1(10)	CONTINUE	2020
S1(9)	CKRN=1B	2030
	TRANSFER TO RESET1	2040
	R	EQUIVALENCE
S1(15)	K1 = 5	2050
	R	SAVE TAPE
S1(14)	TRANSFER TO RESET1	2060
	R	OTHER TYPES OF ERRORS *****
	R	= IN CONNECTIONS STATE.
S(10)	CONTINUE	2200
	R	= IN FUNCTION SUBSTITUTION

S(28)	CONTINUE	
	PRINT COMMENT \$OILLEGAL EQUAL SIGN ABOVE*****\$	
	IFERR1=1B	2270
	TRANSFER TO RESET1	2280
	RSYNONYM SECTION K1=2 *****	2830
	R SYNONYM STATEMENT DECOMPOSITION	2840
	R COMMA IN SYNONYMS	
S(12)	WHENEVER .NOT. SEL .AND. .NOT. S2NDPN	2850
	SSUB = 22	2860
	SEL = 1B	2870
	OR WHENEVER .NOT. SAT .AND. S2NDPN	2880
	SSUB = 24	2890
	SAT = 1B	2900
	WHENEVER NUWORD .E. \$O\$, NUWORD = \$ \$	2910
	OR WHENEVER .NOT. SEID .AND. .NOT. S2NDPN	2920
	SSUB = 23	2930
	SEID = 1B	2940
	END OF CONDITIONAL	2950
	TRANSFER TO SENTER	2960
	R PERIOD IN SYNONYMS	
S(13)	SSIDE = 0	2970
	SISTEQ = 0B	2980
SPRIOD	WHENEVER .NOT. SPAR, A(26,SCNT) = NUWORD	2990
	SCNT = SCNT+1	3000
	WHENEVER SCNT .G. 398	3010
	SCNT = SCNT -1	3020
	PRINT COMMENT \$OSYNONYM TABLE LENGTH EXCEEDED\$	
	END OF CONDITIONAL	3050
SRESET	SPAR = 0B	3060
	SEID = 0B	3070
	SAT = 0B	3080
	SEL = 0B	3090
	S1STPN= 0B	
	S2NDPN = 0B	
	BOOLEAN S1STPN	
	TRANSFER TO RESET1	3100
	R LEFT PAREN IN SYNONYMS	
S(14)	CONTINUE	
	WHENEVER S1STPN, S2NDPN= 1B	
	S1STPN= 1B	
	WHENEVER NUWORD.E. \$ \$	
	WHENEVER .NOT. SPAR, SPAR= 1B	
	TRANSFER TO RESET1	
	END OF CONDITIONAL	
	WHENEVER .NOT. SPAR	3130
	SSUB = 21	3140
	SPAR = 1B	3150
	S2ND PN = 0B	3160
	OR WHENEVER .NOT. SEL	3170
	SSUB = 22	3180
	SEL = 1B	3190
	OR WHENEVER .NOT. SEID	3200
	SSUB = 23	3210
	SEID = 1B	3220
	END OF CONDITIONAL	3230
	TRANSFER TO SENTER	3240
SENER	A(SSUB+SSIDE,SCNT) = NUWORD	3250
	TRANSFER TO RESET1	3260
	R RIGHT PAREN IN SYNONYMS	

S(15)	WHENEVER NUWORD .E. \$ \$, TRANSFER TO RESET1	3270
	WHENEVER .NOT. SEL	3280
	WHENEVER .NOT. S2NDPN	3290
	SEL = 1B	
	SSUB= 22	
	CR WHENEVER .NOT. SAT	3310
	SSUB = 24	
	SAT = 1B	
	OTHERWISE	3330
	SSUB = 25	
	END OF CONDITIONAL	3350
	CR WHENEVER .NOT. SEID .AND..NOT. S2NDPN	3360
	SSUB = 23	
	SEID = 1B	
	OR WHENEVER .NOT. SAT	3380
	SAT = 1B	
	SSUB = 24	
	OTHERWISE	3400
	SSUB = 25	3410
	END OF CONDITIONAL	3420
	TRANSFER TO SENTER	3430
	R	
	EQUAL SIGN IN SYNONYMS	
S(16)	WHENEVER S1STEQ	3440
	THROUGH SREWRT, FOR SSUB= 21,1, SSUB .G. 25	3450
SREWRT	A(SSUB,SCNT+1) = A(SSUB,SCNT)	3460
	TRANSFER TO SPRIOD	3470
	OTHERWISE	3480
	WHENEVER .NOT. SPAR, A(21, SCNT) = NUWORD	3490
	SSIDE = 5	3500
	S1STEQ = 1B	3510
	TRANSFER TO SRESET	3520
	ENDOF CONDITIONAL	3530
	R INPUT PARAMETERS K1=3 K2=4 K3=0*****	3540
	R DESIRED RESULTS K1 = 3 K2=9 K3 = 1	3550
	R COMBINED LOADER	3560
	R MAXIMUM OF 400 INPUT PARAMETERS	3570
	R MAXIMUM OF 400 DESIRED RESULTS	3580
	R	
	COMMA IN IPOR DR	
S(18)	CONTINUE	
	PEQCNT=PEQCNT+1	
	WHENEVER NUWORD.E.\$ \$,TRANSFER TO RESET1	
	W'R PEQL	
	A(14,ECNT)=NUWORD	
	T'H PEQU, FOR PJ=0,1, PJ .G. 4	
PEQU	A(15+PJ,ECNT)=A(K2+PJ,PCNT(K3)+PEQCNT)	
	T'O ECNTIC	
	E'L	
	WHENEVER PLP1	
	WHENEVER .NOT. PEL	
	PEL= 1B	
	CJ=1	
	OR WHENEVER .NOT. PAT	
	PAT= 1B	
	CJ=3	
	OR WHENEVER .NOT. PEID	
	PEID = 1B	
	CJ=2	
	OTHERWISE	
	CJ=4	

	END OF CONDITIONAL	
	A(K2+CJ,PCNT(K3))= NUWORD	
	TRANSFER TO RESET1	
	END OF CONDITIONAL	
	TRANSFER TO PNCNT	3670
	R	PERIOD IN IP OR DR
S(19)	PPER=1B	
	W'R PEQL, T.O S(18)	
PERET	T.H PCOPY, FOR PI=1,1, PI .GE. PNUM(K3)	
	THROUGH PCOPY, FOR PJ=1,1, PJ.G.4	3690
PCOPY	A(K2+PJ,PCNT(K3)+PI)=A(K2+PJ,PCNT(K3))	3700
	PCNT(K3)=PCNT(K3)+PNUM(K3)	3710
	WHENEVER PCNT(K3) .G. 379	3720
POVR	PRINT COMMENT \$ODRIPORS	
	END OF CONDITIONAL	3740
	PAT=0B	3780
	PEID= 0B	
	PEL=0B	3770
	PEQL=0B	
	PLP1=0B	3750
	PLP2=0B	3760
	PNUM(K3)=0	3790
	PPER=0B	
	TRANSFER TO RESET1	3800
	R	LEFT PAREN IN IP OD DR
S(20)	CONTINUE	
	WHENEVER PLP1	
	PLP2=1B	3820
	CJ=1	3830
	WHENEVER PEL	
	CJ=2	
	PEID= 1B	
	END OF CONDITIONAL	
	PEL =1B	3850
	WHENEVER NUWORD .E. \$ \$, TRANSFER TO RESET1	
	TRANSFER TO OMTD	3860
	OTHERWISE	3870
	PLP1=1B	3880
	WHENEVER NUWORD.E. \$ \$, TRANSFER TO RESET1	
PNCNT	A(K2, PCNT(K3)+PNUM(K3))= NUWORD	3890
	PNUM (K3)=PNUM(K3)+1	3900
	WHENEVER PCNT(K3)+PNUM(K3).G.399,TRANSFER TO POVR	3910
	END OF CONDITIONAL	3920
	TRANSFER TO RESET1	3930
	R	RIGHT PAREN IN IP OR DR
S(21)	CONTINUE	
	WHENEVER PLP2 .AND. PLP1	
	CJ=3	3950
	WHENEVER PAT, CJ=4	3960
	PLP1=0B	3970
	OR WHENEVER PLP1	3980
	CJ=1	3990
	WHENEVER PEL, CJ=2	4000
	OTHERWISE	4010
	TRANSFER TO RESET 1	4020
	END OF CONDITIONAL	4030
OMTD	A(K2+CJ, PCNT (K3))= NUWORD	4040
	TRANSFER TO RESET1	4050
	R	EQUAL SIGN IN IP OR DR.

S(22)	PEQL=1B PEQNT=-1 T'H PEQINT, FOR PI=1,1, PI .GE. PNUM(K3) T'H PEQINT, FOR PJ=1,1, PJ .G. 4 PEQINT A(K2+PJ,PCNT(K3)+PI)=A(K2+PJ,PCNT(K3)) T'O RESET1	
	R R FUNCTION SUBSTITUTIONS ***** R	4060
	COMMA IN FUNCTION SUBSTIT.	
S(24)	WHENEVER .NOT. FSNEW FSSUB = 0 FSNEW = 1B OR WHENEVER .NOT. FSOLD	4070 4080 4090 4100
	R LEFT PAREN IN FSB	
S(26)	TRANSFER TO S26P	
S26P	CONTINUE W'R NUWORD .E. \$ \$, T'O RESET1 FSSUB = 1 FSOLD = 1B OTHERWISE FSEL = 1B FSSUB = 2 END OF CONDITIONAL TRANSFER TO FENTER	4120 4130 4140 4150 4160 4170
	R RIGHT PAREN IN FSB	
S(27)	WHENEVER .NOT. FSEL FSEL = 1B FSSUB= 2 OTHERWISE FSSUB = 3 END OF CONDITIONAL TRANSFER TO FENTER	4180 4190 4200 4210 4220 4230
	R PERIOD IN FSB	
S(25)	WHENEVER .NOT. FSOLD,A(1,FSCNT)= NUWORD FSNEW = 0B FSOLD = 0B FSEL = 0B FSSUB=0 FSCNT = FSCNT +1 WHENEVER FSCNT .G. 399 FSCNT = FSCNT -1 PRINT COMMENT \$OFUNCTION SUBSTITUTION TABLE LENGTH EXCEEDED\$ END OFCONDITIONAL TRANSFER TO RESET1	4240 4250 4260 4270 4280 4290 4300 4340 4350
FENTER	A(FSSUB,FSCNT)= NUWORD TRANSFER TO RESET1 RCONNECTION SECTION K1=1 ***** R CONNECTIONS IN A(30,0)...A(38,399) RMAX OF 400 CONNECTIONS	4360 4370 2290 1460 2300
	R COMMA IN CONNECTIONS	
S(6)	WHENEVER NUWORD.E.\$TOS CTO = 1B WHENEVER CSIDE.E. 0 CSIDE = 4 OTHERWISE THROUGH RECOPY, FOR CSUB=31,1,CSUB.G. 34 A(CSUB,CCNT+1) = A(CSUB+4,CCNT) WHENEVER A(37,CCNT).E. \$ \$,A(37,CCNT) = \$1\$ WHENEVER A(33,CCNT+1) .E.\$ \$,A(33,CCNT+1)=\$2\$	2310 2320 2340 2350 2360 2370 2380 2390
RECOPY		

CRESET	CCNT = CCNT+1	2400
	WHENEVER CCNT .G. 399	2410
	CCNT =CCNT-1	2420
	PRINT COMMENT \$CONOVRS	
	END OF CONDITIONAL	2440
	END OF CONDITIONAL	2450
	CEL = 0B	2460
	CEID = 0B	2470
	CAT = 0B	2480
	TRANSFER TO RESET1	2490
	OR WHENEVER NUWORD .E. \$ \$	2500
	TRANSFER TO RESET1	2510
	OR WHENEVER .NOT. CEL	2520
	CSUB = 31	2530
	CEL = 1B	2540
	OR WHENEVER .NOT. CEID	2550
	CSUB = 32	2560
	CEID = 1B	2570
	OTHERWISE	2580
	CAT= 1B	
	CSUB= 33	
	END OF CONDITIONAL	2600
	TRANSFER TO CENTER	2610
	R	
	PERIOD IN CONNECTIONS	
S(7)	CONTINUE	
	WHENEVER CTO	
	WHENEVER .NOT. CEL	
	A(35,CCNT)= NUWORD	
	OR WHENEVER .NOT. CEID	
	A(36,CCNT)= NUWORD	
	END OF CONDITIONAL	
	CSIDE = 0	
	CSUB = 31	
	END OF CONDITIONAL	
	TRANSFER TO CRESET	2670
	R	
	LEFT PAREN IN CONNECTIONS	
S(8)	WHENEVER NUWORD .E. \$ \$,TRANSFER TO RESET1	2680
	WHENEVER .NOT. CEL	2690
	CSUB = 31	2700
	CEL = 1B	2710
	CEID = 1B	
	OTHERWISE	
	CSUB= 32	
	CEID = 1B	
	END OF CONDITIONAL	2720
	TRANSFER TO CENTER	2730
	R	
	RIGHT PAREN IN CONNECTIONS	
S(9)	WHENEVER .NOT. CAT	2740
	CSUB = 33	2750
	CAT = 1B	2760
	OTHERWISE	2770
	CSUB = 34	2780
	END OF CONDITIONAL	2790
	TRANSFER TO CENTER	2800
CENTER	A(CSUB+CSIDE,CCNT) = NUWORD	2810
	TRANSFER TO RESET1	2820
	R EQUIVALENCE DECOMPOSITION *****	4380
	R	
	COMMA IN EQU	
S(30)	CONTINUE	

	WHENEVER NUWORD.E.\$ \$,TRANSFER TO RESET1	
	WHENEVER .NOT. EPAR	
	ESUB= 15	
	EPAR=1B	
	OR WHENEVER .NOT. EEL	
	ESUB = 16	4400
	EEL = 1B	4410
	OR WHENEVER .NOT. EEID .AND. .NOT. E2NDPN	4450
	ESUB = 17	4460
	EEID = 1B	4470
	OR WHENEVER .NOT. EAT .AND. E2NDPN	4420
	ESUB = 18	4430
	EAT = 1B	4440
	END OF CONDITIONAL	4480
	TRANSFER TO EENTER	4490
	R	
	PERIOD IN EQU	
S(31)	WHENEVER .NOT. EPAR, A(15,ECNT) = NUWORD	4500
	EPAREN = 0	
	E2NDPN=0B	
	EPAR = 0B	4510
	EEL= 0B	4520
	EEID = 0B	4530
	EAT = 0B	4540
ECNTIC	ECNT=ECNT+1	
	WHENEVER ECNT .G. 399	4570
	ECNT = ECNT -1	4580
	PRINT COMMENT \$OEQUIVALENCE TABLE LENGTH EXCEEDED\$	
	END OF CONDITIONAL	4610
	W'R PPER, T'O PERET	
	TRANSFER TO RESET1	4620
	R	
	LEFT PAREN IN EQU	
S(32)	CONTINUE	
	WHENEVER EPAREN.E. 1, E2NDPN=1B	
	EPAREN=1	
	W'R NUWORD .E. \$ \$, T'O RESET1	
	WHENEVER .NOT. EPAR	4640
	ESUB = 15	4650
	EPAR = 1B	4660
	OR WHENEVER .NOT. EEL	4670
	ESUB = 16	4680
	EEL = 1B	4690
	OR WHENEVER .NOT. EEID	4700
	ESUB = 17	4710
	EEID = 1B	4720
	END OF CONDITIONAL	4730
	TRANSFER TO EENTER	4740
	R	
	RIGHT PAREN IN EQU	
S(33)	WHENEVER NUWORD .E. \$ \$, TRANSFER TO RESET1	4750
	WHENEVER .NOT. EEL	4760
	ESUB = 16	4770
	OR WHENEVER .NOT. EEID .AND..NOT. E2NDPN	4780
	ESUB = 17	4790
	OR WHENEVER .NOT. EAT	4800
	ESUB = 18	4810
	EAT = 1B	4820
	OTHERWISE	4830
	ESUB = 19	4840
	END OF CONDITIONAL	4850
	TRANSFER TO EENTER	4860

EENTER	A(ESUB, ECNT) = NUWORD	4870
	TRANSFER TO RESET1	4880
	R	
	EQUAL SIGN IN EQU	
S(34)	ESUB = 14	4890
	TRANSFER TO EENTER	4900
	R	
	R N = 0 STATEMENT COLLECTION	N = 4 ELEMENT DESCRIPTION 0170
	R N = 1 DESCRIPTION FINISHED	N = 5 NAME OF ELEMENT 0180
	R N = 2 PERMANENT	N = 6 BROAD SCOPE PARA. 0190
	R N = 3 ANOTHER DESCRIPTION	N = 7 ATTACHMENT NAME 0200
	R S(36) COMMA	S(39) RIGHT PAREN 0320
	R S(37)) PERIOD	S(40) EQUAL SIGN 0330
	R S(38) LEFT PAREN	S(41) MORE THAN 6 CHARACTERS 0340
S(41)	DONE=0B	
	W'R ABRSW .AND. DCNT .L. 4, T'O ABRINS	
	DWORD(DCNT)=NUWORD	
	DCNT = DCNT +1	0370
	WHENEVER DCNT .G. 4	0380
	PRINT COMMENT \$(ELEMENT DESCRIPTION WORDY****)\$	
	IFERR2=1B	
	TRANSFER TO NUSTAT	
	END OF CONDITIONAL	0420
	TRANSFER TO RESET1	
S(37)	CONTINUE	0450
	R PERIOD ENCOUNTERED	0460
	WHENEVER .NOT. DONE	0480
	DWORD(DCNT)= NUWORD	
	LOW = 0	490
	HIGH = 4	0500
LISTA	THROUGH LISTA, FOR N= LOW, 1, (DWORD(0) .E. PART1(N)	0510
	1 .AND. DWORD(1) .E. PART2(N))	0520
	2 .OR. N.G. HIGH	0530
	WHENEVER N .G. HIGH	0540
LISTB	T'H LISTB, FOR N=LOW,1, ABR2(N) .E. DWORD .OR. N .G. HIGH	
	W'R N .G. HIGH	
	PRINT COMMENT \$0INCORRECT STATEMENT OR SPELLING\$	
	TRANSFERTO NUSTAT	
	E'L	
	END OF CONDITIONAL	0580
	R(**WHEN N IS LESS THAN OR EQUAL TO HIGH, STATEMENT TYPE	
	R IS DETERMINED.) IF N.LE. 4, STATEMENT NEEDS NO FURTHER SCAN	
	DONE = 1B	0610
	WHENEVER N.LE.4, TRANSFER TO S2(N)	
	TRANSFER TO RESET1	0620
	R	0640
	OTHERWISE	0630
	DONE = 0B	0650
	R(**TRANSFER TO STATEMENT TYPE N. SEE ABOVE.*)	
	TRANSFER TO S2(N)	0750
	R	0760
	R	0770
	END OF CONDITIONAL	0780
	R	0790
	R	0800
	N = 0 1 2 3	
	VECTOR VALUES PART1 = \$STATEMS, \$DESCRIS, \$PERMANS, \$ANOTHES,	0810
	1\$ELEMENS, \$NAMEOFS, \$BROADSS, \$ATTACHS, \$ \$	0830
	R 4 5 6 7	0820
	VECTOR VALUES PART2=\$NTCOLLS, \$TIONFIS, \$NTS, \$DESCRIS,	
	1\$DESCRIS, \$LEMENTS, \$OPEPARS, \$ENTNAM\$	0850

	V'S ABR2= \$ST'N\$, \$DS'D\$, \$PR'T\$, \$AN'\$\$, \$EL'N\$, \$NM'T\$	
	1, \$BR'\$\$, \$AT'\$\$, \$ \$	
	R COMMA IN ELEMENT DESCRIPTION.	0870
S(36)	CONTINUE	0880
	WHENEVER .NOT. DONE	
	1, TRANSFER TO S(37)	
	TRANSFER TO S2(N)	0900
S(40)	WHENEVER .NOT. DONE	0910
	R EQUAL SIGN ENCOUNTERED - STATEMENT TYPE NOT DETERMINED	0920
	LOW = 5	0930
	HIGH =7	0940
	DWORD(DCNT)= NUWORD	
	R USE THE SAME SEARCH AS WHEN A PERIOD IS ENCOUNTERED	0950
	TRANSFER TO LISTA	0970
	END OF CONDITIONAL	0980
	PRINT COMMENT \$OILLEGAL EQUAL SIGN ABOVE****\$	
	TRANSFER TO NUSTAT	
	R LEFT OR RIGHT PAREN ENCOUNTERED IN ELEMENT DESCRIPTION	1030
S(38)	CONTINUE	1040
S(39)	PRINT COMMENT \$OPARENTHESIS IMPROPERLY USED IN ELEMENT DESCR	
	PTIONS	
	TRANSFER TO NUSTAT	
	R	1090
	R DESCRIPTION FINISHED.	1100
S2(1)	CONTINUE	1110
	R ANOTHER DESCRIPTIONS FOLLOWS	1130
S2(3)	NOSC(7) = ELNAME	1140
	P'T DWORDA, NOSC...NOSC(9)	
	VECTOR VALUES NOSC=\$NO STATEMENT COLLECTIONS INCLUDED WITH \$,	1160
	1\$ \$, \$ DESCRIPTIONS	1170
	R GO TO DESCRIPTION FINISHED AFTER A STATEME-	1180
	R MENT COLLECTION.	1190
	SK=2	
	TRANSFER TO FINISH	
	R NEW ELEMENT DESCRIPTION FOUND BEFORE LAST	1210
	R ONE FINISHED	1220
	R	1230
S2(4)	PRINT COMMENT \$ONEW DESCRIPTION BEGUN BEFORE LAST ONE FINISHE	
	1D\$	
	SK=2	
	TRANSFER TO FINISH	
	R	1300
	R	1270
	R PERMANENT. ENCOUNTERED WITHIN ELEMENT	1280
	R DESCRIPTION	1290
	R ELEMENT DESCRIPTION IS ASSUMED TEMPORARY UNLESS	1310
	R STATED PERMANENT	1320
S2(2)	WHENEVER TELCNT .E. 1	1330
	TELCNT = 0	1350
	PELCNT = PELCNT +1	1360
	OTHERWISE	1370
	PRINT COMMENT \$OPERMANENT ELEMENT CANNOT FOLLOW TEMPORARY ELE	
	1MENTS	
	END OF CONDITIONAL	1410
	TRANSFER TO NUSTAT	
	R	1430
	R ELEMENT NAME.	1440
S2(5)	ELN(PELCNT + TELCNT) = NUWORD	1450
	ELNAME = NUWORD	1480

	P'IT \$S110,C6,HA *****A*\$,ELNAME	1490
	TELCNT = TELCNT +1	1500
	R ELEMENT IS ASSUMED TO BE TEMPORARY	1510
	R	1520
	TRANSFER TO NUSTAT	1530
	R	1540
	R BROAD SCOPE PARAMETER.	1550
S2(6)	EXECUTE BSPARM.	
	TRANSFER TO NUSTAT	1580
	R	1600
	R	1590
S2(7)	ATTACHMENT NAME STATEMENT.	1630
	ATS(ATTCNT)= NUWORD	1640
	ATTCNT = ATTCNT +1	1650
	WHENEVER ATTCNT .G. 20	
	ATTCNT = 20	
	PRINT COMMENT \$OATTACHMENT NAME LIST EXCEEDS TWENTYS	
	END OF CONDITIONAL	1680
	TRANSFER TO NUSTAT	
	R STATEMENT COLLECTION N=0	1700
	R	1710
S2(0)	CONTINUE	1720
	TRANSFER TO NUEND(NUSCAN.(NUCHAR,NOATS))	
	R RESETS FOR ELEMENT DESCRIPTION RETURNS	4200
	R	4210
FINISH	CONTINUE	
	TRANSFER TO NUEND(ENDDES.(0,NOATS))	
NUEND(0)	CONTINUE	
NUEND(1)	CONTINUE	
	TRANSFER TO NXTSET	
	R(**RETURN TO DECLARATION SECTION**)	
NUEND(2)	CONTINUE	
	WHENEVER NUCAR .E. \$. \$	
	I = 72	
	OTHERWISE	
	I = -1	
	END OF CONDITIONAL	
	K1=0	
	R(**OTHERWISE REMAIN IN THE ELEMENT DESCRIPTION SECTION **)	
NUEND(3)	CONTINUE	
NUSTAT	DCNT = 0	4320
	HOLCNT = 1	4330
	NUWORD = \$ \$	4340
	TRANSFER TO LOOP	4350
	R	
	R ** COMPILOR CONTROL CARDS **	
	R	
	VECTOR VALUES DKIND1=\$ELEMENS,\$INPUTS,\$SYNONYS,\$CONNECS,\$DES	2080
	1IRES,\$FUNCTIS,\$NEWELES,\$CALCUL,\$NEXTSES,\$CHECKOS	2090
	2,\$CHECKRS,\$CONNECS,\$SYNONYS,\$CALCUL\$	2100
	3,\$SAVETAS,\$EQUIVAS	2110
	4,\$SUBSTIS,\$CONFIGS	
	VECTOR VALUES DKIND2=\$DESCRIS,\$RAMETES,\$SS,\$SIONSS,\$RESULTS	2120
	1,\$NSUBST,\$SENTTAPS,\$TIONSS,\$OFDATAS,\$TS	2130
	2,\$NS,\$SIONS,\$\$, \$,\$SIONS	2140
	3,\$ES,\$ENCES	2150
	4,\$UTIONSS,\$RATIONS	
	VIS ABR1=\$ELINS,\$INISS,\$SNISS,\$SCNISS,\$SDSISS,\$FNISS	
	1,\$NWIES,\$CLISS,\$NXIAS,\$CHITS,\$CHINS,\$SCNINS,\$SNIM	

```

25 , $CLINS , $SVIES , $SEQES , $SBISS , $CNIOS
R      CALCULATIONS ON TAPE THREE
R      PROLOGUE ON TAPE FOUR
R      FUNCTION DICTIONARY.
R      CONNec. INTPR. EQUIVn. SYNONM. SUBSPT. (IN BUFFER DECK) R      1
RCONCK. EQVGEN. INPUT. INSRTE. OCTDEC. R      2
R      OUTPUT. PLACE. PROLOG. SEQPGM. SYNELM. R      3
R      SAVET4. SYSTEM. SETEOF. SSORDR. IPRELM. R      4
R      R      5
R      R      6
R      TEMPORARY ROUTINES IN CORE1.
R      (CONCK. ) (SAVET4.)
R(MAIN) FOR FIRST CORE 0010
RINPUT FOR CONNECTIONS, SYNONYMS, INPUT- 000
RPARAMETERS, DESIRED RESULTS AND DECLARATIONS 0020
R
PROGRAM COMMON INDEX(99),A,CH
NORMAL MODE IS INTEGER 0070
DIMENSION B(12)
DIMENSION WORD(11),
1S(42,DIMS), DWORD(4), PNUM(1), AMB(7),CH(71)
DIMENSION BIT(1),DCNT(4),EQUER(9),PART1(7),PART2(7),PART(7)
1      ,S2(8),TYPE(7)
2      ,BITS(80,DIM3),MAD(50)
DIMENSION SBIT(1),SCOPB1(1),ATS(20),EQUER(9)
DIMENSION PELT(300),ELN(95),FPAR(71),ZPAR(69)
DIMENSION A(15899,DIM1),PCNT(1)
DIMENSION SBIT(1)
VECTOR VALUES THREE= 3
VECTOR VALUES FOUR = 4
VECTOR VALUES DIMS = 2,7,6
VECTOR VALUES DIM1 = 2,401,400 0030
VECTOR VALUES DIM3=2,5,4
VECTOR VALUES A(15600)=SELTAPES
EQUIVALENCE ( A(15601),PELCNT ),( A(15602),TELCNT ),
1      ( A(15603),PARCNT ),( A(15604),ELN ) ,
2      ( A(15700),FPAR ) ,( A(15772),ZPAR ) ,
3      ( FPAR(70),SCOPB1 ),( FPAR(71),SCOPB2 ),
4      ( A(15852),TAPEOK ),( A(15842),CCNT ) ,
5      ( A(15843),SCNT ) ,( A(15844),ECNT ) ,
6      ( A(15845),PCNT ) ,( A(15847),FSCNT ) ,
7      ( A(15848),ID1 ) ,( A(15849),ID2 ) ,
8      ( A(15850),IFERR1 ),( A(15851),PRLCNT )
9      ,( A(15853),DIM1 ),( A(15889),FIVE)
EQUIVALENCE ( A(15856),CKRN ) ,( A(15857),THREE )
1      ,( A(15858),FOUR ) ,( A(15859),CLDONE)
EQUIVALENCE(A(15870),B),(A(15860),NUWORD),(A(15861),SBIT)
EQUIVALENCE(A(8000),BITS),(A(8080),ATS),(A(8100),MAD)
1      ,(ELTPOK,TAPEOK),(ATTCNT,NOATS)
2      ,(T4,FOUR),(SCRAP,FIVE)
EQUIVALENCE (A(15891),ELNAME)
EQUIVALENCE (PELT,A(8200))
DIMENSION ELNAME(1)
EQUIVALENCE (A(15890),NUCHAR)
DIMENSION VCK(10)
EQUIVALENCE (A(15900),VCK)
EQUIVALENCE ( TIME,A(15999))
BOOLEAN ABRSW
BOOLEAN PEQL
BOOLEAN PPER

```

BOOLEAN PEID		
BOOLEAN CLDONE		
BOOLEAN CAT,CEL,CTO,CEID,PLP1,		0110
1PLP2, PEL, PAT, IFERR1, IFERR2, TAPEOK, OUT, FEL, FSN		0120
2 ,CLC , NTAPE, EOFB , SET4, CKRN		0130
BOOLEAN EPAR, EEL, EEID, EAT, E2NDPN		0140
BOOLEAN SPAR, SEL, SEID, SAT, S2NDPN, S1STEQ		0160
BOOLEAN FSNEW,FSOLD, FSEL		0190
BOOLEAN CPABLE,DONE,EQUAL,ELTPOK,PAREN,THEN		
BOOLEAN NUELTP	B	4
BOOLEAN ALLSET, PERIOD		
BOOLEAN WITH		
END OF PROGRAM		*

SCOMPILE MAD,PUNCH OBJECT,PRINT OBJECT

BSPARM00

EXTERNAL FUNCTION

R'N

ENTRY TO BSPARM.

I = LOCATE.(NUWORD, FPAR, PARCNT)

WHENEVER I.L. 0

R (PARAMETER IS NOT ON F-PAR LIST YET.)

I = PARCNT

J= I/36

SBIT(J)= INBIT.(SBIT(J),I-J*36,1)

EXECUTE PARCK.(I)

R(**PARCNT IS INCREMENTED IN PARCK. SUBROUTINE.)

R (SBIT IS TEMPORARY SCPE BIT FOR THE ELEM.)

OTHERWISE

J= I/36

I= I-J*36

WHENEVER .NOT. IFBIT.(SCOPB1(J),I)

PREV = NUWORD

P'IT FOMAT,PREV...PREV(7)

OTHERWISE

SBIT(J)= INBIT.(SBIT(J),I,1)

END OF CONDITIONAL

END OF CONDITIONAL

VECTOR VALUES PREV=\$ \$,\$ IS NARROW SCOPE BY PREVIOUS ASSERTIO

IN.\$

FUNCTION RETURN

R *****COMPILER CONTROL CARDS*****

PROGRAM COMMON A

V'S FOMAT=\$1H0,22C6*\$

DIMENSION A(15999),SBIT(1),SCOPB1(1), FPAR(71),ZPAR(69)

D

1

NORMAL MODE IS INTEGER

EQUIVALENCE(A(15700),FPAR) ,(FPAR(70),SCOPB1)

E

1

1 ,(A(15603),PARCNT),(A(15861),SBIT)

2 ,(A(15860),NUWORD),(A(15772),ZPAR)

BOOLEAN IFABIT.

BOOLEAN IFBIT.

END OF FUNCTION

*

SCOMPILE MAD,PUNCH OBJECT,PRINT OBJECT
EXTERNAL FUNCTION

CONCK000

R'N

	R CHECK FOR AMBIGUOUS OR REDUNDENT CONNECTIONS *****	4920
	R EXECUTE BEFORE CORE CHANGE AND AFTER EOF1	4930
	ENTRY TO CONCK.	4940
	THROUGH CK, FOR VALUES OF CKA=0,4	4950
	THROUGH CK, FOR CKB=0,1,CKB.GE. CCNT	
	WHENEVER A(33+CKA,CKB).E.\$ \$,A(33+CKA,CKB)=\$1\$	4970
	THROUGH CK, FOR CKC=0,4,CKC.G. 4.OR. CKA .GE. 4	
	THROUGH CK, FOR CKD= CKB+1,1, CKD.GE. CCNT	
	THROUGH CK3, FOR CKI=0,1, CKI.GE. 4	
	WHENEVER A(31+CKA+CKI,CKB).NE.A(31+CKA+CKI+CKC,CKD),TRANSFER	5010
	1 TO CK	5020
	WHENEVER CKI.E.3, EXECUTE CKPRNT.	5021
CK3	CONTINUE	5030
	THROUGH CK4, FOR CKI=0,1, CKI.GE. 4	
	WHENEVER A(35+CKI-CKA,CKB).NE. A(35+CKI-CKA-CKC,CKD)	5050
	P'T FORM,CKB,CKD	
	V'S FORM=\$H*QINCONSISTENT CONNECTION LINES- *,2I6*\$	
	IFERR1=1B	5100
	TRANSFER TO EQIN	5110
	END OF CONDITIONAL	5120
CK4	CONTINUE	5130
EQIN	CCNT=CCNT-1	5140
	P'T FORM1,CCNT	
	V'S FORM1=\$H*0CCNT LOWERED TO *,I6*\$	
	THROUGH CK5, FOR CKI=31,1, CKI.GE. 39	
CK5	A(CKI,CKD)=A(CKI,CCNT)	5160
CK	CONTINUE	5170
	FUNCTION RETURN	5180
	INTERNAL FUNCTION	
	ENTRY TO CKPRNT.	
	P'T FOMAT,CK1A...CK1A(4)	
	EXECUTE DPRINT.(\$COUNTS\$,CKA,4)	
	FUNCTION RETURN	
	END OF FUNCTION	
	VECTOR VALUES CK1A=\$ \$,\$CKA\$, \$CKB\$, \$CKC\$, \$CKD\$	
	EQUIVALENCE (V(0),CKA),(V(1),CKB),(V(2),CKC),(V(3),CKD)	
	DIMENSION V(4)	
	R ***COMPILER CONTROL CARDS *****	
	R DEFINITION OF A(39,0)...A(39,399)	
	VECTOR VALUES A(15600)=\$ELTAPES\$	
	EQUIVALENCE (A(15601),PELCNT),(A(15602),TELCNT),	
1	(A(15603),PARCNT),(A(15604),ELN),	
2	(A(15700),FPAR),(A(15772),ZPAR),	
3	(FPAR(70),SCOPB1),(FPAR(71),SCOPB2),	
4	(A(15852),TAPEOK),(A(15842),CCNT),	
5	(A(15843),SCNT),(A(15844),ECNT),	
6	(A(15845),PCNT),(A(15847),FSCNT),	
7	(A(15848),ID1),(A(15849),ID2),	
8	(A(15850),IFERR1),(A(15851),PRLCNT)	
9	,(A(15853),DIM1)	
	EQUIVALENCE (PELT,A(8200))	
	VECTOR VALUES DIM1 = 2,401,400	-030
	NORMAL MODE IS INTEGER	0070
	DIMENSION A(15999,DIM1),PCNT(1)	
	BOOLEAN IFERR1, TAPEOK	
	PROGRAM COMMON A	

V'S FOMAT=\$1H0,22C6*\$
DIMENSION PELT(300),ELN(95),FPAR(71),ZPAR(69)
R
END OF FUNCTION

*

\$ASSEMBLE,PUNCH OBJECT

DP-PACK0

	ENTRY	DFMP,DFDP,DPFSB,DPFAD,DPFDP,DPFMP
DFMP	CALL	ERROR
DFDP	EQU	DFMP
DPFSB	EQU	DFMP
DPFAD	EQU	DFMP
DPFDP	EQU	DFMP
DPFMP	EQU	DFMP
	END	

*

SASSEMBLE,PUNCH OBJECT

IFINBITO

```

ENTRY INBIT
ENTRY IFBIT
INBIT  SLN      1      RSLT = INBIT.(WORD,J,BIT)
        CLA 3,4      THE VALUE OF BIT  PLACED IN THE JTH POSITION OF WORD.
        STA BIT
        SLT 1
IFBIT  SLN 1
        SXD XRA,1
        CLA 2,4
        STA *+1
        LXA **,1
        CLA 1,4
        STA WORD1
        STA WORD2
        CAL MASK1
        SLT 1
XRA    TXL BIT,0,0
        ALS 35,1
WORD1  ANA **
        TZE EXIT+1
        CLA MASK1
        TXL EXIT+1,0,0
BIT    ANA **
        SLW COMMON
WORD2  CAL **
        LRS 35,1
        ANA MASK2
        ORA COMMON
        LLS 35,1
        SLW COMMON
EXIT   CLA COMMON
        LXD XRA,1
        TRA 3,4
MASK1  OCT C00000000001
MASK2  OCT -377777777776
COMMON SYN -1
LAST   BSS 0
        END

```

*

SASSEMBLE,PUNCH OBJECT

INPUT000

	ENTRY INPUT	
INPUT	CLA 1,4	EXECUTE INPUT.(CH...EOFLBL,WORD)
	STD EOFA	EOFLBL
	ADD INPUT	ADD ONE
	STA CHA	CH
	CLA 2,4	
	STA WORDA	
	SXD XRC,4	
	CALL SCARDS	READ ONE CARD
EOFA	TIX B,0,**	
	CALL SPRINT	PRINT ONE CARD
	TIX B-1,0,13	
	AXT 0,2	
	AXT 72,1	
NEXT	AXT 6,4	
	CAL B+11,2	NEXT WORD FROM B REGION
SPLIT	LDQ BLNK	
	LGR 6	
CHA	STQ **,1	CH+1 WILL BE IN ADDRESS.
	TXI *+1,1,-1	
	TIX SPLIT,4,1	CONUNTS LETTERS.
	TXL OUT,1,1	
	TXI NEXT,2,1	
OUT	AXT 0,1	PUTCARD INTO WORD REGION
	AXT 12,2	
WM	CLA B+12,2	
WGRDA	STO **,1	
	TXI *+1,1,1	
	TIX WM,2,1	
	LXD XRC,4	RETURN
	TRA 3,4	
XRC	PZE 0,,**	
BLNK	BCD 1	
B	BSS 14	
	END	

*

\$ASSEMBLE,PUNCH OBJECT

INSRTC00

```

ENTRY INSRTC
ENTRY EXTRC
INSRTC SLN 1      RSLT = INSRTC.(WORD, J, CHAR)
          CLA      3,4    CHAR IS PLACED IN THE JTH POSITION OF WORD.
          STA CHAR
          SLT 1
EXTRC  SLN 1      RSLT = EXTRC.(WORD, J)
          SXD XRB,2    RSLT IS THE CHARACTER EXTRACTED FROM THE JTH
          CLA*      2,4
          PAX      0,2
          CAL*      1,4
          SLT 1
XRB     TXL IN1,0,0
          LDQ BLANKS
BACK1   TXH CJT1,2,5
          ARS 6
          TXI BACK1,2,1
OUT1    LGL 30
          SLW COMMON
          TXL EXIT,0,0
IN1     SLW COMMON
CHAR    LDQ **
          CAL MASK
          RQL 6
COUNT  TXH OUT2,2,5
          ALS 6
          TXI COUNT-1,2,1
OUT2    STQ COMMON-1
          COM
          ANS COMMON
          COM
          ANA COMMON-1
          ORS COMMON
EXIT    CLA COMMON
          LXD XRB,2
          TRA 3,4
BLANKS  BCD 1
MASK    OCT 000000000077
COMMON  SYN -1
LAST    BSS 0
          END

```

*

EXTERNAL FUNCTION

R'N

ENTRY TO IPRELM.

R SET UP BIT PATTERNS FOR DESIRED RESULTS AND INPUT PARAMETERS

R

R BRIEF DESCRIPTION OF IPRELM (INPUT PARAMETER-DESIRED
R RESULT ELEMINATION). THE LIST OF PARAMETERS IS SEARCHED

R FOR THE PARAMETER ON THE INPUT PARAMETER LIST. WHEN THE

R PARAMETER IS KNOWN, THE CONNECTION MATRIX IS THEN

R SEARCHED UNTIL A ROW IN THE CONNECTION MATRIX MATCHES

R THE ROW IN THE INPUT PARAMETER LIST. A BIT IS PLACED IN

R/A/, WHICH CODES THE ROW OF THE CONNECTION MATIIX WITH

R THE LOCATION OF THE PARAMETER ON THE PARAMETER LIST

R THE SAME PROCESS IS REPEATED FOR THE DESIRED RESULT LIST

PROGRAM COMMON A

V'S FOMAT=\$1H0,22C6*\$

DIMENSION A(15999,DIM1), ERR(8)

VECTOR VALUES DIM1 = 2,401,400

DIMENSION PCNT (1)

R

BOOLEAN OCCURS

R DEFINITION OF A(39,0)...A(39,399)

VECTOR VALUES A(15600)=\$ELTAPES

EQUIVALENCE (A(15601),PELCNT),(A(15602),TELCNT),

1 (A(15603),PARCNT),(A(15604),ELN),

2 (A(15700),FPAR),(A(15772),ZPAR),

3 (FPAR(70),SCOPB1),(FPAR(71),SCOPB2),

4 (A(15852),TAPEOK),(A(15842),CCNT),

5 (A(15843),SCNT),(A(15844),ECNT),

6 (A(15845),PCNT),(A(15847),FSCNT),

7 (A(15848),ID1),(A(15849),ID2),

8 (A(15850),IFERR1),(A(15851),PRLCNT)

DIMENSION PELT(300),ELN(95),FPAR(71),ZPAR(69)

NORMAL MODE IS INTEGER

BOOLEAN IFERR1

EQUIVALENCE (A(12400),A31),(A(12800),A32),(A(13200),A33),

1(A(13600),A34),(A(14000),A35),(A(14400),A36),(A(14800),A37),

2(A(15200),A38) ,(A(15600),A39)

DIMENSION A31(1),A32(1),A33(1),A34(1),A35(1),A36(1),A37(1),

1A38(1) ,A39(1)

DIMENSION FPAR(70)

DIMENSION CHECK(2)

EXECUTE PLACE.(A(23,0), 3200,0)

R COLUMN 4 - INPUT PARAMETERS

COL = 4

THROUGH LISTS, FOR VALUES OF K=23,25

WHENEVER K.E.25,COL=9

THROUGH LISTS, FOR I=0,1,I.GE. PCNT(K/25)

LOOPA THROUGH LOOPA, FOR J=0,1, J.GE. PARCNT .OR. A(COL,I) .E.

1FPAR(J)

WHENEVER J.E.PARCNT

VECTOR VALUES ERR = \$UNKNOWN PARAMETER ATS

ERR(4) = A(COL,I)

ERR(5) = A(COL+1,I)

ERR(6) = A(COL+2,I)

ERR(7) = A(COL+3,I)

ERR(8) = A(COL+4,I)

P'IT FOMAT,ERR...ERR(8)

IFERR1 = 1B	0570
TRANSFER TO LISTS	0580
END OF CONDITIONAL	0590
TEL = A(COL+1,I)	0600
TEID = A(COL+2,I)	0610
TAT = A(COL+3,I)	0620
TAID = A(COL+4,I)	0630
OCCURS = 0B	631
THROUGH LISTS1, FOR L=0,1,L.GE. CCNT	
WHENEVER (TAID.E. A34(L) .OR. TAID .E. \$ \$)	0690
1.AND. (TAT .E. A33(L) .OR. TAT .E. \$ \$)	0700
2.AND. (TEID.E. A32(L) .OR. TEID .E. \$ \$)	0710
3.AND. (TEL .E. A31(L) .OR. TEL .E. \$ \$)	0720
R	740
JMOD = J/36	0750
A(K+JMOD,L) = INBIT.(A(K+JMOD,L),J-JMOD*36, 1)	0760
OCCURS = 1B	0770
END OF CONDITIONAL	0790
WHENEVER (TAID.E. A38(L) .OR. TAID .E. \$ \$)	0800
1 .AND. (TAT .E. A37(L) .OR. TAT .E. \$ \$)	0810
2 .AND. (TEID.E. A36(L) .OR. TEID .E. \$ \$)	0820
3.AND. (TEL .E. A35(L) .OR. TEL .E. \$ \$)	0830
JMOD = J/36	0840
A(K+JMOD,L) = INBIT.(A(K+JMOD,L),J-JMOD*36, 1)	0850
OCCURS = 1B	0880
END OF CONDITIONAL	0890
LISTS1 CONTINUE	
W'R .NOT. OCCURS, P:T FOMAT,NO...NO(9)	
LISTS CONTINUE	900
VECTOR VALUES NO(5) = \$DOES NOT OCCUR ON CONNECTION LISTS	0920
EQUIVALENCE (NO,TEL),(NO(1),TEID),(NO(2),TAT),(NO(3),TAID)	0930
EQUIVALENCE (PELT,A(8200))	
FUNCTION RETURN	0940
END OF FUNCTION	*

\$ASSEMBLE,PUNCH OBJECT

LOCATE00

ENTRY LOCATE

LOCATE.(WORD, LIST, COUNT)

LIST ADDRESS PLUS ONE

WORD

WORD FOUND AT THIS LOCATION PLUS 1
IF WORD NOT ON LIST ACCUMULAROR WILL BE NEG

LOCATE CLA* 3,4
ADD ONE
PAX 0,2
CLA 2,4
STA COMP2
ADD ONE
STA COMP
CLA* 1,4
COMP2 CAS **
TRA COMP
TRA FIRST
COMP CAS **,2
TIX *-1,2,1
TRA EQUAL
TIX *-3,2,1
EQUAL PXA 0,2
SUB ONE
TNZ 4,4
SUB ONE
TRA 4,4
FIRST CLA ZERO
TRA 4,4
ZERO PZE 0
TRA 4,4
ONE PZE 1
END

*

\$ASSEMBLE,PUNCH OBJECT

LOGIC.00

```

ENTRY AND
ENTRY NOT
ENTRY OR
ENTRY XOR
ENTRY THEN
ENTRY EQV
HOOK    CAL*    2,4
        SLW      C-2
HOOK1   CAL*    1,4
        SLW C-1
        TRA 1,2
EQV     TSX HOOK,2
        ORA C-2      W1.OR.W2
        COM          .NOT. (W1.OR.W2)
        SLW C
        CAL C-1
        ANA C-2      W1.AND.W2
        ORS C        (W1.AND.W2).OR.(.NOT.(W1.OR.W2))
        TRA R2
THEN    TSX HOOK,2
        COM          .NOT.W1
        ORA C-2      .NOT.W1.OR.W2
        TRA R1
XOR     TSX HOOK,2
        ANA C-2      W1.AND.W2
        COM          .NOT.(W1.AND.W2)
        SLW C
        CAL C-1
        ORA C-2      (W1.OR.W2)
        ANS C        (W1.OR.W2).AND.(.NOT.(W1.AND.W2))
        TRA R2
NOT     TSX HOOK1,2
        COM          NOT.W1
        TRA R1
OR      TSX HOOK,2
        ORA C-2      W1.OR.W2
        TRA R1
AND     TSX HOOK,2
        ANA C-2      W1 AND W2
R1      SLW C
R2      CLA C
        TRA 2,4
LAST    BSS 0
C       SYN -1
        END

```

*

SCOMPILE MAD,PUNCH OBJECT,PRINT OBJECT
EXTERNAL FUNCTION(HEAD, LIST, COUNT)

LPRINT00

R'N

ENTRY TO DPRINT.

K=12

P'IT FOMAT,HEAD

THROUGH B, FOR I=0,12, I.GE. COUNT

WHENEVCOUNT-I.LE. 12, K= COUNT -I

THROUGH C, FOR J=I,1, J.GE. K

LINE(J) = LIST(J)

P'IT FOMAT,LINE(I)....LINE(I+K-1)

FUNCTION RETURN

NORMAL MODE IS INTEGER

DIMENSION LINE(12)

PROGRAM COMMON A

V'S FOMAT=\$1H0,22C6*\$

DIMENSION A(15999)

EQUIVALENCE (A(15858), FOUR),(A(15601), PELCNT),

1 (A(15602),TELCNT),(A(15603),PARCNT)

EQUIVALENCE(A(15700),FPAR),(A(15772),ZPAR)

1,(A(15604),ELN)

END OF FUNCTION

*

\$ COMPILE MAD,PUNCH OBJECT,PRINT OBJECT
EXTERNAL FUNCTION(ARG1,ARG2)

NUSCAN00

R'N
ENTRY TO NUSCAN.
NUCHAR=ARG1
NOATS = ARG2
IFERR2 = 0B
CPABLE = 0B
MADCNT = 0
SCCNT = 0
PERIOD = 0B
PAREN = 0B
ALLSET = 0B
THEN = 0B
SAT = 0B
WITH = 0B
EOFB = 0B
DWORD = \$ \$
NOATT = \$ \$
BIT(0) = 0
BIT(1) = 0
J = 0
TRAP = 0B
BOOLEAN TRAP
TRAP1= 1B
BOOLEAN TRAP1
ABRSW=0B
BOOLEAN ABRSW
W'R CKRN
PRINT COMMENT \$ONUSCANS
E'L

R SCAN USED IN STATEMENT COLLECTIONS.

1740

R

1750

EXECUTE PLACE.(BITS,80,0)
WHENEVER NUCCHAR.NE. \$.\$.TRANSFER TO CARDIN
R(** IF ABOVE IS TRUE8 THE NEXT CARD IS ALREADY IN CORE.

READ2

EXECUTE INPUT.(CH...END1, WORD)

1760

W'R CH .E. \$1\$

CPABLE=0B

TRAP1=1B

E'L

CARDIN

CONTINUE

HOLCNT = 1

NUWORD = \$ \$

DCNT = 0

WHENEVER CH .E. \$/\$. TRANSFER TO READ2

1770

THROUGH STMT, FOR SI = 0,1, SI .G. 71

1780

SCPCN1

CONTINUE

NUCHAR = CH(SI)

1790

W'R SI.E.0 .AND .NUCHAR .E.\$1\$

NUCHAR = \$ \$

CPABLE = 0B

TRAP1= 1B

E'L

WHENEVER NUCCHAR .E. \$ \$, TRANSFER TO STMT

1800

SCPCNT

T'H SCPCNT, FOR SKK = 0,1,NUCHAR.E.PUNCT(SKK).OR.SKK.G.4

R(**PUNCT LIST IS , . () =

W'R SKK.E.1, TRAP1 = 0B

W'R SKK .LE.3

	W'R SKK.E.3, TRAP1 = 1B	
	WHENEVER .NOT. CPABLE, TRANSFER TO PART(SKK)	
	TRANSFER TO TYPES	
R		1940
R		1950
	O'R SKK.E.4	
	TRAP1 = 0B	
	TRANSFER TO MADOUT	1980
R		1990
	CTHERWISE	2000
	NUWORD = INSRTC.(NUWORD,HOLCNT,NUCHAR)	2010
	HOLCNT = HOLCNT+1	2020
	W'R NUCHAR .E. \$' \$	
	ABRSW=1B	
	O'R ABRW	
	ABRSW=0B	
	HOLCNT=7	
	E' L	
	WHENEVER HOLCNT.G. 6	2030
	W'R CKRN	
	W'R TRAP1, PRINT COMMENT \$OTRAP17\$	
	E' L	
	TRAP = 1B	
	DWORD(DCNT) = NUWORD	2040
	DCNT = DCNT +1	2050
	SI=SI+1	2070
	NUWORD = \$ \$	2080
	HOLCNT = 1	2090
	WHENEVER .NOT. CPABLE, TRANSFER TO TYPEC	
	TRANSFER TO TYPES	
	END OF CONDITIONAL	2100
	END OF CONDITIONAL	2110
	TRANSFER TO STMNT	2120
R		1870
R		1880
TYPES	THROUGH TYPES, FOR SK=0,1,(STYPE1(SK).E.DWORD(0))	
2	.OR. SK .G. 2	1910
	W'R SK .G. 2	
ABRF1	T'H ABRF1, FOR SK=0,1, ABR(SK) .E. DWORD .OR. SK .G. 2	
	E' L	
	W'R SK.E.0	
	TRAP1 = 1B	
	W'R CKRN	
	PRINT COMMENT \$OTRAPS\$	
	E' L	
	OTHERWISE	
	TRAP1 = 0B	
	E' L	
	WHENEVER SK.G. 2, TRANSFER TO MADOUT	
	W'R SK.E.1	
	NUCHAR = \$.\$	
	TRAP1 = 0B	
	E' L	
	TRANSFER TO FINISH	
R		2130
R		2140
R	SK = 0 1 2	2150
	VECTOR VALUES STYPE1=\$STATEM\$, \$DESCRIS\$, \$ANOTHES\$, \$ \$,	
	1\$ESTIMAS\$, \$COMPUT\$, \$WITHOUS	2180

	1,\$ELEMEN\$		
	R 4 5 6		2170
	VECTOR VALUES STYPE2=\$NTCOLLS,\$TIONFIS,\$DESCRIS		2190
	R		2200
	R CAPABILITY STATEMENT, COMMA ENCOUNTERED		2240
	VIS ABR=\$STIN\$, \$SDSID\$, \$ANIS\$, \$ \$, \$ESIES , \$CMIES ,		
	ISWTIT\$, \$ELIN\$		
PART(0)	CONTINUE		2250
	WIR TRAP.AND.TRAP1		
	TRAP = 0B		
	NUWORD = DWORD(0)		
	DCNT = 0		
	DWORD(0) = \$ \$		
	WIR SKK.E.2, TIO PARNAM		
	TIO TRAPED		
	EIL		
	WHENEVER NUWORD . E. \$ \$		2270
	TRANSFER TO RESET2		
	OR WHENEVER NUWORD .E. \$THENS		2280
	THEN = 1B		2290
	TRANSFER TO MUSTA2		
	R CHECK ESTIMATE, COMPUTE, AND WITHOUT 4 5 6		2300
	OTHERWISE		
TYPEC	TIO TYPEC, FOR JK=4,1, STYPE1(JK) .E. DWORD .OR. JK .G. 6		
	WIR JK .LE. 6		
	SI=SI+1		
	WIR JK .E. 4, SI=SI+1		
SCAN	TIO SCAN, FOR SKK=0,1, SKK .G. 4 .OR. CH(SI) .E. PUNCT(SKK)		
	WIR SKK .G. 4, SI=SI-1		
	TRAP1=0B		
	DCNT=0		
	THEN=1B		
	DWORD=0		
	TIO PART(JK)		
	EIL		
ABRF2	TIO ABRF2, FOR JK=4,1, ABR(JK) .E. DWORD .OR. JK .G. 6		
	WIR JK .LE. 6, TIO SCAN		
	WIR CPABLE, TIO TYPES		
	WIR NUCHAR .NE. \$,\$, TIO SCPCN1		
	END OF CONDITIONAL		
	WIR TRAP.AND. TRAP1		
	NUWORD = DWORD(0)		
	DCNT = 0		
	TRAP = 0B		
	EIL		
	WHENEVER .NOT. PAREN		
	R A PARAMETER NAME IN CAPABILITY STATEMENT		2420
PARNAM	CONTINUE		
	SAT= 0B		
	WHENEVER NUWORD.E.\$ \$, TRANSFER TO RESET2		
TRAPED	CONTINUE		
	WIR CKRN		
	WIR TRAP1, PRINT COMMENT \$OTRAPED\$		
	EIL		
	J=PARCK.(J)		2430
	JMOD = J/36		2460
	BIT(JMOD) = INBIT.(BIT(JMOD), J-36*JMOD, 1)		2470
	R		2480
	OTHERWISE		2490

ATTNAM	R(** AN ATTACHMENT NAME**)	
	CONTINUE	
	W'R NUWORD .E. \$ \$.AND..NOT.TRAP,T'O RESET2	
	W'R TRAP .AND. DWORD(0) .E. \$ \$	
	TRAP = 0B	
	T'O RESET2	
	E'L	
	WHENEVER SAT	
R	AN ATTACHMENT NAME FOLLOWED BY AN IDENTIFIER	2500
	BITS(IAT,1)=INBIT.(BITS(IAT,1),35,1)	
	SAT=0B	
	OTHERWISE	
	SAT=1B	
	W'R TRAP	
	I = LOCATE.(DWORD(0),ATS,NOATS)	
	TRAP = 0B	
	OTHERWISE	
	I = LOCATE. (NUWORD, ATS, NOATS)	2510
	E'L	
	WHENEVER I .GE. 0	2550
	IAT=I	
	WHENEVER .NOT. THEN	2560
	BITS(I,0)=OR.(BIT(0),BITS(I,0))	2570
	BITS(I,1)=OR.(BIT(1),BITS(I,1))	2580
	OTHERWISE	2590
	BITS(I,2)=OR.(BIT(0),BITS(I,2))	2600
	BITS(I,3)=OR.(BIT(1),BITS(I,3))	2610
	END OF CONDITIONAL	2620
	BIT(0) = 0	2630
	BIT(1) = 0	2640
	OTHERWISE	2650
ATTERR	CONTINUE	
	NOATT = NUWORD	2660
	P'IT FOMAT,NOATT...NOATT(4)	
	R(**SKIP TO NEXT ELEMENT DESCRIPTION.)	
	SK=2	
	EXECUTE SKPDES.	
	TRANSFER TO FINISH	
	END OF CONDITIONAL	2680
	END OF CONDITIONAL	
	END OF CONDITIONAL	2690
	VECTOR VALUES NOATT= \$ \$,\$ NOT ON ATTACHMENT LISTS	2700
	TRANSFER TO RESET2	2710
	R	2720
	R	CAPABILITY STATEMENT DECOMPOSITION
	R	2730
	R	2740
	R	PERIOD,CAPABILITY FINISHED.
		2750
PART(1)	CONTINUE	
	TRAP = 0B	
	CPABLE = 1B	2760
	PERIOD = 1B	
	DWORD=\$ \$	
	T'O READ2	

PART(2)	R	LEFT PAREN	2810
		CONTINUE	
		W'R TRAP.AND.TRAP1	
		PAREN = 1B	
		T'O PART(0)	
		OTHERWISE	
		TRAP = 0B	
		PAREN = 1B	
		WHENEVER WITH, TRANSFER TO RESET2	
		TRANSFER TO PARNAM	2840
		E'L	
PART(3)	R	RIGHT PAREN	2850
		CONTINUE	2860
		WHENEVER .NOT. PAREN	2880
		PRINT COMMENT \$OERRORS	
		TRANSFER TO NUSTAZ	
		OTHERWISE	2910
		PAREN = 0B	
		WHENEVER WITH, TRANSFER TO WITHOT	
		TRANSFER TO ATTNAM	2920
		END OF CONDITIONAL	2930
	R	EQUAL SIGN NOT OBSERVED HERE.	2940
	R	COMPUTE	2950
PART(5)		CONTINUE	
		TRAP = 0B	
		BIT(1)=INBIT.(BIT(1),34,1)	
		THEN=1B	
		TRANSFER TO RESET2	
PART(4)	R	ESTIMATE	2980
		CONTINUE	
		TRAP = 0B	
		THEN = 1B	
		W'R CH(SI) .E.\$.\$	
		CPABLE = 1B	
		PERIOD = 1B	
		E'L	
		TRANSFER TO RESET2	3010
PART(6)	R	WITHOUT	3020
		CONTINUE	
		WITH = 1B	
		W'R CH(SI) .E. \$(\$, SI=SI-1	
		TRANSFER TO RESET2	
WITHOT		CONTINUE	
		WITH=0B	
		W'R TRAP	
		J = LOCATE.(DWORD(0),ATS,NOATS)	
		TRAP = 0B	
		OTHERWISE	
		J = LOCATE.(NUWORD,ATS,NOATS)	
		E'L	
		WHENEVER J.GE. 0	
		BITS(J,3)=INBIT.(BITS(J,3),35,1)	
		NUWORD = \$ \$	3060
		DCNT = 0	3070
		OTHERWISE	
		TRANSFER TO ATTERR	
		END OF CONDITIONAL	
		TRANSFER TO RESET2	3080
	R		3090

R

3100

ENTRY TO ENDDDES.

FINISH

```

NOATS = ARG2
W'R CKRN
PRINT COMMENT $OENDDDES$
E'L
CONTINUE
W'R CKRN
PRINT COMMENT $OFINISH$
E'L
THEN = 0B
WHENEVER NOATS.LE. 1, NOATS=1
K = (NOATS ) *4-1
WHENEVER .NOT. ALLSET
ALLSET = 1B
WRITE BINARY TAPE T4, ELNAME, NOATS
WRITE BINARY TAPE T4,ATS(0)...ATS(NOATS-1)
SCCNT=0
SCOPB1 = SBIT(0)
SCOPB2 = SBIT(1)
P'T FOMAT,ELNAME
P'T BITP,$SCOPBT$,SCOPB1,SCOPB2
END OF CONDITIONAL
WRITE BINARY TAPE T4, BITS(0)...BITS(K)
T'H BITPR, FOR I=0,1,I.G.NOATS -1
PRINT FORMAT BITP, ATS  (I),BITS(I,0)...BITS(I,3)
V'S BITP = $1H C6,4(S2,K12)*$
CONTINUE

```

BITPR

```

R
SCCNT = SCCNT +1
MAD(SCCNT ) = MADCNT
MADCNT = 0
EXECUTE PLACE.(BITS,K+1,0)

```

R NEXT STATEMENT COLLECTION SETUP

TYPE(0)

```

TRANSFER TO TYPE(SK)
BIT(0) = 0
BIT(1) = 0
CPABLE = 0B
PERIOD = 0B
DWORD(1) = $ $
DWORD(0) = $ $
TRAP = 0B
TRANSFER TO READ2

```

R DESCRIPTION FINISHED OR ANOTHER FOLLOWS.

TYPE(1)
TYPE(2)

```

CONTINUE
CONTINUE
END OF FILE TAPE SCRAP
REWIND TAPE SCRAP
PRINT COMMENT $ORCNO COLNOS$
ALLSET = 0B
END OF FILE TAPE T4
NELNAM = NOT.(ELNAME)
WRITE BINARY TAPE T4,NELNAM
EXECUTE SETEOF.(ERRT3)
THROUGH CARDMA, FOR I = 1,1, I .G. SCCNT
WRITE BINARY TAPE T4, MAD(I), I
P'T $216*$,MAD(I),I
THROUGH CARDMA, FOR J=1,1, J.G. MAD(I)
READ BINARY TAPE SGRAP, WORD(0)...WORD(11)

```

	W'R CKRN	
	P'T FOMAT,WORD...WORD(11)	
	E'L	
CARDMA	WRITE BINARY TAPE T4, WORD(0) ... WORD (11)	3410
FILE	END OF FILE TAPE T4	
	REWIND TAPE SCRAP	3430
	PRINT COMMENT \$6\$	
	NOATS=0	
	SCCNT = 0	3440
	I=SI	
	WHENEVER EOFB, FUNCTION RETURN 1	
	FUNCTION RETURN 2	
	DIMENSION DWORD(4)	
	R RESETS FOR ELEMENT DESCRIPTION RETURNS	4200
	R	4210
NUCAR2	SI=71	
NUSTA2	DCNT=0	
RESET2	HOLCNT = 1	4240
	NUWORD = \$ \$	4250
STMNT	CONTINUE	3890
	W'R CPABLE, T'O MADOUT	
	WHENEVER .NOT. PERIOD.AND..NOT.CPABLE, TRANSFER TO PART(1)	
	TRANSFER TO READ2	3900
ERRT3	CONTINUE	
	PRINT COMMENT \$OEND OF FILE ON SCRATCH TAPE - IMPROPERLY FOUN	
	1D IN FINISHING ELEMENT DESCRIPTION.\$	
	IFERR1= 1B	3457
	EXECUTE SETEOF.(0)	
	TRANSFER TO FILE	
MADOUT	WRITE BINARY TAPE SCRAP, WORD(0) ... WORD (11)	3460
	W'R MADCNT .E. 0	
	P'T \$T106,H*STATEMENT COLLECTION *13*\$,SCCNT+1	
	E'L	
	TRAP1 = 0B	
	MADCNT = MADCNT+1	3480
	TRANSFER TO READ2	3490
	R	2210
	R	2220
	INTERNAL FUNCTION	
	EQUIVALENCE(EOF2,EOFL2)	
	ENTRY TO SKPDES.	
	W'R CKRN	
	PRINT COMMENT \$OSKPDESS	
	E'L	
	I=0	
READ3	EXECUTE INPUT.(CH...EOF2,WORD)	
	I= I+1	
	WHENEVER CH.E.\$/\$,TRANSFER TO READ3	
	THROUGH ST1, FOR SI=0,1,SI.GE. 72	
	NUCHAR= CH(SI)	
	WHENEVER NUCCHAR.E. \$ \$,TRANSFER TO ST1	
PUNCT3	THROUGH PUNCT3, FOR K=0,1,K.G. 4.OR.PUNCT(K).E. NUCCHAR	
	W'R NUCCHAR .E. \$'\$	
	ABRSW=1B	
	C'R ABRWS	
	ABRSW=0B	
	HOLCNT=6	
	E'L	
	WHENEVER K.LE.4.OR.HOLCNT.G.6	

```

THROUGH TYPECK, FOR VALUES OF K=1,2,7
    WHENEVER STYPE1(K).E. NUWORD, TRANSFER TO RETRN
W'R ABR(K) .E. NUWORD, T'O RETRN
TYPECK    CONTINUE
          TRANSFER TO RESET3
END OF CONDITIONAL
    NUWORD= INSRTC.(NUWORD, HOLCNT,NUCHAR)
    HOLCNT = HOLCNT+1
ST1       CONTINUE
RESET3    NUWORD= $ $
          HOLCNT = 1
          TRANSFER TO READ3
EOFL2     EOFB= 1B
RETRN     P'T SKIPPD,I
          V'S SKIPPD=$I5,H* CARDS SKIPPED IN THIS ELEMENT DESCRIPTION**
1$
SI=0
IFERR2=1B
FUNCTION RETURN
END OF FUNCTION
R
PROGRAM COMMON A,CH
V'S FOMAT=$1H 21C6*$
NORMAL MODE IS INTEGER
DIMENSION B(12)
DIMENSION END1(12)
    DIMENSION WORD(11),
1S(42,DIMS), DWORD(4), PNUM(1), AMB(7),CH(71)
    DIMENSION BIT(1),EQER(9),PART1(7),PART2(7),PART(7)
1          ,S2(8),TYPE(7)
2          ,BITS(80,DIM3),MAD(50)
    DIMENSION SBIT(1),SCOPB1(1),ATS(20),EQUER(9)
    DIMENSION PELT(300),ELN(95),FPAR(71),ZPAR(69)
    DIMENSION A(15999,DIM1),PCNT(1)
    VECTOR VALUES THREE= 3
    VECTOR VALUES FOUR = 4
    VECTOR VALUES DIMS = 2,7,6
    VECTOR VALUES DIM1 = 2,401,400
    VECTOR VALUES DIM3=2,5,4
    VECTOR VALUES A(15600)=$ELTAPES
    EQUIVALENCE ( A(15601),PELCNT ),( A(15602),TELCNT ),
1          ( A(15603),PARCNT ),( A(15604),ELN ),
2          ( A(15700),FPAR ),( A(15772),ZPAR ),
3          ( FPAR(70),SCOPB1 ),( FPAR(71),SCOPB2 ),
4          ( A(15852),TAPEOK ),( A(15842),CCNT ),
5          ( A(15843),SCNT ),( A(15844),ECNT ),
6          ( A(15845),PCNT ),( A(15847),FSCNT ),
7          ( A(15848),ID1 ),( A(15849),ID2 ),
8          ( A(15850),IFERR1 ),( A(15851),PRLCNT )
9          ,( A(15853),DIM1),(A(15889),FIVE)
    EQUIVALENCE (PELT,A(8200))
    EQUIVALENCE (A(15890),NUCHAR)
    EQUIVALENCE (A(15891),ELNAME)
    DIMENSION ELNAME(1)
    DIMENSION VCK(10)
    EQUIVALENCE (A(15900),VCK)
    EQUIVALENCE ( A(15856),CKRN ),( A(15857),THREE )
1,          (A(15858),FOUR ),(A(15859),CLDONE)
    EQUIVALENCE(A(15870),B),(A(15860),NUWORD),(A(15861),SBIT)

```

70

030

E 1

EQUIVALENCE(A(8000),BITS),(A(8080),ATS),(A(8100),MAD)		
1	,(ELTPOK,TAPEOK),(ATTCNT,NOATS)	
2	,(T4,FOUR),(SCRAP,FIVE)	
	BOOLEAN CLDONE	
	BOOLEAN CAT,CEL,CTO,CEID,PLP1,	0110
1	PLP2, PEL, PAT, IFERR1, IFERR2, TAPEOK, OUT, FEL, FSN	0120
2	,CLC , NTAPE, EOFB , SET4, CKRN	0130
	BOOLEAN EPAR, EEL, EEID, EAT, E2NDPN	0140
	BOOLEAN SPAR, SEL, SEID, SAT, S2NDPN, S1STEQ	0160
	BOOLEAN FSNEW,FSOLD, FSEL	0190
	BOOLEAN CPABLE,DONE,EQUAL,ELTPOK,PAREN,THEN	
	BOOLEAN NUELTP	B 4
	BOOLEAN ALLSET, PERIOD	
	BOOLEAN WITH	
R		
R	** COMPILOR CONTROL CARDS **	
R		
	VECTOR VALUES PUNCT=\$,\$, \$. \$, \$(\$, \$) \$, \$=\$	
	END OF FUNCTION	*

\$COMPILE MAC,PRINT OBJECT, PUNCH OBJECT

PARCK000

EXTERNAL FUNCTION(Q)

R'N

DEFINE UNARY OPERATOR .BLANK. , PRECEDENCE SAME AS .ABS.

MODE STRUCTURE 1 = .BLANK. 1

CLA B
ARS 18
ALS 18
OUT AC
END

ENTRY TO PARCK.

3540

I= Q

W'R CKRN

PRINT COMMENT \$OPARCK\$

E'L

R

3550

R

IF PAR NOT ON LIST PUT IT ON AND GENERATE

3560

R

A UNIQUE NAME ON ZPAR LIST

3570

PI = LOCATE.(NUWORD, FPAR, PARCNT)

3580

WHENEVER PI .L. 0

3590

FPAR(PARCNT) = NUWORD

3600

PI = PARCNT

3610

NUWORD= .BLANK. (NUWORD)

R

CHECK UNIQUENESS OF NUWORD ON ZPAR LIST.

3630

K= 3

3640

I = 0

3650

ZLIST

J = LOCATE.(NUWORD, ZPAR, PARCNT)

3660

WHENEVER J .GE. 0

3670

WHENEVER I .L. 26

3680

I = I+1

3690

NUWORD = INSRTC.(NUWORD,K,NUM(I))

OTHERWISE

3710

I=10

K = 2

3730

END OF CONDITIONAL

3740

TRANSFER TO ZLIST

3750

OTHERWISE

3760

ZPAR(PARCNT) = NUWORD

3770

PARCNT = PARCNT+1

3780

W'R PARCNT .G. 70, PRINT COMMENT \$OPARAMETER TABLE LENGTH EXCEEDED.\$

END OF CONDITIONAL

3810

OTHERWISE

3811

J=PI/36

3812

I= PI-J*36

3813

ASSERT = NUWORD

3814

WHENEVER IFBIT.(SCOPB1(J),I).AND..NOT.IFBIT.(SBIT(J),I)

SCOPB1(J)= INBIT.(SCOPB1(J),I,0)

P'IT FOMAT, ASSERT...ASSERT(7)

END OF CONDITIONAL

END OF CONDITIONAL

3820

FUNCTION RETURN (PI)

3830

VECTOR VALUES NUM= \$1\$, \$2\$, \$3\$, \$4\$, \$5\$, \$6\$, \$7\$, \$8\$, \$9\$, \$A\$, \$

3860

1B\$, \$C\$, \$D\$, \$E\$, \$F\$, \$G\$, \$H\$, \$I\$, \$J\$, \$K\$, \$L\$, \$M\$, \$N\$, \$O\$, \$P\$, \$Q

3870

1\$, \$R\$, \$S\$, \$T\$, \$U\$, \$V\$, \$W\$, \$X\$, \$Y\$, \$Z\$

3880

VECTOR VALUES ASSERT=\$ \$, \$ IS NOT BROAD SCOPE IN THIS ELEMENT

1 DESCRIPTION.\$

R

3850

PROGRAM COMMON A

```

V'S FOMAT=$1H0,22C6*$
DIMENSION A(15999),SBIT(1),SCOPB1(1), FPAR(71),ZPAR(69)      D      1
NORMAL MODE IS INTEGER
EQUIVALENCE(A(15700),FPAR) ,(FPAR(70),SCOPB1)                  E      1
1      ,(A(15603),PARCNT),(A(15861),SBIT)
2      ,(A(15860),NUWORD),(A(15772),ZPAR)
3      ,(A(15856),CKRN)
BOOLEAN IFABIT., IFBIT
BOOLEAN CKRN
END OF FUNCTION

```

```

$ASSEMBLE,PUNCH OBJECT                                           PLACE000
ENTRY PLACE      CALLING SEQUENCE EXECUTE PLACE.(A(N),L,$B$)

```

PLACE	NOP		
IN	CLA	1,4	
	ADD	IN	
	STA	BASE	
	CLA*	2,4	
	PAX	0,1	
	CAL*	3,4	SIX CHARACTER WORD TO BE STORED IN
BASE	SLW	** ,1	A(N),...,A(L)
	TIX	BASE,1,1	
	TRA	4,4	
	END		

\$ASSEMBLE,PUNCH OBJECT

RDTP4 00

```

RDTP4  ENTRY  RDTP4
      CAL*   1,4
      ALS    18
      STD    UNIT
      STD    UNITA
      STD    UNITB
      CAL    2,4
      STA    SCRATCH
      CAL    3,4
      STA    WORD1
      CAL    4,4
      STA    WORD2
      SXA    RET,4
      CALL   RDSBIN
UNIT   BLK    RET,**
      BLK    WORD1
      BLK    EOF,,RTT
      CALL   CHEKIO
UNITA  BLK    **,
      CALLIO .PRINT
      FMT    FORMAT
      ENDIC
      CALL   SYSTEM
FORMAT BCI    *,H*OHELP - IO TRAP DOESNT UNDERSTAND ME.**
WORD1  IOBP   **,1
WORD2  IOBT   **,1
EOF     CALLIO .PRINT
      FMT    EOFMSG
      ENDIC
SCRATCH TRA   **
EOFMSG  BCI    *,H*OEOF DETECTED ON FOUR - ASSUME IT IS SCRATCH.**
RTT     CALL   RDSDEC
UNITB   BLK    **,
      BLK    WORD1
      BLK    0,0,0
      CALLIO .PRINT
      FMT    REDMSG
      ENDIC
      TRA    SCRATCH
REDMSG  BCI    *,H*OTAPE FOUR REDUNDANCY - ASSUME SCRATCH.**
RET     AXT    **,4
      TRA    5,4
      END

```

\$COMPILE MAD,PRINT OBJECT,PUNCH OBJECT

ST END00

```

      EXTERNAL FUNCTION(ZZ)
      R'N
      V'S NONS=$PADPADPADPADPAD$
      ENTRY TO START.
      DONE = 0B
      OUT = 0B
      IFERR2=0B
STCC(0) REWIND TAPE FOUR
      REWIND TAPE THREE
      REWIND TAPE FIVE
      EXECUTE RDTP4.(FOUR,HA,ELTCK,IDENT)
      W'R (ELTCK .NE. $ELTAPES) .OR. (ZZ .NE. IDENT)
HA      REWIND TAPE FOUR
      WRITE BINARY TAPE FOUR,$ELTAPES,ZZ,NONS...NONS(2)

```

```

WRITE BINARY TAPE FOUR,PELCNT,TELCNT,PARCNT,NONS...NONS(2)
WRITE BINARY TAPE FOUR,A(15604)...A(15853)
END OF FILE TAPE FOUR
PRINT COMMENT $OELTAPE FILE 1 WRITTEN.$
WRITE BINARY TAPE FOUR,$PROLOG$,NONS...NONS(2)
C'E
TAPEOK=1B
READ BINARY TAPE FOUR,PELCNT,TELCNT,PARCNT
READ BINARY TAPE FOUR,A(15604)...A(15853)
E'L
TELCNT = 0
FUNCTION RETURN
ENTRY TO END.
X= ZZ
WHENEVER .NOT. (X.GE. 3 .OR. X .L. 0), TRANSFER TO PART(X)
PRINT COMMENT $QX DOES NOT EQUAL 0,1,2, ERROR IN END.$
PART(0)
CONTINUE
SW=0
WHENEVER .NOT.DONE,TRANSFER TO END1
PRINT COMMENT $2NORMAL TERMINATION - DATA EXHAUSTED$
R   MUST SAVE ELEMENT TAPE IF OURS USED
EXECUTE SYSTEM.
PART(1)
CONTINUE
SW=1
OUT= 1B
END1
CONTINUE
EXECUTE CONCK.
WHENEVER IFERR2
PRINT COMMENT $QAMBIGUOUS DATA, PROBLEM CANNOT CONTINUE$
OR WHENEVER IFERR1
PRINT COMMENT $OMINOR ERROR, PROGRAM CONTINUING$
END OF CONDITIONAL
EXECUTE SYNONM.(SCNT)
R   ELIMINATE SYNONYMS
EXECUTE SYNELM.
R   PRINT OUT CONNECTIONS LIST
EXECUTE CONNEC.(CCNT)
WHENEVER PCNT .G. PCNT(1)
    COUNT = PCNT
OTHERWISE
    COUNT = PCNT(1)
END OF CONDITIONAL
EXECUTE INTPR.(COUNT)
WHENEVER FSCNT .G. ECNT
    COUNT = FSCNT
R   PRINT OUT EQUIVALENCE LISTS
OTHERWISE
    COUNT = ECNT
END OF CONDITIONAL
DONE=1B
FUNCTION RETURN
CONTINUE
EXECUTE EQUIVN.(COUNT)
EXECUTE IPRELM.
PRINT COMMENT $OPARS$
P:T FOMAT,FPAR...FPAR(PARCNT-1)
PRINT COMMENT $OZPAR$
P:T FOMAT,ZPAR...ZPAR(PARCNT-1)
EXECUTE SEQPGM.

```

1330
1350

1370
1380

1410

1440

```

INTERNAL FUNCTION (KOUNT)
FORMAT VARIABLE FV1,FV2
ENTRY TO SYNONM.
PRINT COMMENT $1SYNONYMS.$
LOW=21
HIGH=30
FV1=1
FV2=5
LIST P/T FORM,TITLE...TITLE(6),TITLE(2),...,TITLE(6)
V'S FORM=$1H0,12(C6,S4)*$
V'S TITLE=$COUNT$, $TRUNAM$, $PARAM$, $ELEM$, $EID$, $ATT$, $AID$
T'H LIST1, FOR K=0,1, K .G. KOUNT-1
LIST1 P/T FORM1,K,(J=LOW,1, J .G. HIGH, A(J,K))
V'S FORM1=$1H0,13,S7,2('FV1'S10,'FV2'(C6,S4),S-10),T122*$
F'N
ENTRY TO CONNEC.
PRINT COMMENT $-CONNECTIONS.$
LOW=31
HIGH=38
FV1=2
FV2=4
T'O LIST
ENTRY TO INTPR.
PRINT COMMENT $-INPUT PARAMETERS AND DESIRED RESULTS.$
LOW=4
HIGH=13
FV1=1
FV2=5
T'O LIST
ENTRY TO EQUIVN.
PRINT COMMENT $-EQUIVALENCES.$
LOW=14
HIGH=19
FV1=0
FV2=6
T'O LIST
E'N
PROGRAM COMMON A
NORMAL MODE IS INTEGER
DIMENSION FPAR(71),ZPAR(69)
DIMENSION A(15999,DIM1),PCNT(1)
VECTOR VALUES TWO = 2
VECTOR VALUES THREE= 3
VECTOR VALUES FOUR = 4
VECTOR VALUES FIVE =9
VECTOR VALUES DIM1 = 2,401,400
V'S FOMAT=$10(S6,C6)*$
VECTOR VALUES A(15600)=$ELTAPES$
EQUIVALENCE ( A(15601),PELCNT ),( A(15602),TELCNT ),
1 ( A(15603),PARCNT ),( A(15604),ELN ),
2 ( A(15700),FPAR ),( A(15772),ZPAR ),
3 ( FPAR(70),SCOPB1 ),( FPAR(71),SCOPB2 ),
4 ( A(15852),TAPEOK ),( A(15842),CCNT ),
5 ( A(15843),SCNT ),( A(15844),ECNT ),
6 ( A(15845),PCNT ),( A(15847),FSCNT ),
7 ( A(15848),ID1 ),( A(15849),ID2 ),
8 ( A(15850),IFERR1 ),( A(15851),PRLCNT )
9 ( A(15853),DIM1)
EQUIVALENCE (A(15889),FIVE)

```

70

V 5

30

EQUIVALENCE (A(15856),CKRN),(A(15857),THREE)	E	1
1 , (A(15858), FOUR),(A(15859),CLDONE)		
EQUIVALENCE(A(8000),BITS),(A(8080),ATS),(A(8100),MAD)		
1 , (ELTPOK,TAPEOK),(ATTCNT,NOATS)		
BOOLEAN CLDONE		
BOOLEAN CPABLE,DONE,EQUAL,ELTPOK,PAREN,THEN		
BOOLEAN CAT,CEL,CTO,CEID,PLP1,		0110
1PLP2, PEL, PAT, IFERR1, IFERR2, TAPEOK, OUT, FEL, FSN		0120
2 ,CLC , NTAPE, EOFB , SET4, CKRN		0130
BOOLEAN EPAR, EEL, EEID, EAT, E2NDPN		0140
BOOLEAN SPAR, SEL, SEID, SAT, S2NDPN, S1STEQ		0160
BOOLEAN FSNEW,FSOLD, FSEL		0190
BOOLEAN NUELTP	B	4
EQUIVALENCE (SCRAP,FIVE),(T4,FOUR)		
END OF FUNCTION		*

SCOMPILE MAD,PRINT OBJECT,PUNCH OBJECT
EXTERNAL FUNCTION

SYNELMOO

	R'N	
	R SYNONYM ELIMINATION	10
	ENTRY TO SYNELM.	0400
	R -BRIEF DESCRIPTION OF FOLLOWING PROGRAM-	0020
	R THE I-TH ROW OF THE SYNONYM TABLE IS COMPARED BY SEARCHS TO	0030
	R 1. FUNCTION SUBSTITUTION LIST - SEARCH 1	0040
	R 2. (A)INPUT PARAMETERS LIST SEARCH 2	0050
	R (B)DESIRED RESULTS LIST	0060
	R (C)EQUIVALENCE LIST	70
	R 3.CONNECTIONS LIST. -SEARCH 3	0080
	R WHEN A MATCH IS FOUND IN THE ABOVE LISTS, THE TRUE NAMES	0090
	R REPLACE THE SYNONYMOUS NAMES IN THE LIST.	0100
	R	110
	THROUGH LOOPA, FOR I=0,1,I.GE. SCNT	
	OCCURS = 0B	1170
	TPAR = A(21,I)	0490
	TEL = A(22,I)	0500
	TEID = A(23,I)	0510
	TAT = A(24,I)	0520
	TAID = A(25,I)	0530
	SYPAR = A(26,I)	0540
	SYEL = A(27,I)	0550
	SYEID = A(28,I)	0560
	SYATT = A(29,I)	0570
	SYAID = A(30,I)	0580
	EXECUTE SRCH1.	0600
	COUNT = PCNT(0)	0610
	EXECUTE SRCH2.(A(4,0),A(5,0),A(6,0),A(7,0),A(8,0))	0630
	COUNT = PCNT(1)	0640
	EXECUTE SRCH2.(A(9,0),A(10,0), A(11,0), A(12,0) ,A(13,0))	0660
	COUNT = ECNT	0670
	EXECUTE SRCH2.(A(14,0), A(15,0),A(16,0) ,A(17,0) ,A(18,0))	0690
	EXECUTE SRCH3.	0710
	W'R .NOT. OCCURS, P'IT FOMAT, NO...NO(9)	
LOOPA	CONTINUE	0720
	FUNCTION RETURN	0730
	INTERNALFUNCTION(AA,AB,AC,AD,AE)	0740
	R SEARCH OF INPUT PARAMETER,DESIRED RESULTS,EQUILIALENCE	0750
	ENTRY TO SRCH2.	0760
	THROUGH LOOPS, FOR L=0,1,L.GE.COUNT	
	WHENEVER (SYAID.E.AE(L).OR. SYAID.E.\$ \$)	0780
	1.AND.(SYATT.E.AD(L) .OR. SYATT .E. \$ \$)	0790
	2.AND.(SYEID.E.AC(L) .OR. SYEID .E. \$ \$)	0800
	3.AND.(SYEL.E.AB(L) .OR. SYEL .E. \$ \$)	0810
	4.AND.(SYPAR.E.AA(L) .OR. SYPAR .E. \$ \$)	0820
	WHENEVER TPAR .NE. \$ \$, AA(L) = TPAR	0830
	WHENEVER TEL .NE. \$ \$, AB(L) = TEL	0840
	WHENEVER TEID .NE. \$ \$, AC(L) = TEID	0850
	WHENEVER TAT .NE. \$ \$, AD(L) = TAT	0860
	WHENEVER TAID .NE. \$ \$, AE(L) = TAID	0870
	OCCURS = 1B	0910
LOOPS	END OF CONDITIONAL	0920
	FUNCTION RETURN	0930
	END OF FUNCTION	094
	R SEARCH OF SUBSTITUTION TABLE.	0950
	INTERNAL FUNCTION	0960
	ENTRY TO SRCH1.	0970

	WHENEVER TPAR.E.\$ \$,FUNCTION RETURN	0980
	THROUGH LOOPSB, FOR L =0,1, L .GE. FSCNT	0990
	EQUIVALENCE (A(800),ACOL2),(A(1200),ACOL3)	1000
	WHENEVER (SYEID .E.ACOL3(L) .OR. SYEID .E. \$ \$)	1010
	1.AND.(SYEL.E.ACOL2(L) .OR. SYEL .E. \$ \$)	1020
	WHENEVER TEL .NE. \$ \$, ACOL2(L) = TEL	1030
	WHENEVER TEID .NE. \$ \$, ACOL3(L) = TEID	1070
	OCCURS = 1B	1080
LOOPSB	END OF CONDITIONAL	1100
	BOOLEAN OCCURS	1110
	FUNCTION RETURN	1120
	END OF FUNCTION	1130
R	SEARCH OF CONNECTIONS	1150
	INTERNAL FUNCTION	1160
	ENTRY TO SRCH3.	
	WHENEVER TPAR .NE. \$ \$, FUNCTION RETURN	1180
	THROUGH LOOPC, FOR L=0,1,L.GE.CCNT	1190
	WHENEVER (SYAID .E. A34(LCNT).OR. SYAID .E. \$ \$)	1200
	1.AND.(SYATT.E. A33(LCNT) .OR. SYATT .E. \$ \$)	1210
	2.AND.(SYEID.E. A32(LCNT) .OR. SYEID .E. \$ \$)	1220
	3.AND.(SYEL .E. A31(LCNT) .OR. SYEL .E. \$ \$)	1230
	WHENEVER TEL .NE. \$ \$, A31(L) = TEL	1240
	WHENEVER TEID.NE. \$ \$, A32(L) = TEID	1250
	WHENEVER TAT .NE. \$ \$, A33(L) = TAT	1260
	WHENEVER TAID.NE. \$ \$, A34(L) = TAID	1310
	END OF CONDITIONAL	1320
	WHENEVER (SYAID .E. A38(LCNT) .OR. SYAID .E. \$ \$)	1330
1	.AND.(SYATT .E. A37(LCNT) .OR. SYATT .E. \$ \$)	1340
2	.AND.(SYEID .E. A36(LCNT) .OR. SYEID .E. \$ \$)	1350
3	.AND.(SYEL .E. A35(LCNT) .OR. SYEL .E. \$ \$)	1370
	EQUIVALENCE(LCNT,L)	1380
	WHENEVER TEL .NE. \$ \$, A35(L) = TEL	1390
	WHENEVER TEID.NE. \$ \$, A36(L) = TEID	1400
	WHENEVER TAT .NE. \$ \$, A37(L) = TAT	1410
	WHENEVER TAID.NE. \$ \$, A38(L) = TAID	1460
	OCCURS = 1B	1470
LOOPC	END OF CONDITIONAL	1520
	FUNCTION RETURN	1530
	END OF FUNCTION	1500
	DIMENSION NO(10)	
	EQUIVALENCE(NO,SYPAR),(NO(1),SYEL),(NO(2),SYEID)	
	1,(NO(3),SYATT),(NO(4),SYAID)	
	VECTOR VALUES NO(5)=\$ DOES NOT OCCUR ON ANY LIST.\$	
	NORMAL MODE IS INTEGER	0120
	DIMENSION ERR(8)	0130
	DIMENSION A(15999,DIM1), PCNT(1)	0170
	VECTOR VALUES DIM1 = 2,401,400	0180
R	DEFINITION OF A(39,0)...A(39,399)	0190
	VECTOR VALUES A(15600)=SELTAPES	0200
	EQUIVALENCE (A(15601),PELCNT),(A(15602),TELCNT),	0210
1	(A(15603),PARCNT),(A(15604),ELN),	0220
2	(A(15700),FPAR),(A(15772),ZPAR),	0230
3	(FPAR(70),SCOPB1),(FPAR(71),SCOPB2),	0240
4	(A(15852),TAPEOK),(A(15842),CCNT),	0250
5	(A(15843),SCNT),(A(15844),ECNT),	0260
6	(A(15845),PCNT),(A(15847),FSCNT),	0270
7	(A(15848),ID1),(A(15849),ID2),	0280
8	(A(15850),IFERR1),(A(15851),PRLCNT)	0290
	EQUIVALENCE (PELT,A(8200))	

DIMENSION PELT(300),ELN(95),FPAR(71),ZPAR(69)	0300
PROGRAM COMMON A	0310
V'S FOMAT=\$1H0,22C6*\$	
BOOLEAN IFERR1	0320
DIMENSION A31(1),A32(2),A33(1),A34(1),A35(1),A36(1),A37(1),	0330
1A38(1),A39(1)	340
EQUIVALENCE (A(12400),A31),(A(12800),A32),(A(13200),A33),	0350
1(A(13600),A34),(A(14000),A35),(A(14400),A36),(A(14800),A37),	0360
2(A(15200),A38)	370
EQUIVALENCE (A39,A(15600))	0380
DIMENSION ACOL2(1),ACOL3(1)	0390
END OF FUNCTION	*

\$ASSEMBLE,PUNCH OBJECT

TAPE1000

	ENTRY	SKFILE
	ENTRY	SKRECD
	ENTRY	BKSPCE
SKFILE	AXT	0,1
	TRA	START1
SKRECD	AXT	1,1
START1	CLA*	2,4
	TZE	3,4
	TMI	BACKS
	STA	BLK1
	CLA	TYPE,1
	STA	CALL
	CLA*	1,4
	PAX	0,1
	SXD	BLK1,1
	SXA	XRC1,4
CALL	TSX	**,4
BLK1	BLK	**,,**
XRC1	AXT	**,4
	TRA	3,4
	CALL	SKPREC
TYPE	CALL	SKPFIL
BKSPCE	AXT	1,1
	CLA*	2,4
	TZE	3,4
BACKS	PAX	0,2
	CLA	TYPE2,1
	STA	CALL2
	CLA*	1,4
	PAX	0,1
	SXD	BLK2,1
	SXA	XRC2,4
CALL2	TSX	**,4
BLK2	BLK	**,,**
	TIX	*-2,2,1
XRC2	AXT	**,4
	TRA	3,4
	CALL	BSRTAP
TYPE2	CALL	BSFTAP
	END	

*

\$COMPILE MAD,EXECUTE,DUMP,PUNCH OBJECT ,PRINT OBJECT

CORE2000

R'N
PRINT COMMENT \$1CORE 2\$
EXECUTE OUTPUT.(\$PROLOG\$,1)
P'S PELCNT,TELCNT
THREE=3
FOUR=4
A(15853)=2
A(15854)=401
A(15855)=400
REWIND TAPE FOUR
EXECUTE COLIN.
EXECUTE SSORDR.
EXECUTE SUBSPT.(A(27,0),CCNT)
EXECUTE PROLOG.
P'S PELCNT,TELCNT
R PRINT OUT AND PUNCH EQUIVALENCE CARDS*****

EXECUTE EQVGEN.
SQUEEZ.
SNOW.
EXECUTE SEQPGM.
INTERNAL FUNCTION
ENTRY TO PROLOG.
EXECUTE OUTPUT.(\$PROLOG\$,1)
REWIND TAPE FOUR
TAPCO = 0
REP1 READ BINARY TAPE FOUR, IDENT
EXECUTE OUTPUT.(IDENT,1)
WHENEVER IDENT .E. \$PROLOG\$.OR. IDENT.E.\$ PROLOS\$,TRANSFER TO
1G01

EXECUTE SKFILE.(FOUR, 1)
TAPCO = TAPCO+1
EXECUTE OUTPUT.(TAPCO,1)
WHENEVER TAPCO .G. 12, EXECUTE ERR.(10000)
TRANSFER TO REP1

GO1 CONTINUE
SETEOF.(G022)
REWIND TAPE THREE
TAPCO1 = 0
REP2 CONTINUE
G022

READ BINARY TAPE THREE, IDENT
EXECUTE OUTPUT.(IDENT,1)
WHENEVER IDENT.E. \$ CALC\$,TRANSFER TO G02
W'R IDENT.E.\$FSUB.\$
TAPCO1 = TAPCO1 +1
W'R TAPCO1.G. 2
PRINT COMMENTS\$ HELP I'M CAUGHT IN A CHINESE IBM FACTORY\$
EXECUTE ERROR.

E'L
REWIND TAPE THREE
T'O REP2

E'L
EXECUTE SKFILE.(THREE,1)
TAPCO = TAPCO+1
EXECUTE OUTPUT.(TAPCO,1)
WHENEVER TAPCO .G. 12, EXECUTE ERR.(13000)
TRANSFER TO REP2
GO2 CONTINUE

```

UNIT = FOUR
CKRN = 1B
EOFBLN = 0B
CCOUNT = 0B
PLSFLG = 0B
MINFLG = 0B
MCOUNT = 0B
R (**CALCULATIONS ARE ON TAPE THREE, FILE ONE.)
SCANS
CONTINUE
EXECUTE OUTPUT.($SCANS$,1)
EXECUTE PSCAN.
WHENEVER MINFLG
    EXECUTE CALCS.
PLSFLG = 0B
    MINFLG = 0B
    WHENEVER MCOUNT, TRANSFER TO PRON
    MCOUNT=1B
    TRANSFER TO SCANS
END OF CONDITIONAL
TRANSFER TO SCANS
PRON
CONTINUE
EXECUTE OUTPUT.($PRON $,1)
EXECUTE PSCAN.
WHENEVER MINFLG
    MINFLG = 0B
    EXECUTE OUTPUT.(JACK,6)
    EXECUTE OUTPUT.(JACK2,3)
    V'S JACK2 = $ PROLOG RETURNED    $
    FUNCTION RETURN
    END OF CONDITIONAL
    ERR.(2090)
    VECTOR VALUES JACK = $ LAST MINUS SIGN IN NEW ELEMENT TAPES$
    END OF FUNCTION
R
INTERNAL FUNCTION
    ENTRY TO CALCS.
    EXECUTE OUTPUT.($CALCS$,1)
    WHENEVER .NOT. CLDONE, FUNCTION RETURN
    UNIT = THREE
    MINFLG = 0B
    PLSFLG = 0B
    EXECUTE PSCAN.
    W'R MINFLG
    UNIT = FOUR
    FUNCTION RETURN
    O'R PLSFLG
    PRINT COMMENT $ CALCULATIONS ENDED    $
    UNIT = FOUR
    FUNCTION RETURN
    OTHERWISE
        EXECUTE OUTPUT.(CERR,5)
        VECTOR VALUES CERR=$CALCULATIONS IMPROPERLY TERMINATED.$
    END OF CONDITIONAL
    UNIT = FOUR
    FUNCTION RETURN
    END OF FUNCTION
R (**PROLOGUE SECTION STORAGE ASSIGNMENTS.)
PROGRAM COMMON A,B
PROGRAM COMMON C

```

```

DIMENSION C(200)
NORMAL MODE IS INTEGER
BOOLEAN IFERR1, CKRN, FSUBIN, EPFLG, MCOUNT, BOOLN
1,ONCEB, ONCEBB,MINFLG,PLSFLG,SLBFLG,FINFLG
2,CLDONE,EOFBLN,ELFLG,CCOUNT,SCANDL
VECTOR VALUES DIM1 = 2, 401,400
VECTOR VALUES PDIM =2,6,5
VECTOR VALUES THREE= 3
VECTOR VALUES FOUR = 4
VECTOR VALUES FIVE = 9
DIMENSION A(15999,DIM1),PELT(299,PDIM),T(300),ATTNO(50)
1,BFR(200),FPAR(71),ZPAR(69),B(1999 ,PDIM),PCNT(1)
EQUIVALENCE (PCNT(0),IPCNT),(PCNT(1),DRPCNT)
2,(T(100),FPAR),(T(172),ZPAR) ,(T(259),CLDONE)
3,(T(276),FCNO),(T(245),PCNT),(T(273),UNIT),(T(294),LOCLBL)
4,(T(248),ID1),(T(250),IFERR1),(A(6600),BFR)
EQUIVALENCE (T,A(15600)) ,(T(1),PELCNT)
1,(T( 3),PNUMBR),(T(245),IPCNT) ,(T(246),DRPCNT)
2,(T(247),FSCNT) ,(T(253),DIM1) ,(T(256),ERRNO )
3,(T(257),THREE) ,(T(258),FOUR) ,(T(263),FSUBIN)
4,(T(264),M) ,(T(265),CCA) ,(T(266),CARDS )
5,(T(267),MA) ,(T(268),EPFLG) ,(T(269),EPCNT )
6,(T(270),MCOUNT),(T(271),REM) ,(T(272),BOOLN )
7,(T(274),PXSIT) ,(T(275),LINE) ,(T(277),J )
8,(T(278),I3) ,(T(279),I2) ,(T(284),LBWORD)
9,(T(287),MINFLG),(T(288),PLSFLG),(T(289),FIVE )
EQUIVALENCE (T(242),CCNT),(T(002),TELCNT),(T(004),ELN)
1,(T(290),SLBFLG),(T(291),FINFLG)
6,(A( 7100),ATTNO) ,(A( 8200),PELT(0))
EQUIVALENCE(SCANDL,T(296)),(ELFLG,T(297)),(CCOUNT,T(295)),
1(EOFBLN,T(260)),(CCA,J2),(PARCNT,PNUMBR)
END OF PROGRAM

```

\$ASSEMBLE,PUNCH OBJECT

BITLGK00

ENTRY BLANK
ENTRY LGR
ENTRY LGL
ENTRY IFABIT
ENTRY ANDBIT
ENTRY SIGN
ENTRY EXTNAM

C=SIGN.(A,B) CAUSES C TO HAVE SIGN OF A*B

ANDBIT CAL* 1,4
ANA* 2,4
TZE 3,4
CLA ONE
TRA 3,4
IFABIT CAL* 1,4
TZE 2,4
CLA ONE
TRA 2,4
ONE PZE 1
EXTNAM NOP
BLANK CLA* 1,4
ARS 18
ALS 18
TRA 2,4
LGR CAL* 1,4
LGR 18
SLW TEMP
CLA LBLNK

	ORA	TEMP
	TRA	2,4
LGL	CAL*	1,4
	LDQ	RBLNK
	LGL	18
	SLW	TEMP
	CLA	RBLNK
	TRA	2,4
LBLNK	BCD 1	000
RBLNK	BCD 1000	
SIGN	CLA*	1,4
	LDQ*	2,4
	LLS	0
	TRA	3,4
TEMP	SYN	-1
	END	

CK

*

\$ASSEMBLE,PUNCH OBJECT			BUFFER00	
	ENTRY	EQUIVN		30
	ENTRY	CONNEC		40
	ENTRY	SYNONM		50
	ENTRY	INPTPR		60
	ENTRY	SUBSPT		70
	ENTRY	BITPRT	CALLING SEQUENCE))) (B(FSTCOL),NOLINES)	
EQUIVN	TSX	SETUP,2	EQUIVALENCE TABLE BUFFER SETUP	
	TSX	OUTPUT,4		
	TXH	VECE		
	TXH	TWO		
	CALL	SPRINT	LABEL FOR EQUIVALENCE	
	BLK	LABELC,,19		
ERESET	TSX	RESET,4		90
ELSIDE	TSX	ALPHA,4		0100
	TSX	BETA,4		110
	TXL	ELSIDE,2,2399		0120
	TSX	STOBNK,4		
	TXI	*+1,2,-8000	SUBSTUTION TABLE STARTS AT A(0)	
ERSIDE	TSX	ALPHA,4		0160
	TSX	BETA,4		0170
	TXL	ERSIDE,2,-4001		0180
	TSX	STOBNK,4		
	TSX	WRITE,4		0190
	TRA	ERESET		0200
CONNEC	TSX	SETUP,2	CONNECTIONS TABLE BUFFER SETUP	
	TSX	OUTPUT,4		
	TXH	VECC		
	TXH	TWO		
	CALL	SPRINT	LABEL FOR CONNECTIONS	
	BLK	LABELC,,19		
CRESET	TSX	RESET,4		220
	TSX	STOBNK,4		
CLSIDE	TSX	BETA,4		260
	TSX	ALPHA,4		270
	TXL	CLSIDE,2,1599	FIRST FOUR COLUMNS 4*400-1	0280
	TSX	STOBNK,4		
	CAL	TO		
	TSX	STOACC,4		
	TSX	STOBNK,4		
CRSIDE	TSX	ALPHA,4		330
	TSX	BETA,4		340
	TXL	CRSIDE,2,3199	LAST FOUR COLUMNS 8*400-1	0350
	TSX	STOBNK,4		
	TSX	WRITE,4		360
	TRA	CRESET		0370
TO	BCD	1 TO		380
SYNONM	TSX	SETUP,2	SYNONYMS TABLE BUFFER SETUP	
	TSX	OUTPUT,4		
	TXH	VECS		
	TXH	TWO		
	CAL	EQUAL		
	TRA	PNT		
INPTPR	TSX	SETUP,2	INPUT PARAMETERS AND DESIRED RESULTS	
	TSX	OUTPUT,4		
	TXH	VECI		
	TXH	TWO		
	CAL	BLANK		
PNT	SLW	PUNCT		

	CALL	SPRINT	LABEL FOR SYNONYMS AND INPUT PARAMETER LIST	
	BLK	LABELS,,19		
SRESET	TSX	RESET,4		450
SLSIDE	TSX	ALPHA,4		-470
	TSX	BETA,4		480
	TXL	SLSIDE,2,1599		
	TSX	ALPHA,4		
	CAL	PUNCT		
	TSX	STOACC,4		510
SRSIDE	TSX	BETA,4		520
	TSX	ALPHA,4		0530
	TXL	SRSIDE,2,3599		
	TSX	BETA,4		
	TSX	STOBNK,4		
	TSX	WRITE,4		0550
	TRA	SRESET		0560
EQUAL	BCD	1 =		0570
PUNCT	BCD	1		580
SUBSPT	TSX	SETUP,2	SUBSCRIPT BITS-INPUT PARAM.BITS-DES. RESULT	
	STA	BASE4		
	STA	BASE3		610
	TSX	OUTPUT,4		
	TXH	VECST		
	TXH	TWO		
	CALL	SPRINT	LABEL FOR SUBSCRIPT BITS	
	BLK	LABBIT,,19		
TRESET	TSX	RESET,4		630
	SLW	PARAMA		
BASE4	CLA	#,2	GET NEXT NUMBER OFF THE LIST.	
	STO	PARAMA		
	TSX	OCTDEC,4	CONVERT TO DECIMAL INTEGER	
	BLK	PARAMA		
	STO	BUFER,1		
	TXI	*+1,2,400		
	TXI	*+1,1,1		
	TXL	BASE4,2,1599		
	TXI	*+1,2,-3200		
	TSX	STOBNK,4		
	TXL	*-1,1,10		
	TSX	STOBNK,4		0820
	TSX	STOBNK,4	0	0830
	TSX	STOBNK,4		
	TSX	STOBNK,4		0860
	TSX	STOBNK,4		0870
	TSX	STOBNK,4		0880
	TRA	TRESET		0890
PARAMA	BSS	1		
LOCA	BSS	1		0910
ONE	PZE	1		
LOCB	BSS	1		930
SETUP	CLA*	2,4		
	STA	COUNT		
	CLA	COUNT		
	TZE	3,4	NO LINES IN THE TABLE TO BE PRINTED	
	STZ	ROWCNT		1000
	SUB	F400		
***** TPL *+3 CHANGED TO TMI *+3 *****				
	TMI	*+3		
	CALL	ERR		

	PAR	F400			
	CLA	1,4			0970
	STA	ALPHA			0980
	STA	BASE			0990
	SXA	IRC,4			1010
	TRA	1,2			
RESET	SXA	RIRC,4	RESETS THE INDEX REGISTERS	INSERTS	1040
	LXA	ROWCNT,2	CARRIAGE CONTROL		1050
	LXA	ONE,1			1060
	TSX	OCTDEC,4			1070
	TXH	ROWCNT			1080
	ORA	CARCON			1090
	TSX	STOACC,4			1100
	TSX	STOBNK,4			
	LXA	RIRC,4			1130
	TRA	1,4			1140
ALPHA	CAL	**,2	TAKES THE NEXT WORD OFF THE LIST		1150
ALPHA1	SLW	BUFER,1	AND STORES THE WORD IN THE NEXT		1160
	TXI	**+1,2,400	AVAILABLE LOCATION IN THE BUFFER		1170
	TXI	**+1,1,1			1180
	TRA	1,4			1190
BETA	LDQ	BLANK	TAKES THE NEXT WORD OFF THE LIST		1200
BASE	CAL	**,2	AND STORES THE WORD IN THE MIDDLE		
	LGR	18	OF THE NEXT TWO AVAILABLE WORDS		1220
	ORA	HFBLNK	IN THE BUFFER		1230
	SLW	BUFER,1			1240
	TXI	**+1,1,1			1250
	STQ	BUFER,1			1260
	TXI	**+1,2,400			1270
	TXI	**+1,1,1			1280
	TRA	1,4			1290
STOBNK	CAL	BLANK	STORES A BLANK IN BUFFER		
STOACC	SLW	BUFER,1	STORES THE CONTENTS OF ADC IN BUFFER		1300
	TXI	**+1,1,1			1310
	TRA	1,4			1320
WRITE	SXA	WIRC,4	PRINTS OUT NINETEEN COLUMNS OF THE BUFFER		1330
	TSX	OUTPUT,4	TRANSFER BACK TO CALLING PART IF MORE ROWS.		
	TXH	BUFER-1			
	TXH	NINET	BACK TO CALLIN PROGRAM IF PRINTING IS FINIS		
	CLA	ROWCNT			1370
	ADD	ONE			1380
	STO	ROWCNT			1390
	SUB	COUNT			1400
	LXA	WIRC,4			
	TMI	1,4	TO SUBROUTINE -MORE ROWS TO BE PRINTED		1420
	LXA	IRC,4			1430
	TRA	3,4	TO CALLING PROGRAM-FINISHED PRINTING		1440
WIRC	BSS	1			1450
CARCON	BCD	10 000			1460
IRC	BSS	1			1470
HFBLNK	BCD	1 000			1490
BLANK	BCD	1			1500
STOBIT	SXA	SIRC,4	STORE BITS OF SUBSCRIPT LISTS		
BASE3	CAL	**,2			
	SLW	PARAMB			
	TSX	OCT,4	CONVERT BITS INTO OCTAL EQUIVALENT		
	TXH	PARAMB			
	TXH	LOCBUF			
	CAL	LOCBUF			

```

      TNZ      OUT1
      CAL      PRD
OUT1   TSX      STOACC,4
      CAL      LOCBUF-1
      TNZ      OUT2
      CAL      PRD
OUT2   TSX      STOACC,4
      TXI      *+1,2,400
SIRC   AXT      **,4
      TRA      1,4
BITPRT TSX      SETUP,2
      STA      BASE3
      CLA      BLANK
      AXT      0,1
      STO      BUFER,1
      TXI      *+1,1,1
      TXL      *-2,1,19
      TSX      OUTPUT,4
      PAR      BITLBL
      PAR      =9
BRESET TSX      RESET,4
      TSX      STOBNK,4
      TXL      *-1,1,3
      TSX      STOBIT,4
      TXI      *+1,2,-399
      TSX      STOBIT,4
      TXI      *+1,2,-399
      TSX      STOBNK,4
      TSX      STOBIT,4
      TXI      *+1,2,-399
      TSX      STOBIT,4
      TXI      *+1,2,-399
      TSX      WRITE,4
      TRA      BRESET
      BCD      1000000
      BCD      1E00000
      BCD      1T ABOV
      BCD      1ELEMEN
      BCD      1S FOR
      BCD      1ON BIT
      BCD      1LLECTI
      BCD      1ENT CO
BITLBL BCD      1STATAM
PARAMB PZE      0
LOCBUF BES      2
      CALL     OUTPT
      CALL     OUTPUT
      CALL     OCT
      CALL     OCTDEC
PRD    BCD      1 . .
BUFER  BES      20
RIRC   BSS      1
ROWCNT BSS      1
F400   PZE      400
NINET  PZE      19
TWO    PZE      2
LABELC BCD      9
      BCD      9D
LABLE  BCD      1

```

IF NO BITS ARE PRESENT, REPLACE THE ZEROS
WITH PERIODS.

PRINT FOR BITS IN B-REGION OF CORE 3

1670

ELEM EID ATT AI
ELEM EID ATT AID

	BCD 9	TRUE NAME	PAR	ELEM	EID	ATT	
	BCD 9	AID SUBST.	NEW	OLD	ELEM	EID	
LABELS	BCD 1						
	BCD 9	PARA	ELEM	EL ID	ATT	ATT ID	
	BCD 9	PARA	ELEM	EL ID	ATT	ATT ID	
LABBIT	BCD 1						
	BCD 9						
	BCD 9	R	L	R-L	L-R		
		INPUT	PARAMETER	BITS	DESIRED	RESULT	BITS
	BCD 1	LENCE.					CK
VECE	BCD 1	EQUIVA					
	BCD 1	TION.					CK
VECC	BCD 1	CONNEC					
	BCD 1	PARAM					
VECI	BCD 1	INPUT					
	BCD 1	BITS.					CK
VEGST	BCD 1	SUBST					
	BCD 1	MS.					CK
VECS	BCD 1	SYNONY					
COUNT	PZE	399					
N	PZE	19					
	END						

*

\$	COMPILE MAD,PRINT OBJECT,PUNCH OBJECT	COLIN.00	
	EXTERNAL FUNCTION		1230
	R'N		
	R		1240
	R ** PELT=PRESENT ELEMENT TABLE IS SET UP		1260
	R ** ALL COLLECTIONS ARE IN CORE		1270
	R ***** UPON COMPLETION OF THIS ROUTINE*****		
	ENTRY TO COLIN.		1280
	EXECUTE PLACE.(PELT,298,0)		
	EXECUTE PLACE.(B,1800,0)		
	PRLCNT = PELCNT + TELCNT		
	J = 0		
	THROUGH MOVELP, FOR I = 0,1,I.G. PRLCNT		
MOVELP	PELT(J) = ELN(I)		
	J = J +5		
	ALLIN = 1B		
	EXECUTE OUTPUT.(\$COLIN.\$,1)		
	REWIND TAPE FOUR		
	EXECUTE OUTPUT.(LABEL , 4)		
	VECTOR VALUES LABEL = \$ PELCNT TELCNT PRLCNT\$		
	CHECK = OCTDEC.(PELCNT)		
	CHECK(1) = OCTDEC.(TELCNT)		
	CHECK(2) = OCTDEC.(PRLCNT)		
	DIMENSION CHECK(3)		
	EXECUTE OUTPUT.(CHECK,3)		
	M = PRLCNT		
	MC=0		1310
	PELT(1)=0		1320
	PELT(3)=0		1330
	THROUGH LP1, FOR I=0,5,I.E.5*PRLCNT		1340
NXT	WHENEVER I .E. 0		
	EXECUTE SKFILE.(FOUR,2)		
	OTHERWISE		
	EXECUTE SKFILE.(FOUR,1)		
	END OF CONDITIONAL		
	READ BINARY TAPE FOUR, ELN1, NOATS		
	WHENEVER ELN1 .NE. PELT(I)		
	MC=MC+1		1390
	WHENEVER MC.G. M		
	NO = ELN1		
	EXECUTE OUTPUT.(NO, 8)		
	VECTOR VALUES NO = \$ \$,SELEMENT DESCRIPTION NOT ON TAPE FOUR\$		
	EXECUTE ERR.(3017)		
	END OF CONDITIONAL		
	TRANSFER TO NXT		1410
	END OF CONDITIONAL		1420
	PELT(I+2)=NOATS		1430
	J=PELT(I+1)		1440
	J1=J+NOATS-1		1450
	PELT(I+6)=J1+1		1460
	READ BINARY TAPE FOUR, ATS(J),...,ATS(J1)		
	J2=4*NOATS		1480
	EXECUTE SETEOF.(END1)		1490
	NOCOL=0		1500
	J=PELT(I+3)		1510
RDCOL	J1=J+J2-1		1520
	COUNT = J1 - J		
	READ BINARY TAPE FOUR, B(J)... B(COUNT+J)		
	J=J+J2		1540

	NOCOL=NOCOL+1	1550
	TRANSFER TO RDCOL	1560
END1	PELT(I+4)=NOCOL	1570
	PELT(I+8)=PELT(I+3)+J2*NOCOL	1580
	WHENEVER PELT(I+8).G.8399.OR.PELT(I+6).G.399	1590
	COLOVR(6)=PELT(I)	1600
	VECTOR VALUES COLOVR=\$ELEMENT COLLECTIONS EXCEED STORAGE \$,0	1610
	EXECUTE OUTPUT.(COLOVR,7)	1620
	IFERR1=1B	1630
	ALLIN = 0B	
	TRANSFER TO PRNOUT	
	END OF CONDITIONAL	1650
LP1	CONTINUE	1660
PRNOUT	T'H BBLIST ,FOR K=0,1,K.G.PRLCNT-1	
BBLIST	P'T PELTL,PELT(K,0)...PELT(K,4)	
	V'S PELTL = \$1H C6,S4,4(S2,I12) *\$	
	EXECUTE OUTPUT.(BREG,8)	
	THROUGH BLIST, FOR I=0,1,I.GE. M	
	P'T PELTL, PELT(I,0)...PELT(I,4)	
	EXECUTE OUTPUT.(ATTHD,3)	
	EXECUTE LPRINT.(\$ \$,ATS(PELT(I,1)),PELT(I,2))	
	T'H BLIST,FOR K=0,4,K.GE.PELT(I+1,3) - PELT(I,3)	
	SUBSC1 = PELT(I,3)+K	
	SUBSC2 = PELT(I,3)+K +3	
	PRINT FORMAT BITPR,SUBSC1 ,B(PELT(I,3)+K)...B(PELT(I,3)+K+3)	
	1,SUBSC2	
	V'S BITPR = \$1H S4,I12,S4,4(S2,K12),S2,I12*\$	
BLIST	CONTINUE	
	VECTOR VALUES BREG=\$ B-REGION. BITS FOR DESIRED RESULTS REDUCTION	
	ITION.\$	
	VECTOR VALUES ATTHD=\$ATTACHMENT NAMES.\$	
	FUNCTION RETURN	1670
	R	1680
	NORMAL MODE IS INTEGER	1690
	DIMENSION ATS(199)	1700
	BOOLEAN ALLIN	
	EQUIVALENCE (ALLIN,A(15872))	
	R	1710
	R	1720
	R ** STANDARD PROGRAM COMMON FOR CORE NO.2	1730
	PROGRAM COMMON A,B	1740
	DIMENSION A(15999,DIM1)	
	DIMENSION B(1999, PDIM)	
	R	1760
	R ***COMPILER CONTROL CARDS *****	1770
	R	1780
	R DEFINITION OF A(39,0)...A(39,399)	1790
	VECTOR VALUES A(15600)=\$ELTAPES	1800
	EQUIVALENCE (A(15601),PELCNT),(A(15602),TELCNT),	1810
1	(A(15603),PARCNT),(A(15604),ELN),	1820
2	(A(15700),FPAR),(A(15772),ZPAR),	1830
3	(FPAR(70),SCOPB1),(FPAR(71),SCOPB2),	1840
4	(A(15852),TAPEOK),(A(15842),CCNT),	1850
5	(A(15843),SCNT),(A(15844),ECNT),	1860
6	(A(15845),PCNT),(A(15847),FSCNT),	1870
7	(A(15848),ID1),(A(15849),ID2),	1880
8	(A(15850),IFERR1),(A(15851),PRLCNT)	1890
9	,(A(15853),DIM1)	1900
	EQUIVALENCE (CKRN,A(15856)),(THREE,A(15857)),(FOUR,A(15858)) CK	

1,	(FIVE,A(15889))	CK
	EQUIVALENCE (ATS,A(8600)), (PELT,A(8200))	
	DIMENSION ELN(95), FPAR(71), ZPAR(69)	
	DIMENSION PELT(299,DIM2)	
	BOOLEAN IFERR1, TAPEOK	1940
	BOOLEAN CLRN	
	VECTOR VALUES DIM1 = 2,401,400	1090
	VECTOR VALUES DIM2 = 2,6,5	
	VECTOR VALUES PDIM = 2,6,5	
	EQUIVALENCE (M,A(15864))	
	END OF FUNCTION	*

\$ COMPILE MAD,PUNCH OBJECT,PRINT OBJECT

ERR00000

```
EXTERNAL FUNCTION(Z)
R'N
ENTRY TO ERR.
ERRNO = Z
BOOLEAN RESTAR
RESTAR = 1B
EXECUTE OUTPUT.(DFRCOM,18)
VECTOR VALUES DFRCOM=$ERROR ENCOUNTERED. EOF,ERRNO ON TAPE TH
1REE. THREE AND FOUR REWOUND. EOF REMOVED FROM FIVE.SYSTEM EXE
2CUTED.$
END OF FILE TAPE THREE
WRITE BINARY TAPE THREE, ERRNO
CKOUT=BNBCD.(ERRNO)
EXECUTE OUTPUT.(CKOUT,3)
DIMENSION CKOUT(2)
VECTOR VALUES CKOUT(1)=$=ERRNO$
END OF FILE TAPE THREE
REWIND TAPE THREE
W'R RESTAR
REWIND TAPE FOUR
READ BINARY TAPE FOUR, IDENT
W'R IDENT .E. $ELTAPE$
BACKSPACE RECORD OF TAPE FOUR
WRITE BINARY TAPE FOUR,$RESTARRESTARRESTARRESTAR$
EXECUTESKFILE.(FOUR,1)
WRITE BINARY TAPE FOUR,$RESTARRESTARRESTARRESTARRESTAR$
E'L
E'L
SEQPGM.
PROGRAM COMMON A,B
DIMENSION A(15999),B(2000)
NORMAL MODE IS INTEGER
VECTOR VALUES THREE= 3
VECTOR VALUES FOUR = 4
VECTOR VALUES FIVE = 9
ERASABLE BLAST
END OF FUNCTION
```

*

\$	COMPILE MAD,EXECUTE,DUMP,PUNCH OBJECT	EQVGEN00	10
	EXTERNAL FUNCTION		
	R'N		
	ENTRY TO EQVGEN.		0280
	PRINT COMMENT \$ EQVGEN\$		
	R		290
	THROUGH LISTE, FOR L=0,1,L.GE. ECNT		
	PAR = A(15,L)		
	EXECUTE OUTPUT.(OCTDEC.(L),1)		0330
	EXECUTE OUTPUT.(PAR,1)		0340
	R		350
	OCCURS = 08		0360
	TEL = A(16,L)		
	TEID = A(17,L)		
	TAT = A(18,L)		
	TAID = A(19,L)		
	EXECUTE SRCHFP.		0410
	EXECUTE SRCHCN.		0420
LISTE	CONTINUE		0430
	FUNCTION RETURN		0440
	INTERNAL FUNCTION		0450
	ENTRY TO SRCHFP.		0460
	R		0470
	R CHECK LIST OF PARAMETERS TO SEE IF -PAR- IS ON LIST.		0480
F	THROUGHFPARVC, FOR I=0,1,I.E. PARCNT .OR. PAR .E. FPAR(I)		0490
PARVC	WHENEVER I.L. PARCNT, FUNCTION RETURN		0500
	VECTOR VALUES FERR = \$****THE PARAMETER \$,\$ \$,\$ IS NOT ON PAR		0510
	PARAMETER LIST\$		520
	FERR(3) = PAR		0530
	EXECUTE OUTPUT.(FERR,9)		0540
	TRANSFER TO LISTE		0550
	END OF FUNCTION		0560
	INTERNAL FUNCTION		0570
	R SEARCH BOTH SIDES OF CONNECTIONS LIST		0580
	ENTRY TO SRCHCN.		0590
	THROUGH CLIST, FOR J=0,1,J.GE. CCNT		
	WHENEVER (A34(J) .E. TAID .OR. TAID .E. \$ \$)		0610
	1.AND. (A33(J) .E. TAT .OR. TAT .E. \$ \$)		0620
	2.AND. (A32(J) .E. TEID .OR. TEID .E. \$ \$)		0630
	3.AND. (A31(J) .E. TEL .OR. TEL .E. \$ \$)		0640
	SIDE =0		650
	TRANSFER TO PRNT		0660
	OR WHENEVER (A38(J) .E. TAID .OR. TAID .E. \$ \$)		0670
	1.AND. (A37(J) .E. TAT .OR. TAT .E. \$ \$)		0680
	2.AND. (A36(J) .E. TEID .OR. TEID .E. \$ \$)		0690
	3.AND. (A35(J) .E. TEL .OR. TEL .E. \$ \$)		0700
	SIDE =1		710
PRNT			
	ENUM=BNBCD.(A(27+SIDE,J))		
	EQV(7) =OR.(ZPAR(I),ENUM)		
	EXECUTE OUTPUT.(OCTDEC.(J),1)		
	EQV (5) = A(14,L)		0740
	EXECUTE OUTPUT.(EQV,9)		
	PUNCH FORMAT FORMPU,EQV...EQV(8)		
	V'S FORMPU=\$9C6*\$		
	VECTOR VALUES EQV = \$ \$,\$ \$,\$EQUIVALENCE (\$,\$ \$,\$ \$,\$		0780
	1\$ \$,\$ \$)		0790
	OCCURS = 18		0800
	END OF CONDITIONAL		0810

CLIST

CONTINUE	830
WHENEVER .NOT. OCCURS, EXECUTE OUTPUT.(NO,10)	0820
FUNCTION RETURN	0840
VECTOR VALUES NO(4)=\$ DOES NOT OCCUR ON THE CONNECTION LIST\$	
EQUIVALENCE (TEL,NO),(TEID, NO(1)),(TAT,NO(2)),(TAID,NO(3))	0860
DIMENSION NO(10)	0870
END OF FUNCTION	0880
PROGRAM COMMON A	20
DIMENSION A(15999,DIM1),PCNT(1)	0030
VECTOR VALUES DIM1 = 2,401,400	040
NORMAL MODE IS INTEGER	0070
R DEFINITION OF A(39,0)...A(39,399)	0050
VECTOR VALUES A(15600)=\$ELTAPES	0060
EQUIVALENCE (A(15601),PELCNT),(A(15602),TELCNT),	0070
1 (A(15603),PARCNT),(A(15604),ELN),	0080
2 (A(15700),FPAR),(A(15772),ZPAR),	0090
3 (FPAR(70),SCOPB1),(FPAR(71),SCOPB2),	0100
4 (A(15852),TAPEOK),(A(15842),CCNT),	0110
5 (A(15843),SCNT),(A(15844),ECNT),	0120
6 (A(15845),PCNT),(A(15847),FSCNT),	0130
7 (A(15848),ID1),(A(15849),ID2),	0140
8 (A(15850),IFERR1),(A(15851),PRLCNT)	0170
DIMENSION PELT(300),ELN(95),FPAR(71),ZPAR(69)	0180
R EQUIVALENCE OUTPUT GENERATOR	0190
DIMENSION A31(1),A32(2),A33(1),A34(1),A35(1),A36(1),A37(1),	0200
1A38(1)	210
DIMENSION ACOL2(1), ACOL3(1)	0220
EQUIVALENCE (A(12400),A31),(A(12800),A32),(A(13200),A33),	0230
1(A(13600),A34),(A(14000),A35),(A(14400),A36),(A(14800),A37),	0240
1(A(15200),A38)	*250
BOOLEAN OCCURS	0260
R	270
END OF FUNCTION	*

\$ASSEMBLE,PUNCH OBJECT

INSRTC00

```
ENTRY INSRTC
ENTRY EXTRC
INSRTC SLN 1      RSLT = INSRTC.(WORD, J, CHAR)
          CLA      3,4      CHAR IS PLACED IN THE JTH POSITION OF WORD.
          STA CHAR
          SLT 1
EXTRC  SLN 1      RSLT = EXTRC. (WORD, J)
          SXD XRB,2      RSLT IS THE CHARACTER EXTRACTED FROM THE JTH
          CLA*      2,4
          PAX      0,2
          CAL*      1,4
          SLT 1
XRB     TXL IN1,0,0
          LDQ BLANKS
BACK1   TXH OUT1,2,5
          ARS 6
          TXI BACK1,2,1
OUT1    LGL 30
          SLW COMMON
          TXL EXIT,0,0
IN1     SLW COMMON
CHAR    LDQ **
          CAL MASK
          RQL 6
COUNT  TXH OUT2,2,5
          ALS 6
          TXI COUNT-1,2,1
OUT2    STQ COMMON-1
          COM
          ANS COMMON
          COM
          ANA COMMON-1
          ORS COMMON
EXIT    CLA COMMON
          LXD XRB,2
          TRA 3,4
BLANKS  BCD 1
MASK    OCT 000000000077
COMMON  SYN -1
LAST    BSS 0
END
```

*

\$ASSEMBLE,PUNCH OBJECT

LOGIC.00

```

ENTRY AND
ENTRY NOT
ENTRY OR
ENTRY XOR
ENTRY THEN
ENTRY EQV
HOOK  CAL*      2,4
      SLW      C-2
HOOK1  CAL*      1,4
      SLW C-1
      TRA 1,2
EQV    TSX HOOK,2
      ORA C-2      W1.OR.W2
      COM          .NOT. (W1.OR.W2)
      SLW C
      CAL C-1
      ANA C-2      W1.AND.W2
      ORS C        (W1.AND.W2).OR.(.NOT.(W1.OR.W2))
      TRA R2
THEN   TSX HOOK,2
      COM          .NOT.W1
      ORA C-2      .NOT.W1.OR.W2
      TRA R1
XOR    TSX HOOK,2
      ANA C-2      W1.AND.W2
      COM          .NOT.(W1.AND.W2)
      SLW C
      CAL C-1
      ORA C-2      (W1.OR.W2)
      ANS C        (W1.OR.W2).AND.(.NOT.(W1.AND.W2))
      TRA R2
NOT    TSX HOOK1,2
      COM          NOT.W1
      TRA R1
OR     TSX HOOK,2
      ORA C-2      W1.OR.W2
      TRA R1
AND    TSX HOOK,2
      ANA C-2      W1 AND W2
R1     SLW C
R2     CLA C
      TRA 2,4
LAST   BSS C
C      SYN -1
      END

```

*

\$ASSEMBLE,PUNCH OBJECT
ENTRY LOCATE

LOCATE00

LOCATE CLA* 3,4 LOCATE.(WORD, LIST, COUNT)

ADD ONE

PAX 0,2

CLA 2,4

STA COMP2

ADD ONE

STA COMP

CLA* 1,4

COMP2 CAS **

TRA COMP

TRA FIRST

COMP CAS **,2

TIX *-1,2,1

TRA EQUAL

TIX *-3,2,1

EQUAL PXA 0,2

SUB ONE

TNZ 4,4

SUB ONE

TRA 4,4

FIRST CLA 7FRO

TRA 4,4

ZERO PZE 0

TRA 4,4

ONE PZE 1

END

LIST ADDRESS PLUS ONE

WORD

WORD FOUND AT THIS LOCATION PLUS 1
IF WORD NOT ON LIST ACCUMULAROR WILL BE NEG

*

```

SCOMPILE MAD,PUNCH OBJECT,PRINT OBJECT
      EXTERNAL FUNCTION(HEAD, LIST, COUNT)
      ENTRY TO LPRINT.
      K= 12
      EXECUTE OUTPUT.(HEAD,1)
      THROUGH A, FOR I=0,12, I.GE. COUNT
      WHENEVER COUNT-I.LE. 12, K= COUNT-I
A     EXECUTE OUTPUT.(LIST(I),K)
      FUNCTION RETURN
      ENTRY TO DPRINT.
      K=12
      EXECUTE OUTPUT.(HEAD,1)
      THROUGH B, FOR I=0,12, I.GE. COUNT
      WHENEVERCOUNT-I.LE. 12, K= COUNT -I
      THROUGH C, FOR J=I,1, J.GE. K
C     LINE(J) = LIST(J)
B     EXECUTE OUTPUT.(LINE(I),K)
      FUNCTION RETURN
      NORMAL MODE IS INTEGER
      DIMENSION LINE(12)
      PROGRAM COMMON A
      DIMENSION A(15999)
      EQUIVALENCE (A(15858), FOUR),(A(15601), PELCNT),
1      (A(15602),TELCNT),(A(15603),PARCNT)
      EQUIVALENCE(A(15700),FPAR),(A(15772),ZPAR )
1,(A(15604),ELN)
      END OF FUNCTION

```

LPRINT.0

*

\$ASSEMBLE,PUNCH OBJECT
ENTRY OCTDEC

OCTDEC00

CK
CK

D NOP
OCTDEC CLA* 1,4
TNZ **3
CLA BZP
TRA 2,4
SLW OCTINT
SXA IRA,1
TMI MI
CLA BZP
TRA STO
MI CLA BZM
STO STO WORD
CLA OCTINT
SUB =9999
TMI **3
CLA LARGE
TRA 2,4
LXD THREE,1
SLN 1
BACK CLA TEN
LDQ OCTINT
TLQ FINI
PXD 0,0
DVP TEN
STQ OCTINT
BACK2 ORS WORD
LDQ WORD
RQL 30
STQ WORD
TIX BACK,1,1
FINI2 LDQ WORD
TXH RTN,1,3
RQL 6
TXI *-2,1,1
RTN STQ WORD
CLA WORD
IRA AXT **,1
TRA 2,4
FINI SLT 1
THREE TXL FINI2,0,3
CLA OCTINT
TXI BACK2,1,1
WORD SYN -1
OCTINT SYN -2
TEN PZE 10
LARGE BCD 1*LARGE
BZP BCD 1 + 000
BZM BCD 1 - 000
LAST BSS 0
END

CORRECTION TO OCTDEC

*

\$ASSEMBLE,PUNCH OBJECT

OCT.0000

```
ENTRY OCT
OCT  CLA      1,4
      SXA      IRA,1
      STA L1
      CLA 2,4
      STA S1
      SSP
      SUB CNE
      STA CUT
L1    LDQ **
      SLN 1
S1    STO **
      LXA A=6,1
T1    ALS 3
      LGL 3
      TIX T1,1,1
      SLT 1
      TRA *+2
      TRA S1
OUT   STO **
IRA   AXT      **,1
      TRA 3,4
A=6   PZE 6,0,0
ONE   PZE 1
LAST  BSS 0
      END
```

*

\$ASSEMBLE,PUNCH OBJECT			OUTPUT00	
	ENTRY OUTPUT			730
	ENTRY OUTPT			
	ENTRY OUT		CK	
OUTPT	NOP			
OUT	NOP		CK	
OUTPUT	CLA 1,4	EXECUTE OUTPUT.(NAME,COUNT)		740
	STA OT			750
	LDQ =20			
	CLA* 2,4			760
	TLQ THEAXE			
XRC	PAX 0,3			770
	TXI *+1,1,1			
	SXD LG,1		CK	
	TXI *+1,1,C	ADD.OF C IS ADD. OF D MINUS ONE	CK	
	SXA ST,1			810
	AXT 0,1			820
OT	CLA **,1		CK	
ST	STO **,2	PICK UP FROM NAME REGION,STORE IN D BUFFER-FORWARD	CK	
	TXI *+1,1,1			850
	TIX OT,2,1			860
	SXA XRC,4			870
	CALL SPRINT			880
LG	BLK C,**,		CK	
	LXA XRC,4			900
	TRA 3,4			910
THEAXE	CALL SPRINT			
	BLK PAGE,,N			
	CALL SYSTEM			
ONE	PZE 1			
N	PZE 0,,0			
	CLA COUNT	LINES OF OUTPUT		
	SUB ONE			
	TMI THEAXE			
	STO COUNT			
COUNT	PZE 250			
PAGE	BCD 91	MAXIMUM OF 250 LINES OF OUTPUT PER JOB EXCEEDED. SOR		
	BCD 5RRY	MAC...YOU ARE BEING CUT OFF.		
C	BCD 10			920
D	BSS 20			930
	END			*

\$COMPILE MAD,PRINT OBJECT,PUNCH OBJECT

OUTPT100

EXTERNAL FUNCTION(X)

ENTRY TO OUTPT1.

EXECUTE OUTPUT.(CKH,2)

CK

VECTOR VALUES CKH=\$*OUTPT1. SUBROUTINES

CK

R (**ARGUMENT X IS NUMBER OF COLUMNS TO BE PUNCHED.)

XX=X

WHENEVER EXTRC.(BFR(1),5) .F. SR\$, FUNCTION RETURN

EXECUTE PNCH12.(BFR1)

EXECUTE OUTPUT.(BFR1,12)

WCNT= 11

WORDS = (XX-1)/6

THRO U GH LIST, FOR J=1,1, WCNT.GE. WORDS .OR. J.G. 9

BFR1(WCNT+2)= INSRTC.(BFR1(WCNT+2),5, NUM(J))

VECTOR VA LU ES NUM=\$1\$, \$2\$, \$3\$, \$4\$, \$5\$, \$6\$, \$7\$, \$8\$, \$9\$

EXECUTE OUTPUT.(BFR1(WCNT+1),12)

EXECUTE PNCH12.(BFR1(WCNT+1))

LIST

WCNT = WCNT + 12

FUNCTION RETURN

DIMENSION A(15999)

PROGRAM COMMON A

NORMAL MODE IS INTEGER

EQUIVALENCE (A(5320),BFR1),(A(5200),BFR)

DIMENSION BFR1(120),BFR(120)

EQUIVALENCE (T(265),CCA(0))

EQUIVALENCE (CCA,J2)

EQUIVALENCE (T(0),A(15600))

END OF FUNCTION

*

```

$COMPILE MAD,PRINT OBJECT,PUNCH OBJECT                                PELTOT00
EXTERNAL FUNCTION (COUNT)
ENTRY TO PELTOT.
EXECUTE OUTPUT. ($PELTOT$,1)
K=0
NORMAL MODE IS INTEGER
EXECUTE OUTPUT.(PRIN1,4)
VECTOR VALUES PRIN1 = $ PRESENT ELEMENT TABLE $
P CONTINUE
EXECUTE OUTPUT.(PRIN2,6)
VECTOR VALUES PRIN2 = $ ELN FSTAT NOATS FSTSC NOCOL $
TRANSFER TO S(K)
S(0) CONTINUE
T'H PRIN6, FORI=0,1,I.G. COUNT
OUTPUT.(PELT(I,0),5)
PRIN6 CONTINUE
FUNCTION RETURN
ENTRY TO PELTLN.
EXECUTE OUTPUT.($PELTLN$,1)
K=1
I= COUNT
TRANSFER TO P
S(1) CONTINUE
THROUGH PRIN4, FOR I = 0,1,I.G.PRLCNT
EXECUTE OUTPUT.(PELT(I*5),5)
PRIN4 CONTINUE
FUNCTION RETURN
ENTRY TO ELNOUT.
EXECUTE OUTPUT.($ELNOUT$,1)
EXECUTE OUTPUT.(PRIN3,3)
VECTOR VALUES PRIN3 = $ ELEMENT NAMES $
EXECUTE OUTPUT.(ELN(0),COUNT)
FUNCTION RETURN
PROGRAM COMMON A,B
EQUIVALENCE ( A(15601),PELCNT ),( A(15602),TELCNT ),           1810
1      ( A(15603),PARCNT ),( A(15604),ELN ),                 1820
2      ( A(15700),FPAR ),( A(15772),ZPAR ),                 1830
3      ( FPAR(70),SCOPB1 ),( FPAR(71),SCOPB2 ),             1840
4      ( A(15852),TAPEOK ),( A(15842),CCNT ),               1850
5      ( A(15843),SCNT ),( A(15844),ECNT ),                 1860
6      ( A(15845),PCNT ),( A(15847),FSCNT ),               1870
7      ( A(15848),ID1 ),( A(15849),ID2 ),                 1880
8      ( A(15850),IFERR1 ),( A(15851),PRLCNT )             1890
9      ,( A(15853),DIM1)                                     1900
EQUIVALENCE (CKRN,A(15856)),(THREE,A(15857)),(FOUR,A(15858)) CK
1,      (FIVE,A(15889))                                     CK
EQUIVALENCE (ATS,A(8600)), (PELT,A(8200))
DIMENSION A(15999),PELT(400),ELN(95)
DIMENSION B(1999,PDIM2)
VECTOR VALUES PDIM2 = 2,6,5
EQUIVALENCE(PELT,A(8200)),(ELN,A(15604))
EQUIVALENCE (M,A(15864))
END OF FUNCTION
$ASSEMBLE,PUNCH OBJECT                                PLACE000
ENTRY PLACE CALLING SEQUENCE EXECUTE PLACE.(A(N),L,$B$)
PLACE NOP
IN CLA 1,4
ADD IN
STA BASE

```

	CLA*	2,4	
	PAX	0,1	
	CAL*	3,4	SIX CHARACTER WORD TO BE STORED IN
BASE	SLW	**,1	A(N),...,A(L)
	TIX	BASE,1,1	
	TRA	4,4	
	END		

*

SCOMPILE MAD,EXECUTE,DUMP,PUNCH OBJECT ,PRINT OBJECT
EXTERNAL FUNCTION
ENTRY TO PSCAN.

PSCAN000

M=0

CARDS = 0

READ READ BINARY TAPE UNIT, BFR(M),...,BFR(M+11)
CHAR= EXTRC.(BFR(M+1),5)
WHENEVER CHAR.E. \$R\$

DOLLAR

PUNCH FORMAT FORMPU,BFR(M)...BFR(M+11)
V'S FORMPU = \$ 12C6*\$
TRANSFER TO READ
O'R CHAR.E.\$\$.OR.(CHAR.GE.\$0\$.AND.CHAR.LE.\$9\$)
COL1=EXTRC.(BFR(M),1)
W'R COL1.E. -\$=\$, T'O DOLLAR
WHENEVER COL1.E.\$+\$,OR. COL1.E.\$-\$
BFR(M)= INSRTC.(BFR(M),1,\$ \$)
WHENEVER COL1.E. \$+\$
PLSFLG= 1B
OTHERWISE
MINFLG= 1B
END OF CONDITIONAL

PUNCH FORMAT FORMPU,BFR(M)...BFR(M+11)
T'O SCANS
OTHERWISE
PUNCH FORMAT FORMPU,BFR(M)...BFR(M+11)
T'O READ

E'L
OTHERWISE
PRINT COMMENT \$ THIS CARD NOT PUNCHED \$
OUTPUT.(BFR(M),12)
E'L

SCANS

FUNCTION RETURN
PROGRAM COMMON A,B
D'N DIM1(2)
DIMENSION A(15999,DIM1),PELT(299,PDIM),T(300),ATTNO(50)
1,BFR(200),FPAR(71),ZPAR(69),B(1999 ,PDIM),PCNT(1)
EQUIVALENCE (T,A(15600)),(A(6600),BFR),(T(276),RCNO)
1,(T(3),PNUMBR),(T(245),IPCNT),(T(246),DRPCNT)
2,(T(247),FSCNT),(T(253),DIM1),(T(256),ERRNO)
3,(T(257),THREE),(T(258),FOUR),(T(263),FSUBIN)
4,(T(264),M),(T(265),CCA),(T(266),CARDS)
5,(T(267),MA),(T(268),EPFLG),(T(269),EPCNT)
6,(T(270),MCOUNT),(T(271),REM),(T(272),BOOLN)
EQUIVALENCE(T(273),UNIT), (T(294),LOCLBL)
7,(T(274),PXSIT),(T(275),LINE),(T(277),J)
8,(T(278),I3),(T(279),I2),(T(284),LBWORD)
9,(T(287),MINFLG),(T(288),PLSFLG),(T(289),FIVE)
1,(T(290),SLBFLG),(T(291),FINFLG),(T(292),J1)
6,(A(7100),ATTNO),(A(8200),PELT)
NORMAL MODE IS INTEGER
BOOLEAN IFERR1, CKRN, FUBSIN,EPFLG, MCOUNT, BOOLN
1,ONCEB, ONCEBB,MINFLG,PLSFLG,SLBFLG,FINFLG,EOFBLN,SCANDL
BOOLEAN ELFLG, CCOUNT
VECTOR VALUES PDIM=2,6,5
EQUIVALENCE (CCA,J2),(T(260),EOFBLN),(RCNO,RCNO)
EQUIVALENCE(PARCNT,PNUMBR)
END OF FUNCTION

*

\$ASSEMBLE,PUNCH OBJECT,PRINT OBJECT

SEARCH00

* SEARCH SIMULATOR CORE4

BOYD NOV 64

* CALLED BY SSCAN1

* ROUTINE TO SEARCH A LIST LOOKING FOR A SPECIFIC ENTRY

* RETURNS IN AC LIST SUBSCRIPT WHERE ENTRY IS FOUND

* IF THE SAME ENTRY OCCURS MORE THAN ONCE ONLY THE LAST WILL BE FOUN

*

* ARGUMENTS

* 1,4 ZERO LOCATION OF LIST

* 2,4 LAST ENTRY ON LIST

* 3,4 WORD TO BE FOUND

* 4,4 EXIT IF ENTRY NOT FOUND

*

* CALLED FROM SSCAN1 TO SCAN FPAR LIST FOR VARIABLE NAMES, SO THAT

* THE UNIQUE THREE CHARACTER NAME CAN BE FOUND ON THE ZPAR LIST

*

ENTRY SEARCH

SEARCH CLA 1,4 ZERO LOCATION

STA CAS

CLA* 2,4 LAST ENTRY

PAX ,1

CLA* 3,4

CAS CAS **,1

TRA **2

TRA OUT WORD FOUND

TIX CAS,1,1 LOOP TO SEARCH BACKWARDS

AXT 0,1

CAS* CAS CHECK ZERO LOCATION

TRA* 4,4 WORD NOT FOUND, ERROR RETURN

TRA **2

TRA* 4,4 WORD NOT FOUND, ERROR RETURN

OUT PXA ,1 WORD FOUND, PUT INDE IN AC

TRA 5,4 NORMAL RETURN

END

*

SCMPPILE MAD, PRINT OBJECT, PUNCH OBJECT
EXTERNAL FUNCTION
ENTRY TO SNOW.

SNOW000

R ROUTINE TO RECORD THE FIRST OCCURENCE AND FUTURE
R OCCURENCES OF ALL ATTACHMENTS

N'S INTEGER
P'R F(200)
D'N CONL(F),CONR(F),BROAD(F)
PROGRAM COMMON A(15999),B(1999),BROAD(F)

R *****

DEFINE UNARY OPERATOR .M., PRECEDENCE SAME AS .E.
MODE STRUCTURE 2 = .M. 1
CLA B
TMI LOC+3
PXA ,0
TRA LOC+2
CLA =1
OUT AC
END

DEFINE UNARY OPERATOR .STZ., PRECEDENCE SAME AS =
MODE STRUCTURE .STZ. 1
STZ B
OUT Z
END

MODE STRUCTURE .STZ.2, SAME SEQUENCE AS .STZ.1

R *****

R CCONT = THE NO. OF LINES IN THE CONNECTIONS MATRIX
R CONL = LEFT HALF OF THE CONNECTIONS MATRIX
R CONR = RIGHT HALF OF THE CONNECTIONS MATRIX
R BROAD = THE END RESULT OF THIS ROUTINE

.STZ. BROAD
T'H SEEK, FOR Q = 1, 1, Q .GE. CCONT
QP = Q - 1
FIRST = BLB.(SUCH,CONL(Q))
SECOND = BLB.(NOISE,CONR(Q))
BOOLEAN NOISE, SUCH
W'R FIRST .L. SECOND

R SAVE THE SMALLEST SUBSCRIPT AND THEN REPLACE THE
R OCCURENCES OF THE LARGER, IF THERE ARE ANY, BY THE
R SMALLER IN THE BROAD ARRAY

BROAD(Q) = FIRST
W'R NOISE .AND. (SECOND .NE. Q), FAKED.(SECOND)
O'R SECOND .L. FIRST
BROAD(Q) = SECOND
W'R SUCH .AND. (FIRST .NE. Q), FAKED.(FIRST)
C'E

BROAD(Q) = FIRST
E'L

SEEK

F'N
INTERNAL FUNCTION(NUMB)
E'O FAKED.

R FUNCTION TO FIND AND REPLACE THE OCCURENCES OF A
R NUMBER

(QP=QP,-1, QP .L. 0, QP = SEARCH.(BROAD,QP,NUMB,SEEK),BROAD
1(QP) = FIRST)

T'O SEEK

E'N

EQUIVALENCE (CONR,A(6000)),(CONL,A(5600)),(CCNT, A(15842))

EQUIVALENCE (CCONT,CCNT)

INTERNAL FUNCTION (MUNCH,CON)

E'O BLB.

W'R .M. CON

MUNCH = 1B

BOOLEAN MUNCH

F'N BROAD(SEARCH.(CONR,QP,CON,NEWL))

C'E

.STZ. MUNCH

SAME

F'N Q

E'L

NEWL

F'N BROAD(SEARCH.(CONL,QP,CON,SAME))

E'N

E'N

*

SCOMPILE MAD, PRINT OBJECT,PUNCH OBJECT

SQUEEZ000

EXTERNAL FUNCTION

R'N

E'0 SQUEEZ.

R FUNCTION TO COMPRESS CONECTIONS MATRIX

ROPERATOR TO STORE SUBSCRIPTS INTO A 9 BIT PACKAGE IN THE
RPRODUT VECTOR

R(ARRANGED AS FOLLOWS ... /ATT ID/ ELM ID/ ATT/ ELN/)

DEFINE UNARY OPERATOR .SQUEEZ., PRECEDENCE HIGHER THAN =
MODE STRUCTURE .SQUEEZ.1

PXA ,0
AXT 3,4
LGL 9
ADD =SUB+1,4
TIX LOC-2,4,1
STO B
OUT Z
END

IQ = 0

ID1 = \$\$

LONG=1

SQUEAK.(LFT ELM, LFTATT,L ELM ID,L ATT ID, CON L)

SQUEAK.(RIT ELM,RIT ATT, R ELM ID, R ATT ID, CON R)

F'N

INTERNAL FUNCTION (ELEMNT,ATMNT,ELM ID,ATT ID, PRODUT)

E'0 SQUEAK.

T'H HALF, FOR I=0,1, I .GE. CCNT

SUB = SEARCH.(ELMNO,NUMBER-1,ELEMNT(I),MISING(IQ))

R SUB = SUBSCRIPT OF ELEMENT IN ELN MATRIX

BEGIN=SUB*5+1

SUB(2) = SEARCH.(ATTMNT(PELT(BEGIN)),PELT(BEGIN+1)-1,ATMNT(I)
1,MISING(IQ))

R SUB(2) = SUBSCRIPT OF ATTACHMENT IN ATS MATRIX

SUB(1) = SEARCH.(ID1,LONG-1,ELM ID(I),ADD)

R SUB(1) = SUBSCRIPT OF ELEMNT IDENTIFER IN

R THE UNIQUE LIST OF IDENTIFERS

R UNIQUE LIST OF IDENTIFIERS

GO ON

SUBR=SEARCH.(ID1,LONG-1,ATT 'D(I), ADD1)

R SUBR= SUBSCRIPT OF ATTACHMENT IN UNIQUE LIST OF

R IDENTIFIERS

QUIK

.SQUEEZ. PRODUT(I)

W'R SUBR .G. 0, PRODUT(I)= - PRODUT(I)

HALF

```

F'N
ADD      R      TO HERE IF THE ELM ID NOT YET ON THE LIST
          ID1(LONG) = ELM ID(I)
          SUB(1)=LONG
          LONG = LONG + 1
          T'O GO ON

ADD1     R      TO HERE IF THE ATT ID NOT YET ON THE LIST
          ID1(LONG) = ATT ID(I)
          SUBR=LONG
          LONG = LONG + 1
          T'O QUIK

MISING   P'IT $H86***** ERROR -- COULD NOT FIND ONE OF THE FOLLOWING N
          1AMES/LINE(S)8*$
          IQ = 1

R        TO HERE IF THE ELEMENT IS NOT IN THE ELN OR PELT
R        VECTORS
MISING(1) P'IT $1H C6,5H, OR C6*$,ELEMNT(I),ATMNT(I)
          PRODUT(I) = 0
          T'O HALF
          E'N

          N'S INTEGER
          D'N SUB(2)
          PROGRAM COMMON A(15999)
          EQUIVALENCE (LFTELM,A(12400)),(LFTATT,A(13200)),
1          (LELMID,A(12800)),(LATTID,A(13600)),(RITELM,A(14000))
2          ,(RITATT,A(14800)),(RELMID,A(14400)),(RATTID,A(15200))
3          ,(CCNT,A(15842)),(ELMNO,A(15604)),(NUMBER,A(15851)),
4          (ATTMNT,A(8600)),(PELT,A(8200))
5          ,(CONL,A(5600)),(CONR,A(6000))
          PARAMETER L(399),N(95),M(199),R(299),U(399)
          D'N LFT ELM(L),LFT ATT(L), L ELM ID(L), L ATT ID(L),
1RIT ELM(L), RIT ATT(L), R ELM ID(L), R ATT ID(L)
          D'N ELMNO(N), ATTMNT(M), PELT(R), ID1(U)
          DIMENSION CONL(L),CONR(L)
          E'N
          *
$COMPILE MAD,PUNCH OBJECT,PRINT OBJECT          SSORDR00
          EXTERNAL FUNCTION
          R'N
          ENTRY TO SSORDR.
          PRLCNT=0
          ELCNT=PELCNT+TELCNT-1
          EXECUTE ELNOUT.(ELCNT+1)
          R          ELN,LINENO ARE SCRATCH          0050
          T'H LOOPHH,FOR VALUES OF RL = 31,35
          THROUGH LOOPH, FOR I=0,1,I.GE. CCNT
          ELEMI = A(RL,I)
          THROUGH LOOPI,FOR J=0,1,J.G.ELCNT          0070
          W'R ELEMI.E.ELN(J),T'O LOOPH
          VECTOR VALUES ERR2=$ ELEMENT DESCRIPTION NOT AVAILABLE $,$ $          0190
          ERR2(6)=ELEMI
          EXECUTE OUTPUT.(ERR2,7)
          IFERR1=1B
          CONTINUE
          LOOPH          CONTINUE
          LOOPHH          CONTINUE
          PRLCNT=PELCNT+TELCNT

```

	THROUGH LOOPA, FOR VALUES OF RL=31,35	0410
	RLMOD = RL/35	
	TELID=0	420
	THROUGH LOOPB, FOR J=0,1, J.GE. PRLCNT	
	LNO=0	440
	ELID=0	450
	ELNJ = PELT(5*J)	
	THROUGH LOOPC, FOR I=0,1,I.GE. CCNT	
	WHENEVER A(RL,I).NE.ELNJ, TRANSFER TO LOOPC	
	LINENO(LNO)=I	0480
	LNO=LNO+1	0490
LOOPD	EIDA = A(RL+1,I)	
	THROUGH LOOPD, FOR K=0,1,K.GE.ELID.OR.EIDA.E. EID(K)	
	WHENEVER K.GE. ELID	
	EID(ELID) = EIDA	
	ELID=ELID+1	0530
	END OF CONDITIONAL	0540
LOOPC	CONTINUE	550
	RL1= RL+1	
	THROUGH LOOPG, FOR I1=TELID, I1.GE. TELID+LNO	
	THROUGH LOOPG, FOR I2=0,1, I2.GE. ELID	
	EID1= EID(I2)	
	THROUGH LOOPG, FOR I3=0,1, I3 .GE. LNO	
	WHENEVER A(RL1, LINENO(I3)).NE. EID1, TRANSFER TO LOOPG	
	A(27+ RLMOD*I1) = LINENO(I3)	
	I1=I1+1	610
LOOPG	CONTINUE	0620
LOOPB	TELID=I1	0630
LOOPA	CONTINUE	0640
R	** R AND L ARE ORDERED	0650
	THROUGH LOOPZ, FOR VALUES OF RL=31,35	0660
	RLMOD = RL/35	
	THROUGH LOOPZ, FOR I=0,1,I.GE. CCNT	
	SUB1 = A(27+RLMOD,I)	
	EL1=A(RL,SUB1)	0690
	ID1=A(RL+1,SUB1)	0700
	THROUGH LOOPW, FOR J=0,1,J.GE. CCNT	
	SUB2=A(28-RLMOD,J)	
	EL2 = A(35-4*RLMOD,SUB2)	
	ID2 = A(36-4*RLMOD,SUB2)	
	WHENEVER EL1.E.EL2.AND.ID1.E.ID2	0750
OUT1	A(29+RLMOD,I) = J	
	TRANSFER TO LOOPZ	0770
	END OF CONDITIONAL	0780
LOOPW	CONTINUE	0790
	THROUGH LOOPV, FOR J=0,1, J.GE. CCNT	
	SUB3 = A(27 +RLMOD,J)	
	EL3=A(RL,SUB3)	0820
	ID3=A(RL+1,SUB3)	0830
	WHENEVER EL1.E.EL3.AND.ID1.E.ID3	0840
	J=-J	0850
	TRANSFER TO OUT1	0860
	END OF CONDITIONAL	0870
LOOPV	CONTINUE	0880
LOOPZ	CONTINUE	0890
	FUNCTION RETURN	0900
	EQUIVALENCE (A(8400),LINENO),(A(8800),EID)	0910
	DIMENSION EID(400),LINENO(400)	0920
R		930

R		940
R		950
R		960
R	***COMPILER CONTROL CARDS *****	0970
R		980
R	DEFINITION OF A(39,0)...A(39,399)	0990
	VECTOR VALUES A(15600)=\$ELTAPES	1000
	EQUIVALENCE (A(15601),PELCNT),(A(15602),TELCNT),	1010
1	(A(15603),PARCNT),(A(15604),ELN),	1020
2	(A(15700),FPAR),(A(15772),ZPAR),	1030
3	(FPAR(70),SCOPB1),(FPAR(71),SCOPB2),	1040
4	(A(15852),TAPEOK),(A(15842),CCNT),	1050
5	(A(15843),SCNT),(A(15844),ECNT),	1060
6	(A(15845),PCNT),(A(15847),FSCNT),	1070
8	(A(15850),IFERR1),(A(15851),PRLCNT)	1090
9	,(A(15853),DIM1)	1100
	DIMENSION PELT(299,DIM2),ELN(95),FPAR(71),ZPAR(69)	
	V'S DIM2 = 2,6,5	
	EQUIVALENCE (ATS,A(8600)), (PELT,A(8200))	1130
	VECTOR VALUES DIM1 = 2,401,400	1140
	NORMAL MODE IS INTEGER	1150
	DIMENSION A(15999,DIM1),PCNT(1)	1160
	BOOLEAN IFERR1, TAPEOK	1170
	PROGRAM COMMON A	1180
R		1190
R		1200
	E'N	*

\$ASSEMBLE,PUNCH OBJECT

TAPE1000

	ENTRY	SKFILE
	ENTRY	SKRECD
	ENTRY	BKSPCE
SKFILE	AXT	0,1
	TRA	START1
SKRECD	AXT	1,1
START1	CLA*	2,4
	TZE	3,4
	TMI	BACKS
	STA	BLK1
	CLA	TYPE,1
	STA	CALL
	CLA*	1,4
	PAX	0,1
	SXD	BLK1,1
	SXA	XRC1,4
CALL	TSX	**,4
BLK1	BLK	**,,**
XRC1	AXT	**,4
	TRA	3,4
	CALL	SKPREC
TYPE	CALL	SKPFIL
BKSPCE	AXT	1,1
	CLA*	2,4
	TZE	3,4
BACKS	PAX	0,2
	CLA	TYPE2,1
	STA	CALL2
	CLA*	1,4
	PAX	0,1
	SXD	BLK2,1
	SXA	XRC2,4
CALL2	TSX	**,4
BLK2	BLK	**,,**
	TIX	*-2,2,1
XRC2	AXT	**,4
	TRA	3,4
	CALL	BSRTAP
TYPE2	CALL	BSFTAP
	END	

*

\$ASSEMBLE,PUNCH OBJECT

TEMPC200

```
ENTRY IDSET
ENTRY PNCH
ENTRY PNCH12
PNCH NOP
PNCH12 NOP
IDSET CLA CK
      TZE 1,4
      SXA IRC,4
      CALL SPRINT
      BLK WORD,7
      STZ CK
IRC AXT 0,4
   TRA 1,4
WORD BCD 78( PNCH,PNCH12, IDSET ARE DUMMY ROUTINES.)
QPQ CALL SPRINT
    BLK ERR,8
    CALL SYSTEM
ERR BCD 78(QPQSUBROUTINE ILLEGALLY ENTERED.SYSTEM EXEQ.)
   CK PZE
      END
```

*

\$ASSEMBLE,PUNCH OBJECT
ENTRY SYSTEM,ERROR

SYS/ER00

SYSTEM NOP

ERROR PXA 0,4

PAC 0,4

SXA IN,4

PRINT FORMAT,IN,0

CALL PANEL

8CALL SDUMP,10000,....,77777

CALL SELRCD

BLK 1,2

IN PZE

FORMAT BCI *.H*6SYSTEM/ERROR CALLED FROM *.K5*

END

*

\$COMPILE MAD,EXECUTE, PRINT OBJECT,PUNCH OBJECT,DUMP,I/O DUMP CORE3000CORE3000
R SYSTEM SIMULATOR - CORE 3.

R'N

RDEFINED OPERATORS.

DEFINE BINARY OPERATOR .NID., PRECEDENCE SAME AS .NE.

MODE STRUCTURE 2 = 1 .NID. 1

JMP *+1,AC,*+2
STO T
JMP *+1,MQ,*+2
STQ T
JMP *+1,LA,*+2
SLW T
CAL A
LAS B
TRA LOC+2
TRA LOC+3
CLA =18
TRA LOC+2
CLA =0B
OUT AC
END

ROPERATOR TO TEST FOR PARAMETER BITS.

DEFINE BINARY OPERATOR .FPAR., PRECEDENCE SAME AS .E.

MODE STRUCTURE 2=1 .FPAR. 1

JMP *+1,AC,*+2
STO T
JMP *+1,MQ,*+2
STQ T
JMP *+1,LA,*+2
SLW T
CAL B
ARS 2
ORA A
TZE LOC+2
CLA =18
OUT AC
END

DEFINE UNARY OPERATOR .ELPOS., PRECEDENCE SAME AS .ABS.

MODE STRUCTURE 1= .ELPOS. 1

JMP *+1,AC,*+2
STO T
JMP *+1,MQ,*+2
STQ T
CAL B
ANA =777K
OUT AC
END

DEFINE UNARY OPERATOR .EIDPOS., PRECEDENCE SAME AS .ELPOS.

MODE STRUCTURE 1= .EIDPOS. 1

JMP *+1,AC,*+2
STO T
JMP *+1,MQ,*+2
STQ T
CAL B
ANA =777K3
ARS 9
OUT AC
END

DEFINE UNARY OPERATOR .ATPOS., PRECEDENCE SAME AS .ELPOS.

```

MODE STRUCTURE 1= .ATPOS. 1
JMP    **1,AC,**2
STO     T
JMP     **1,MQ,**2
STQ     T
CAL     B
ANA     =777K6
ARS     18
OUT     AC
END

DEFINE UNARY OPERATOR .ANEG., PRECEDENCE SAME AS .ABS.
MODE STRUCTURE 2= .ANEG. 1
JMP     **1,AC,**2
STO     T
JMP     **1,MQ,**2
STQ     T
CLA     B
TMI     LOC+3
CLA     =0B
TRA     LOC+2
CLA     =1B
OUT     AC
FND

DEFINE BINARY OPERATOR .COUNT., PRECEDENCE SAME AS *
MODE STRUCTURE 1=1 .COUNT. 1
JMP     **1,AC,**2
STO     T
JMP     **1,MQ,**2
STQ     T
JMP     **1,LA,**2
SLW     T
CLA     A
STO     =Z1
CLA     B
STO     =Z2
AXT     0,1
AXT     36,2
CAL     =Z1
TSX     LOC+5,4
CAL     =Z2
ARS     2
AXT     34,2
AXC     LOC+6,4
LBT
TRA     LOC+2
TXI     LOC+1,1,1
ARS     1
TIX     LOC-4,2,1
TRA     1,4
PXA     0,1
OUT     AC
END

DEFINE BINARY OPERATOR .IND., PRECEDENCE SAME AS +
MODE STRUCTURE 1= 1 .IND. 1
JMP     **1,AC,**2
STO     T
JMP     **1,MQ,**2
STQ     T
JMP     **1,LA,**2

```

```

SLW      T
CLA      A
STO      =Z1
CLA      B
PAX      0,4
SXD      =Z1,4
CLA      =Z1
OUT      AC
END

```

```

DEFINE UNARY OPERATOR .DECR., PRECEDENCE SAME AS .ABS.
MODE STRUCTURE 1= .DECR. 1

```

```

JMP      *+1,AC,*+2
STO      T
JMP      *+1,MQ,*+2
STQ      T
JMP      *+1,LA,*+2
SLW      T
CLA      B
PDX      0,4
PXA      0,4
OUT      AC
END

```

```

DEFINE UNARY OPERATOR .ADR., PRECEDENCE SAME AS .ABS.
MODE STRUCTURE 1= .ADR. 1

```

```

JMP      *+1,AC,*+2
STO      T
JMP      *+1,MQ,*+2
STQ      T
JMP      *+1,LA,*+2
SLW      T
CAL      B
PAX      0,4
PXA      0,4
OUT      AC
END

```

RINITIALIZATIONS.

```

PRINT COMMENT $ICORE 3$
AIDS(0)=777777777777K
A(CONPNT+CCNT)=777777777777K
A(CONPNT(1)+CCNT)=777777777777K
P'S CCNT

```

```

PRINT OCTAL RESULTS SCOPB,SCOPB(1)
NOCNT=0

```

```

REWIND TAPE FIVE
EXECUTE SETEOF.(NASTY)
EXECUTE SETERR.(NASTY)
T'H QQQ50, FOR QQ=0,1, QQ.GE. CCNT
A(GRPNT+QQ)=A(IPPNT+QQ)
A(GRPNT(1)+QQ)=A(IPPNT(1)+QQ)

```

QQQ50

```

PRINT OCTAL RESULTS A(5600+QQ),A(6000+QQ)
RREMOVE ANY DESIRED RESULTS WHICH ARE INPUT PARAMETERS
T'H KTCHIP, FOR J=0,1, J.GE. CCNT
MADBUG=(A(IPPNT+J).A.A(DRPNT+J)).FPAR.(A(IPPNT(1)+J).A.A(
1      DRPNT(1)+J))

```

W'R MADBUG

RDESIRED RESULTS FOUND ON INPUT PARAMETER LIST

REDIP

```

T'H REDIP, FOR VALUES OF K=0,1
A(DRPNT(K)+J)=A(DRPNT(K)+J) .A. (.N. A(IPPNT(K)+J))
E'L

```

```

KTCHIP      CONTINUE
            EXECUTE SUBSPT.
            P'T $H*0TIME TRAP SET FOR* I4,H* SEC.**$,MAXTIM
            TIMTRP.(1,60*MAXTIM,TRAP)
            RAND=0.
            SOLCNT=0
            LLINE=-1
            DRSW=0B
            SOLSW=0B
            T'H SETUP, FOR I=0,1, I .G. CCNT-1
SETUP        LTOR( A(LRPNT(1)+I) ) = I
RELOOP      RWRITE CURRENT SOLUTION COUNT AND DESIRED RESULT MATRIX
            WRITE BINARY TAPE FIVE, SOLCNT, CCNT, A(DRPNT)...A(DRPNT+CCNT-1)
            1  ,A(DRPNT(1))...A(DRPNT(1)+CCNT-1), A(GRPNT)...A(GRPNT+CCNT
            2  -1), A(GRPNT(1))...A(GRPNT(1)+CCNT-1), NOCNT
            T'O LOOPDR
START        RCHOOSE A CONNECTION LINE WITH DESIRED RESULTS
            DRSW=0B
            SOLSW=0B
            T'H LOOPDR, FOR LLINE=0,1, LLINE .G. CCNT-1
HUNTDR      RIDLIN=A(LRPNT+LLINE)
            MADBUG=A(DRPNT+RIDLIN).FPAR.A(DRPNT(1)+RIDLIN)
            W'R MADBUG, T'O GOTDR
LOOPDR      CONTINUE
            W'R .NOT. DRSW, T'O SOLVED
            W'R SOLSW, T'O START
RESET       RNO POSSIBLE SOLUTION EXISTS DOWN THIS PATH - BACK UP
            END OF FILE TAPE FIVE
            REWIND TAPE FIVE
            RRANDOMLY DECIDE WHERE TO RESTART
            I=SOLCNT*RANDOM.(RAND)
            LLINE=CCNT*RANDOM.(RAND)
            P'T FORM1, I, LLINE
            T'H FRESH, FOR J=0,1, J .G. I
FRESH       READ BINARY TAPE FIVE, SOLCNT, CCNT, A(DRPNT)...A(DRPNT+CCNT-1),
            1  A(DRPNT(1))...A(DRPNT(1)+CCNT-1)
            2  ,A(GRPNT)...A(GRPNT+CCNT-1), A(GRPNT(1))...A(GRPNT(1)+CCNT
            3  -1), NOCNT
            SOLSW=1B
            T'O HUNTDR
GOTDR       RDESIRED RESULTS ON RIDLIN TO BE ELIMINATED
            DRSW=1B
            P'T $H*-ATTEMPTING TO REMOVE DESIRED RESULTS ON LINE*, I5*$,
            1  RIDLIN
            AIDCNT=1
            MBECNT=0
            LINE=LLINE
            LINE(1)=LTOR(A(LRPNT+LINE))
            MAXWT=0.
            RGATHER ALL COLLECTIONS WHICH REDUCE THE DESIRED RESULTS.
            T'H GATHER, FOR VALUES OF SIDE=0,1
            CONETY=A(CONPNT(SIDE)+RIDLIN)
            ELPOS=.ELPOS. CONETY
            PELBAS= 5*ELPOS
            ATPOS= .ATPOS. CONETY
            EIDPOS=.EIDPOS. CONETY
            RBUILD A TABLE POINTING FROM ATTACHMENTS TO CONNECTION LINES
            T'H ATFND1, FOR I=0,1, I .GE. A(PELPNT+PELBAS+2)
ATFND1      ATINFO(I, SIDE)=$ $

```

```

W'R .ANEG. CONETY
J=RIDLIN
XZ=ATPOS
EXECUTE AIDBLD.(SIDE)
E'L
COMP=A(LRPNT(2+SIDE)+LINE(SIDE))
W'R .ANEG. COMP
TEST=.ABS. COMP
O'E
TEST=A(LRPNT(3-SIDE)+COMP)
T'H ATFND3, FOR I=COMP,1,A(LRPNT(3-SIDE)+I) .NID. TEST
J=A(LRPNT(1-SIDE)+I)
XZ=.ATPOS. A(CONPNT(1-SIDE)+J)
W'R .ANEG. A(CONPNT(1-SIDE)+J)
W'R ATINFO(XZ,SIDE) .E. $ $
EXECUTE AIDBLD.(1-SIDE)
E'L
O'E
W'R .ANEG. A(CONPNT(SIDE)+J)
EXECUTE AIDBLD.(SIDE)
ATINFO(XZ,SIDE)= .ABS. ATINFO(XZ,SIDE)
O'E
ATINFO(XZ,SIDE)=J
E'L
ATFND3 E'L
E'L
T'H ATFND2, FOR I=TEST,1,A(LRPNT(2+SIDE)+I) .NID. COMP
J=A(LRPNT(SIDE)+I)
XZ=.ATPOS. A(CONPNT(SIDE)+J)
W'R .ANEG. A(CONPNT(SIDE)+J)
W'R ATINFO(XZ,SIDE) .E. $ $
EXECUTE AIDBLD.(SIDE)
E'L
O'E
W'R .ANEG. A(CONPNT(1-SIDE)+J)
EXECUTE AIDBLD.(1-SIDE)
ATINFO(XZ,SIDE)= .ABS. ATINFO(XZ,SIDE)
O'E
ATINFO(XZ,SIDE)=J
E'L
ATFND2 E'L
T'H QQQ1, FOR QQ=0,1, QQ .GE. A(PELPNT+PELPNT+2)
W'R ((.DECR. ATINFO(QQ,SIDE)) .NE. 0) .AND. (ATINFO(QQ,SIDE)
1 .NE. $ $ )
EXECUTE BPSPRD.(ATINFO(QQ,SIDE))
PRINT OCTAL RESULTS ATINFO(QQ,SIDE)
QQQ1 E'L
PRINT OCTAL RESULTS AIDS...AIDS(AIDCNT-1)
T'H GATHER, FOR COLPAS=0,1, COLPAS .GE. A(PELPNT+PELBAS+4)
COLBAS=A(PELPNT+PELBAS+3)+COLPAS*A(PELPNT+PELBAS+2)*4
RTEST TO SEE IF THE COLLECTION REMOVES ANY BITS ON THE LINE
MADBUG=(A(DRPNT+RIDLIN).A.B(COLBAS+4*ATPOS+2)).FPAR.(A(DRPNT
1 (1)+RIDLIN).A.B(COLBAS+4*ATPOS+3))
W'R MADBUG
PRINT OCTAL RESULTS B(COLBAS+4*I)...B(COLBAS+4*I+3)
RTHIS COLLECTION DOES REMOVE BITS FROM THE LINE.
RNOW DETERMINE A WEIGHT FOR THE COLLECTION
IPCNT=0.
IPHERE=0

```

```

DRCNT=0.
IN34=0B
ESTNOK=0B
BITS=0
BITS(1)=0
BITS(2)=0
BITS(3)=0
T'H ADDUP, FOR I=0,1, I .GE. A(PELPNT+PELBAS+2)
WTOUT=0B
W'R ATINFO(I,SIDE) .E. $ $, WTOUT=1B
NOPAR=(.NOT.(B(COLBAS+4*I).FPAR.B(COLBAS+4*I+1))).AND.(.NOT.(
1 B(COLBAS+4*I+2).FPAR.B(COLBAS+4*I+3)))
W'R NOPAR .AND. IFBIT.(B(COLBAS+4*I+3),35) .AND. .NOT. WTOUT
1 , T'O GATHER
W'R WTOUT .AND. .NOT. NOPAR, T'O GATHER
W'R NOPAR .OR. WTOUT, T'O ADDUP
T'H ESTPEN, FOR QQ = 0,1, QQ .G. 3
ESTPEN BITS(QQ) = BITS(QQ) .V. B(COLBAS+4*I+QQ)
W'R IFBIT.(B(COLBAS+4*I+1),35)
W'R .ANEG. ATINFO(I,SIDE)
RATTACHMENT IDENTIFIER AND SUMMATION COLLECTION.
SUMUP CONTINUE
W'R ATPOS .NE. 1
W'R (B(COLBAS+4*I+2).A.(.N.SCOBP)).FPAR.(B(COLBAS+4*I+3).A.
1 (.N.SCOBP(1))), T'O GATHER
IPCNT=IPCNT+((.N.A(IPPNT+.ADR.ATINFO(I,SIDE))).A.B(COLBAS+4*I
1 ).A.(.N.SCOBP)) .COUNT. ((.N.A(IPPNT(1)+.ADR. ATINFO
2 (I,SIDE))).A.B(COLBAS+4*I+1).A.(.N.SCOBP(1)))
E'L
CONETY=COLBAS.IND. RIDLIN
CKNOTB T'H CKNOTB, FOR XI=0,1,(XI .GE. NOCNT) .OR. (CONETY .E.
1 NOTAB(XI))
W'R XI .L. NOCNT, T'O GATHER
RCOUNT THE BROAD-SCOPE PARAMETER BITS.
IPCNT=IPCNT+((.N.A(IPPNT+.ADR.ATINFO(I,SIDE))).A.B(COLBAS+
1 4*I).A.SCOBP) .COUNT. ((.N. A(IPPNT(1)+.ADR. ATINFO
2 (I,SIDE))).A. B(COLBAS+4*I+1) .A. SCOB(1))
RCOUNT THE NARROW-SCOPE PARAMETER BITS.
T'H SUM, FOR XI=.DECR.ATINFO(I,SIDE),-XI+.DECR.AIDS(XI),.ANEG
1 . AIDS(XI)
SUM IPCNT=IPCNT+((.N.A(IPPNT+AIDS(XI))).A.B(COLBAS+4*I).A.(.N.
1 SCOB)) .COUNT. ((.N.A(IPPNT(1)+AIDS(XI))).A.B(COLBAS
2 +4*I+1).A.(.N. SCOB(1)))
T'O SUM2
O'E
RSUMMATION COLLECTION BUT NO ATTACHMENT IDENTIFIER.
T'O SUMUP
E'L
O'E
W'R .ANEG. ATINFO(I,SIDE)
RATTACHMENT IDENTIFIERS BUT NO SUMMATION COLLECTION.
R*** CAN GET BURNED BY NARROW-SCOPE PARAMETERS ***
T'O NOATBR
C'E
NOATBR RNO ATTACHMENT IDENTIFIERS AND NO SUMMATION COLLECTION
CONTINUE
IPCNT=IPCNT+((.N.A(IPPNT+.ADR.ATINFO(I,SIDE))).A.B(COLBAS+
1 4*I)) .COUNT. ((.N. A(IPPNT(1)+.ADR.ATINFO(I,SIDE)))
2 .A. B(COLBAS+4*I+1))

```

```

SUM2      CONTINUE
          DRCNT=DRCNT+(A(DRPNT+.ADR.ATINFO(I,SIDE)).A.B(COLBAS+4*I+2))
1          .COUNT. (A(DRPNT(1)+.ADR.ATINFO(I,SIDE)).A.B(COLBAS+4
2          *I+3))
          IN34=IN34 .OR. IFBIT.(B(COLBAS+4*I+3),34)
          ESTNOK=ESTNOK.OR.((B(COLBAS+4*I+2).A.(.N.A(GRPNT+.ADR.
1          ATINFO(I,SIDE))))).FPAR.(B(COLBAS+4*I+3).A.(.N.
1          A(GRPNT(1)+.ADR.ATINFO(I,SIDE))))))
          IPHERE=IPHERE+B(COLBAS+4*I) .COUNT. B(COLBAS+4*I+1)
          E'L
          END OF CONDITIONAL
ADDUP     CONTINUE
          P'S IPCNT,IPHERE,DRCNT
          SOLSW=1B
          W'R IPCNT .NE. 0.
          QQ=(BITS.A.BITS(2)).COUNT.(BITS(1).A.BITS(3))
          IPCNT = IPCNT + 4*QQ
          E'L
          W'R IPHERE .E. 0
          RSTATEMENT COLLECTION IS FOR COMPUTE OR ESTIMATE
          W'R IN34
          RTHIS IS A COMPUTE STATEMENT - GIVE IT A LARGE WEIGHT
GOODUN    CONTINUE
          WHGHT=100.*DRCNT
          O'E
          RTHIS IS AN ESTIMATE STATEMENT - GIVE IT A LOW WEIGHT
          W'R ESTNOK, T'O GATHER
          WHGHT=DRCNT/PARCNT
          E'L
          O'E
          RTHIS IS A NORMAL STATEMENT - GIVE IT A REGULAR WEIGHT
          W'R IPCNT .E. 0. , T'O GOODUN
          WHGHT=DRCNT/IPCNT
          E'L
          RNOW SAVE THE WEIGHTS, ETC.
          MAXWT=MAXWT+WHGHT
          CHOICE(MBECNT)=MAXWT
          CHINFO(MBECNT,1)=SIDE
          CHINFO(MBECNT,2)=COLBAS
          CHINFO(MBECNT,3)=PELBAS
          CHINFO(MBECNT,4)=ELPOS
          CHINFO(MBECNT,5)=COLPAS
          CHINFO(MBECNT,6)=EIDPOS
          MBECNT=MBECNT+1
          P'T $S11,H*SIDE*I2,H*, ELEMENT *C6,H*, COLLECTION *I3,H*, W
1EIGHT *F9.3,$I0,I3*$, SIDE,A(PELPNT+PELBAS),COLPAS+1,WHGHT,
2          MBECNT
          E'L
GATHER    CONTINUE
          W'R MBECNT .E. 0
          P'T $HZO***** NO BITS CAN BE REMOVED FROM THIS LINE.Z*$
          W'R ((A(DRPNT+RIDLIN).A.(.N.SCOB)).FPAR.(A(DRPNT(1)+RIDLIN)
1          .A.(.N.SCOB(1))))
          P'T $HZ ***** NARROW SCOPE PARAMETERS CANNOT BE REMOVED.Z*$
          T'O RESET
          C'E
          T'O LOOPDR
          E'L
          E'L

```

```

RNOW CHOOSE A COLLECTION
WHGHT=MAXWT*RANDOM.(RAND)
ATTACK   T'H ATTACK, FOR I=0,1, CHOICE(I) .GE. WHGHT
RWE HAVE DECIDED TO USE THE I'TH STATEMENT COLLECTION
PIT $H$OWE HAVE ELECTED TO USE CHOICE*,I3,H* FROM THE ABOVE.*
1*$, I+1
RNOW OUTPUT THE STRING OF INFORMATION FOR CORE 4.
A(SOLPNT+SOLCNT)=RIDLIN
PUTWAY   T'H PUTWAY, FOR J=1,1, J .G. 3
A(SOLPNT(J)+SOLCNT)=CHINFO(I,J+3)
A(SOLPNT(2)+SOLCNT)=A(SOLPNT(2)+SOLCNT)+1
SIDE=CHINFO(I,1)
COLBAS=CHINFO(I,2)
PELBAS=CHINFO(I,3)
W'R SIDE .E. 1.A(SOLPNT(1)+SOLCNT)=-A(SOLPNT(1)+SOLCNT)
SOLCNT=SOLCNT+1
RNOW UPDATE THE DESIRED RESULT BITS
T'H FIXUP, FOR J=0,1, J .GE. A(PELPNT+PELBAS+2)
W'R ATINFO(J,SIDE) .E. $ $, T'O FIXUP
A(DRPNT+.ADR.ATINFO(J,SIDE))=A(DRPNT+.ADR.ATINFO(J,SIDE)).V.
1      B(COLBAS+4*J)
A(DRPNT(1)+.ADR.ATINFO(J,SIDE))=A(DRPNT(1)+.ADR.ATINFO(J,SIDE)
1      ).V.B(COLBAS+4*J+1)
A(DRPNT+.ADR.ATINFO(J,SIDE))=A(DRPNT+.ADR.ATINFO(J,SIDE)).A.
1      (.N.B(COLBAS+4*J+2))
A(DRPNT(1)+.ADR.ATINFO(J,SIDE))=A(DRPNT(1)+.ADR.ATINFO(J,SIDE)
1      ).A.(.N.B(COLBAS+4*J+3))
A(DRPNT+.ADR.ATINFO(J,SIDE))=A(DRPNT+.ADR.ATINFO(J,SIDE)).A.
1      (.N.A(IPPNT+.ADR.ATINFO(J,SIDE)))
A(DRPNT(1)+.ADR.ATINFO(J,SIDE))=A(DRPNT(1)+.ADR.ATINFO(J,SIDE)
1      ).A.(.N.A(IPPNT(1)+.ADR.ATINFO(J,SIDE)))
A(GRPNT+.ADR.ATINFO(J,SIDE))=A(GRPNT+.ADR.ATINFO(J,SIDE)).V.
1      B(COLBAS+4*J+2)
A(GRPNT(1)+.ADR.ATINFO(J,SIDE))=A(GRPNT(1)+.ADR.ATINFO(J,SIDE)
1      ).V.B(COLBAS+4*J+3)
W'R IFBIT.(B(COLBAS+4*J+1),35)
RSPRAY NARROW-SCOPE DESIRED RESULT BITS FOR SUMMATIONS.
T'H SUM1, FOR XI=.DECR.ATINFO(J,SIDE),-XI+.DECR.AIDS(XI),
1      .A.NEG. AIDS(XI)
NOTAB(NOCNT) = COLBAS .IND. AIDS(XI)
NOCNT=NOCNT+1
A(DRPNT+AIDS(XI))=A(DRPNT+AIDS(XI)).V.(B(COLBAS+4*J)).A.(.N.
1      SCOPB))
A(DRPNT(1)+AIDS(XI))=A(DRPNT(1)+AIDS(XI)).V.(B(COLBAS+4*J+1)
1      .A.(.N. SCOPB(1)))
A(DRPNT+AIDS(XI))=A(DRPNT+AIDS(XI)).A.(.N.A(IPPNT+AIDS(XI)))
SUM1    A(DRPNT(1)+AIDS(XI))=A(DRPNT(1)+AIDS(XI)).A.(.N.A(IPPNT(1)+AI
1      DS(XI)))
E'L
W'R (.DECR. ATINFO(J,SIDE)) .NE. 0, EXECUTE BPDELT.(ATINFO
1      (J,SIDE))
FIXUP   CONTINUE
PIT $S6,H*THE STATUS AS OF SOLUTION LINE*,I3,H* IS AS FOLLOWS
1 **$, SOLCNT
EXECUTE SUBSPT.
T'O RELOOP
NASTY   EXECUTE ERROR.
RTIME TRAP - GO TO NEXT PROBLEM.
TRAP    PRINT COMMENT $0 NO SOLUTION FOUND IN THE ALLOTTED TIME.$

```

```

PRINT COMMENT $OPROBLEM STATUS FOLLOWS BELOW.$
T'H BBB, FOR I=0,1, I .GE. SOLCNT
BBB P'T FORM3,A(SOLPNT+I),A(SOLPNT(1)+I),A(SOLPNT(2)+I),
1 A(SOLPNT(3)+I)
EXECUTE SUBSPT.
EXECUTE PANEL.
EXECUTE SDUMP.(A(15999)...TOPCOR)
EXECUTE SELPGM.(1)
SOLUTION FOUND - RESET TIME TRAP AND NOTE TIME
SOLVED TIME = MAXTIM - RESTIM.(1)/60.
P'T FORM2, TIME
PRINT COMMENT $1SOLUTION FOLLOWS ( IN REVERSE ORDER ).$
T'H AAA, FOR I=0,1, I .GE. SOLCNT
AAA P'T FORM3, A(SOLPNT+I),A(SOLPNT(1)+I),A(SOLPNT(2)+I),A(SOLPNT
1(3)+I)
EXECUTE SEQPGM.
RFORMAT STRINGS FOR ALL I/O STATEMENTS
V'S FORM1=$H*OBLIND ALLEY - NEW SOLCNT,LLINE ARE *,215*$
V'S FORM2=$H*OSOLUTION FOUND IN *F6.2,H* SEC.**$
V'S FORM3=$S1,4I10*$
RVARIBLE MODE DECLARATIONS
NORMAL MODE IS INTEGER
BOOLEAN DRSW,SOLSW,IN34,WTOUT,IFBIT.,NOPAR
BOOLEAN ESTNOK
BOOLEAN MADBUG
BOOLEAN Y4
FLOATING POINT RANDOM.,RAND,IPCNT,DRCNT,MAXWT,CHOICE
FLOATING POINT TIME
FLOATING POINT WHGHT
RDIMENSIONING AND COMMON ALLOCATIONS
PROGRAM COMMON INDEX(99),A(15899),B(1999),BROAD(200)
ERASABLE TOPCOR
DIMENSION DRPNT(1),LRPNT(3),CONPNT(1),LTOR(399),IPPNT(2),
1 CHOICE(40),CHINFO((0...40)*6),ATINFO((0...20)*
2 (0...1)),SOLPNT(3),LINE(1)
DIMENSION AIDS(40), XLST((0...10)*2)
DIMENSION BITS(5),SCOPB(1)
DIMENSION NOTAB(400)
EQUIVALENCE (A(15600),SOLCNT)
EQUIVALENCE (A(15842),CCNT)
EQUIVALENCE (SCOPB,A(15770))
EQUIVALENCE (PARCNT,A(15603))
EQUIVALENCE ( MAXTIM,INDEX(99))
V'S FIVE=9
V'S DRPNT=10000,10400
V'S LRPNT=10800,11200,11600,12000
V'S CONPNT=5600,6000
V'S IPPNT=9200,9600
V'S SOLPNT=100,500,900,1300
V'S PELPNT=8200
V'S GRPNT=12400,12800
INTERNAL FUNCTION
ENTRY TO SUBSPT.
P'T $1H0,T7,1HL,T13,1HR,T18,2HLR,T24,2HRL,T35,3HIP1,S11,3HIP2
1S11,3HDR1S11,3HDR2S11,3HGR1S11,3HGR2S14,4HLINE*$
T'H QQQ10, FOR QQ=0,1, QQ .G. CCNT-1
QQQ10 P'T FORM9,A(LRPNT+QQ),A(LRPNT(1)+QQ),A(LRPNT(2)+QQ),
1A(LRPNT(3)+QQ),A(IPPNT+QQ),A(IPPNT(1)+QQ),A(DRPNT+QQ),
2A(DRPNT(1)+QQ),A(GRPNT+QQ),A(GRPNT(1)+QQ),QQ

```

```

F'N
V'S FORM9=$S1,4I6,$S9,6(K12,$2),I6*$
E'N
INTERNAL FUNCTION (XSIDE)
ENTRY TO AIDBLD.
ATINFO(XZ,SIDE)= - ( J .IND. AIDCNT)
XLST(0,1)=J
XLST(0,2)=XSIDE
XLPNT=0
T'H XBATC, FOR XI=0,1, XI .G. XLPNT
CONETY=A(CONPNT(XLST(XI,2))+XLST(XI,1))
XAT= .ATPOS. CONETY
XSRCH T'H XSRCH, FOR XJ=0,1, A(LRPNT(XLST(XI,2))+XJ) .E. XLST(XI,1)
XCOMP=A(LRPNT(2+XLST(XI,2))+XJ)
W'R .ANEG. XCOMP
XTEST=.ABS. XCOMP
C'E
XTEST=A(LRPNT(3-XLST(XI,2))+XCOMP)
T'H XFND3, FOR TI=XCOMP,1,A(LRPNT(3-XLST(XI,2))+TI) .NID.
1 XTEST
J=A(LRPNT(1-XLST(XI,2))+TI)
W'R XAT .E. ( .ATPOS. A(CONPNT(1-XLST(XI,2))+J))
XCHK T'H XCHK, FOR TJ=0,1, TJ .G. XLPNT .OR. J .E. XLST(TJ,1)
W'R TJ .E. 0 .AND. XI .E. 0, T'O OK1
W'R TJ .G. XLPNT
AIDS(AIDCNT)= J .IND. (AIDCNT+1)
W'R XI .NE. 0, AIDS(AIDCNT)=-AIDS(AIDCNT)
AIDCNT=AIDCNT+1
OK1 CONTINUE
W'R .ANEG. A(CONPNT(XLST(XI,2))+J)
XLPNT=XLPNT+1
XLST(XLPNT,1)=J
XLST(XLPNT,2)=XLST(XI,2)
E'L
E'L
XFND3 E'L
E'L
T'H XFND2, FOR TI=XTEST,1,A(LRPNT(2+XLST(XI,2))+TI) .NID.
1 XCOMP
J=A(LRPNT(XLST(XI,2))+TI)
W'R XAT .E. ( .ATPOS. A(CONPNT(XLST(XI,2))+J))
XCHK2 T'H XCHK2, FOR TJ=0,1, TJ .G. XLPNT .OR. J .E. XLST(TJ,1)
W'R TJ .E. 0 .AND. XI .E. 0, T'O OK2
W'R TJ .G. XLPNT
AIDS(AIDCNT)= J .IND. (AIDCNT+1)
W'R XI .NE. 0, AIDS(AIDCNT)=-AIDS(AIDCNT)
AIDCNT=AIDCNT+1
OK2 CONTINUE
W'R .ANEG. A(CONPNT(1-XLST(XI,2))+J)
XLPNT=XLPNT+1
XLST(XLPNT,1)=J
XLST(XLPNT,2)=1-XLST(XI,2)
E'L
E'L
XFND2 E'L
XBATCH CONTINUE
AIDS(AIDCNT)= 777777777777K
AIDCNT=AIDCNT+1
F'N

```

```

E'N
INTERNAL FUNCTION (ARG)
ENTRY TO BPSPRD.
Y4=1B
T'H Y1, FOR VALUES OF XI=0,1
BITS(XI)=A(IPPNT(XI)+.ADR. ARG)
BITS(XI+4)=A(GRPNT(XI)+.ADR. ARG)
Y1 BITS(XI+2)=A(DRPNT(XI)+.ADR. ARG)
T'H Y2, FOR XJ=.DECR. ARG, -XJ+.DECR. AIDS(XJ), AIDS(XJ) .E.
1 777777777777K
T'H Y2, FOR VALUES OF XI=0,1
BITS(XI)=A(IPPNT(XI)+.ADR.AIDS(XJ)).V.BITS(XI)
BITS(XI+4)=A(GRPNT(XI)+.ADR.AIDS(XJ)).V.BITS(XI+4)
Y2 BITS(XI+2)=A(DRPNT(XI)+.ADR.AIDS(XJ)).V.BITS(XI+2)
T'H Y3, FOR VALUES OF XI=0,1
A(IPPNT(XI)+.ADR. ARG)=(BITS(XI).A.SCOBP(XI)).V.A(IPPNT(XI)+
1 .ADR. ARG)
A(GRPNT(XI)+.ADR. ARG)=(BITS(XI+4).A.SCOBP(XI)).V.A(GRPNT(XI)+
1 .ADR. ARG)
Y3 A(DRPNT(XI)+.ADR. ARG)=((BITS(XI).A.SCOBP(XI)).V.A(DRPNT(XI)+
1 .ADR. ARG)).A.(.N.A(IPPNT(XI)+.ADR. ARG))
ENTRY TO BPDELT.
T'H T1, FOR XJ=.DECR. ARG, -XJ+.DECR. AIDS(XJ), AIDS(XJ) .E.
1 777777777777K
T'H T1, FOR VALUES OF XI=0,1
W'R Y4
A(IPPNT(XI)+.ADR.AIDS(XJ))=A(IPPNT(XI)+.ADR.AIDS(XJ)).V.(A(
1 IPPNT(XI)+.ADR. ARG).A.SCOBP(XI))
E'L
A(GRPNT(XI)+.ADR.AIDS(XJ))=A(GRPNT(XI)+.ADR.AIDS(XJ)).V.(SCOP
1 B(XI).A.A(GRPNT(XI)+.ADR. ARG))
T1 A(DRPNT(XI)+.ADR.AIDS(XJ))=(A(DRPNT(XI)+.ADR.AIDS(XJ)).V.SCOBP
1 B(XI)).A.(.N.SCOBP(XI)).V.A(DRPNT(XI)+.ADR. ARG)).A.
2 (.N.A(IPPNT(XI)+.ADR.AIDS(XJ)))
Y4=0B
F'N
E'N
END OF PROGRAM

```

\$ASSEMBLE,PUNCH OBJECT

IFBIT000IFBIT000

```

*** SUBROUTINE IFBIT.
*** A TYPICAL CALL FROM MAD WOULD BE AS FOLLOWS
*** WHENEVER IFBIT.(A,N)
*** IFBIT. RETURNS 1B IF A HAS A BIT IN LOCATION N.
*** OTHERWISE 0B IS RETURNED.

```

```

ENTRY IFBIT
IFBIT CLA* 2.4
STA SHIFT
CAL =K4K11
SHIFT ARS **
ANA* 1.4
TRA 3.4
END

```

\$ASSEMBLE,PUNCH OBJECT

RESTIM00RESTIM00

*** SUBROUTINE RESTIM.

*** A TYPICAL MAD CALLING SEQUENCE WOULD BE AS FOLLOWS

*** TIME=TIMAX-RESTIM.(CELL)/60.

*** THIS SUBROUTINE WILL RESET TIME TRAP 'CELL', AND RETURN THE

*** TIME DURATION THAT REMAINED AS A FULL WORD INTEGER, IN 60TH'S OF

*** A SECOND.

ENTRY RESTIM

RESTIM CLA* 1,4
PAX 0,2
SXD CELL2,2
SXD CELL3,2

SXA A,4

CLA 1,4

STA CELL1

CALL ITMLFT

CELL1 PAR **

STO SVTIM

CALL SETTIM

CELL2 BLK =0,,**

CALL SETTRP

CELL3 BLK =0,,**

A AXI **,4

CLA SVTIM

TRA 2,4

SVTIM NOP **

END

GET TIME REMAINING.

RESTORE TIME TRAP

CLEAN UP TRAP CELL

DUMMY STORAGE BOX

*

\$ASSEMBLE,PUNCH OBJECT

TIMTRPOOTIMTRPOO

*** SUBROUTINE TIMTRP.

*** A TYPICAL MAD CALLING SEQUENCE WOULD BE AS FOLLOWS

*** EXECUTE TIMTRP.(CELL,TIME,WHERE TO)

*** CELL IS THE LOGICAL TIME CELL NUMBER.

*** TIME IS THE DESIRED TIME DURATION IN 60TH'S OF SECONDS.

*** WHERE TO IS THE STATEMENT LABEL TO WHICH CONTROL IS TRANSFERRED ON

*** A TRAP.

	ENTRY	TIMTRP	
TIMTRP	CLA*	1,4	
	PAX	0,2	
	SXD	TIME,2	INIT TIMER NUMBER
	SXD	WHTO,2	
	CLA	2,4	
	STA	TIME	INIT TIME INTERVAL
	CLA	3,4	
	STA	WHTO	INIT TRAP ADDRESS
	SXA	A,4	
	CALL	SETTIM	
TIME	BLK	**,**	
	CALL	SETTRF	
WHTO	BLK	**,**	
A	AXT	**,4	
	TRA	4,4	RETURN TO MAIN
	END		

*

\$ASSEMBLE,PUNCH OBJECT
ENTRY SYSTEM,ERROR

SYS/ER00

SYSTEM NOP
ERROR PXA 0,4
PAC 0,4
SXA IN,4
PRINT FORMAT,IN,0
CALL PANEL
8CALL SDUMP,10000,....,77777
CALL SELRCD
BLK 1,2
IN PZE
FORMAT BCI *,H*6SYSTEM/ERROR CALLED FROM *,K5*
END

*

```

R'N
P'R QQ(100)
D'N ELMID(QQ),ELM(QQ),LINENO(QQ),COLLNO(QQ)
EQUIVALENCE (A(15603),PARCNT),(A(15602),TELCNT),
1      (A(15601),PELCNT),(A(15851),NOELMS),
2      (A(100 ),LINENO),(A(500 ),ELM),
3      (A(900 ),COLLNO),(A(1300),ELMID),
4      (A(15600),NOSOL)
EQUIVALENCE (PELT,A(8200)),(CONL,A(5600)),(CONR,A(6000))
EQUIVALENCE(A(8600),ATS)
D'N ATS(100)
D'N PELT(300),CONL(400),CONR(400),ELMLST(100),STMNT(100)
DEFINE UNARY OPERATOR .M.,PRECEDENCE SAME AS .E.
M'E 2 = .M.1

```

```

CLA      B
TMI      LOC+3
PXA      ,0
TRA      LOC+2
CLA      =1
OUT      AC
END

```

```

EQUIVALENCE (NOELM,NOELMS)
DEFINE UNARY OPERATOR .STZ.,PRECEDENCE SAME AS =
M'E .STZ. 1
STZ      B
OUT      Z
END

```

```

M'E .STZ. 2, S'S .STZ.1
N'R
B'N TWICE
V'S ONE = 1
V'S TWO = 2
V'S FOUR = 4
V'S ELTAPE = SELTAPES
V'S PROLOG = $PROLOG$
P'N A(15999),C(1999),BROAD(200)
EQUIVALENCE (C,B)
D'N B(7599)
V'S MASKED = 777K
V'S ZICK = $H90***** CORE4 -- CHECK NO. -915,5H*****$
V'S TSTFOR = 0B

```

```

PRINT COMMENT $1***** CORE4 HAS ARRIVED. *****$
(I=0,1,I.GE.NO SOL, ELM(I)=ELM(I).V.(ELMID(I).LS.9).V.(COLLNO
1(I).LS.18))
P'T $4(S10,K12,S2,I6)*$, (I=0,1,I.GE.NOSOL,ELM(I),LINENO(I))
PRINT BCD RESULTS ATS...ATS(100)
P'T$H90***** PELT TABLE*****9*$
PRINT OCTAL RESULTS PELT...PELT(NOELMS*5)
TWICE = 1B
.STZ. COUNT
AGAIN    REWIND TAPE FOUR
W'R TSTFOR, P'T ZICK, 1
SETEOF.(C1)
AGAIN2   READ BINARY TAPE FOUR,WORD
W'R TSTFOR, P'T ZICK, 2

```

```

W'R WORD .E. ELTAPE
W'R TSTFOR, P'T ZICK, 3
READ BINARY TAPE FOUR, PELCNT, TELCNT, PARCNT
NOELMS = PELCNT + TELCNT
W'R TSTFOR, PRINT BCD RESULTS PELCNT, PARCNT, TELCNT, NOELMS
SKIP.(TWO,0,FOUR)
T'O READY
E'L
W'R TSTFOR, P'T ZICK, 4
W'R TWICE
.STZ. TWICE
T'O AGAIN2
E'L
C1 TIME = -1
W'R TSTFOR, P'T ZICK, 5
SETEOF.(C2)
TWICE = 1B
SKIP.(ONE,0,FOUR)
C READ BINARY TAPE FOUR, WORD
W'R TSTFOR, P'T ZICK, 6
W'R WORD .E. PROLOG
W'R TSTFOR, P'T ZICK, 7
SKIP.(ONE,0,FOUR)
T'O READY
E'L
C(1) COUNT = COUNT + 1
W'R COUNT .L. 10, T'O AGAIN
PRINT COMMENT $6***** TAPE FOUR COULD NOT BE POSITIONED CORRE
CTLY FOR READING STATEMENT COLLECTIONS.$
SEQPGM.
C2 TIME = TIME + 1
W'R TSTFOR, P'T ZICK, 8
T'O C(TIME)
READY
W'R TSTFOR, P'T ZICK, 9
.STZ. CNT
.STZ. STMNT
.STZ. COLCNT
.STZ. ELMLST
.STZ. SCNT
.STZ. SZ
UPHS T'H SETABL, FOR ELCNT = 0,1,ELCNT .GE. NOELMS
SETEOF.(BAD)
READ BINARY TAPE FOUR, WORDS, NOATS
W'R TSTFOR, PRINT BCD RESULTS WORDS, NOATS, PELT(ELCNT*5), ELCNT
W'R WORDS .NE. PELT(ELCNT*5)
SZ = 1
E'L
MUCH SKIP.(1,1,FOUR)
SETEOF.(NXTONE)
READ BINARY TAPE FOUR, NOCRDS, NOCOL
W'R TSTFOR, PRINT BCD RESULTS NOCRDS, NOCOL
SCNT = SCNT + 1
SETEOF.(BAD)
W'R NOCRDS .G. 0
START = COLCNT
COLCNT = COLCNT + NOCRDS * 12
READ BINARY TAPE FOUR, B(START)...B(COLCNT-1)

```

```

      W'R TSTFOR,
1P'T $1H 12C6*$,(I=START,12,I.GE.COLCNT,B(I)...B(I+1))
      E'L
      STMT(SCNT) = COLCNT
      W'R TSTFOR, P'T ZICK, 10
      T'O MUCH
NXTONE  CNT = CNT + 1
      ELMLST(CNT) = SCNT
SETABL
      W'R TSTFOR, P'T ZICK, 11
      W'R SZ .G. 0, T'O CHECK
DONCHK
      W'R TSTFOR
      PRINT FORMAT $(S5,I10)*$,(I=0,1,I.G.NOELM,ELMLST(I)),(I=0,1,
1I.G.ELMLST(NOELM),STMT(I))
      E'L
      T'H GENALL, FOR I = NOSOL-1,-1,I .L. 0
      (JKQ=NOSOL-1,-1,(LINENO(JKQ).E.LINENO(I).AND.ELM(JKQ).E.
1ELM(I)).OR.(JKQ.LE.I))
      W'R JKQ .LE. I
      J = ELMLST(ELM(I) .A. MASKED) + COLLNO(I)
      P'T $H91HERE WE GO LOOP-DE-LOOP.9*$
      SSCAN.(STMT(J),STMT(J-1),ELM(I),LINENO(I))
      O'E
      PRINT COMMENT $0 BANG, BANG, BANG ... YOU'R DEAD. (STATEMENT
1COLLECTION GENERATION DELETED)$
      E'L
GENALL
      W'R TSTFOR, P'T ZICK, 13
      PUNCH FORMAT $T80,1H**$
      SEQPGM.
BAD      PRINT COMMENT $0***** UPHS.....$
      SZ = 2
      W'R ELCNT .E. 0, T'O UPHS
      T'O SETABL
CHECK    P'T $1H 12C6*$,A(5600)...A(STMT(SCNT))
      T'O DONCHK
      E'M
$ASSEMBLE, PRINT OBJECT, PUNCH OBJECT
      ENTRY  GETBRD
      GETBRD000
*****
***  ROUTINE TO SELECT A BIT BETWEEN 1 AND 36 * NUMBER OF BROAD-  ****
***  SCOPE BIT WORDS ----- TEST AND RETURN A 1B IF ON, A 0B IF  ****
***  OFF *****
*****
GETBRD LDQ* 1,4 BIT NUMBER
      DVP THREE6 DETERMINE WORD BIT IS IN
      XCA
      PAX ,1
      XCA
      MPY THREE6
      ADD =1
      SUB 1,4 DETERMINE BIT TO BE EXAMINED
      STA **4
      CLA 2,4
      STA **1
      CAL* **,1
      ALS ** MOVE BIT TO END OF AC
      TMI **3 IS BIT ON .....

```

TRA *+3
CLA =1 YES ..
TRA 1.4
PXA .0 NO
TRA 1.4
THREE6 DEC 36
END

*

\$ASSEMBLE,PRINT OBJECT,PUNCH OBJECT

PMBCDN000

```

ENTRY SET
ATS MACRO /LFS/TAG,A,B
TAG CAL A,4
IRP B
STA B
IRP
ENDM
SET ATS 1(PSTO+1,BCDNUM,MOVE-2,MOVE+5,ZQ-2)
ATS 2,BCDNUM+2
ATS 3(ZQ-5,AA+2,ZERO)
ATS 4(BCDNUM-3,ZQ1-2,ZQ1,ZQ1+3,ZQ-4)
ADD =2
STA ZQ1+1
STA ZQ1+2
TRA 1,4
ENTRY PSTO,MOVE,BCDNUM
PSTO LDQ* 1,4
LXA J,1
AXT 3,2
TRA *+2
-3 LGL 6
-2 TXI *+1,1,1
-1 STQ OUTPUT,1
TIX *-3,2,1
TRA *+2
BCDNUM LXA J,1
AXT 3,2
LDQ QQ
PXA ,0
TRA *+2
-6 LDQ ZQ
-5 TXI *+1,1,1
-4 DVP =10
-3 STQ ZQ
LGR 6
-1 STQ OUTPUT,1
TIX *-6,2,1
ZQ1 CLA OUTPUT,1
LDQ OUTPUT-2,1
STO OUTPUT-2,1
STQ OUTPUT,1
SXA J,1
TRA 1,4
MOVE CLA* 1,4
PAX ,2
CLA* 2,4
PAX ,1
SXD *+3,1
LXA J,1
-5 TXI *+1,2,1
-4 TXH *+5,2,**
-3 TXI *+1,1,1
-2 CLA CARD,2
-1 STO OUTPUT,1
TRA *-5
SXA J,1
TRA 1,4
ZQ PZE

```

J	PGMCCM	1000	
CARD	EQU	J-1	
OUTPUT	EQU	J-73	
QQ	EQU	J-74	
	ENTRY	FIND,FIND1,FIND2,FIND3	
FIND	CLA*	5,4	STARTING SUB VALUE
	STA	FIND3	
	CLA*	4,4	LAST POSITION TO BE LOOKED AT
	ALS	18	
	STD	DONE	
FIND1	CLA	3,4	ADD OF VEC TO SAVE SUBS IN
	STA	VEC	
	STA	SAVE	
	AXT	0,2	
FIND2	CLA	2,4	ADD OF VEC TO BE SEARCHED
	STA	ZERO	
FIND3	AXT	**,1	
	CAL*	1,4	WORD SOUGHT
	SLW	ZQ	
ZERO	LAS	**,1	
	TXI	**+3,1,1	
	TXI	SAVE-1,2,1	
	TXI	**+1,1,1	
DONE	TXL	ZERO,1,**	IF NOT DONE,TRY NEXT ONE
VEC	SXA	**,2	ZERO LOC HOLDS LENGTH OF VECTOR
	TRA	1,4	
	PXA	,1	
SAVE	STO	**,2	
	CAL	ZQ	
	TXI	DONE,1,1	
	ENTRY	WORD	
WORD	CLA*	2,4	SUBSCRIPT OF RIGHT END
	PAX	,1	
	TXI	**+1,1,-1	
	SXD	AA,1	
	CLA*	1,4	SUBSCRIPT OF LEFT END
	PAX	,1	
	AXT	36,2	
	REM		
	TXI	**+1,1,1	
AA	TXH	OUT,1,**	
	XCL		
	CAL	CARD,1	
	LAS	BLANK	
	TRA	**+2	
	TRA	AA,1,1	
	XCL		
	LGL	6	
NEXT	TIX	AA-1,2,6	
	TRA	QUIT	
OUT	SXA	**+2,2	
	LDQ	BLANK	
	LGL	**	
QUIT	XCL		
	XCA		
	TRA	1,4	
BLANK	BCD 1		
	END		

*

\$ COMPILE MAD,PRINTOBJECT,PUNCHOBJECT,EXECUTE
EXTERNAL FUNCTION (BOTTOM,TOP,EL,NOLINE)

SSCAN000

```

R'N
E'0 SSCAN.
N'R
P'R AQ(500)
P'R BQ(299)
P'R CQ(69)
P'R EQ(7599)
PARAMETER MQ(100)
PARAMETER NQ(199)
P'N A(15999),C(1999),BROAD(200)
EQUIVALENCE (C,B)
D'N B(7599)
D'N PLACE(2),SAVE(MQ),SLIST(NQ),ATS(NQ)
DIMENSION AREA(AQ),PELT(BQ),ZPAR(CQ)
D'N CARD(71)
EQUIVALENCE (A(8600),ATS),(A(8200),PELT),(A(15700),FPAR),
1 (A(15603),PARCNT),(A(15772),ZPAR),(FPAR(70),BWORDS),
2 (A(5600),CONL),(A(6000),CONR),(A(15842),ELMCNT)
EQUIVALENCE (A(15600),NOSOL)
EQUIVALENCE (A(15842),CCNT)
EQUIVALENCE (PLACE,LP),(PLACE(1),RP),(PLACE(2),CM)
D'N SAVEC(29)
DEFINE UNARY OPERATOR .M., PRECEDENCE SAME AS .E.
MODE STRUCTURE 2 = .M. 1
CLA      B
TMI      LOC+3
PXA      ,0
TRA      LOC+2
CLA      =1
OUT      AC
END

DEFINE UNARY OPERATOR .STZ., PRECEDENCE SAME AS =
MODE STRUCTURE .STZ. 1
STZ      B
OUT      Z
END

W'R MINTST, PRINT OCTAL RESULTS BOTTOM,TOP,EL,NOLINE
W'R MINTST, P'IT $$15,12C6*$,(I=BOTTOM-12,-12,I.L.TOP,B(I)...B
1(I+11))
(I=0,1,I.G.18,.STZ.MINUS(I),.STZ.PLUS(I))
COLS = 72 - STATE
.STZ. QQZ
SLIST = 0
ELQ = EL .A. (777777K)
ELI = ELQ .A. (777K)
BEGIN = ELI * 5 + 1
BEGIN1 = BEGIN + 1
J = 77777K
SET.(J,QQ,CARD,OUTPUT)
SELM = $$
T'H COLECT, FOR Q = BOTTOM -12, -12, Q .L. TOP
.STZ. SAVEC
R          LOOP TO STEP THROUGH THE REGION WHERE THE CARD
R          IMAGES ARE STORED, Q BEING THE BEGINNING OF EACH
R          SEPARATE CARD IMAGE
SPREAD.(B(Q),$72C1*$,CARD...CARDE)
W'R MINTST, P'IT $1H 12C6*$,B(Q)...B(Q+11)

```

```

R          CHECK TO SEE IF IT'S A COMMENT CARD
W'R CARD(COL) .E. COMLET
R          IS IT A COMMENT CARD
P'T FUN, CARD...CARDE
PUNCH FORMAT $72C1*$,CARD...CARDE
T'O COLECT
O'R CARD(COM) .NE. $$
W'R MINTST, P'T ZICK, 001
R          IS IT A CONTINUATION CARD
START = STATE
START1 = STATE
O'E
W'R MINTST, P'T ZICK, 002
START = 77777K
START1 = 0
E'L
R          AT THIS POINT, COMMENT CARDS
R          HAVE BEEN PRINTED OUT, CONTINUATION
R          CARDS FLAGED
FIND.(MINO,CARD,MINUS,71,START1)
R          LOCATE ALL PLUS AND MINUS ZEROS
FIND1.(PLUSO,CARD,PLUS)
W'R PLUSF .OR. MINUSF: P'T $6615*$,MINUS...MINUS(18),PLUS...
1PLUS(18)
W'R MINUS .E. 0
W'R MINUSF, P'T ZICKM, 001
MINM = 77777K
O'E
MINM = MINUS(1)
NEXTM = 3
E'L
NEXTP = 1

SCAT
W'R PLUS .L. NEXTP
W'R PLUSF, P'T ZICKP, 001
MINP = 77777K
O'E
MINP = PLUS(NEXTP)
NEXTP = NEXTP + 2
E'L
T'O TEST

LABEL
W'R MINUS .L. NEXTM
W'R MINUSF, P'T ZICKM,002
R          HAVE ALL MINUS ZEROS BEEN PROCESSED
MINM=77777K
O'E
R          IF NOT, GET THE NEXT ONE TO BE PROCESSED
MINM = MINUS(NEXTM)
NEXTM = NEXTM + 2
E'L

TEST
W'R MINP .L. MINM
W'R PLUSF, P'T ZICKP, 002
R          ENTER HERE IF NEXT SUBSTITUTION IS TO BE
R          FOR A PLUS ZERO
MOVE.(START,MINP-1)
ELM = ELQ
STIP = PLUS(NEXTP-1)

```

```

START = STIP
KKL = 0
T'H CHECK1, FOR VALUES OF LETTER = LPAREN, RPAREN, COMMA
W'R PLUSF, P'T ZICKP, 003
R          FIND OCCURENCES OF -(-, -)-, --, --
R          BETWEEN PLUS ZEROS
COUNT = 0
COUNT(1) = 0
FIND.(LETTER,CARD,COUNT,STIP-1,MINP+1)
W'R COUNT .G. 1
MOVE.(MINP-1,STIP)
PRINT COMMENT $0**** ILLEGAL NO. OF CHARACTERS ON FOLLOWING C
1ARD.$
T'O SCAT
O'E
PLACE(KKL) = COUNT(1)
E'L
KKL = KKL + 1

CHECK1
W'R .NOT. RPK, RP = STIP
W'R .NOT. LPK
W'R PLUSF, P'T ZICKP, 004
LP = MINP
PAR = $$
C'E
PAR = WORD.(MINP,LP)
E'L
R          NOW HAVE PARAMETER NAMES IN PAR ---
W'R PAR .E. $$, PAR = 0
W'R PLUSF,P'T $H8 PARAMETER NAME = 8C6*$,PAR
W'R CMK
W'R WORD.(CM,RP) .NE. $$, ELM = -ELM
W'R PLUSF, P'T $H9ATTACHMENT IDENTIFIER IS 9C6*$,
1WORD.(CM,RP)
R          IF THERE IS AN ATTCHMENT IDENTIFER, FLAG
R          IT
C'E
CM = RP
E'L
ELM = ELM .V. (SEARCH.(ATS(PELT(BEGIN)),PELT(BEGIN1))-1,WORD.
1(LP,CM),NOAT).LS.18)
W'R PLUSF,P'T$H90 CM = 9I6,H8 RP = 8I6,H8 THE ATTACHMENT NA
1ME IS 8C6,H8 THE ELM WORD = 8K12*$,CM,RP,WORD.(LP,CM),ELM
R          WE NOW HAVE A PACKED WORD CONTAINING
R          /+ OR -/ ATT / EID / ELM /
W'R .M. ELM
W'R ELM .E. SELM, T'O OKK1
PLUSF = 1B
SELM = ELM
FIND.(ELM,CONR,SAVEC,NOSOL,0)
FIND3.(-ELM)
FIND2.(ELM,CONL)
FIND3.(-ELM)
W'R PLUSF, P'T ZICKP, 005
W'R SAVEC .E. 0
P'T $H9 ***** ERROR -- ELEMENT NOT FOUND....9*$
QQ = 988
O'E
W'R PLUSF, P'T ZICKP, 007

```

```

JQJ = SAVEC - 1
K = 1
TSTLIN
  QQ = SEARCH.(SAVEC(K),JQJ,NOLINE,OKK) + K
  W'R PLUSF, P'T ZICKP, 008
  (I=QQ,1,I.G.SAVEC,SAVEC(I)=SAVEC(I+1))
  SAVEC = SAVEC - 1
  K = QQ
  JQJ = JQJ - K
  T'O TSTLIN

OKK
  W'R SAVEC .E. 1
  P'T $H9 ***** ONLY ONE OCCURENCE OF ELEMENT.9*$
  SAVEC = 0
R      TURN OFF THE MULTI-SUBSTITUTION SWITCH
R      AND PROCEED

OKK1
  W'R PLUSF, P'T ZICKP, 006
  QQ = SAVEC(1)
  T'O QQERR
  O'R SAVEC .E. 0
  W'R PLUSF
  P'T $H90***** ELEMENT OCCURES ONLY ON THE PRESENT SOLUTION LI
INE.9*$
  E'L
  QQ = NOLINE
  T'O QQERR
  E'L
  SAVE = 1 + SAVE
  W'R PLUSF, P'T ZICKP, 009
R      IF ELEMENT HAD AN IDENTIFER FOR THE ATT.
R      SAVE THE POSITIONS FOR LATER SUBSTITUTIONS
  W'R PARK
  W'R PLUSF, P'T ZICKP, 010
  SAVE(SAVE) = J + 3
  C'E
  SAVE(SAVE) = J
  E'L
  QQ = SAVEC(1)
  TOOK = 1
  E'L
  O'E
  SECOND = 0B

OKK2
  W'R PLUSF, P'T ZICKP, 011
  QQ = FINDEL.(ELM,CCNT-1,BAD)
  E'L

QQERR
  W'R PLUSF, P'S QQ
  W'R PLUSF, P'S SAVEC...SAVEC(SAVEC)
  W'R PARK
R      IF THERE WAS A PARAMETER, CHECK FOR BEING
R      BROAD SCOPE HERE
  ZQQ = SEARCH.(FPAR,PARCNT,PAR,NOPAR)
  W'R PLUSF, P'S ZQQ
  W'R GETBRD.(ZQQ,BWORDS)
  QQ = BROAD(QQ)
  W'R PLUSF, P'S QQ
  E'L

```

NOPAR

```
PAR = ZPAR(ZQQ)
PSTO.(PAR)
W'R PLUSF, P'T ZICKP, 012
O'E
R          IF NO PARAMETER NAME, JUST STORE NUMBER(QQ)
BCDNUM.
E'L
W'R PLUSF, P'T ZICKP, 013
T'O SCAT
O'R MINM .L. MINP
W'R MINUSF, P'T ZICKM,003
R          ENTER HERE IF MINUS ZERO IS THE NEXT SUB.
COUNT = 0
MOVE.(START,MINM-1)
START = MINM
NXTM = NEXTM - 1
FIND.(ASTRIC,CARD,SC,MINUS(NXTM)-1,MINM+1)
R          COUNT NO. OF ASTRICS
W'R SC
W'R MINUSF, P'T ZICKM,004
SSC = COUNT(COUNT)
C'E
SSC = MINM
E'L
PAR = WORD.(SSC,MINUS(NXTM))
W'R COUNT .E. 0
W'R MINUSF, P'T ZICKM,005
R          PAR CONTAINS A FUNCTION NAME
PRINT COMMENT $Q**** HEY ... YOO HOO ... HERE'S ONE.....$
START = MINUS(NXTM)
MOVE.(MINM,START-1)
T'O LABEL
O'R COUNT .E. 2
W'R MINUSF, P'T ZICKM,006
R          PAR CONTAINS A STATEMENT LABEL NAME
QQ = SEARCH.(SLIST(1),SLIST(SLIST),PAR,GEN) * MULT
PSTO.(PREFIX)
T'O LABEL
GEN
W'R MINUSF, P'T ZICKM,007
SLIST = SLIST + 1
SLIST(SLIST) = PAR
R          ADD THE LABEL TO THE LIST
QQ = SLIST * MULT
PSTO.(PREFIX)
T'O LABEL
O'E
MOVE.(MINM+1,MINUS(NXTM-1))
PRINT COMMENT $Q**** ILLEGAL NUMBER OF ASTRICS ON THE FOLLOWI
ING CARD.$
R          AN ILLEGAL NUMBER OF ASTRICS APPEARED
E'L
O'E
W'R MINTST, P'T ZICK, 003
MOVE.(START,71)
QQT = Q - 12
W'R B(QQT+CONW).A.MASK.E.MB.OR.QQT.L.TOP
W'R MINTST, P'T ZICK, 004
```

CRD OUT

```

W = 1
P'T COUT,OUTPUT...OUTPUT(J)
W = 0
PUNCH FORMAT COUT, OUTPUT...OUTPUT(J)
W'R FLAG
W'R MINTST, P'T ZICK, 005
TOOK = TOOK + 1
W'R TOOK .LE. SAVEC
W'R MINTST, P'T ZICK, 006
QQ = SAVEC(TOOK)
Z = J
(I=1,1,I.G.QQZ,J=SAVE(I),BCDNUM.)
J = Z
T'O CRDOUT
E'L
W'R MINTST, P'T ZICK, 007
.STZ. QQZ
SELM = $$
E'L
W'R MINTST, P'T ZICK, 008
J = 77777K
E'L
W'R MINTST, P'T ZICK, 009
E'L
W'R MINTST, P'T ZICK, 010
COLECT
W'R MINTST, P'T ZICK, 011
MULT = MULT + 1
F'N
P'R ZICK(ZICKI)
I'N FINDEL.(SEL,CNT)
W'R TSTSCN, P'T ZICK, 032
SSD = 1B
E'O FINDL.
W'R TSTSCN, P'T ZICK, 033
W'R SSD
W'R TSTSCN, P'T ZICK, 034
F'N SEARCH.(CONL,CNT,SEL,NELM1)
NELM1
SSD = 0B
W'R TSTSCN, P'T ZICK, 035
E'L
W'R TSTSCN, P'T ZICK, 036
F'N SEARCH.(CONR,CNT,SEL,NOELM2)
NOELM2
ERROR RETURN
E'N
BAD
W'R SECOND
PRINT COMMENT $0**** COULDN'T FIND ELEMENT ON THE LIST.$
QQ = 987
T'O QQERR
E'L
SECOND = 1B
ELM = -ELM
T'O OKK2
NOAT
PRINT COMMENT $90***** ATTACHMENT COULD NOT BE FOUND IN ATS T
TABLE.$
P'S PELT(BEGIN)...PELT(BEGIN+4),ATS(PELT(BEGIN))...ATS(PELT(
1BEGIN1)-1)
MOVE.(Z-1,STIP)

```

```

T'O SCAT
P'R ZICK(ZICK)
V'S FUN = $1H 72C1*$
V'S ASTRIC = $*$
V'S COL = 10
V'S COM = 10
V'S COMLET = $R$
V'S COMMA = $,$
V'S CONW = 1,0
V'S COUT = $'W'((1H )72C1,'FLAG'(/8('W'((1H )T'COL',1H1'COLS'C1)
1 )*$
V'S ERK = $H90***** TOO MANY ASTRICS (*) IN SPECIFICATION ON
1CARD 9/12C6*$
V'S MINO = $0$
V'S LPAREN = $( $
V'S MULT = 1
V'S PLUSO = $0$
V'S PREFIX = $ZZW$
V'S RPAREN = $) $
V'S STATE = 11
V'S TSTSCN = 1B
V'S MASK = 7700K
V'S MB = 6000K
FORMAT VARIABLE COL, COLS, W
FORMAT VARIABLE FLAG
BOOLEAN PARK,CMK,RPK,LPK
BOOLEAN GETBRD.
BOOLEAN FLAG
BOOLEAN SC
B'N SSD,SSSD
B'N TSTSCN
EQUIVALENCE (CARD(71),CARDE)
EQUIVALENCE (PARK,PAR),(RP,RPK),(LP,LPK),(CM,CMK)
EQUIVALENCE (COUNT,SC)
EQUIVALENCE (SAVE,QQZ,FLAG)
EQUIVALENCE (SSD,SSSD)
D'N MINUS(18),PLUS(18)
D'N OUTPUT(659)
D'N COUNT(4)
EQUIVALENCE (Z,MINP)
B'N SECOND
V'S MINTST = 0B
V'S PLUSF = 0B
V'S MINUSF = 1B
V'S ZICK = $H9 ***** SSCAN -- GENERAL TRACE STATION 915*$
V'S ZICKP = $H9 ***** SSCAN -- PLUS TRACE STATION 915*$
V'S ZICKM = $H9 ***** SSCAN -- MINUS TRACE STATION 915*$
V'S ZICKI = $H9 ***** SSCAN -- INTERNAL FUNCTION STATION 915
1*$
V'S TSTSCN = 0B
E'N

```

*

\$ASSEMBLE,PUNCH OBJECT

SEARCH00

* SEARCH SIMULATOR CORE4
* CALLED BY SSCAN1
*

BOYD NOV 64

* ROUTINE TO SEARCH A LIST LOOKING FOR A SPECIFIC ENTRY
* RETURNS IN AC LIST SUBSCRIPT WHERE ENTRY IS FOUND
* IF THE SAME ENTRY OCCURS MORE THAN ONCE ONLY THE LAST WILL BE FOUN
*

* ARGUMENTS

* 1,4 ZERO LOCATION OF LIST
* 2,4 LAST ENTRY ON LIST
* 3,4 WORD TO BE FOUND
* 4,4 EXIT IF ENTRY NOT FOUND
*

* CALLED FROM SSCAN1 TO SCAN FPAR LIST FOR VARIABLE NAMES, SO THAT
* THE UNIQUE THREE CHARACTER NAME CAN BE FOUND ON THE ZPAR LIST
*

ENTRY SEARCH

SEARCH	CLA	1,4	ZERO LOCATION
	STA	CAS	
	CLA*	2,4	LAST ENTRY
	PAX	,1	
	CLA*	3,4	
CAS	CAS	** ,1	
	TRA	*+2	
	TRA	OUT	WORD FOUND
	TIX	CAS,1,1	LOOP TO SEARCH BACKWARDS
	AXT	0,1	
	CAS*	CAS	CHECK ZERO LOCATION
	TRA*	4,4	WORD NOT FOUND, ERROR RETURN
	TRA	*+2	
	TRA*	4,4	WORD NOT FOUND, ERROR RETURN
OUT	PXA	,1	WORD FOUND, PUT INDES IN AC
	TRA	5,4	NORMAL RETURN
	END		

*

PART B

REGRESSION PROGRAM

TABLE OF CONTENTS

Section		Page No.
	INTRODUCTION	B-3
I	GENERATION OF A PREDICTING EQUATION	B-5
II	ARTIFICIAL INTELLIGENCE APPLIED TO THE STEPWISE REGRESSION	B-12
III	COMMUNICATION OF INFORMATION TO THE REGRESSION PROGRAM	B-20

INTRODUCTION

In every system for which simulation is an important method of analysis there are many components or elements whose behavior must be modeled. In some cases, there may exist sufficient theoretical background in physics, mathematics, and allied sciences to allow an analytical model to be derived. But in most cases the real world physical element will behave somewhat differently from the performance expected by theoretical considerations alone. Real world "noise" such as gearing friction, surface roughness, and variations in physical properties conspire to cause the actual components to deviate from ideal behavior.

In other cases, particularly in new fields of exploration, there may not yet exist sufficient evidence to complete an analytical modeling. In fact, it is the true purpose of experimentation and test to build the basis of facts required for such analysis.

The Regression Program described in this report can be of major assistance in each of the foregoing situations. This procedure -- developed over a period of several years and applied in many diverse areas of physical and sociological systems -- enables the careful researcher, engineer, or scientist to explore an apparently unrelated set of data for relationships that can be of significant help in deriving both an analytical model of the phenomenon and, later, an insight into the nature of the process itself.

The Regression Program builds such a model with the assistance of the experimenter and tests this model against the set of observations supplied. In the process a heuristic model of the phenomenon is constructed and used to explore the problem further beyond the initial help supplied by the experimenter. Because most realistic systems problems permit so many possible terms and variables, the heuristic procedure reduces an almost impossible, if not economically prohibitive, exhaustive search through all possibilities to a much smaller problem in adaptive processes.

The following three sections are designed to (1) provide an understanding of this procedure and (2) give detailed instructions on the method of using this computer program and preparing the data for this analysis. Those readers desiring to use the program in specific problems may use the third section as a manual. It should be pointed out that very little information is absolutely mandatory. The "standard" values

supplied by the program in the absence of specific instructions from the user have been selected after many, many uses of the program. These values will allow a new user to get some results almost immediately. But with experience in particular problems the user may wish to supply other values to get improved performance.

In addition, new features allow the user to insert special arbitrary functions to be used in the regression analysis. Prior to this development the user who had some insight leading him to wish, say, an exponential or sinusoidal function to be employed did so with considerable difficulty.

These and other innovations are detailed in the following sections.

SECTION I

GENERATION OF A PREDICTING EQUATION

Consider a simple example; suppose that an experiment has been made consisting of a set of observations of six variables. Regarding one of the six as a dependent variable and the remaining five as predictor or independent variables, the analysis determines the "minimal" set of variables which may be used in a relation of the form

$$Y = b_0 + b_1 X_1 + b_2 X_2 + \dots + b_p X_p \quad (1)$$

where, in this case, $p = 5$.

The first step is to find that variable X_i which best predicts Y . This is done by correlating each of the X_i to Y and selecting that X_i which has the greatest "correlation coefficient"* in absolute value. If more

*The correlation coefficient is defined as the product-moment coefficient of correlation: Let

$$X_i X_j = \sum_t W_t X_{it} X_{jt} - \frac{(\sum_t W_t X_{it})(\sum_t W_t X_{jt})}{\sum_t W_t}$$

where

t	=	number of observations
n	=	number of independent variables
j	=	$i, i+1, i+2, \dots, n+1$
i	=	$1, 2, \dots, n+1$.

Then let

$$a_i = \sqrt{X_i X_i} \quad i = 1, 2, \dots, n+1$$

and the correlation coefficient r is then

$$r_{ij} = \frac{(X_i X_j)}{(a_i)(a_j)}$$

with the properties

$$\begin{aligned} r_{ji} &= r_{ij} & i, j &= 1, 2, \dots, n+1 \\ r_{ii} &= 1.000 \\ -1 &\leq r_{ij} \leq 1. \end{aligned}$$

than one X_i shares the largest value, take the X_i with the lowest subscript i . That is, take the first such X_i encountered. Suppose that in this instance that best i is 4. The first predicting equation is then

$$Y = b_0 + b_4 X_4 . \quad (2)$$

The b_0 and b_4 satisfy the least-squares criterion.

Succeeding steps are of slightly different form. First, the X_i are sorted into two subsets $X_{i,1}$ and $X_{i,2}$. The set $X_{i,1}$ consists of all those variables that are in the predicting equation at the time of sorting. The set $X_{i,2}$ consists of all those variables that are not yet in the predicting equation.

For each of the members of $X_{i,1}$ the analysis computes an "importance factor" ** which is a measure of the relative contribution of the variable to the predicted equation. The smallest of these importance factors is

** The "importance factor" is found by using the variance contribution for each variable. Initially, the correlation matrix r_{ij} defined earlier is equal to the regression matrix a_{ij}

$$a_{ij} = r_{ij} ; \quad i, j = 1, 2, \dots, n+1.$$

Then the variance contribution for the i -th variable is:

$$V_i = \frac{a_{iy} a_{yi}}{a_{ii}} \quad i = 1, 2, \dots, n$$

and where the subscript y is understood to be the dependent variable subscript $(n+1)$.

NOTE: If $V_i > 0$, then X_i is not yet in the regression equation and the $V_i > 0$ may be regarded as the relative contribution by the respective X_i in explaining the as yet unexplained variance in the dependent variable Y .

If $V_i < 0$, then X_i is currently in the regression equation. The $|V_i|$ for all $V_i < 0$ may be regarded as the relative contribution by the respective X_i to the regression prediction of Y .

As each term is added or deleted from the regression equation, the regression matrix a_{ij} is modified to contain the corresponding effect.

isolated. If the variable associated with this factor is less important than the user requires for the variable to be retained in the equation, then that variable is removed from the equation before continuing.

The scale used to determine whether a variable meets the "importance" criterion is simply the probability or chance that the user is willing to take that a variable may be left in the predicting equation that should have been removed. The F-test measures the extent to which a variable will contribute toward explaining the dependent variable behavior, and tests this contribution against a purely chance correlation by comparing the variance with and without the term. The hypothesis tested is that the variance is equal in both cases and that any difference is due only to chance. Thus, the term will be used only when the difference in variance cannot be explained by chance alone. Thus, if one selects a probability of committing an insertion error (that is, inserting a term into the predicting equation that really does not belong in the equation) and finds the number of degrees of freedom (roughly the number of weighted observations), the value of F indicated by the surface is such that if the value of F displayed by the "best" term exceeds the value on the surface, then the risk is less than the probability chosen. As might be expected, the value of F on the "threshold" surface goes to zero as the probability goes to 1 (certainty of committing an error). In that case, any non-negative value of F equals or exceeds the "threshold" and the result would be to insert every term whether correlated or not. Conversely, if one goes toward zero probability (certainty of not committing an error) the threshold value grows, approaching infinity in the case of zero probability. Thus no value of F can exceed this threshold and so no terms can be inserted. For any reasonable probability, the effect of the number of data can be assessed. As the number of data grows large, the "threshold" value approaches a constant dependent only on the probability. As the number of data approach zero, the risk of error is held constant by requiring larger and larger F values with infinity as the value corresponding to "no data." With such a test, the Stepwise Regression Program can control the generation of a predicting equation so that each term possesses a maximum risk of appearing incorrectly. Of course, many, perhaps most, terms actually appearing in the final equation exceed the threshold by substantial amounts and thus represent greatly reduced risks. The test insures that every term is at least as good as the risk specified.

If the user takes a probability of error for removing variables of 0.05 then the odds are 1 in 20 that a term may be left in the predicting equation incorrectly. Obviously, if the user wants to make this error

very rarely, he may set the probability of that error very low, say 0.01 or 0.001. This situation requires one additional remark. When one asks that the chance of committing an error be made small, the chance of committing the converse error must become large. In this case, one increases the risk of removing variables that really belong in the equation by decreasing the risk of leaving variables that do not belong in the equation. If the chance of leaving a variable incorrectly were set by the user at 1 in 10,000, it is also possible that insufficient data may have been accumulated to allow the retention of any variables in the equation, and the analysis can do no more than predict the average value of the dependent variable Y by the appropriate b_0 . The remedy is clear: if one wishes to set high standards, the price is additional experimentation to produce additional evidence to support the case.

If and when all the importance factors exceed the minimum value required for retention of the set $X_{1,1}$, the analysis then examines the set $X_{1,2}$. For each element of this set a "potential importance factor" (as defined earlier), is determined and the largest of these isolated. These factors measure the relative contribution which each variable not presently in the predicting equation might make to the equation if it were put in. The largest of these is associated with the "best" variable at this stage. Once again a comparison is made to insure that the risk of inserting a variable incorrectly is in agreement with the significance of the "best" variable before the insertion is allowed to occur. Again, the user specifies the risk he is willing to take of a variable being incorrectly inserted into the predicting equation, understanding 0.05 to mean 1 chance in 20 of the error occurring and recognizing that reducing the chances of incorrectly inserting variables increases the chances of omitting correct variables from lack of evidence.

Suppose that, in the example considered, X_4 has been retained, and that of X_1 , X_2 , X_3 , X_4 , the variable X_1 best explains the behavior of Y not explained by X_4 . If there is sufficient evidence to support the insertion of X_1 , then a new predicting equation is formed by least squares:

$$Y = b_0^{(1)} + b_1 X_1 + b_4^{(1)} X_4 . \quad (3)$$

The superscripts on b_0 and b_4 indicate that these coefficients have undergone one modification in the process and are new values. At this point the variables are again sorted and checked for importance and the procedure repeated. The analysis ceases when either all the X variables have been inserted into the predicting equation or none of the X variables that remain as possible candidates for the equation is sufficiently important to allow insertion.

Continuing the example, suppose that on the third step X_5 is introduced, yielding:

$$Y = b_0^{(2)} + b_1^{(1)} X_1 + b_4^{(2)} X_4 + b_5 X_5. \quad (4)$$

Further suppose that X_1 and X_5 behave together in such a way that the results is like having X_4 in the equation twice. In such a case, the importance of X_4 might be considerably reduced. Suppose that this is the case and that the importance of X_4 falls below the limit set by the user. Then X_4 is removed and the equation becomes:

$$Y = b_0^{(3)} + b_1^{(2)} X_1 + b_5^{(1)} X_5. \quad (5)$$

In step (5) suppose that X_2 is added giving:

$$Y = b_0^{(4)} + b_1^{(3)} X_1 + b_2 X_2 + b_5^{(2)} X_5. \quad (6)$$

Now suppose that neither X_3 nor X_4 are sufficiently significant to allow their insertion. The final prediction equation produced is (6). The analysis makes available a number of statistics at each step which may be interpreted as a measure of goodness of fit or prediction as well as the b values and the importance level for the term considered at that step.

The Statistical Model

Suppose that the physical system giving rise to the preceding example was such that it could be hypothesized that the system could be characterized or described by the mathematical model:

$$Y = B_0 + B_1 X_1 + B_2 X_2 + B_3 X_3 + B_4 X_4 + B_5 X_5 + E, \quad (7)$$

where the B_i ($i=0, 1, 2, \dots, 5$) are unknown and possibly some of them may be zero. E is a random error variable term which accounts for the inability to obtain strictly reproducible data when observing the

physical system. Setting aside the consideration of E for the moment, the problem is that of obtaining the best estimates of the B_i . It may be observed immediately that this is the problem just considered, resulting in Equation (6), and that the B_i are estimated by b_i , respectively. The best estimates of B_3 and B_4 are zero.

Turning attention once again to E in Equation (7), it is clear that Equation (6) is not quite complete. The randomness of E makes the prediction of E impossible. What is possible is the determination of the likelihood of E being inside a range of values. In other words, because of E the measurements obtained are not exactly repeatable even if all the X 's could be set at exactly their former values. Therefore, the estimates are possibly, but not necessarily, in error due to the influence of E . A more nearly complete treatment of Equation (7) would

- (1) estimate the B_i as before;
- (2) estimate the possible errors in the B_i ; and
- (3) estimate the variability of E .

The Stepwise Regression Program automatically estimates each of the three items desired. The estimate of the B_i has already been discussed. The possible errors in the B_i are indicated by quantities S_{B_i} called the "Standard Error of the Coefficient" for each i . These values are such that if one forms the interval

$$B_i - S_{B_i} \leq B \leq B_i + S_{B_i} \quad (8)$$

then the "true" value of B may be expected to be included by this interval in about 68% of all cases. If one extends the interval to form

$$B_i - 2S_{B_i} \leq B \leq B_i + 2S_{B_i}, \quad (9)$$

then this interval should include the true value in about 95% of all cases.

The variability of E is measured by a statistic called "The Standard Error of Estimate". This is roughly the standard deviation of the E . Adding additional terms to the predicting equation usually results in reducing the standard error of estimate. The amount of reduction is a measure of the contribution made by that variable toward the explanation of Y . When the analysis is completed, this statistic measures the behavior of Y not explained by the predicting equation and reflects the remaining observational errors and, of course, possible errors in the

hypothetical model. The precision of the predicting equation is reflected by the magnitude of the Standard Error of Estimate (S_y) such that if one uses the predicting Equation (6) to estimate Y and then forms a band about the curve predicted by (6) of plus and minus S_y (i. e. , the band is $2S_y$ in width and centered on the curve from (6)), then the "true" value of Y may be expected to be included by this band in about 68% of all cases. Again, doubling the band width to $\pm 2S_y$ raises the expectation to about 95% of all cases. In other words, when enough experimental observations of a physical system are made accurately on good instruments so as to minimize observational errors, and when the hypothetical model correctly describes the physical system, then S_y will be small and the predicting equation may be used to estimate Y with a measure of the precision of this estimate interpreted as indicated.*

The analysis produces two other valuable statistics at each step of the estimation process. The "Coefficient of Determination" ** is interpreted as the proportion of the total variation in Y that is explained by the predicting equation. The possible values lie in the range from +1.00 (perfect prediction) to 0.0 (no prediction). Statisticians familiar with the "Multiple Correlation Coefficient", which is the positive square root of the Coefficient of Determination, will find it displayed also.

*It should be mentioned in passing that the interpretation of S_y and S_{B_i} should be as stated here and that it is not correct to say that about 68% of all observed values will lie within the intervals indicated for $\pm S$ and so on.

**The Coefficient of Determination (R^2) is found by subtracting the regression matrix element a_{yy} (which measures the dependent variable variance) from unity. That is, $R^2 = 1. - a_{yy}$.

SECTION II
ARTIFICIAL INTELLIGENCE APPLIED TO THE
STEPWISE REGRESSION METHOD

Section I of this discussion treated the use of the Stepwise Regression Method as it applied to those cases in which the entire set of variables and functions of variables can be represented by a single collection of small enough size to allow complete retention within the memory of the machine. While some expansion might be realized by adroit programming, a little study of the nature of the problem indicates that an expansion in capacity of several orders of magnitude together with a new concept of programming will be required to handle problems of the types commonly encountered in research.

To understand the nature of the problem encountered, consider the following example. Suppose that, as in the example of Section I, an experiment has been made consisting of a set of observations of six variables. Once again we regard one of the variables as a dependent variable and the remainder as predictor or independent variables. Assuming for the moment that only linear behavior is to be expected from any variable (a drastic simplification), it is apparent that the formal relation

$$Y = b_0 + b_1 X_1 + \dots + b_p X_p \quad (1)$$

is not completely descriptive of even this simplified case. This is because of the possible existence of interactions between variables. Extending the example proposed to include interaction requires the inclusion of sets of terms of the forms

- (1) X_i ,
- (2) $X_i X_j$,
- (3) $X_i X_j X_k$,
- (4) $X_i X_j X_k X_l$

and, in this case, the single term $X_1 X_2 X_3 X_4 X_5$ as possible candidates for the predicting equation. The number of such terms is found in the following way.

Let there be K groups of n_i distinct objects ($i = 1, 2, \dots, k$) and let there be selected j objects

$$j \leq \sum_{i=1}^K n_i$$

at a time to form combinations. The number of such combinations is readily obtained for the case $n_i = 1$ for all i (the present example case). The number is (for $n_i = 1$)

$$N_j = K! / (K - j)! j!$$

and the case of $K = 5$ produces the table.

TABLE OF N_j FOR $n_i = 1$ and $K = 5$

j	N_j	$\sum N_j$
1	5	5
2	10	15
3	10	25
4	5	30
5	1	31

The table shows that even the simple example chosen has expanded the required storage capacity from a $6 + 1$ square matrix of 49 locations to a $32 + 1$ square matrix of 1089 locations. Furthermore, the usual problem does not permit the assumption of $n_i = 1$. The more general case may be determined if

- (1) n_i is constant for all i ;
- (2) selection occurs always between groups and not within groups.

Then

$$N_j = \left[K! / (K - j)! j! \right] n_i^j . \quad (13)$$

Condition (1) is not unreasonable and condition (2) simply requires that n_i be large enough to include whatever terms might be desired to be generated within the smaller group. That is, if one considers X^2 and X^3 and wishes also to consider $X^5 = X^2 * X^3$, then condition (2) requires that X^5 be made a member of the n_i (and not generated from X^2 and X^3).

Suppose that in the example, 10 function choices are suggested for each of the five variables (the use of 20 or more is not uncommon in problems concerning a single independent variable). Neglecting interactions the problem requires $52 * 52$ locations. Considering interactions and using (13), one obtains the table:

TABLE OF N_j FOR $n_i = 10$ AND $K = 5$

j	N_j	N_j
1	50	50
2	1000	1050
3	10000	11050
4	50000	61050
5	100000	161050

Obviously this is outside the range of even projected computers since the matrix now requires $(161052)^2$ locations.

Conventionally, work has progressed in this field by the expedient of setting the coefficients of all but a very few of these terms identically equal to zero. The formal relation (1) is such a reduction. This method, while enabling some attack to be made on otherwise nearly hopeless problems, suffers greatly for several reasons. First of all, the choice of omitted terms is a process of discarding thousands of terms to retain one. Secondly, the usual practice of relying on apparent fit to select terms before the regression process begins may result in the omission of exactly the terms needed.*

*Experience with the regression program on single independent variable problems indicates that the terms added successively to the predicting equation bear little relation after the first step to their partial correlation coefficients with respect to the dependent variable. This is because the added terms are always charged with explaining the as yet unexplained variation in the dependent variable. Consequently, if the first term entered explains the dependent variable behavior quite well, the next term may be of quite different character in order to explain what is left by the first term.

A procedure is needed to conduct a search through thousands of possible terms engaging only a few dozen at a time to produce the predicting equation. To be as effective as possible, it would be very desirable to use each experience with the problem, whether successful or not, to learn more about the nature of the terms that are generally useful and thereby accelerate the search. Such techniques as "learning" and the "acquiring of experience" are generally associated with nonmechanistic organisms. Since it is proposed that these techniques be simulated by the computer, this is the application of artificial intelligence to the problem.

The program has been written so that the machine is not presented with the condensed subset, as usually happens, but instead is given access to all possible terms and interactions within the bounds of the number of variables considered and the number of function choices per variable allowed. As usual in problems of this type, no straightforward procedure can be given to proceed to the solution that does not also appear economically prohibitive. It is not a matter of instructing the machine how to solve the problem, but instead of instructing the machine how to "learn" to solve the problem. Specifically, the machine must "learn" how to select terms so that the set of terms chosen contain those terms needed to produce predicting equations of high precision. Much remains to be done in this new and vital area. The present effort contains only the most rudimentary learning but is written in such a way that more sophisticated learning models can be inserted fairly easily. Experience with the simple learning mechanism has been extremely encouraging.

Turning to the example of 5 variables and 10 functions per variable, the following discussion describes the nature of the learning scheme used by the program. Suppose that no knowledge of the nature of the more likely terms nor of the relative importance of the various term classifiers are known a priori. A "term classifier" is one of the set of (1) interaction order identifier, (2) variable identifier, or (3) function identifier, and is used to classify terms as to the degree of interaction, variables involved, and functions of variables involved in the term. If such knowledge is presumed known before commencing the solution, means are provided to suggest either the initial set of terms to try or the initial distribution of weight among the term classifiers or both or neither. In the present case, neither are assumed to be supplied so the discussion may be understood for any other case where more information is given initially.

Since term classifiers are not assumed to be supplied, the program assumes no previous experience with the problem and accordingly sets the relative likelihood of all terms equal. This is accomplished by considering each of the classifiers as an array the elements of which are the lengths of the components of a vector. Each component is initially set to a unit length.

Next the initial set of terms must be generated by the program. Each of the 161050 possible terms in this example are equally likely at this state. The program uses a pseudo-random number generator to select (1) an interaction classifier, (2) variables to satisfy the interaction selected, and (3) a function for each variable chosen for the interaction. As each term is selected, a check is made to be sure that it is not a duplicate of an earlier term chosen for the current pass. When the number of terms (less than 60) requested by the user for each pass have been chosen and entered in a term matrix, the program calls upon an editor program to examine the data and the term matrix and thereby generate the set of edited data required by the regression analysis program. The editing process consists of operating on the raw data by referring to the term matrix for the definitions of the terms and to subroutines to carry out the generation of the terms. Each raw observation is converted into the edited data and a magnetic tape recording of the result is made. When all the data have been edited, the program turns to the Stepwise Regression Program to carry out the analysis exactly as before with respect to the set of terms chosen by the program. Upon completion of the Regression Program for this selection of terms, a check of the generated predicting equation is made to see if:

- (1) the Coefficient of Determination is as large as the user specified,
- (2) the Standard Error of Estimate is as small as the user specified,
- (3) the number of passes executed have not exceeded the limit by the user.

If further work is allowed as the result of these checks, the program proceeds to examine the results of the pass just completed and in so doing acquires "experience" concerning the types of terms most suitable for future use.

This "experience" is acquired by the student program as follows. Each term is checked against the list of terms included in the predicting equation. If a term has been successfully used in the relation, the student (1) retains the term to be used again, and (2) increases the probability of trying similar terms by incrementing the lengths of the vector components of the classifier arrays that chose the terms. If the term was not successful, the student decrements the lengths of the vector components that selected the term. By modifying the vectors by amount proportional to their current size, no term will ever be reduced to zero probability but may have its probability made arbitrarily small but positive. In this way the arbitrary setting of huge blocks of coefficients to zero is avoided and any term may at any time be used successfully and thereby become a member of the predicting equation until supplanted by a still better term.

After the student program completes the study of the previous run, the "experience" gained is utilized to select a new set of trial terms for the next pass. This is to say, the previously successful terms are retained from the former pass and the term matrix is filled out with terms chosen by using the modified classifier arrays and the random selection process. Since the classifier arrays have been modified, the selection of new terms no longer occurs with equal probability for all interactions, variables, and functions. Thus the search is less random and becomes more nearly stepwise as success and failure direct the modification of the classifier arrays. So long as it is possible to retain terms used successfully on the previous pass and still select some additional term or terms, the program retains the previously successful terms. In this way, the new pass will always be at least as "successful" as the last pass. If, however, a new pass is called for and there is no room for additional terms, the program has encountered a "traffic jam" since a new pass would not be requested if the old selection had been good enough. In this situation, a fresh start is needed but old "experience" may still provide valuable assistance in the selection of term. The student program discards the old selection of terms (printout of the discarded set is automatic so that human study can be made of it) and selects a complete new set while retaining the "experience" imbedded in the classifier arrays. In this case the machine is completely able to handle the "traffic jam" without outside help.

Another pitfall which might be encountered by the program concerns the case in which the solution has progressed to a locally maximally successful predicting equation. In this case, any change appears to make the predicting equation less useful and yet the present predicting equation is not good enough. An interesting property of the Stepwise

Regression method for choosing the most desirable terms results in the ability of the program to work itself out of such a situation. In fact, several instances have been observed in which the program accepted somewhat poorer overall fits for one or two trials in order to retain particularly good terms and on a succeeding trial found the fitted predicting equation to be several times better than the best previous equation.

In any case the process repeats itself, studying, grading, selecting, editing, fitting, until the conditions on the goodness of fit are met or until the desired number of passes have been used whichever comes first. While one cannot be certain that the very best predicting equation possible has been found after any predetermined number of passes (a characteristic of iterative process generally,) the procedure insures that the best solution to date is preserved and that all trials contribute to the improvement of the selection process.

The learning scheme employed by the student program embodies many of the principles discussed by Friedburg, Dunham, and North in their articles on "learning Machines" in the I. B. M. Journal. The student program extends these ideas and incorporates the advantages of both random search and stepwise search. Initial passes search rather randomly looking for promising leads. As evidence accumulates, the mode of search becomes increasingly stepwise as the number of "good" terms retained grow. Thus the search narrows itself into promising areas and progress is made toward solution until either a solution is found or the allowable number of passes is exceeded or until a "traffic jam" forces the random search to begin again. Random searching of the early stages is most promising since a poor start does not inhibit progress. Later stages have experienced some success and therefore the modifications are less drastic to allow the previous leads to be followed as far as they may prove to be profitable.

During the solution of any particular problem, it may happen that, when the data are operated upon by the editor program to produce the edited data, the size of the numbers generated may overflow or underflow the size of the computer word. In floating point arithmetic this may occur whenever the editor produces a non-zero number with absolute value outside the range 10^{-18} to 10^{+18} because of a later production of the sums of the squares and cross products of the terms by the Stepwise Regression Program. In these circumstances, the student program cannot experience "learning" for those terms of correct size since they have not yet been tried for the actual curve fitting, but the student program must "learn" about the selection of terms acceptable

to the computer. Occasionally, it has been observed that the terms suggested by the curve fitting process and the terms acceptable in size to the computer may not agree. The present learning mechanism is capable of correcting itself in this case without requiring human intervention.

Some final remarks may be of assistance in understanding the analysis. First of all, a given set of data may result in more than one predicting equation of a specified goodness of fit. This corresponds to the existence of several mathematical models of the system. Classically, this situation leads to the development of experiments capable of distinguishing between the models and the retention of those models which best describe the greatest variety of consistent circumstances accurately. By randomly restarting the problem this possibility may be investigated. If the program produces different predicting equations upon random restarting, more evidence is needed. Failure to produce different equations, however, does not guarantee freedom from such difficulty but decreases the probability of this difficulty. Secondly, if previous experience with a problem is available, prudence usually dictates that the initial pass make full use of it. The program provides ready means for saving previous results and for restarting with any or all of the previous classifier arrays and term selections intact. This same philosophy may be extended to initial runs in which the user's training and experience or previous encounters with similar problems may serve to generate an initial selection and/or weighting. The penalty for a poor guess is an increased number of passes, but a good guess results in considerable saving.

The Stepwise Regression Program with Simple Learning has been used successfully on many test problems and actual physical component modeling problems. In addition, interest in this technique has been generated in many diverse areas of the physical and social sciences. The ability to know precisely the worth of each and every term in a predicting equation, as well as the worth of the equation as a whole, as it is supported by actual evidence, should enable extensions of knowledge in many fields.

SECTION III
COMMUNICATION OF INFORMATION TO THE
REGRESSION PROGRAM

Information governing output (printed and punched), type of analysis to be executed, amount of initial experience, and descriptions of the observations and variables, is conveyed to the program through several control cards. Parameter values are punched on these cards according to the write-up that follows. These control cards are placed directly after the \$ DATA specification card (blue). Information on how to read the control card write-up follows.

With every run there must be four control cards in this order:

1. Title Card
2. Second Control Card
3. Third Control Card
4. Learning Control Card

Depending on the contents of the problem control cards, one, several, or all of the following groups of cards may be required:

5. Ordered Term Insertion Cards
6. Data Deck
 - 6A. Format Specification Card
 - 6B. Observed Data Deck
 - 6C. Blank Data Set
7. Accumulated Learning Deck
8. Initial Pass Terms Deck

ITEM #6 MUST BE PRESENT IN EVERY RUN. Others are optional, but if included, must be placed in the exact order listed.

Mode

This is the mode of the parameter as used in the program. If integer, its value must be integer, and must be punched on the control cards without a decimal point.

If fixed point, its value may be any number within the RANGE as stated, and should be punched on the control cards with the decimal point in its proper place (i. e. XXX.XX)

If floating point, its value must be punched on the control cards with the floating point format (i. e. X.XXEXX)

If Binary Coded Decimal, any legal alpha-numeric character allowed by the MAD translator may be used.

Columns

The numbers referred to by columns are the actual column numbers on the control cards that contain value of the applicable parameter.

Standard Values

Certain of the parameters required on the following control cards may be left blank and the program will insert its own standard values. These values are listed as STD Value in the control card write up. If no STD Value is listed, and the card column pertaining to the parameter is left blank, the University of Michigan input routine will interpret it as a negative zero. This is different from zero. Hence if parameter (17) on control card 3 is left blank, the -0 will be interpreted the same as a -1 by the program. In most other instances, if an integer zero is desired, the field may be left blank.

Parameter Name

This is the name assigned to the parameter in the symbolic listing. These are of interest to those who have occasion to make an actual change in the program statements.

First Control Card: Title Card

<u>MODE:</u>	Binary Coded Decimal
<u>RANGE:</u>	Any legal alpha-numeric characters
<u>COLUMNS:</u>	2-72

The title card allows the user to present any title that may be desired to be printed at the beginning of a new problem. Only one card may be used and the title may appear anywhere within columns 2-72. Column 1

is reserved for a carriage control character. Consult the executive system write-up for the selection of carriage controls. Ordinarily the user will punch a numeric one in column 1 so that the problem print out will begin on a new page.

If a blank card is used, the printer will simply single space the paper.

Second Control Card

1) Estimate Coefficient of Determination

MODE: Floating Point

RANGE: 0. - 1.

STD. VALUE: .96

COLUMNS: 1 - 10

PAR. NAME: CODTRM

This parameter is the users estimate of the expected goodness of fit between the predicting equation surface and the data. Perfect agreement is represented by a value of 1.0; no agreement by 0.0.

2) Standard Error of Dependent Variable

MODE: Floating Point

COLUMNS: 11-20

PAR. NAME: SIGMAY

This parameter is the users estimate of the standard error of the dependent variable represented in this data. The value reflects the probable errors present in the data in units of the same kind as the data. The smaller the value of SIGMAY, the more difficult is the task of finding an acceptable function.

3) Probability of Insertion Error

MODE: Floating Point

RANGE: 0. - 1.0

STD. VALUE: .05

COLUMNS: 21 - 30

PAR. NAME: PRBIN

This value is the probability allowed by the user that the least significant term inserted into the predicting equation is erroneous. A value of .05 represents a risk of 1 chance in 20.

4) Probability of Deletion Error

MODE: Floating Point

RANGE: 0. - 1.0

STD. VALUE: .05

COLUMNS: 31 - 40

PAR. NAME: PRBOUT

This value is the probability allowed by the user that a term removed from the predicting equation for lack of support should have been allowed to remain in the equation.

The probability of deletion error must not exceed the probability of insertion error. If it does, the program may reject every term offered.

5) Tolerance for Division and Round Off Error

MODE: Floating Point

STD. VALUES: .0001

COLUMNS: 41 - 50

PAR. NAME: TOL

This value is a bound such that if the magnitude of any division is less than this value, no division will occur. This value is also used to limit round off error in the matrix manipulations.

6) Number of Independent (Predictor) Variables

MODE: Integer

RANGE: 1 - 59

COLUMNS: 51 - 55

PAR. NAME: NOIND

This value plus one is the total number of variables in the problem.

7) Number of Passes Allowed for this Problem

MODE: Integer

STD. VALUE: 3 (or 1 if the entire problem can be treated in one pass)

COLUMNS: 56 - 60

PAR. NAME: NOTRYS

This value is the allowed number of complete passes made on the problem. In general, several passes will be required. However, if the problem is simple enough (in terms of the number of possible terms that may be generated) it may be completed in 1 pass.

8) Number of Functions to be Considered for Each Independent Variable

MODE: Integer

STD. VALUE: 10

COLUMNS: 61 - 65

PAR. NAME: NOFNCT

The choice of functions to be used by the program is determined by the subroutine PFNCT.

Standard operation (no specially designated functions) provides for access to integer powers, integer roots, and their reciprocals. The extent of the set so generated depends upon the number of functions allowed. The order of functions is:

FUNCTION No. 1	-----	X (I) . P. 1
2	-----	X (I) . P. -1
3	-----	X (I) . P. 2
4	-----	X (I) . P. -2
5	-----	X (I) . P. 1/2
6	-----	X (I) . P. -1/2

and so on repeating the pattern of functions 3, 4, 5, and 6 above for each increasing integer.

If special functions are suggested by the user, the number of special functions must be subtracted from the total number of functions to obtain the number of standard functions that will be considered, for that particular run.

<u>Parameter Value</u>	<u>Standard Functions</u>
10	thru X (I) . P. -1/3
22	thru X (I) . P. -1/6
38	thru X (I) . P. -1/10

9) Number of Terms to be Tried at Each Solution Pass

<u>MODE:</u>	Integer
<u>RANGE:</u>	1 - 59
<u>STD. VALUE:</u>	40 (or the size of the set of all terms, if possible)
<u>COLUMNS:</u>	66 - 70
<u>PAR. NAME:</u>	NOTRMS

In general, the set of possible terms (i. e. products of functions of the independent variables) is very large. Thus it is not generally possible to try all possible terms at one time. The value of NOTRMS gives the size of the subset of terms to be investigated on each pass.

If it should be possible, because of the simplicity of a given problem to try all of the possible terms at one time, this value will be selected for NOTRMS. Otherwise, the most desirable value is one which permits the generation of enough terms to provide the adaptive simple learning feature of the program with some record of success and failure on which to base the choice of future terms.

Third Control Card

The first Ten Parameters Are Output Control

A numeric 1 suppresses the particular calculations and printing specified by the parameter. A zero or blank will represent no suppression.

Since in most cases, this represents a very sizeable output, the user is cautioned to select only those items of real interest. All ten parameters are integer.

	<u>OUTPUT</u>	<u>COLUMN NO.</u>	<u>PAR. NAME</u>
1)	Raw data	1	INFRWDA
2)	Raw sums of squares and cross products	2	IFRAW
3)	Average (mean) values	3	IFAVE
4)	Residual sums of squares and cross products	4	IFRES
5)	Standard deviations	5	IFSIG
6)	Partial correlation coefficients	6	IFCOEN
7)	Intermediate steps in regression process	7	IFSTEP
8)	Predictions using the intermediate step	8	IFPRDI
9)	Predictions using the final equation	9	LEPRED
10)	Values of terms for each set of observations	10	IFPRNT

If all of the above are suppressed, the output will consist of:

- 1) A listing of the special functions included, if any, and their function number in PFNCT.
- 2) Definitions of terms used for each pass.
- 3) Final equation found for each pass, with pertinent statistics.

- a) The F-level of the last term treated
 - b) The standard error of the independent variable
 - c) The coefficient of determination
 - d) The multiple correlation coefficient
 - e) The constant term, if any
 - f) The coefficients and their standard errors for all terms finally retained in the equation
- 4) The diagonal elements of the regression matrix. Use to determine whether the analysis terminated because of tolerance (TOL) problems or because of F-level of remaining terms not yet in regression equation.

For most applications, the automatically produced output is sufficient; however, the accumulated learning deck and the output external functions are generally desirable. The user is again cautioned against requesting all of the printout due to its size.

11) Output External Function Option

<u>MODE:</u>	Integer
<u>RANGE:</u>	0 - 3
<u>COLUMN:</u>	11
<u>PAR. NAME:</u>	IFOUT

The program punches and prints an external function as output that contains the predicting equation obtained after the last regression step. A numeric zero will give the subroutine in the MAD language, a numeric one, in the FORTRAN language, a numeric two, in both languages; and a numeric three will suppress the printing and punching of this output.

12) Number of Times Regression is to be Done With Random Selection of Terms

MODE:

TYPICAL VALUES: 1-4

COLUMN: 12
PAR. NUME: NORAND

Under normal operation, the regression analysis chooses, from the set of terms not yet in the predicting equation, the locally best term (i. e. with the highest F-level) tests it and, if successful, enters it into the equation.

However, there is the possibility of two or more terms, each of whose F-level is not quite as high as the best term, but whose combined effect better describes the data than does the single "locally best" term. Hence setting this parameter to a positive integer n will cause the regression to repeat itself n times randomly choosing terms with high F-levels on a integrated basis, and saving the best trial result.

The two parameters that might serve to indicate which trial was better are the coefficient of determination and the standard error of the dependent variable. The standard error of the dependent variable is too greatly influenced by large deviations of relatively few data points and makes a poor "goodness of fit" test. Generally, standard error will be better (i. e. lower) if the coefficient of determination gets better (i. e. higher). Therefore, the measure of which trial result is better is selected on the basis of the change in the coefficient of determination. The "best" result is that one with the largest coefficient of determination.

13) Learning Deck Printed and Punched Output Option

MODE: Integer
RANGE: 0, 1
COLUMN: 13
PAR. NAME: IFPCH

A numeric one suppresses the printing and punching of the final status of the selector arrays and the terms, both of which comprise the experience to be fed into future runs of similar data.

14) Learning Arrays After Each Pass

MODE: Integer

COLUMNS: 14

PAR. NAME: IFLRN

An integer one suppresses the printing; blank or zero allows printing. The learning arrays will be printed after the last pass depending upon the value of IFPCH and will occur independently of this IFLRN.

15) Number of Interactions

MODE: Integer

STD. VALUE: 3

COLUMNS: 15 - 16

PAR. NAME: NOINTR

This integer governs the order of interaction of the terms in the predicting functions.

16) Number of Terms Whose Order of Insertion is Specified by the User

MODE: Integer

RANGE: 0- numbers of terms tried at each solution pass

COLUMNS: 19 - 23

PAR. NAME: NOINIT

A non-zero value forces complete control of the order of insertion of terms, by the user.

In general, this can be done safely and properly only when the user also controls the definitions of the terms to be generated. The effect is to cause the insertion of terms, in the order specified, whether significant or not until the complete set (as specified) has been inserted.

At this point, normal regression analysis is allowed to evaluate the situation and remove any terms found to be below the specified significant level.

The control allows the creation of a multiple term relation (perhaps suggested by theory) in which the individual terms are not as significant as the combination.

17) Number of Terms Initially Defined by User

<u>MODE:</u>	Integer
<u>RANGE:</u>	0 - number of terms tried at each solution pass
<u>COLUMNS:</u>	24 - 28
<u>PAR. NAME:</u>	NOINT

Defined terms under this control will be used subject to the statistical analysis of the program unless overridden by control parameter (16). If less than the total terms tried at each solution pass are defined by the user, the program will attempt to generate enough new terms to satisfy that limit.

18) Parameter Controlling the Type of Regression Analysis Executed by the Program

<u>MODE:</u>	Integer
<u>RANGE:</u>	-1, 0, +1
<u>STD. VALUE:</u>	-1
<u>COLUMNS:</u>	29 - 33
<u>PAR. NAME:</u>	IFCNST

(A) If greater than zero, the data is treated with respect to the coordinate axes and the constant term is always suppressed to zero.

(B) If equal to zero, the data is treated with respect to a set of axes translated to the means of the variables. The constant term is not suppressed.

(C) If less than zero, the data is treated as in part (A) but the constant term is not suppressed. The constant term is treated just like every other term, except that the constant is always inserted as the first term in the relation and held until the next term is tried.

After this point, all constraints are removed.

Type A - Most useful when other information dictates a zero constant.

Type B - Most useful when dealing with data that tends to group itself about the means (biological and sociological problems).

Type C - Most useful in dealing with physical data.

19) Weighted Data Parameter

MODE: Integer

RANGE: 0, 1

COLUMNS: 34

PAR. NAME: IFWT

If data is all of unit weight, set this parameter equal to zero or leave column blank; if data is weighted individually, set the parameter equal to one.

If the parameter is one, each set of data must carry its weight.

20) Extra Format Specification Parameter

MODE: Integer

RANGE: 1 - 9 (never punch a zero)

STD. VALUE: 1

COLUMNS: 35 - 38

PAR. NAME: NUMCDS

If more than 1 and up to 9 format cards are needed to specify the format for the data deck, place the total number of format cards used in these columns. A zero punched here is an error; however, a one will be assumed by the program if these columns are left blank.

21) Problem Number

MODE: Integer
RANGE: MODULO 32768
STD. VALUE: 1
COLUMNS: 39 - 43
PAR. NAME: NOPROB

22) Output Function Name

MODE: Binary Coded Decimal
RANGE: 6 alpha-numeric characters the first of which must be alphabetic
STD. VALUE: Y Value
COLUMNS: 44 - 49
PAR. NAME: FNAME

This parameter is the name assigned to the output external function. The rules for allowable function names are those of the MAD translator (or FORTRAN or both) depending on the value of IFOUT (parameter 11).

23) Previous Experience Parameter

MODE: Integer
RANGE: 0, 1
COLUMNS: 50, 51
PAR. NAME: IFTRWT

If the parameter is equal to one, the program assumes the accumulated learning deck is present. If the accumulated learning deck is present, the user must make certain that any special functions suggested on that previous run that punched the learning deck, are again included in that same order.

24) Number Of Special Functions

MODE: Integer

RANGE: 0 - number of functions to be considered for each independent variable

COLUMNS: 52, 53

PAR. NAME: NOSPFN

When set to zero, standard PFNCT operation giving integer powers and roots and their reciprocals, is assumed. If the user wishes to suggest other special functions such as (in MAD notation) SIN., ATAN., ELOG. or any other function for which there is an evaluation routine available. This parameter is set to the total number of special functions suggested.

Special Functions

The user, upon observing past runs or from physical theory or intuition, might wish to make other functions besides integer powers, roots, and reciprocals available to the program.

This can be done in the following manner:

1. Set the control parameter NOSPFN (control Card #3) equal to the number of special functions suggested.
2. Include a special function card before the data format card (s), listing the functions by their proper name as they would appear in a MAD calling statement, calling on the subroutine that evaluates them (i.e. include the suffixing dots (periods) required by the MAD translator). List these names consecutively in fields of seven columns starting in column one and not extending beyond column 70. Columns 71 - 80 may be used for identification. For example, if the user wishes to make the sine and arc tangent function available to the program, as well as a function he has supplied called FUNCT., this card would have SIN. in column 1 - 7, ATAN. in columns 8 - 14, and FUNCT. in columns 15 - 21.

3. Include a subroutine to evaluate any functions suggested which are not in the system library.

For example, consider the function FUNCT, which the user wishes to define as $(\cos(X_1)) * (\log_e(X_1))$ and supply to the program. His evaluation routine might be of the following form.

\$ COMPILE MAD

EXTERNAL FUNCTION (X1)

ENTRY TO FUNCT.

FUNCTION RETURN (COS. (X1))*(ELOG. (X1))

END OF FUNCTION

These subroutines must be placed before the binary deck (before the \$BINARY specification card)

4. Include a subroutine exactly as follows:

\$ COMPILE MAD

EXTERNAL FUNCTION

ENTRY TO DATA.

EXECUTE LIST. (V)

VECTOR VALUES V = —, —, —

FUNCTION RETURN

END OF FUNCTION

The sole purpose of this subroutine is to get the special function evaluation subroutines loaded into core. The function names must appear in the VECTOR VALUES statement in the same order and form that they appeared on the data card in (2).

For example:

VECTOR VALUES V = SIN. , COS. , ELOG.

Fourth Control Card: Learning Control

The user may, at this stage in the development of artificial intelligence programs, control some of the characteristics of the learning mechanism. Use of the external function structure for the program allows fairly easy modification of the various parts of the program. With the "standard learning mechanism" as it is now used, data is accumulated concerning three kinds of selections.

- 1) Order of Interaction
- 2) Variables Entering Interaction
- 3) Functions of the Variables

A term is generated by selecting interaction order, next the variable to be concerned in the interaction and finally the functions of the variables selected. The term is the cross product of the functions of the variables selected. The program must "learn" which interactions are most useful in explaining the data, which variables are most useful, and which functions of those variables are most useful. The program "learns" by trying to use terms selected by the program to explain the data. If the mechanism has selected a term which is supported by the data and retained by the regression analysis, the mechanism that selected that term is modified so as to be more likely to select a term of a similar character. On the other hand, if a term is not supported by the data and is, thus, of no apparent utility in the equation it is cast out and the mechanism is adjusted to be less likely to select terms of similar character. Since the probability of selection of any component of any allowed term should be bounded positive, the program uses a "half-life" constant to modify the probabilities. In this way, the relative probability of any term may be made arbitrarily small but remains positive.

- 1) Number of Constants

<u>MODE:</u>	Integer
<u>RANGE:</u>	3 - 12
<u>STD. VALUE:</u>	3
<u>COLUMNS:</u>	1 - 5
<u>PAR. NAME:</u>	NOGRDS

The standard mechanism uses three constants. If the mechanism is modified to require more constants, succeeding cards (up to 2) are of format (E 21.8, 3E16.8). The format of the first card is (15, 4E16.8).

2) Half Life Of Interaction Selector

MODE: Floating Point

RANGE: See * below

STD. VALUE: 3.0E00

COLUMNS: 6 - 21

PAR. NAME: Grades (1)

If the standard value of 3.0E00 is used, three consecutive successes will double the present probability (or conversely, three consecutive failures results in a halving of the present probability).

3) Half Life Of Variable Selector

MODE: Floating Point

RANGE: See * below

STD. VALUE: 3.0E00

COLUMNS: 22 - 37

PAR. NAME: Grades (2)

4) Half Life Of Function Selector

MODE: Floating Point

RANGE: See * below

STD. VALUE: 1.5E00

COLUMNS: 38 - 55

PAR. NAME: GRADES (3)

* Here 1.5E000 is used since any function may be used relatively infrequently and it is somewhat desirable to take more powerful action on each encounter.

A great deal of work remains to be done in exploring "learning" mechanisms. It should be observed that as the constants are made larger, the mechanism "learns" more slowly. In fact, for very large values of the constants the mechanism is essentially deactivated. Very small values of the constants, on the other hand, may cause wildly erratic behavior of the mechanism since each encounter so strongly distorts the relative probabilities.

Ordered Term Insertion Card(s)

If the parameter (16) of the Third Control Card is non-zero, the user must supply a set of cards to define the order in which terms are to be inserted. This allows an arbitrary equation to be generated without regard to the statistical analysis after which the statistical analysis may be used to discard those terms that are not sufficiently important to meet the deletion error criterion. If a theoretical relation is available for which a study is being made to determine how the relation may be improved, this feature may be useful. Otherwise, one must assume the risk that some of the theoretical terms will be displaced in the search. The user must be aware that the use of these term order cards is a severe constraint on the analysis and treat the results accordingly.

The format is 14I5. Each five columns contains a integer whose value is the number of a term to be inserted. The first number is the first term to be inserted, the second number is the second term and so on. For example, if parameter (16) on the Third Control Card were three and column five on the Ordered Term Insertion Card were six, column ten were three and column fifteen were one, the effect would be to insert term six, then term three and then term one after which the Stepwise Regression Program would examine the equation to be sure that these terms meet the deletion error criterion. If any of these terms fail the test they will be discarded. When all of the terms in the equation meet the deletion error test, the remaining terms not yet in the equation will be tested for insertion. From this point on the standard analysis is followed:

Data Deck Preparation

1. Format Specification Card

Since the data may come from various sources, the data deck allows the data format to be specified at execution time. This is done

by using a standard FORTRAN format statement beginning with the word FORMAT (beginning in column seven and ending in column thirteen, followed by any allowable format specification that can be placed on one card and terminating with a ")" right parenthesis in or before column 72. For example, the following format statements would be acceptable:

Column

7

FORMAT (5F10.2, E16.8)

or

FORMAT (4E16.7, F10.1/4E15.8)

This card must immediately precede the data deck, and is known as the Format Specification Card.

2. Data Cards

Following this card are the data cards. Arbitrary formats are allowed as described above. The data must be listed for each observation in the following order however. (All values are floating point numbers).

- 1) Observation Number. There must be a positive observation number for every observation. Run number one must appear once and only once.
- 2) Independent Variables. These values are listed in order following the observation number. The values must correspond to one observation group.
- 3) Dependent Variable. The variable whose value is to be predicted must follow the predictor or independent variables.
- 4) Weight of this Observation group. The weight of the group may be specified or can be assumed to be unity depending on the parameter (11) of Problem Control Card.

3. End of Data Card

The actual data is then followed by a complete blank data set which acts as a termination for the data. If any Observation Number is blank or less than or equal to zero, the data input is terminated at that point. Therefore, the user must take care in preparing the data deck so that the entire set of data will be read into machine storage.

The program automatically counts the weighted data sets to establish the degrees of freedom for the analysis. In this way, new data can be added to the data deck and/or old data can be deleted very easily.

The Accumulated Learning Deck

Whenever a multiple independent variable problem occurs in which a large number of functions are allowed for each independent variable and interactions of all orders are admitted, the result is the generation of a very large set of possible terms that may appear in a predicting equation. Since the most desirable equation consists of a "minimal" set of these terms comprising those terms most significant in explaining the dependent variable behavior, it becomes apparent that the analysis must usually perform a selection process while dealing with a segment of the entire set of terms at each encounter.

If it is possible to verify the validity of terms independently from their method of initial selection then it becomes feasible to allow the machine to select the terms using some heuristic method. The terms so selected will not always be the correct ones or even the "best" ones although the method of selection should certainly tend to operate in this way. The important point to observe is that the validity of the term is tested by the regression analysis independently of the selection and the regression analysis is, therefore, not affected in any way by the heuristic method of selection. Because of this, the heuristic term selection method is free to select terms using any convenient scale for choosing the terms. If the terms so selected are shown to have validity by the regression analysis then the heuristic method that selected the valid term is modified so as to be more likely to select similar terms. A converse action occurs whenever the term is shown to be invalid.

At the completion of each solution pass the current status of the selector mechanism is represented by a set of values which give:

- 1) The relative probability of each Interaction Order.
- 2) The relative probability of each Independent Variable.
- 3) The relative probability of each function of each variable.

Whenever no accumulated learning deck is available, parameter (12) of the problem control parameter card is set equal to zero. The result is that the program will then assign equal unit relative probability to all interactions, variables and functions of variables.

If previous encounters with similar problems have occurred, however, the program has already had "experience" with a similar problem and

can be allowed to take advantage of these encounters by providing the accumulated learning deck that was automatically produced at the conclusion of the former problem together with the new problem data. If the user desires to transmit this information to the program, parameter (12) of the problem control parameter card is set equal to one and the accumulated learning deck is placed after the data deck.

The user can also suggest his own experience to the program by preparing an accumulated learning deck. The format is 5E14.7. The relative probabilities are inserted in the following order:

- 1) Relative Probabilities for Interactions from first order to the maximum order for the problem.
- 2) The sum of the preceeding probabilities (1).
- 3) Relative Probabilities for Independent Variables, from the first to the last in the same order as they appear in the data deck.
- 4) The sum of the preceeding probabilities (3).
- 5) Relative probabilities for each function of the first variable followed by the sum of these probabilities.
- 6) Relative probabilities as in (5) for the second, third, etc. variables.

The preceeding items are punched successively in the available fields as specified by the format. No blank fields are permitted between groups since every field is interpreted consecutively.

The accumulated learning produced by the machine program has each relative probability normalized so that the mean relative probability is unity. In this way, a problem may be easily expanded to more variables, functions, etc. and still retain previous "experience" by making all new entries of unit value.

Because of the independent regression analysis of the terms chosen from the accumulated learning it must be emphasized that this mechanism cannot force the adoption of incorrect terms. Rather, such an incorrect set of "experience" would be modified progressively by the program. If the "experience" supplied is valid for the current problem the result is to speed the generation of the desired equation but invalid "experience" can only temporarily delay this generation.

In general, if good experience is available from previous similar problems or from the user's background the user is strongly advised to make use of it.

Initial Pass Terms Deck

The user may suggest any initial terms that may be desired for the first pass. If the suggested terms stem from theoretical considerations and the theory is in agreement with the data, such a suggestion will speed the generation of the predicting equation by insuring an early treatment of likely terms. Any number of terms may be given initially up to the total number of terms allowed for each pass. The number of terms to be so defined by the user is given by the parameter (17) on control card number three.

At the end of each problem and immediately following the production of the accumulated learning deck, the program produces a set of terms to begin the next encounter with the problem. If the problem is continued later, these terms may be supplied by simply including these cards following the accumulated learning deck and setting parameter (17) control card three.

If the user wishes to suggest terms the procedure is the following:

(The card format is 14F5.0)

- 1) Produce a card (or cards, if sufficient variables are present) as described below for each term desired.
- 2) Treat each consecutive field in the above format specification as a one to one correspondence with the independent variables in the problem.
 - 2A) Insert the appropriate exponent corresponding to the particular function that is desired.
 - 2B) Leave blank (or zero) every variable field not associated with the term.
- 3) Insert the interaction order in the field immediately following the last variable field.

For example:

Suppose the user is dealing with two independent variables and the "standard" definition of function and desires to form the term: $X(1).P.3 X(2).P. - 1/2$

The term is specified by punching a three (3.0) in column five, a negative one-half (-0.5) in column ten, and a two (2.0) in column fifteen. The two in column fifteen declares the term to be a second order interaction and thus produces the desired multiplication of the two components of the term considered above.

```

C
C
C CORE 7 -- PGEN
C (1) GENERATES THE FINAL EQUATION.
C (2) CALLED FROM CORE 3 AND CORE 4.
C PROCESS -- PRINTS AND PUNCHES THE SPECIFICATION CARDS, DIMENSION, AND
C IF APPLICABLE, THE INTEGER DECLARATION.
C CTERM -- PRINTS AND PUNCHES THE CONSTANT FOR EACH TERM.
C GENTRM -- PRINTS AND PUNCHES THE COMPONENTS OF EACH TERM (FOR FORTRAN
C PROGRAMS).
C GTERM -- PRINTS AND PUNCHES THE COMPONENTS OF EACH TERM (FOR MAD
C PROGRAMS).
C SUMEQN --
C (1) PRINTS AND PUNCHES STATEMENTS WHICH WILL SUM THE EQUATION.
C (2) PRINTS AND PUNCHES THE TERMINATING STATEMENTS OF THE PROGRAM.
C CORE 1 -- THE STARTER PROGRAM
C (1) STARTS NEW PROBLEMS.
C (2) PUNCHES THE LEARNING ARRAYS AND TERMS ARRAY IF REQUESTED.
C
C

```

SUBROUTINE CORE7

```

COMMON NO,J1,ERASE,ICOUNT,DMNY,IFOUT,ERAS,NOPASK,
1NOINKP,EERAS,NBEGIN,DDMMYY,ITAPE9,DM,NORAND,FNAME,
1NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
2,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRESO
3,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE
4DIT,NORGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN
5DEX,X,Z,NOIN,NOEXIT,
6ARRAY5,DEFR,SIGMA,SIGMCO,NINIT,N,SIGY,RSQ,R,VMAX,VMIN,VAR,NOMIN,NO
7 MAX,K,NOENT,FLEVEL,CNST,FLEVL,TERMS,DMY,FUNC,NOSPFN
DIMENSION ERASE(7),ERAS(3),EERAS(2),DDMMYY(7),TERMS(60,60),
1GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),
2 DATA(60),Z(60),FUNC(60),DMY(1000),
6ARRAY5(61,61),AVE(60),SIGMA(60),COEN(60),SIGMCO(60),FLEVL(60)
EQUIVALENCE(DATA,Z)
EQUIVALENCE(STDEV,COEN,X),(X(61),AVE(1))

```

```

C
C
IF (NBEGIN) 712,711,712
711 WRITE OUTPUT TAPE 6,3333
3333 FORMAT(60H0NO TERMS HAVE BEEN ENTERED INTO THE REGRESSION EQUATION
- )
GO TO 22
712 IF(NORAND) 713,713,714
713 IF(NOIN) 21,21,715
715 NIN = NOIN
GO TO 717
714 IF(NOINKP) 21,21,92
92 NIN = NOINKP
717 CONTINUE
IF(CNST)1,2,1
1 NIN =NIN+1
2 IF(IFOUT-2)400,400,21
400 IF(IFOUT-1)401,402,401
401 IFFOUT=0
GO TO 300
402 IFFOUT =1
300 CALL PROCES(FNAME,NOIND,NIN,IFFOUT)

```

```

      I=1
      IF(CNST)3,4,3
3     CALL CTERM(I,CNST)
      I=I+1
4     DO 20 I1=I,NIN
      I3 = I1 - I + 1
      CALL CTERM(I1,COEN(I3))
5     IF(TERMS(I3,60))8,20,8
8     J1=TERMS(I3,60)
      J2=0
      DO 11 J=1,J1
10    J2=J2+1
      IF(TERMS(I3,J2))9,10,9
9     A3=TERMS(I3,J2)
      J3=A3
      J3NFNS=J3-NOFNCT+NOSPFN
      IF(J3NFNS)332,332,331
331   FUNCT=FUNC(J3NFNS)
      IN1=1
      GO TO 333
332   FUNCT = A3
      IN1=0
333   IF(IFFOUT)350,350,250
250   CALL GENTRM(I1,J2,FUNCT,IN1)
      GO TO 11
350   CALL GTERM(I1,J2,FUNCT,IN1)
      11 CONTINUE
      20 CONTINUE
      CALL SUMEQN(FNAME,NIN,IFFOUT)
      IF(IFOUT-2)21,406,21
406   IFFOUT=1
      IFOUT=3
      GO TO 300
      21 CONTINUE
      22 CONTINUE
      NOEXIT=2
23    CALL CORE1(NIN)
      RETURN
      END

```

*

```
$ COMPILE FORTRAN,PRINT OBJECT,PUNCH OBJECT
```

```

C
C
C
C CORE 1 -- THE STARTER PROGRAM
C   (1) STARTS NEW PROBLEMS.
C   (2) PUNCHES THE LEARNING ARRAYS AND TERMS ARRAY IF REQUESTED.
C PARAM -- READS IN THE LAST THREE CONTROL CARDS AND ESTABLISHES
C   STANDARD VALUES FOR COMMON VARIABLES.
C DATARD -- READS IN THE NAMES OF THE EXTERNALLY SUPPLIED FUNCTIONS.
C
C DATAIN -- READS IN THE FORMAT FOR THE DATA AND THE DATA.
C
C RDTRWT -- READS IN THE TERM WEIGHTS IF SUPPLIED BY THE USER.
C
C CMTRWT -- COMPUTES THE TERM WEIGHTS IF NOT SUPPLIED BY USER.
C
C TRMCHK -- CHECKS THE TERMS IN THE EQUATION TO INSURE INDEPENDENCE.
C PKTRM -- PICKS NEW TERMS, I.E. IT RANDOMLY PICKS INTERACTION ORDERS,
C   VARIABLES, FUNCTIONS, AND WHEN APPLICABLE, THE EXTERNAL
C   FUNCTIONS TO FILL IN THE TERMS ARRAY.
C CORE4 -- THE EDITOR PROGRAM.
C   (1) EVALUATES THE EQUATION AND CHECKS FOR COMPATIBILITY WITH THE
C   MACHINE WORD SIZE.
C   (2) EVALUATES THE EQUATION FOR ALL DATA SETS.

```

```

C
C
C
C
C
C
      REWIND 3
      IF (LRAM) 21,20,21
21  IF(NOEXIT-1)22,20,22
22  WRITE OUTPUT TAPE 6,103,NOPROB
      103  FORMAT(12H0PROBLEM NO.15,25H ... NORMAL TERMINATION  ////)
      IF(IFPCH)200,200,20
200  CONTINUE
300  CALL TAPE1(ITAPE3,NORCD3(1))
      READ TAPE ITAPE3,ARRAY1,ARRAY2,ARRAY3
      PUNCH 101,((ARRAY1(I),I=1,NOIND),ARRAY1(60),((ARRAY2(I,1),I=1,NOIND)
,ARRAY2(60,1),((ARRAY3(I,J),J=1,NOFNCT),ARRAY3(I,60),I=1,NOIND)
101  FORMAT(5E14.7)
      DO 641 I=1,NNI

```

```

PUNCH 101,(TERMS(I,J),J=1,NOIND),TERMS(I,60)
641 CONTINUE
20 NOEXIT=0
  NODATA=0
  LRAM=1
  DO 30 I=1,60
    DO 30 J=1,60
      30 TERMS(I,J)=0.
C THE FIRST CONTROL CARD.....
C READ AND PRINT HEADING
  READ INPUT TAPE 7,102
102 FORMAT(72H
-
  WRITE OUTPUT TAPE 6,102
  DOIN = PARAM(0)
  IF(NOSPFN)114,114,112
112 NPFN=NOSPFN-1
  DOIN = DATARD(NPFN,FUNC) + DOIN
  WRITE OUTPUT TAPE 6,620,NOSPFN,((I,FUNC(I)),I=1,NOSPFN)
620 FORMAT(10H6THERE ARE,I3,38H EXTERNALLY SUPPLIED SPECIAL FUNCTIONS/
1/4(13H FUNCTION NO.,I3,4H IS ,A6,5X)/)
114 CALL DATAIN
  IF(IFTRWT)1,2,1
1 CALL RDTRWT(ARRAY1,ARRAY2,ARRAY3,ITAPE3,NORCD3(1),NOIND,NOFNCT)
  GO TO 3
2 CALL CMTRWT
3 NOCNT=0
  IF(IFCNST)4,7,7
7 IF(NOINT)5,5,10
10 J=1
11 K=NOINT+J-1
  DO 6I1=J,K
    READ INPUT TAPE 7,101,(TERMS(I1,I),I=1,NOIND),TERMS(I1,60)
C READS IN THE TERMS IF INITALLY SUPPLIED BY THE USER.
6 CALL TRMCHK(N1,NOCNT,TERMS,NOIND,NOSTRT)
5 J1=NOCNT+1
  NOPASS=1
  IF(J1-NOTRMS)12,12,13
12 ISTL = 1
  DO 8 I1=J1,NOTRMS
9 CALL PKTRM(TERMS,LRAM,ARRAY1,ARRAY2,ARRAY3,NOIND,NOFNCT)
  CALL TRMCHK(N1,NOCNT,TERMS,NOIND,NOSTRT)
  ISTL=ISTL+1
  IF (ISTL-1000) 31,31,32
32 NOTRMS=I1-1
  WRITE OUTPUT TAPE 6,110, NOTRMS
110 FORMAT (41H0SELECTOR MECHANISM REDUCED NO. TERMS TO ,I5)
  GO TO 13
31 GO TO (9,8),N1
8 CONTINUE
13 IF (DOIN) 20,14,20
14 CALL CORE4
  RETURN
4 CALL TRMCHK(N1,NOCNT,TERMS,NOIND,NOSTRT)
  J=2
  IF(NOINT)5,5,11
  END

```

*

```

5  COMPIL FCRTAN, PRINT OBJECT, PUNCH OBJECT                                VARSRT00
C    VARSRT COMBINED WITH VINSRT. ALLOWS S.O. OR
C    TERM INSERTION, AS DESIRED.
    SUBROUTINE VARSRT
      COMMON NO,NSW,NVP,I,I1,ERASE,ICOUNT,ERAS,
1     1NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
2     2,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRES
3     3,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE
4     4DIT,NCRGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN
5     5DEX,X,Z,NOIN,NOEXIT,
6     6ARRAY5,DEFR,SIGMA,SIGMCO,NINIT,N,SIGY,RSQ,R,VMAX,VMIN,VAR,NOMIN,NO
7     7MAX,K,NOENT,FLEVEL,CNST,FLEVL
      DIMENSION ERASE(4),ERAS(21),FLVMAX(61),IVEC(60),
2     2DATA(60),Z(60)
      1GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),
6     6ARRAY5(61,61),AVE(60),SIGMA(60),COEN(60),SIGMCO(60),FLEVL(60)
      EQUIVALENCE(DATA,Z)
      EQUIVALENCE(STDEV,COEN,X),(X(61),AVE(1))
      IFRAND=1
      IF(NINIT)1,2,1
1     1 NSW=2
      NUP=(NOINIT-NINIT)+1
3     3 VMAX=0.
      VMIN=0.
      NOIN=0
      FLVMAX(61)=0.
      ITERA =1
      FLVIN=FLVL(PRBIN,DEFR)
301   DO 4 I=1,NUP
      GO TO (5,6),NSW
5     5 I1 = I
      GO TO 7
6     6 I1=INVAR(I)
7     7 IF(ARRAY5(I1,I1)-TOL)8,9,9
8     8 IF(ARRAY5(I1,I1)+TOL)10,41,41
41    41 ARRAY5(I1,I1)=0.
      GO TO 4
10   10 WRITE OUTPUT TAPE 6,100,I1,I1,ARRAY5(I1,I1)
100  100 FORMAT (18H ERROR *** ARRAY5(I2,1H,I2,3H) =E17.9,15H, TOO NEGATIVE
-.)
      R=0.
      N=1
200  200 RETURN
2     2 NSW=1
      NUP=NCTRMS
      GO TO 3
9     9 VAR=(ARRAY5(I1,NOZ)*ARRAY5(NOZ,I1))/ARRAY5(I1,I1)
      IF(INDEX(I1))11,12,11
11    11 NOIN=NOIN+1
      COEN(NOIN)=ARRAY5(I1,NOZ)*SIGMA(NOZ)/SIGMA(I1)
      SIGMCO(NOIN)=(SIGY/SIGMA(I1))*SQRT (ARRAY5(I1,I1))
      GO TO (13,14),NSW
13    13 IF(VMIN)15,16,16
15    15 IF(VAR-VMIN)14,14,16
16    16 VMIN=VAR
      NOMIN=I1
14    14 FLEVL(NOIN)=VAR*DEFR/ARRAY5(NOZ,NOZ)
      GO TO 4
12    12 VAR=ABSF(VAR)

```

```

      GO TO (120,18),NSW
120  IF(ICOJNT)17,17,870
      17 IF (VAR-VMAX)4,4,18
      18 VMAX=VAR
        NOMAX=I1
        GO TO 4
      870 FLVMAX(ITERA)=VAR*DEFR/(ARRAY5(NOZ,NOZ)-VAR)
        IF (FLVMAX(ITERA)-FLVIN)4,850,850
      850 IVEC(ITERA)=I1
        FLVMAX(61)=FLVMAX(61)+FLVMAX(ITERA)
        ITERA = ITERA +1
      4 CONTINUE
        N=3
        IF (ICOUNT)200,200,801
      801 ZERO=C.
        IF(FLVMAX(61)) 869,869,868
      869 NO =1
        FLEVEL=0.
        GO TO 200
      868 CONTINUE
        NO=0
        FSPOT=FLVMAX(61)*RANDOM(ZERO)
        HOLD=0.
        DO 810 ITERA = 1,60
          HOLD =HOLD+FLVMAX(ITERA)
          IF(HOLD-FSPOT)810,811,811
      810 CONTINUE
      811 I1=IVEC(ITERA)
        VMAX  =(ARRAY5(I1,NOZ)*ARRAY5(NOZ,I1))/ARRAY5(I1,I1)
        NOMAX=I1
        FLEVEL =FLVMAX(ITERA)
        EFIN=FLVIN
        GO TO 200
        END

```

*

C MATRIX TRANSFORMATION ROUTINE.

SUBROUTINE MATRAN

COMMON NO,I,J,ERASE,

1NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT

2,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRESO

3,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE

4DIT,NORGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN

5DEX,X,Z,NOIN,NOEXIT,

6ARRAY5,DEFR,SIGMA,SIGMCO,NINIT,N,SIGY,RSQ,R,VMAX,VMIN,VAR,NOMIN,NO

7MAX,K,NOENT,FLEVEL,CNST,FLEVL

DIMENSION ERASE(28),

1GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),

2DATA(60),Z(60)

6ARRAY5(61,61),AVE(60),SIGMA(60),COEN(60),SIGMCO(60),FLEVL(60)

EQUIVALENCE (DATA,Z)

EQUIVALENCE(STDEV,COEN,X),(X(61),AVE(1))

IF(K)1,1,2

1 WRITE OUTPUT TAPE 6,100,K

100 FORMAT (22H ERROR *** ILLEGAL K =14)

CALL CORE1

2 DO 3 I=1,NOZ

IF (K - 1) 4,3,4

4 DO 5 J=1,NOZ

IF (J - K) 18,5,18

18 ARRAY5(I,J) = ARRAY5(I,J)-ARRAY5(I,K)*ARRAY5(K,J)/ARRAY5(K,K)

5 CONTINUE

3 CONTINUE

DO 7 I=1,NOZ

IF(I-K)8,7,8

8 ARRAY5(I,K) = -ARRAY5(I,K)/ARRAY5(K,K)

ARRAY5(K,I)=ARRAY5(K,I)/ARRAY5(K,K)

7 CONTINUE

ARRAY5(K,K)=1./ARRAY5(K,K)

WRITE OUTPUT TAPE6,102

CALL PRINT2(ARRAY5,NOZ)

102 FORMAT(1H0,42(1H),H1TRANSFORMED MATRIX1//)

RETURN

END

*

```

$ COMPIL FORTRAN, PRINT OBJECT, PUNCH OBJECT                                GRADER00
C   GRADER ROUTINE USING HALF-LIFE VALUES.
C   N=1, IF TERM UNSUCCESSFUL, N=2 IF TERM SUCCESSFUL.
SUBROUTINE GRADER(A1,A2,A3,TRM,NOTRM,NOV,FIN,N,GRDS,NOPROB,NOGRDS)
COMMON B1,NO
DIMENSION A1(60),A2(60,2),A3(60,60),TRM(60,60),GRDS(12),FACTOR(6)
1,B1(180)
IF (NOGRDS) 1,2,2
1 NOGRDS = - NOGRDS
DO 3 I=1,3
FACTOR(2*I-1) = .5** (1./GRDS(I))
3 FACTOR(2*I) = 1./FACTOR(2*I-1)
2 IF (TRM(NO,60)) 9,5,4
9 TRM(NO,60)=-TRM(NO,60)
4 I1=TRM(NO,60)
A1(I1)=A1(I1)*FACTOR(N)
II=0
DO 6 I1=1,NOV
X=TRM(NO,I1)
IF (X-FIN) 11,11,10
10 I2 = X
GO TO 7
11 CONTINUE
IF(X) 12,6,13
C   IF THE EXPONENT IS POSITIVE ADD1, OTHERWISE ADD 2. IF THE
12 X=ABS(X)
NEG=2
GO TO 14
13 NEG=1
14 IF(X-1.) 16,15,17
C   EXPONENT IS AN INTEGER, SUB. 2, OTHERWISE ADD 0.
15 I2=NEG
GO TO 7
16 INTEGR = 0
X=1./X
GO TO 18
17 INTEGR = -2
18 I2=X
K=X+.5
IF(K-I2) 5,21,20
20 I2=K
21 I2 = 4*(I2-1) + NEG + INTEGR
C   NUMBER ASSOCIATED WITH THE EXPONENT IS GIVEN BY
C   4*(EXPONENT - 1) + EXTRA NUMBERS IF POSITIVE OR NEG AND IF
C   AN INTEGER OR A FRACTION.
7 A2(I1,1)=A2(I1,1)*FACTOR(N+2)
A3(I1,I2)=A3(I1,I2)*FACTOR(N+4)
6 CONTINUE
8 RETURN
5 WRITE OUTPUT TAPE 6,100,NOPROB,(I,TRM(NO,I),I=1,60)
100 FORMAT (42H0ERROR *** GRADER ENCOUNTERS ILLEGAL TERM./12H PROBLEM
-NO.15/4(8H TERM(I2,3H) =E15.7))
CALL CORE1
END

```

```

$COMPILE MAD      ,PRINT OBJECT, PUNCH OBJECT          DUMMY000
  EXTERNAL FUNCTION
  ENTRY TO DATA.
  PRINT COMMENT$6****ERROR YOU SUGGESTED USAGE OF SPECIAL FUNC
  TIONS AND FORGOT THE DUMMY SUBROUTINES$
  EXECUTE CORE1.
  FUNCTION RETURN
  END OF FUNCTION

```

```

$ COMPILE FORTRAN,PRINT OBJECT,PUNCH OBJECT          EDITR000 CORE4

```

```

C
C
C CORE4 -- THE EDITOR PROGRAM.
C   (1) CHECKS EACH TERM FOR COMPATIBILITY WITH THE MACHINE WORD SIZE.
C   MACHINE WORD SIZE.
C   (2) EVALUATES EACH TERM FOR ALL DATA SETS.
C   (3) CALLED FROM CORE 1 AND CORE3.
C TAPE1 -- TAPE1(I,J) SPACES TAPE I BY J RECORDS.
C ZFNCT -- CONTROLS THE EVALUATION OF EACH TERM , USING ONE DATA SET
C PER EVALUATION OF THE EQUATION ( THE DATA SETS BEING CONTROLLED
C BY THE EDITOR PROGRAM).
C CORE6 -- CONTROLS THE USE OF REGRESSION AND WINDUP.
C CORE7 -- (PEGEN) GENERATES THE FINAL EQUATION.
C CORE3 -- THE STUDENT.
C   (1) CONTROLS THE LEARNING PROCEDURES.
C   (2) REFILLS THE TERMS ARRAY.
C   (3) CALLED FROM CORE4 AND WINDUP.
C
C
C

```

SUBROUTINE CORE4

```

COMMON NO, ERASE, IFPCH, EERAS,
1NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
2,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRES
3,IFSIG,IFCOEN,IFSTEP,IFPRD1,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE
4DIT,NORGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN
5DEX,X,Z,NOIN,NOEXIT,
9 DMYXXX,TERMS,CRDTA
  DIMENSION ERASE(12), EERAS(17),
1GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),
2DATA(60),Z(60),TERMS(60,60),
5DMYXXX(3916),CRDTA(1000)
  EQUIVALENCE(DATA,Z)
    EQUIVALENCE(STDEV,COEN,X),(X(61),AVE(1))

```

```

WRITE OUTPUT TAPE 6,101,NOPROB,NOPASS,NOIND,NOTRMS
IFERR1=0

```

```

CALL TAPE1(ITAPE3,NORCD3(2))

```

```

  ISW=C

```

```

  ICC=C

```

```

DO 3 I=1,NODATA

```

```

  LOOP EVALUATES THE FUNCTION FOR ALL DATA SETS.

```

```

  MAX=ICC+NOVAR+2

```

```

  IF (MAX-1000) 901,901,902

```

```

  ICC=ICC+1

```

```

  RUN=CRDTA(ICC)

```

```

  DO 903 J=1,NOVAR

```

```

  ICC=ICC+1

```

```

  X(J)=CRDTA(ICC)

```

```

  ICC=ICC+1

```

```

WHT=CRDTA(ICC)
GO TC 904
902 IF(ISW) 905,906,905
906 REWIND ITAPE4
    ISW=1
905 READ TAPE ITAPE4,RUN,(X(J),J=1,NOVAR),WHT
C
C     IF THERE ARE MORE THAN 1000 +ATA POINTS (=(NOVAR*2)*NODATA),
C     THE REMAINING POINTS ARE ON TAPE 4.
C
904 CONTINUE
    CALL ZFNCT(1,X,NOIND,WHT,Z,TERMS,NOTRMS,NOFNCT,NOZ,IFERR
    -,NOSTRT,IFPRNT,NOPASS,RUN)
    IF (IFERR)4,3,4
    4 WRITE OUTPUT TAPE 6,100,RUN
100 FORMAT (23H0ERROR *** DATA SET NO.F5.0)
    IFERR1=1
    3 WRITE TAPE ITAPE3,RUN,(Z(I),I=1,NOZ),WHT
    IF(IFERR1)24,20,24
20 REWIND ITAPE3
    REWIND ITAPE4
    IF (IFCNST)5,6,6
    5 IF(NOPASS-1)7,8,7
    7 NOINIT=1
    GO TO 9
    8 IF(NOINIT)7,7,10
10 DO 11 I=1,NOINIT
    J1=NOINIT+2-I
    J2=J1-1
11 INVAR(J1)=INVAR(J2)
    NOINIT=NOINIT+1
    9 INVAR(1)=1
12 CALL CORE6
    6 IF(NOPASS-1)13,12,13
13 NOINIT=0
    GO TO 12
C     LOOP PUSHES THE GOOD TERMS DOWN INTO THE FIRST N POSITIONS.
24 DO 25 I=1,NOIN
25 INDEX(I)=1
    I=NOIN+1
    DO 26 I=1,NOTRMS
26 INDEX(I)=0
    NOIN = -NOIN
    NOPASS = NOPASS+1
    REWIND ITAPE3
    REWIND ITAPE4
    IF(NOTRYS-NOPASS)30,31,31
30 LRAM = 1.
    NOIN = -NOIN
    IF (IFPCH) 2,2,1
    1 CALL PRINT4(1)
    2 CALL CORE7
31 WRITE OUTPUT TAPE6,102,NOPASS,NOPROB
    NOEXIT =2
    CALL CORE3
    RETURN
102 FORMAT(9HOPASS NO.15,55H BEGUN (AFTER ERROR DETECTED BY EDITOR) FO
    -R PROBLEM NO.15////)
101 FORMAT(15H1EDITOR PROGRAM///12H PROBLEM NO.15/18H SOLUTION PASS NO

```

-•I5/31H NO. OF INDEPENDENT VARIABLES =I5/21H NO. OF TRIAL TERMS =I
-5//)
END

*

\$ASSEMBLE,PUNCH OBJECT
ENTRY STDRD

STDRD000

```

REM
REM
REM      SUBROUTINE TO INSERT STANDARD VALUES FOR
REM      NOFNCT,NOTRYS,NOINTR,IFCNST,NUMCDS,NOTRMS,TOL,PRBIN,
REM      PRBOUT,CODTRM,NOPROB,FNAME
REM
STDRD  REM
      SXA      HOLD1,4
      SXA      HOLD2,2
      AXT      12,2
LOOP   CLA*    1,4
      TPL      **3
      ARS      18
      TZE      BLANK
DECR   TXI     **1,4,-1
      TIX      LOOP,2,1
HOLD1  AXT     **,4
HOLD2  AXT     **,2
      CLA*    12,4
      SUB      BLNK
      TNZ      **3
      CLA      LIST
      STO*    12,4
      TRA      12,4
BLANK  CLA     LIST+1,2
      STO*    1,4
      TRA      DECR
      PZE      ,,10
      PZE      ,,3
      PZE      ,,3
      MZE      ,,1
      PZE      ,,1
      PZE      ,,40
      DEC      .0001,,.05,,.05,,.97
      PZE      ,,1
LIST   BCI     1,YVALUE
BLNK   BCI     1,
      END

```

*

```

$  COMPILER FORTRAN,PUNCH OBJECT,PRINT OBJECT                                MAIN0000
COMMON DMMY,NO,NOGRDS,EFOUT,EFIN,RUN,WHT,IFERR,NVP1,NUMCDS,ICOUNT,
1IFLRN,IFOUT,NOINTR,IFPCH,NOPASK,NOINKP,BLANK1,BLANK2,BLANK3,BLANK4
2,BLANK5,BLANK6,BLANK7,BLANK8,BLANK9,BLANK10,BLANK11,ITAPE9,IFRWDA,N
3ORAND,FNAME,DMMY,LRAM
DIMENSION DMMY(180),DMMY(55)
ITAPE9=9
CALL IOHSIZ(1)
CALL FTRAP
LRAM=0
REWIND ITAPE9
CALL CORE1
END

```

*

```

$  COMPILER FORTRAN,PUNCH OBJECT,PRINT OBJECT
      FUNCTION  ALTRMS (NOINTR,NOVAR,NOFUN)
5      IF (NOINTR-NOVAR) 2,2,1
1      NOINTR = NOVAR
2      IF (NOINTR) 4,3,4
3      NOINTR =NOVAR
      GO TO 5
4      F=NOFUN
      FI=FCT(NOVAR)
      IF(FI-1.E30)7,8,8
8      ALTRMS=999999.
      GO TO 9
7      ALTRMS=0.
      DO 6 J=1,NOINTR
      I=NOVAR-J
6      ALTRMS=ALTRMS+(FI*F**J)/(FCT(I)*FCT(J))
9      RETURN
      END

```

ALTRMS00

*

\$ COMPILE FORTRAN, PRINT OBJECT, PUNCH OBJECT

PRINT200

C. SPECIAL ARRAY PRINTER ROUTINE NO. 2.

```
SUBROUTINE PRINT2(A,N)
  DIMENSION A(61,61)
  NM1=N-1
  WRITE OUTPUT TAPE 6,100,(((I,J,A(I,J),J=1,NM1),I=1,NM1))
100  FORMAT(3(8H  TERM(I2,10H) VS TERM(I2,3H) =E15.7))
  WRITE OUTPUT TAPE 6,101,(I,A(I,N),I=1,NM1)
101  FORMAT(3(8H  TERM(I2,15H) VS      Y      =E15.7))
  WRITE OUTPUT TAPE 6,102,A(N,N)
102  FORMAT(6(1H )19HY      VS      Y      =E15.7)
  RETURN
  END
```

*

```

$ COMPILE FORTRAN, PRINT OBJECT, PUNCH OBJECT
C   SPECIAL ARRAY PRINTER ROUTINE NO. 3.
    SUBROUTINE PRINT3(A,N)
    DIMENSION A(61)
    NM1 =N-1
    WRITE OUTPUT TAPE 6,100,(I,A(I),I=1,NM1)
100  FORMAT(4(10H      TERM(12,3H) = E15.7))
    WRITE OUTPUT TAPE 6,101,A(N)
101  FORMAT(10(1H ),5HY  = E15.7)
    RETURN
    END

```

PRINT300

*

\$ COMPIL FORTRAN, PRINT OBJECT, PUNCH OBJECT

PICKE000

C PICKE PICKS THE EXPONENTS OR SPECIAL FUNCTION NUMBER.

SUBROUTINE PICKE(I,A,N,T)

COMMON B1,NO,B2,NOFNCT,B3,LRAM,DUM,NOSPFN

DIMENSION T(60,60),A(60,60),B1(180),B2(36),B3(48),DUM(8702)

4 A(I,60)=0.

DO 1 J=1,N

1 A(I,60)=A(I,60)+A(I,J)

RN = A(I,60)*RANDOM(LRAM)

T1=0.

DO 2 J=1,N

T1=T1+A(I,J)

IF(RN-T1)3,3,2

2 CONTINUE

GO TO 4

3 IF(J-NOFNCT+NOSPFN) 203,203,14

203 IF (XMODF(J,2))33,44,33

C ODD J FOR INTEGER POWERS AND ROOTS.

C EVEN J FOR RECIP. INTEGER POWERS AND ROOTS.

33 K=J/2+1

K2=1

GO TO 5

44 K=J/2

K2=2

5 K1=K/2+1

IF(XMODF(K,2))6,7,6

C ODD K, EXCEPT K=1, INTEGER ROOTS.

C K = 1, LINEAR TERM OR RECIPROCAL.

C EVEN K, INTEGER POWERS.

6 IF(K-1)8,7,8

7 T(NO,I)=K1

GO TO (12,10),K2

12 RETURN

10 T(NO,I)=-T(NO,I)

GO TO 12

8 T(NO,I)=1./FLOATF(K1)

GO TO (12,10),K2

14 T(NO,I)=J

GO TO 12

END

*

\$ASSEMBLE,PUNCH OBJECT
 REM MAD AND FORTRAN PROCESSING FUNCTION GENERATOR

PROCES00

```

PRINT1 CALL .PRINT
PUNCH1 CALL .PUNCH
ENTRY PROCES
PROCES SXA XRC,4
      SXA XRB,2
      SXA XRA,1
      CLA 4,4
      STA CHECK
      CLA* 4,4
      ARS 18
      STO S
      CLA =1
      SUB S
      STO N
      CLA* 3,4
      ARS 18
      STO NIN
      CLA* 2,4
      ARS 18
      STO NOX
      CLA 1,4
      STA FNAME
      STA FNAMES
      CALL .PRINT
      STR SYMTAB,,FRMT2
      STR
      CALL .PUNCH
      STR SYMTAB,,FRMT1
      STR
      CALL .PRINT
      STR SYMTAB,,FRMT3
      STR
      LXA TWO,2
      CLA PRINT1
RPT   STA OUTPT1
      STA OUTPT2
      STA OUTPT3
      STZ NOCARD
      CLA TWO
      STO VARN0
      LXD NUM1,1
OUTPT1 TSX **,4
      STR SYMTAB,,FRMT4
CHECK  ZET **
FNAMES STR **
      CLA NOX
      SUB ONE
      TZE RPRN
      ADD TWO
      STO NOX
LP1   STR COMMA
      STR X
      STR VARN0
      CLA VARN0
      ADD ONE
      STO VARN0
      SUB NOX
  
```

```

TEST1  TNX NXTCD,1,1
TEST2  TNZ LP1
RPRN   STR RPAREN
        STR
        CLA*    CHECK
        TNZ     **5
OUTPT3 TSX **,4
        STR     FRMT6
FNAME  STR **
        STR
        TNX RTRN,2,1
        CLA PUNCH1
        TRA RPT
NXTCD  STO TEMP
        CLA NOCARD
        ADD CNE
        STO NOCARD
        STR
        LXA NUM1,1
OUTPT2 TSX **,4
        STR     SYMTAB,,FRMT5
        STR NOCARD
        CLA TEMP
        TRA TEST2
RTRN   PRINT  FRMT8,NIN,0
        PUNCH FRMT8,NIN,0
XRC    AXT     **,4
XRB    AXT     **,2
XRA    AXT     **,1
        TRA     5,4
        BCI     1,N
        PZE     N,2
        BCI     1,S
        PZE     S,2
SYMTAB PZE     4
ONE    PZE 1,0,0
TWO    PZE 2,0,0
NUM1   PZE 15,0,10
NIN    PZE
TEMP   PZE 0,0,0
NOX    PZE 0,0,0
N       PZE
S       PZE
VARNO  PZE 0,0,0
NOCARD PZE 0,0,0
COMMA  BCD 1,
X       BCD 1X
RPAREN BCD 1)
FRMT1  BCI , 'N'((13H$ COMPILE MAD/),'S'((17H$ COMPILE FORTRAN/))14H$ PUNCH
        BCI     *,OBJECT/14H$ PRINT OBJECT*
FRMT2  BCI , 'N'((4H1MAD),'S'((H$1FORTRAN$)H$ EXTERNAL FUNCTION STATEMENTS F
        BCI ,OR PREDICTING EQUATION PRODUCED BY LAST REGRESSION STEPS//*
FRMT3  BCI , 'N'((14H $ COMPILE MAD/),'S'((18H $ COMPILE FORTRAN/))15H $ PUNC
        BCI     *,H OBJECT/15H $ PRINT OBJECT*
FRMT4  BCI ,S11,'N'((H$EXTERNAL FUNCTION$),'S'((H$FUNCTION $C6,)3H(X110(C1,
        BCI ,C1,I2)*
FRMT5  BCI ,S5,'N'((S5,)I1,'S'((S5,)15(C1,C1,I2)*
FRMT6  BCD 7S11,9HENTRY TO C6,1H./S11,9HINTEGER I*
FRMT8  BCD 5S11,12HDIMENSION T(I2,1H)*
        END

```

*

\$ COMPILE FORTRAN, PRINT OBJECT, PUNCH OBJECT

TSTLVL00

C F LEVEL TEST ROUTINE

SUBROUTINE TSTLVL

COMMON NO,N1,EFOUT,EFIN,ERASE,

1NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT

2,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRES

3,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE

4DIT,NORGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN

5DEX,X,Z,NOIN,NOEXIT,

6ARRAY5,DEFR,SIGMA,SIGMCO,NINIT,N,SIGY,RSQ,R,VMAX,VMIN,VAR,NOMIN,NO

7MAX,K,NOENT,FLEVEL,CNST,FLEVL

DIMENSION ERASE(27),

1GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),

2DATA(60),Z(60)

6ARRAY5(61,61),AVE(60),SIGMA(60),COEN(60),SIGMCO(60),FLEVL(60)

EQUIVALENCE (DATA,Z)

EQUIVALENCE (STDEV,COEN,X),(X(61),AVE(1))

N1=1

GO TO (1,2,3),N

1 FLEVEL=VMIN*DEFR/ARRAY5(NOZ,NOZ)

EFOUT=FLVL(PRBOUT,DEFR)

IF(EFOUT+FLEVEL)2,2,4

4 K=NOMIN

NOENT=0

INDEX(NOMIN)=0

GO TO 5

2 VX = ARRAY5(NOZ,NOZ) - VMAX

IF (VX) 100,7,101

100 IF (TOL + VX) 9,7,7

101 FLEVEL = VMAX * DEFR / VX

GO TO (6,7),N1

6 EFIN=FLVL(PRBIN,DEFR)

IF(EFIN-FLEVEL)7,8,9

7 K=NOMAX

NOENT=K

INDEX(NOMAX)=1

5 N=4

10 CONTINUE

RETURN

8 IF(EFIN)9,9,7

9 N=1

GO TO 10

3 N1=2

GO TO 2

END

```

$  COMPILER FORTRAN, PRINT OBJECT, PUNCH OBJECT                                RDTRWT00
  SUBROUTINE RDTRWT(ARRAY1,ARRAY2,ARRAY3,ITAPE,NORCD,NOVAR,NOFNCT)
C    READ TERM WEIGHT ROUTINE
      DIMENSION ARRAY1(60),ARRAY2(60,2),ARRAY3(60,60)
      READ INPUT TAPE 7,100,(ARRAY1(I),I=1,NOVAR ),ARRAY1(60),(ARRAY2(I,
-1),I=1,NOVAR),ARRAY2(60,1),((ARRAY3(I,J),J=1,NOFNCT),ARRAY3(I,60),
-I=1,NOVAR)
100  FORMAT (5E14.7)
      CALL TAPE1(ITAPE,NORCD)
      WRITE TAPE ITAPE,ARRAY1,ARRAY2,ARRAY3
      RETURN
      END

```

```

$ COMPILE FORTRAN, PRINT OBJECT, PUNCH OBJECT
C      TERM SELECTOR ROUTINE
      SUBROUTINE PKTRM(TERMS,LRAM,ARRAY1,ARRAY2,ARRAY3,NOVAR,NOFNCT)
      COMMON B1,NO
      DIMENSION TERMS(60,60),ARRAY1(60),ARRAY2(60,2),ARRAY3(60,60)
1,LRAM(3),B1(180)
      DO 1 I=1,60
1  TERMS(NO,I)=0.
      TERMS(NO,60)=PICKV(ARRAY1,NOVAR)
      CALL PICKS(TERMS,ARRAY2,ARRAY3,NOVAR,NOFNCT)
      RETURN
      END

```

*

```

$ COMPILER FORTRAN, PRINT OBJECT, PUNCH OBJECT
C  TERM CHECKER
  SUBROUTINE TRMCHK(N1,NOCNT,TERMS,NOVAR,NOSTRT)
  COMMON B1,NO
  DIMENSION TERMS(60,60),B1(180)
  IF(NOCNT)1,1,2
1  N1=2
  NOCNT=NOCNT+1
3  RETURN
2  DO 4 I=1,NOCNT
  IF (TERMS(NO,60)-TERMS(I,60))4,6,4
6  DO 7 I1=1,NOVAR
  IF (TERMS(NO,I1)-TERMS(I,I1))4,7,4
7  CONTINUE
  N1=1
  GO TO 3
4  CONTINUE
  GO TO 1
  END

```

TRMCHK00

*

\$ COMPILE FORTRAN, PRINT OBJECT, PUNCH OBJECT

PICKV000

```
FUNCTION PICKV(A,N)
COMMON B1,NO,ERASE,NOINTR,ERAS,LRAM
DIMENSION A(60),B1(180),ERASE(11),ERAS(73)
A(60)=0.
10 DO 1 I=1,NOINTR
1  A(60)=A(60)+A(I)
   RN = A(60)*RANDOM(LRAM)
   T1=0.
   DO 2 I=1,NOINTR
   T1=T1+A(I)
   IF (RN-T1)3,3,2
2  CONTINUE
   GO TO 10
3  PICKV=I
   RETURN
   END
```

*

\$ COMPILE FORTRAN, PRINT OBJECT, PUNCH OBJECT

PICKS000

```
SUBROUTINE PICKS(T,A2,A3,NV,NF)
COMMON B1,NO,B2,LRAM
DIMENSION T(60,60),A2(60,2),A3(60,60),B1(180),B2(85)
12 K1=1
NOPCKS =T(NO,60)
IF(NV-NOPCKS)1,1,2
1 T(NO,60)=NV
K=2
DO 7 I1=1,NV
GO TO 8
7 CONTINUE
GO TO 9
2 A2(60,1)=0.
DO 3 I=1,NV
A2(60,1)=A2(I,1)+A2(60,1)
3 A2(I,2)=A2(I,1)
A2(60,2)=A2(60,1)
K=1
DO 4 I=1,NOPCKS
RN = A2(60,2)*RANDOM(LRAM)
T1=0.
DO 5 I1=1,NV
T1=T1+A2(I1,2)
IF(RN-T1)6,6,5
5 CONTINUE
GO TO (2,11),K1
6 K1=2
A2(60,2)=A2(60,2)-A2(I1,2)
A2(I1,2)=0.
8 CALL PICKE(I1,A3,NF,T)
GO TO (4,7),K
4 CONTINUE
9 RETURN
11 DO 13 I=1,59
13 T(NO,I)=0.
GO TO 2
END
```

*

```

C
C
C PFNCT --
C (1) EVALUATES EACH COMPONENT AND PRINTS THE EQUATION.
C (2) CALLED FROM ZFNCT.
C STQUO -- I/O ROUTINE WHICH SAVES THE CURRENT CONTENTS OF THE I/O
C BUFFERS FROM THE PREVIOUS I/O CALL.
C SPEFUN -- FUNCTION WHICH CONNECTS WITH THE EXTERNALLY DEFINED
C FUNCTIONS FOR THE FUNCTION EVALUATION.
C
C
C FUNCTION PFNCT(X,I,A,N,JVAR,NOFNCT,IFER )
C COMMON DMYXXX,FUNC,NOSPFN
C DIMENSION X(120),FUNC(60),DMYXXX(8909)
C IFER=0
C J3=A+.5
C GO TO (1,2),JVAR
C
C JAVAR = 1 IF X REPRESENTS AN INDEPENDENT VARIABLE, = 2 IF X
C REPRESENTS A DEPENDENT VARIABLE.
C
C J7=J3-NOFNCT+NOSPFN
C
C IF J7 IS GREATER THAN ZERO, J7 CORRESPONDS TO AN EXTERNAL
C FUNCTION NUMBER.
C
C IF (J7) 13,13,204
13 GO TO (203,12),N
203 IF (A-1.0) 3,4,3
4 CALL STQUO
WRITE OUTPUT TAPE 6,101,I
101 FORMAT(S1, 2HX(,I2,1H))
GO TO 12
3 CALL STQUO
WRITE OUTPUT TAPE 6, 100,I,A
100 FORMAT(S1 ,2HX(,I2,6H) .P. ,S1,F10.5)
12 PFNCT=X(I) ** A
11 RETURN
2 PFNCT = X(I)
GO TO (16,11),N
16 CALL STQUO
WRITE OUTPUT TAPE 6,103,J3,I
GO TO 11
204 PFNCT=SPEFUN(J7,X,I)
IF(J7)207,208,208
208 GO TO (205,11),N
205 CALL STQUO
WRITE OUTPUT TAPE 6,104,FUNC(J7),I
104 FORMAT(S1 ,A6,3H(X(,I2,2H)))
GO TO 11
103 FORMAT(6HOTHERM(I2,6H) = X(I2,22H), DEPENDENT VARIABLE.)
207 J7=-J7
WRITE OUTPUT TAPE 6,110,FUNC(J7),I
110 FORMAT(1H ,A6,41HCANNOT HANDLE THIS COMPUTED VALUE FOR X(,I3,1H))
IFER=1
GO TO 11
END

```

*

\$ASSEMBLE, PUNCH OBJECT, PRINT OBJECT

SPEFUN000

	ENTRY	SPEFUN
	ENTRY	LIST
SPEFUN	SXA	HOLD,4
	SXA	XRB,2
	SXA	XRA,1
	CALL	DATA
LIST	CLA	1,4
	STA	LOC
	LXA	HOLD,4
	CLA*	1,4
	ARS	18
	SUB	=1
	PAX	,2
	CLA*	3,4
	ARS	18
	SUB	=1
	STO	FARHAD
	CLA	2,4
	SUB	FARHAD
	STA	PAR
LOC	CLA	==,2
	STA	*+1
	TSX	==,4
PAR	PAR	**
	PAR	ERLOC
RTM	LXA	HOLD,4
XRB	AXT	** ,2
XRA	AXT	** ,1
	TRA	4,4
ERLOC	CLA*	LOC
	SSM	
	STO*	LOC
	TRA	RTM
	ZERO	FARHAD,HOLD
	END	

*

\$ COMPILE FORTRAN, PRINT OBJECT, PUNCH OBJECT

PRINT400

C
C
C
C
C
C
C

PRINT4 --

- (1) PRINTS OUT THE SELECTOR ARRAYS.
- (2) CALLED FROM CORE3 AND CORE7.

SUBROUTINE PRINT4

COMMON NO,ERASE,

1NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
2,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRES
3,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE
4DIT,NCRGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN
5DEX,X,Z,NOIN,NOEXIT,
8ARRAY3,ARRAY2,ARRAY1
DIMENSION ERASE(30),
1GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),
2DATA(60),Z(60),
4ARRAY1(60),ARRAY2(60,2),ARRAY3(60,60)
EQUIVALENCE (DATA,Z)
EQUIVALENCE(STDEV,COEN,X),(X(61),AVE(1))

C
C

WRITE OUTPUT TAPE 6, 99

99 FORMAT(35H1PRESENT STATUS OF SELECTOR ARRAYS.)

WRITE OUTPUT TAPE 6,104

104 FORMAT(18H0INTERACTION ARRAY/1H0,4(29HINTERACTION NO. WEIGHT
-))

WRITE OUTPUT TAPE 6,100,(I,ARRAY1(I),I=1,NOIND)

100 FORMAT(4(I8,E21.7))

WRITE OUTPUT TAPE 6,101,ARRAY1(60)

101 FORMAT(17H0SUM OF WEIGHTS =E17.7//)

WRITE OUTPUT TAPE 6,105

105 FORMAT(15H0VARIABLE ARRAY/1H0,4(29H VARIABLE NO. WEIGHT))

WRITE OUTPUT TAPE 6,100,(I,ARRAY2(I,1),I=1,NOIND)

WRITE OUTPUT TAPE 6,101,ARRAY2(60,1)

WRITE OUTPUT TAPE 6,106

106 FORMAT(15H0FUNCTION ARRAY)

DO 1 I=1,NOIND

WRITE OUTPUT TAPE 6,103,I

103 FORMAT(13H0VARIABLE NO.I3/1H0,4(29H FUNCTION NO. WEIGHT))

WRITE OUTPUT TAPE 6,100,(J,ARRAY3(I,J),J=1,NOFNCT)

1 WRITE OUTPUT TAPE 6,101,ARRAY3(I,60)

RETURN

END

*

```

$ COMPILE FORTRAN, PRINT OBJECT, PUNCH OBJECT                                ZFNCT000
C   ZFNCT SUBROUTINE FOR REGRESSION PLUS LEARNER.
    SUBROUTINE ZFNCT (I ,X,NOIND,WHT,Z,TRMATX,NOTRMS,NOFNCT,NOZ,IFERR
- ,NOSTRT,IFPRNT,NOPASS,RUN)
    DIMENSION X(120),Z(60),TRMATX(60,60)
    IFERR=0
    NOVAR=NOIND+1
    ASSIGN 32 TO N
    IF(1) 200,200,201
200  GO TO N,(32)
201  CALL SET8(N)
    IF(I-1)1,2,1
    1 N=2
    GO TO 3
    2 N=1
    3 GO TO (4,5),N
    4 WRITE OUTPUT TAPE 6,100,NOPASS
100  FORMAT (36H TRIAL TERM DEFINITIONS FOR PASS NO.15)
    5 DO 6 I=1,NOTRMS
      Z(I)=1.
      IF(TRMATX(I,60))6,8,9
    8 GO TO (10,6),N
    10 WRITE OUTPUT TAPE 6,102, 1
102  FORMAT (6H0TERM(I2,23H) = 1.0, CONSTANT TERM.)
    GO TO 6
    9 I1=ABSF(TRMATX(I,60))
    GO TO (11,12),N
    11 WRITE OUTPUT TAPE 6,103,I,I1
103  FORMAT (6H0TERM(I2,25H) = INTERACTION OF ORDER I2,43H, WHERE THE C
-OMponents ARE DEFINED TO BE --)
    12 I2=1
    JVAR=1
    DO 13 J=1,I1
    18 IF(I2-NOIND)14,14,7
    14 IF (TRMATX(I,I2))16,15,16
    15 I2=I2+1
    GO TO 18
    16 A3=TRMATX(I,I2)
    GO TO (19,20),N
    19 WRITE OUTPUT TAPE 6,104,J
104  FORMAT (1H ,9X ,10HCOMPONENT(I2,3H) =N)
20  Z(I)=Z(I)*PFNCT(X,I2,A3,N,JVAR,NOFNCT,IFER )
    IF (IFER)32,30 ,32
    30 CONTINUE
    13 I2=I2+1
333  IF(Z(I))33,6 ,33
33  IF(ABSF(Z(I))-1.E18)31,34,34
31  IF(1.E-18-ABSF(Z(I)))6 ,34,34
32  WRITE OUTPUT TAPE 6,108,I,J
108  FORMAT(51H0FLOATING POINT TRAP OCCURED WHILE PROCESSING TERM(,I2,
- 17H) -- COMPONENT(,I2,2H).)
34  IFERR=1
    TRMATX(I,60)=-ABSF(TRMATX(I,60))
    WRITE OUTPUT TAPE 6,107,I,Z(I)
6  CONTINUE
    JVAR=2
    A3 = NOZ
    Z(NOZ)=PFNCT(X,NOVAR,A3 ,N,JVAR,NOFNCT,IFER)
40  IF (IFPRNT)21,22,21

```

```

22 WRITE OUTPUT TAPE 6,105,RUN,WHT,(I,Z(I),I=1,NOTRMS)
105 FORMAT(29H0TERMS COMPUTED FROM OBS. NO.F5.0,9H WEIGHT =F10.5/4(8H
- TERM(I2,3H) =E15.7) )
WRITE OUTPUT TAPE 6,106,Z(NOZ)
106 FORMAT (9(1H ),4HY =E15.7,21H (DEPENDENT VARIABLE))
21 CALL RSET8(0)
RETURN
7 WRITE OUTPUT TAPE6,120,I,(J,TRMATX(I,J),J=1,60)
120 FORMAT(15HUERROR *** TERM14/4(10H ELEMENT(I2,3H) =E15.7))
CALL RSET8(0)
CALL CORE1
107 FORMAT(24H0SCALING ERROR *** TERM(I2,3H) =E15.7)
END

```

*

```

$ COMPILE FCRTAN, PRINT OBJECT, PUNCH OBJECT                                SUMSQ000
C  SUBROUTINE TO LOAD MATRIX AND COMPUTE THE RAW SUMS OF SQUARES
C  AND CROSS PRODUCTS.
  SUBROUTINE SUMSQ
    COMMON NO,NVM1,I,J,RUN,WHT,IFERR,NVP1,ERASE,
1  NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
2  ,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRESO
3  ,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPREL,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE
4  ,DIT,NORGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN
5  ,DEX,X,Z,NOIN,NOEXIT,
6  ,ARRAY5,DEFR,SIGMA,SIGMCO,NINIT,N,SIGY,RSQ,R,VMAX,VMIN,VAR,NOMIN,NO
7  ,MAX,K,NOENT,FLEVEL,CNST,FLEVL
    DIMENSION ERASE(23),
1  GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),
2  DATA(60),Z(60)
6  ,ARRAY5(61,61),AVE(60),SIGMA(60),COEN(60),SIGMCO(60),FLEVL(60)
    EQUIVALENCE (DATA,Z)
    EQUIVALENCE (STDEV,COEN,X),(X(61),AVE(1))
    CALL TAPE1(ITAPE3,NORCD3(2))
    NVP1=NOZ+1
    DO 10 I=1,NVP1
    DO 10 J=1,NVP1
10  ARRAY5(I,J)=0.
    ASSIGN 32 TO N
    IF (1) 222,222,223
222  GO TO N,(32)
223  CALL FSPIII(N)
32  CONTINUE
    DO 11 N=1,NODATA
    READ TAPE ITAPE3,RUN,(DATA(I),I=1,NOZ ),WHT
    DO 12 I=1,NOZ
    ARRAY5(I,NVP1)=DATA(I)*WHT+ARRAY5(I,NVP1)
    DO 12 J=1,NOZ
12  ARRAY5(I,J)=DATA(I)*DATA(J)*WHT+ARRAY5(I,J)
11  ARRAY5(NVP1,NVP1)=ARRAY5(NVP1,NVP1)+WHT
    DEFR=ARRAY5(NVP1,NVP1)
    IF(DEFR)1,1,2
    2 IFERR=1
    WRITE OUTPUT TAPE 6,100,DEFR
100  FORMAT(30H WEIGHTED DEGREES OF FREEDOM =F12.2///)
    IF(IFRAW)4,3,4
    3 WRITE OUTPUT TAPE 6,101
101  FORMAT(53(1H ),14H SUM OF TERMS//)
    NVM1=NOTRMS
    WRITE OUTPUT TAPE 6,102,(I,ARRAY5(I,NVP1),I=1,NVM1)
102  FORMAT(4(10H      TERM(I2,3H) =E15.7))
    WRITE OUTPUT TAPE 6,103,ARRAY5(NOZ ,NVP1)
103  FORMAT(10(1H ),5HY      =E15.7)
    WRITE OUTPUT TAPE 6,104
104  FORMAT(1H0,38(1H ),42HRAW SUM OF SQUARES AND CROSS PRODUCTS//
- )
    CALL PRINT2(ARRAY5,NOZ)
    4 REWIND ITAPE3
    GO TO (5,6),IFERR
    5 RETURN
6  CALL CORE1
    1 WRITE OUTPUT TAPE 6,108,DEFR
108  FORMAT(17H ERROR *** DEFR =E17.9)
    IFERR=2
    GO TO 3
  END

```

```

$ COMPILE FORTRAN, PRINT OBJECT, PUNCH OBJECT                                PRRCRCN00
C   PARTIAL CORRELATION COEFFICIENTS
    SUBROUTINE PRRCRCN
      COMMON NO,NVM1,I,J,IP1,ERASE,
1     NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
2     NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRESO
3     IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE
4     DIT,NORGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN
5     DEX,X,Z,NOIN,NOEXIT,
6     ARRAY5,DEFR,SIGMA,SIGMCO,NINIT,N,SIGY,RSQ,R,VMAX,VMIN,VAR,NOMIN,NO
7     MAX,K,NOENT,FLEVEL,CNST,FLEVL
      DIMENSION ERASE(26),
1     GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),
2     DATA(60),Z(60),STDEV(60),
6     ARRAY5(61,61),AVE(60),SIGMA(60),COEN(60),SIGMCO(60),FLEVL(60)
      EQUIVALENCE (DATA,Z)
      EQUIVALENCE (STDEV,COEN,X),(X(61),AVE(1))
      DEFR=DEFR-1.
      DO 6 I=1,NOZ
        IF (ARRAY5(I,I))1,2,3
1     IF (ARRAY5(I,I)+TOL)4,5,5
4     WRITE OUTPUT TAPE 6,100,I,I
100  FORMAT (18H ERROR *** ARRAY5(I2,1H,I2,15H) TOO NEGATIVE.//)
      CALL PRINT2(ARRAY5,NOZ)
      CALL CORE1
5     IFSIG=0
      IFCOEN=0
      WRITE OUTPUT TAPE 6,101,I,I,ARRAY5(I,I)
101  FORMAT (8H ARRAY5(I2,1H,I2,3H) =E17.9)
2     SIGMA(I)=1.
      GO TO 6
3     SIGMA(I)=SQRT (ARRAY5(I,I))
6     ARRAY5(I,I)=1.
      IF(IFSIG)7,8,7
7     NVM1=NOTRMS
      DO 9I=1,NVM1
        IP1=I+1
        DO 9 J=IP1,NOZ
          ARRAY5(I,J)=ARRAY5(I,J)/(SIGMA(I)*SIGMA(J))
9     ARRAY5(J,I)=ARRAY5(I,J)
      IF(IFCOEN)10,11,10
11  WRITE OUTPUT TAPE 6,102
102  FORMAT (1H0,42(1H ),34HPARTIAL CORRELATION COEFFICIENTS//)
      CALL PRINT2(ARRAY5,NOZ)
10  RETURN
8     SQDF=SQRT (DEFR)
      DO 12 I=1,NOZ
12  STDEV(I)=SIGMA(I)/SQDF
      WRITE OUTPUT TAPE 6,103
103  FORMAT (1H0,49(1H ),20HSTANDARD DEVIATIONS//)
      CALL PRINT3(STDEV,NOZ)
      GO TO 7
      END
$ COMPILE FORTRAN, PRINT OBJECT, PUNCH OBJECT                                RSDSUM00
C   RESIDUAL SUMS OF SQUARES AND CROSS PRODUCTS.
    SUBROUTINE RSDSUM
      COMMON NO,NVM1,I,J,NVP1,IFERR,ERASE,
1     NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
2     NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRESO

```

```

3, IFSIG, IFCOEN, IFSTEP, IFPRDI, IFPRED, IFPRNT, GRADES, NOSTRT, NSTDNT, NOE
4DIT, NORGRS, ITAPE3, NORCD3, ITAPE4, NORCD4, LRAM, INVAR, NOSTEP, NOPASS, IN
5DEX, X, Z, NOIN, NOEXIT,
6ARRAY5, DEFR, SIGMA, SIGMCO, NINIT, N, SIGY, RSQ, R, VMAX, VMIN, VAR, NOMIN, NO
7MAX, K, NOENT, FLEVEL, CNST, FLEVL
  DIMENSION ERASE(25),
1GRADES(12), NORCD3(5), NORCD4(5), LRAM(3), INVAR(60), X(120), INDEX(60),
2DATA(60), Z(60)
6ARRAY5(61,61), AVE(60), SIGMA(60), COEN(60), SIGMCO(60), FLEVL(60)
EQUIVALENCE (DATA,Z)
EQUIVALENCE (STDEV,COEN,X), (X(61),AVE(1))
NVP1=NOZ+1
IFERR=1
DO 10 I=1,NOZ
  AVE(I)=ARRAY5(I,NVP1)/ARRAY5(NVP1,NVP1)
DO 9 J=1,NOZ
9  ARRAY5(I,J)=ARRAY5(I,J)-AVE(I)*ARRAY5(J,NVP1)
  IF (ARRAY5(I,I))1,2,10
1  IF (ARRAY5(I,I)+TOL)4,5,5
4  IFERR=2
5  WRITE OUTPUT TAPE 6,100,I,I,ARRAY5(I,I)
100 FORMAT(8H ARRAY5(I2,1H,I2,14H) IS NEGATIVE.E19.9)
  IFAVE=0
  IFRESO=0
  GO TO 10
2  WRITE OUTPUT TAPE 6,101,I
101 FORMAT (10H0      TERM(I2,14H) IS CONSTANT.)
10  CONTINUE
  IF(IFAVE)6,7,6
7  WRITE OUTPUT TAPE 6,102
102 FORMAT (1H0,46(1H),25HAVERAGE  VALUE  OF  TERMS//)
  CALL PRINT3(AVE,NOZ)
6  IF (IFRESO)11,12,11
12  WRITE OUTPUT TAPE 6,105
105 FORMAT (1H0,35(1H),49HRESIDUAL  SUMS  OF  SQUARES  AND  CROSS PR
-ODUCTS//)
  CALL PRINT 2(ARRAY5,NOZ)
11  GO TO (13,14),IFERR
13  RETURN
14  CALL CORE1
  END

```

*

```

$  COMPILER FORTRAN, PRINT OBJECT, PUNCH OBJECT                                RGRTRM00
  SUBROUTINE RGRTRM
C  ROUTINE TO TERMINATE REGRESSION PROGRAM
    COMMON NO,NVM1,NSW,ERASE,
1  NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
2  ,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRESO
3  ,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE
4  DIT,NORGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN
5  DEX,X,Z,NOIN,NOEXIT,
6  ARRAY5,DEFR,SIGMA,SIGMCO,NINIT,N,SIGY,RSQ,R,VMAX,VMIN,VAR,NOMIN,NO
7  MAX,K,NOENT,FLEVEL,CNST,FLEVL
    DIMENSION ERASE(28),
1  GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),
2  DATA(60),Z(60),
6  ARRAY5(61,61),AVE(60),SIGMA(60),COEN(60),SIGMCO(60),FLEVL(60)
    EQUIVALENCE (DATA,Z)
    EQUIVALENCE (STDEV,COEN,X),(X(61),AVE(1))
    NSW=1
    NVM1=NOTRMS
    IF(IFSTEP)1,2,1
1  CALL PRINT1(NSW)
2  WRITE OUTPUT TAPE 6,100,NOSTEP
100 FORMAT (29HOREGRESSION TERMINATED AFTER 12,7H STEPS.)
    WRITE OUTPUT TAPE 6,101,(I,ARRAY5(I,I),I=1,NVM1)
101 FORMAT (1H0,50(1H ),18HDIAGONAL  ELEMENTS//48(1H ),8HVAR. NO.,10(1
-H ),5HVALUE//151X      ,12,E24.9) )
    IF(IFPRED)3,4,3
4  IF(IFPRDI)5,3,5
5  CALL PREDCT(0)
3  RETURN
    END

```

*

\$COMPILE FORTRAN ,PRINT OBJECT,PUNCH OBJECT

NRAND000

C
C
C NRAND -- RANDOM TRIAL CONTROLLER (CALLED FROM WINDUP)
C (1) TESTS RESULTS OF EACH RANDOM TRIAL DURING THE RANDOM SELECTION
C PROCESS AND SAVES THE BEST TRIAL.
C (2) TESTS ICOUNT AGAINST NORAND TO DETERMINE WHETHER SPECIFIED
C NUMBER OF RANDOM PASSES HAVE BEEN USED.
C SAVER -- WRITES THE BEST RESULT OF THE RANDOM TRIALS ON TAPE 9 TO BE
C RESTORED WHEN SPECIFIED NUMBER OF RANDOM TRIALS HAVE BEEN USED.
C
C

SUBROUTINE NRAND(N)
COMMON NO,ERASE,ICOUNT,ERAS,ITAPE9,DMY,NORAND,FNAME,
1NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
2,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRES
3,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE
4DIT,NORGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN
5DEX,X,Z,NOIN,NOEXIT,
6ARRAY5,DEFR,SIGMA,SIGMCO,NINIT,N,SIGY,RSQ,R,VMAX,VMIN,VAR,NOMIN,NO
7 MAX,K,NOENT,FLEVE,CNST,FLEVL,TERMS,DDMY,SAVRSQ,SAVSIG,SVSGMO,
8SVCOEN,ISVTRM
DIMENSION DDMY(1061)
DIMENSION SVSGMO(60),SVCOEN(60),ISVTRM(60)
DIMENSION ERASE(8),ERAS(17),
1GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),
2DATA(60),Z(60),
6ARRAY5(61,61),AVE(60),SIGMA(60),COEN(60),SIGMCO(60),FLEVL(60)
7,TERMS(60,60)
EQUIVALENCE(DATA,Z)
EQUIVALENCE(STDEV,COEN,X),(X(61),AVE(1))

C
C
NINIT=0
N=1
IF(ICOUNT-1)3,67,7
67 WRITE OUTPUT TAPE 6,1001,NOPASS,NORAND
1001 FORMAT(1H0,H1PASS NO. 1,I5,H1 WILL BE EXECUTED 1,I5,H1 TIMES, USIN
2G RANDOM SELECTION1)
NOTRSV = 0
4 CALL SAVER(1)
6 REWIND ITAPE9
WRITE OUTPUT TAPE 6,103,NOPASS,ICOUNT
103 FORMAT(13H6 PASS NUMBER,I5,S5,19HRANDOM TRIAL NUMBER,I5)
READ TAPE ITAPE9,(DEFR,((ARRAY5(IT1,IT2),IT1=1,NOZ),IT2=1,NOZ))
12 RETURN
3 ICOUNT=0
N=6
GO TO 12
7 WRITE OUTPUT TAPE6,100,SAVSIG,SAVRSQ
100 FORMAT(38H0BEST PREVIOUS FITTED CURVE PROPERTIES, //23H STANDARD
-ERROR OF Y = ,E17.7/ 32H COEFFICIENT OF DETERMINATION = ,E15.7)
TEST = ABSF(RSQ-SAVRSQ)-1.E-04
IF(TEST) 226,227,227
226 IF(SAVSIG-SIGY) 2,2,1
227 IF(SAVRSQ-RSQ) 1,2,2
2 CONTINUE
IF(NOTRSV) 79,78,79
78 WRITE OUTPUT TAPE 6,122

```

122  FORMAT(55H6CURRENT TRIAL REJECTED IN FAVOR OF THE STANDARD TRIAL.)
      GO TO 80
79   WRITE OUTPUT TAPE 6,102,NOTRSV
142  FORMAT (53H6CURRENT TRIAL REJECTED IN FAVOR OF RANDOM TRIAL NO. I3
      - ,1H.)
80   IF (NORAND - ICOUNT) 9,6,6
9     ICOUNT =0
      N=6
      CALL SAVER(-1)
      GO TO 12
1     IF (NOTRSV) 90,91,90
91    WRITE OUTPUT TAPE 6, 2000
2000 FORMAT(55H6STANDARD TRIAL REJECTED IN FAVOR OF THE CURRENT TRIAL.)
      GO TO 105
90    WRITE OUTPUT TAPE 6,101,NOTRSV
101   FORMAT(18H6RANDOM TRIAL NO. I3,39H REJECTED IN FAVOR OF THE CURREN
      -T TRIAL.)
105   NOTRSV = ICOUNT-1
      IF (NCRAND - ICOUNT) 3,4,4
      END

```

*

```

$ ASSEMBLE, PUNCH OBJECT
  REM CONSTANT TERM GENERATOR
  ENTRY CTERM
CTERM  SXA      RTRN,4
      CLA*     1,4
      ARS 18
      STO I
      CLA 2,4
      STA      CNXT1
      STA      CNXT2
      CALL     .PRINT
      IOP      FRMT1
      IOP      I
CNXT1  IOP      **
      ENDIO
      CALL     .PUNCH
      IOP      FRMT1
      IOP      I
CNXT2  IOP      **
      ENDIO
RTRN   AXT      **,4
      TRA 3,4
FRMT1  BCD 4S11,2HT(I2,3H) =E17.8*
!      PZE
      END

```

CTERM000

*

```

$ ASSEMBLE, PUNCH OBJECT
REM MAD AND FORTRAN EQUATION SUMMATION AND RETURN GENERATOR
ENTRY SUMEQN
PRINT1 CALL .PRINT
PUNCH1 CALL .PUNCH
SUMEQN SXA RTRN,4
      SXA GO,4
      SXA XRB,2
      CLA 3,4
      STA CHECK
      CLA* 3,4
      TNZ FORT
      AXT FRMT1,4
      SXA FORM1,4
      TRA GO
FORT  AXT FRMT11,4
      SXA FORM1,4
GO    AXT **,4
      CLA 1,4
      STA FNAME1
      STA FNAME2
      STA FNAME3
      AXT 1,2
      CLA* 2,4
      ARS 18
      STO NIN
      CLA PRINT1
RPT   STA **1
      TSX **,4
FORM1 STR **
CHECK CLA **
      TZE **2
FNAME1 STR **
      STR NIN
      CLA* CHECK
      TZE **3
FNAME2 STR **
FNAME3 STR **
      STR
      TXL RTRN,2,0
      CLA PUNCH1
      TXI RPT,2,-1
RTRN  AXT **,4
XRB   AXT **,2
      TRA 4,4
FRMT1 BCD S11,9HT(0) = 0./S11,31HTHROUGH SUM, FOR I = 1,1,I .G. I2/S7,
      BCD 22HSUM T(0) = T(0) + T(I)/S11,20HFUNCTION RETURN T(0)/S11,15
      BCD 3HEND OF FUNCTION*
FRMT11 BCI *,S11,C6,5H = 0./S11,10HDO 1 I = 1,I2/S4,1H1,S6,C6,3H = C6,
      BCI *,7H + T(I)/S11,6HRETURN/S11,3HEND*
NIN   PZE
      END

```

*

S COMPILE FORTRAN, PRINT OBJECT, PUNCH OBJECT

NRML0000

C NRML -- TOTALS THE ARRAYS AND COMPUTES THE NEW WEIGHTS FOR THE
C VARIABLES, INTERACTION ORDERS, AND FUNCTIONS OF EACH VARIABLE.
C CALLED FROM CORE3.
C

SUBROUTINE NRML

COMMON NO,ERASE,

1NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
2,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRES
3,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE
4DIT,NCRGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN
5DEX,X,Z,NOIN,NOEXIT,
8ARRAY3,ARRAY2,ARRAY1

DIMENSION ERASE(30),

1GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),

2DATA(60),Z(60)

4ARRAY1(60),ARRAY2(60,2),ARRAY3(60,60)

EQUIVALENCE (DATA,Z)

EQUIVALENCE (STDFV,COEN,X),(X(61),AVE(1))

N=1

2 ARRAY1(60) = 0.

ARRAY2(60,1) = 0.

DO 1 I = 1,NOIND

6 ARRAY1(60) = ARRAY1(60)+ARRAY1(I)

ARRAY2(60,1) = ARRAY2(60,1)+ARRAY2(I,1)

ARRAY3(I,60)=0.

DO 1 J=1,NOFNCT

1 ARRAY3(I,60) = ARRAY3(I,60) + ARRAY3(I,J)

GO TO (3,4),N

4 RETURN

3 F1 = NOIND

F2 = NOFNCT

D1 = F1/ARRAY1(60)

D2 = F1/ARRAY2(60)

DO 5 I=1,NOIND

ARRAY1(I) = ARRAY1(I)*D1

ARRAY2(I,1) = ARRAY2(I,1)*D2

D3 = F2/ARRAY3(I,60)

DO 5 J = 1,NOFNCT

5 ARRAY3(I,J) = ARRAY3(I,J)*D3

N=2

GO TO 2

END

*

```
$ COMPILER FORTRAN, PRINT OBJECT, PUNCH OBJECT
C   TAPE HANDLING SUBROUTINE.
    SUBROUTINE TAPE1(I,J)
      REWIND I
      IF(J-1)1,1,2
2    JM1=J-1
      DO 3 K=1,JM1
3    READ TAPE I
1    RETURN
    END
```

TAPE1000

*

```

$ASSEMBLE,PLNCH OBJECT, PRINT OBJECT,EXECUTE,FULL DUMP
      REM      MAD AND FORTRAN GENERAL TERM GENERATOR
      REM      WITH SPECIAL FUNCTION PRINTER
      ENTRY GTERM
      ENTRY GENTRM
PRINT1 CALL      .PRINT
PUNCH1 CALL      .PUNCH
XRB      TXH 2,0,0
GTERM     STZ      F
          CLA      =1
          STO      M          M=1,MAD
          TRA      **4
GENTRM     CLA      =1
          STO      F          F=1,FORTRAN
          STZ      M
          CLA*     4,4
          ARS      18
          STO      U          U=1, SPECIAL FUNCTION
          CLA      =1
          SUB      U
          STO      X          X=1, EXPONENT
          SXA      RTRN,4
          SXA      IX2,2
          SXA      IX1,1
          LXA XRB,2
          SXD XRB,2
          CLA*     1,4
          ARS 18
          STO I
          CLA*     2,4
          ARS 18
          STO J
          CLA NRM
          STO WRD
          CLA NRM+1
          STO WRD+1
          CLA 3,4
          STA EXPNT1
          STA      EXPNT2
          STA EXPT1
          ZET      U
          TRA FLOAT1+2
          CLA*     3,4
          FSB CNE
          TNZ EXPT2
          CAL TXL
          TRA STO
EXPT2     FAD CNE
          UFA A
          LRS 27
          PXD 0,0
          LLS 27
FIX3      STO FIX
          ORA A
          FAD A
EXPT1     FSB **
          TNZ FLOAT1
          CLA FIX3
          STA EXPNT1

```

```

        CLA FIX2
        TRA END1
FLOAT1 NZT      U
        TRA      **+3
        CLA      ENDING+1
        TRA      END1
        ZET      F
        TRA      SKIPAB
        CLA      ABS
        STO WRD
        CLA ABS+1
        STO WRD+1
SKIPAB CLA      FLOAT2
END1   STO ENDING
        CAL XRB
STO    STP SW1
        CLA PRINT1
RPT    STA **+1
        TSX **,4
        STR SYMTAB,,FRMT1
        STR I
        STR I
        NZT      U
        TRA      **+3
EXPNT2 STR      **
        STR      PAREX
        STR J
        ZET      U
        TRA      FINI
SW1    TXL      FINI
        ZET      F
        STR WORD2
        NZT      F
        STR      WORD1
        NZT      U
EXPNT1 STR      **
FINI   STR
        LXD XRB,2
        TNX RTRN,2,1
        SXD XRB,2
        CLA PUNCH1
TXL    TXL RPT
RTRN   AXT      **,4
IX1    AXT      **,1
IX2    AXT      **,2
        TRA      5,4
ONE    DEC 1.
A      OCT +233000000000
FRMT1  BCD 5S11,2HT(I2,6H) = T(I2,4H) *
WRD    BCD 2
        BCI      *, 'X'(1HX,I2,C6), 'U'(C6, 'M'(1H.),C2,I2)
ENDING BCD 1
        BCD 1,1H)*
FIX2   BCD 1,I3
FLOAT2 BCD 1,F10.6
WORD1  BCI      *, .P. (
WORD2  BCI      *, ** (
PAREX  BCI      1,(X
ABS    BCD 2      6H.ABS.

```

```

NRM      BCD 2          1H
        ZERO      F,U,X,I,J,FIX,M
        BCI       1,F
        PZE       F,1
        BCI       1,U
        PZE       U,1
        BCI       1,X
        PZE       X,1
        BCI       1,M
        PZE       M,1
SYMTAB  PZE       8
        END

```

```

$ ASSEMBLE, PUNCH OBJECT
      REM REGRESSION  SUBROUTINE MAINBODY
      ENTRY RGRSSN

```

RGRSSN00

```

RGRSSN  SXD XRC,4
        CLA 1,4
        STA SW1
8A      CALL DGFRDM
SW1     LXD **,4
10A     TRA 10A+5,4
        TRA 17A
        TRA 14A
        TRA 11A
        TRA 20A
11A     CALL VARSRT
        TRA SW1
14A     CALL VARCHK
        TRA SW1
17A     CALL MATRAN
        TRA 8A
20A     LXD      XRC,4
        CLA*     2,4
        TNZ      1,4
        CALL     RGRTRM
        LXD XRC,4
LAST    TRA 1,4
XRC     PZE
        REM
MATRAN  REM
RGRTMN  REM
        REM
        REM
        END

```

```

      TRANSFORMS THE REGRESSION MATRIX.
      (1) PRINTS OUT THE LAST REGRESSION STEP.
      (2) CONTROLS PRINTING OF PREDICTIONS OF EQUATION.

```

\$ COMPILE FORTRAN, PRINT OBJECT, PUNCH OBJECT

PREDCT00

C PREDICTION ROUTINE.

SUBROUTINE PREDCT

COMMON NO,ERASE,

1NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
2,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRES
3,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE
4DIT,NORGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN
5DEX,X,Z,NOIN,NOEXIT,
6ARRAY5=DEFR,SIGMA,SIGMCO,NINIT,N,SIGY,RSQ,R,VMAX,VMIN,VAR,NOMIN,NO
7MAX,K,NOENT,FLEVEL,CNST,FLEVL

DIMENSION ERASE(30),

1GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),

2DATA(60),Z(60)

6ARRAY5(61,61),AVE(60),SIGMA(60),COEN(60),SIGMCO(60),FLEVL(60)

EQUIVALENCE (DATA,Z)

EQUIVALENCE (STDEV,COEN,X),(X(61),AVE(1))

IF(NOIN)20,20,1

1 CALL TAPE1(ITAPE3,NORCD3(2))

WRITE OUTPUT TAPE 6,100

100 FORMAT (1H0,39(1H),40HPREDICTED RESULTS VERSUS DATA POINTS//9

-H OBS. NO.,21(1H),11HPREDICTIONS,34(1H),4HDATA,19(1H),10HDEVIAT

-IONS/12(1H),9HY - SIGMA,14(1H),1HY,16(1H),9HY + SIGMA,13(1H),6

-HPOINTS,12(1H),10H(DATA - Y),5(1H),7HPERCENT)

BIGPCT=0.

NMXPCT=0

BIGDEV = 0.

NMAX = 0

DO 10 N1=1,NODATA

READ TAPE ITAPE3,RUN,(DATA(I),I=1,NOZ),WHT

Y=CNST

I1=0

DO 11 I2=1,NOTRMS

IF(INDEX(I2))25,11,25

25 I1=I1+1

Y=Y+COEN(I1)*DATA(I2)

11 CONTINUE

YPSIG=Y+SIGY

YMSIG=Y-SIGY

DEV=DATA(NOZ)-Y

PCT=(DEV/DATA(NOZ))*100.

ABDEV = ABSF(DEV)

ABPCT=ABSF(PCT)

IF(ABDEV-BIGDEV)4,4,2

2 BIGDEV = ABDEV

NMAX = N1

RUNMX=RUN

4 IF(ABPCT-BIGPCT)10,10,3

3 BIGPCT=ABPCT

NMXPCT=N1

RUNMXP=RUN

10 WRITE OUTPUT TAPE 6,101,RUN,YMSIG,Y,YPSIG,DATA(NOZ),DEV,PCT

101 FORMAT(F7.0,E17.8,E19.8,E21.8,E19.8,E21.8,F10.3)

102 FORMAT(29HOMAXIMUM ABSOLUTE DEVIATION =E15.7,14H,(SEE OBS. NO.F4.0

- ,10H, LINE NO.I3,1H))

103 FORMAT(37HOMAXIMUM ABSOLUTE PERCENT DEVIATION =F10.3,14H,(SEE OBS.

- NO.F4.0,10H, LINE NO.I3,1H))

REWIND ITAPE3

WRITE OUTPUT TAPE 6,102,BIGDEV,RUNMX,NMAX

WRITE OUTPUT TAPE 6,103,BIGPCT,RUNMXP,NMXPCT
20 RETURN
END

*

```

$ COMPILE FORTRAN, PRINT OBJECT, PUNCH OBJECT                                PRINT100
C   SPECIAL PRINTER ROUTINE NO.1
    SUBROUTINE PRINT1(NSW)
    COMMON NO,ERASE,
    1NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
    2,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRESO
    3,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE
    4DIT,NORGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN
    5DEX,X,Z,NOIN,NCEXIT,
    6ARRAY5,DEFR,SIGMA,SIGMCO,NINIT,N,SIGY,RSQ,R,VMAX,VMIN,VAR,NOMIN,NO
    7MAX,K,NOENT,FLEVEL,CNST,FLEVL
    DIMENSION ERASE(30),
    1GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),
    2DATA(60),Z(60)
    6ARRAY5(61,61),AVE(60),SIGMA(60),COEN(60),SIGMCO(60),FLEVL(60)
    EQUIVALENCE (DATA,Z)
    EQUIVALENCE (STDEV,COEN,X),(X(61),AVE(1))
    1 IF(NOENT)3,3,2
    3 WRITE OUTPUT TAPE 6,100,NOSTEP,K
100  FORMAT (9H0STEP NO.15/ 15H   TERM REMOVEDI4)
    GO TO 4
    2 WRITE OUTPUT TAPE 6,101,NOSTEP,K
101  FORMAT (9H0STEP NO.15/ 15H   TERM ENTEREDI4)
    4 WRITE OUTPUT TAPE 6,102,FLEVEL,SIGY,RSQ,R,CNST
102  FORMAT (12H   F LEVEL =E32.9/24H   STANDARD ERROR OF Y = E20.9/27H
    -   COEFF OF DETERMINATION =E17.9/26H   MULTIPLE CORLTN COEFF =E18.
    -9//18H   CONSTANT TERM = E26.9/57,8HTERM NO.8X11HCOEFFICIENT8X17HS
    -TD ERR OF COEFF 8X8HF LEVEL )
    J = 0
    DO 11 I = 1, NOTRMS
    IF (INDEX(I)) 22,11,22
22   J = J + 1
    WRITE OUTPUT TAPE 6,104,I,COEN(J),SIGMCO(J),FLEVL(J)
11   CONTINUE
104  FORMAT(4X5HTERM-I3,E24.9,E22.9,E10.9)
    7 RETURN
    END

```

\$COMPILE MAD,PRINT OBJECT,PUNCH OBJECT,EXECUTE I/ODUMP
 EXTERNAL FUNCTION (NP,VEC)
 E'0 DATARD.
 N'S INTEGER
 D'N V(71)
 NOSPFN = NP .RS. 18 -1

DATARD000

READ R'T \$72C1*\$,V...V(71)
 J = 0
 T'H LL, FOR I = 0, 1, I.G.71
 W'R V(I) .NE. \$\$
 V(J) = V(I)
 J = J + 1
 LL E'L

 (I=J,1, I .G. 71, V(I) = \$\$)

 J1 = 0
 T'H M, FOR K = 0,1, K .G. NOSPFN
 T'H N, FOR L = J1, 1, V(L) .E. \$.
 W'R V(L) .E. \$\$
 W'RL.NE. J1, T'0 ERR
 W'R K .E. NOSPFN, F'N 1.
 T'0 READ
 N E'L
 GATHER.(VEC(K),\$6C1*\$,V(J1)...V(L-1))
 W'R V(L+1) .E. \$,\$, L = L + 1
 M J1 = L + 1
 F'N 0
 ERR PRINT COMMENT \$6***ERROR ... ILLEGAL FORMATION IN SPECIAL FUN
 1CTION SPECIFICATION.\$
 F'N 0
 E'N

*

\$COMPILE FORTRAN,PRINT OBJECT,PUNCH OBJECT

CMTRWT 000

C TERM WEIGHT COMPUTER

SUBROUTINE CMTRWT

COMMON NO, ERASE,

1NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT

2,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRES

3,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE

4DIT,NCRGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN

5DEX,X,Z,NOIN,NOEXIT,

8ARRAY3,ARRAY2,ARRAY1

DIMENSION ERASE(30),

1GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),

2DATA(60),Z(60)

4ARRAY1(60),ARRAY2(60,2),ARRAY3(60,60)

EQUIVALENCE (DATA,Z)

EQUIVALENCE(STDEV,COEN,X),(X(61),AVE(1))

CALL TAPE1(ITAPE3,NORCD3(1))

DO 1 I=1,NOIND

ARRAY1(I)=1.

ARRAY2(I,1)=1.

DO 2 J=1,NOFNCT

2 ARRAY3(I,J)=1.

1 ARRAY3(I,60)=NOFNCT

ARRAY1(60)=NOIND

ARRAY2(60,1)=NOIND

WRITE TAPE ITAPE3,ARRAY1,ARRAY2,ARRAY3

RETURN

END

*

5 COMPILE FORTRAN, PRINT OBJECT, PUNCH OBJECT
C DEGREE OF FREEDOM SUBROUTINE.

DGFRDM00

```
SUBROUTINE DGFRDM
COMMON NO,ERASE,
1NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
2,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRESO
3,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE
4DIT,NORGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN
5DEX,X,Z,NOIN,NOEXIT,
6ARRAY5,DEFR,SIGMA,SIGMCO,NINIT,N,SIGY,RSQ,R,VMAX,VMIN,VAR,NOMIN,NO
7MAX,K,NOENT,FLEVEL,CNST,FLEVL
DIMENSION ERASE(30),
1GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),
2DATA(60),Z(60)
6ARRAY5(61,61),AVE(60),SIGMA(60),COEN(60),SIGMCO(60),FLEVL(60)
EQUIVALENCE (DATA,Z)
EQUIVALENCE (STDEV,COEN,X),(X(61),AVE(1))
IF (ARRAY5(NOZ,NOZ))10,1,2
1 WRITE OUTPUT TAPE 6,100,ARRAY5(NOZ,NOZ)
100 FORMAT (24H ERROR *** ARRAY5(Y,Y) =E15.7)
4 N=1
5 RETURN
10 IF (ABSF (ARRAY5 (NOZ,NOZ)) - TOL)12,12,1
12 WRITE OUTPUT TAPE 6,102,ARRAY5(NOZ,NOZ)
102 FORMAT (15H ARRAY5(Y,Y) = E15.7)
ARRAY5(NOZ,NOZ) = ABSF (ARRAY5 (NOZ,NOZ))
2 NOSTEP=NOSTEP+1
SIGY = SIGMA(NOZ)*SQRT (ARRAY5 (NOZ,NOZ)/DEFR)
RSQ=1.-ARRAY5(NOZ,NOZ)
RSQ = ABSF(RSQ)
R=SQRT(RSQ)
DEFR=DEFR-1.
IF (DEFR)3,3,6
6 N=2
GO TO 5
3 WRITE OUTPUT TAPE 6,101
101 FORMAT (28H NO MORE DEGREES OF FREEDOM.)
GO TO 4
END
```

*

```

$ COMPILE FORTRAN, PRINT OBJECT, PUNCH OBJECT                                WINDUP00
C WINDUP --
C (1) CALLS ON THE ROUTINES WHICH CONTROL THE RANDOM SELECTION
C MECHANISMS.
C (2) TESTS THE RESULTS OF EACH REGRESSION ANALYSIS TO DETERMINE
C WHETHER OR NOT THE GIVEN CRITERION HAVE BEEN MET.
C (3) TESTS THE RESULTS OF EACH PASS AND SAVES THE BEST PASS.
C (4) CALLED FROM CORE 6.
C NRAND -- RANDOM TRIALS CONTROLLER.
C (1) TESTS THE RESULTS OF EACH REGRESSION TRIAL DURING THE RANDOM
C SELECTION PROCESS AND SAVES THE BEST TRIAL.
C (2) TESTS TO DETERMINE WHEN THE SPECIFIED NUMBER OF RANDOM TRIALS
C HAVE BEEN USED.
C CORE3 -- THE EDITOR PROGRAM.
C (1) CONTROLS THE GRADING OF INTERACTION ORDERS, EXPONENTS, AND
C VARIABLES.
C (2) SELECTS NEEDED TERMS TO REFILL THE SPECIFIED NUMBER OF
C TERMS.
C
C
C SUBROUTINE WINDUP
C COMMON NO,ERASE,ICOUNT,ERAS,NOPASK,NOINKP,ER,NBEGIN,DMY,NORAND,
1 FNAME,
1NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
2,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRESO
3,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPREL,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE
4DIT,NORGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN
5DEX,X,Z,NOIN,NOEXIT,
6ARRAY5,DEFR,SIGMA,SIGMCO,NINIT,N,SIGY,RSQ,R,VMAX,VMIN,VAR,NOMIN,NO
7MAX,K,NOENT,FLEVEL,CNST,FLEVL
C DIMENSION SEPCN(60)
C DIMENSION ERASE(8),ERAS(5),ER(2),DMY(9),
1GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),
2DATA(60),Z(60)
6ARRAY5(61,61),AVE(60),SIGMA(60),COEN(60),SIGMCO(60),FLEVL(60)
C EQUIVALENCE (DATA,Z)
C EQUIVALENCE (STDEV,COEN,X),(X(61),AVE(1))
C
C
C IF(ICOUNT) 202,202,246
246 IF(NO) 247,202,247
247 WRITE OUTPUT TAPE 6, 10000
10000 FORMAT(59H0RANDOM TRIALS DELETED BECAUSE NO TERMS WERE SIGNIFICANT
- )
ICOUNT = NORAND +1
GO TO 206
202 WRITE OUTPUT TAPE 6,106
WRITE OUTPUT TAPE 6,107,SIGMAY,CODTRM
WRITE OUTPUT TAPE 6,108
WRITE OUTPUT TAPE 6,107,SIGY,RSQ
106 FORMAT(20H0POSTULATED CRITERIA)
107 FORMAT(22H0STANDARD ERROR OF Y =E17.7/25H COEFF OF DETERMINATION =
-E15.7)
108 FORMAT(24H0FITTED CURVE PROPERTIES)
IS=0
200 N=0
IF(SIGMAY - SIGY)1,2,2
2 N = N + 1
1 IF(CODTRM - RSQ)4,4,3

```

```

3  N = N + 2
   GO TO 55
4  N = N + 4
55  IF(IS)56,56,5
56  IF(ICOUNT)5,5,205
205  IF(N-5)206,9,206
206  IS=1
    CALL NRAND(N)
C    WHEN RETURNING FROM NRAND, N = 6 IF ALLOWED NUMBER OF RANDOM
C    TRIALS HAVE BEEN USED, N = 1 OTHERWISE.
5  GO TO (16,6,7,8,9,202),N
6  WRITE OUTPUT TAPE 6,100
100 FORMAT(37H0FITTED CURVE MEETS NEITHER CRITERIA.)
    GO TO 10
7  WRITE OUTPUT TAPE 6,101
101 FORMAT(46H0FITTED CURVE MEETS ONLY STD. ERROR CRITERION.)
    GO TO 10
8  WRITE OUTPUT TAPE 6,102
102 FORMAT(58H0FITTED CURVE MEETS ONLY COEF. OF DETERMINATION CRITERIO
-N.)
10  CONTINUE
    IF (NBEGIN) 45,444,45
444  NBEGIN = 1
    GO TO 46
45  TEST = ABSF(RSQ-SPRSQ)-1.E-04
    IF(TEST) 226,227,227
226  IF(SPSIGY-SIGY)47,227,44
227  IF(SPRSQ-RSQ) 44,47,47
44  WRITE OUTPUT TAPE 6,1004
1004 FORMAT(33H6CURRENT PASS IS THE BEST SO FAR.)
46  SPRSQ = RSQ
    SPSIGY=SIGY
    NOINKP = NOIN
    NOPASK = NOPASS
    DO 48 I = 1,NOIN
48  SEEPCN(I)=COEN(I)
    GO TO 11
47  CONTINUE
    NOIN = NOINKP
    DO 49 I = 1,NOIN
    INDEX(I) = I
49  COEN(I) = SEEPCN(I)
    I=NOIN+1
    DO 50 I=I,60
50  INDEX(I)=0
    WRITE OUTPUT TAPE 6,111,NOPASK
111 FORMAT(39H0CURRENT PASS REJECTED IN FAVOR OF PASS,15,1H.)
11  IF(NOPASS - NOTRYS)20,12,12
20  NOPASS = NOPASS+1
    WRITE OUTPUT TAPE 6,103,NOPASS,NOPROB,NOTRYS
103 FORMAT(13H0PASS NUMBER 14,22H BEGUN FOR PROBLEM NO.15/1H 14,22H TO
-TAL PASSES ALLOWED.)
13  NOEXIT = 2
14  CALL CORE3
12  WRITE OUTPUT TAPE 6,104,NOPROB,NOPASS
104 FORMAT(12H0PROBLEM NO.15,17H TERMINATED AFTER15, 8H PASSES./1H6)
15  NOEXIT = 1
    GO TO 14
16  RETURN

```

```

9 WRITE OUTPUT TAPE 6,105
  ICOUNT=0
105 FORMAT(34H0FITTED CURVE MEETS BOTH CRITERIA.)
  GO TO 12
  END
$ COMPILE FORTRAN, PRINT OBJECT, PUNCH OBJECT                                PARAM000
C
C
C  PARAM --
C  (1) READS IN THE LAST THREE CONTROL CARDS.
C  (2) ESTABLISHES VALUES FOR PARAMETERS IN COMMON.
C  (3) CALLED FROM CORE1.
C  STDRD -- ESTABLISHES STANDARD VALUES FOR THOSE PARAMETERS WHICH WERE
C          NOT SET BY THE USER.
C  ALTRMS -- COMPUTES THE TOTAL NUMBER OF POSSIBLE TERMS.
C
C
C  FUNCTION PARAM
C    COMMON NO,NDMY,I,DMY,NUMCDS,ERASE,IFLRN,IFOUT,NOINTR,
6  IFPCH,ERA,NBGIN,ERSA,ITAPE9,IFRWDA,NORAND,FNAME,
1  NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
2  ,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRES
3  ,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE
4  ,DIT,NORGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN
5  ,DEX,X,Z,NOIN,NOEXIT,ERAAS,NOSPFN
C    DIMENSION ERA(5),ERSA(7),ERAAS(8576),DMY(5),
1  GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),
2  DATA(60),Z(60)
C    EQUIVALENCE (DATA,Z)
C    EQUIVALENCE (STDEV,COEN,X),(X(61),AVE(1))
C
C
C    READ SECOND CONTROL CARD
C    READ INPUT TAPE 7,100,CODTRM,SIGMAY,PRBIN,PRBOUT,TOL,NOIND,NOTRYS,
1  NOFNCT,NOTRMS,IFMORE
100  FORMAT(5F10.5,4I5,I2)
C    READ THIRD CONTROL CARD
C    READ INPUT TAPE 7,101,IFRWDA,IFRAW,IFAVE,IFRES,IFSIG,
3  IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,IFOUT,NORAND,IFPCH,IFLRN,NOINTR
1  ,NOINIT,NOINT,IFCNST,IFWT,NUMCDS
2  ,NOPROB,FNAME,IFTRWT,NOSPFN
101  FORMAT(14I1,I4,3I5,I1,I4,I5,A6,2I2)
C    READ FOURTH CONTROL CARD
C    READ INPUT TAPE 7,103,NORGRS,(GRADES(I), I=1,NORGRS)
103  FORMAT(I5,4E16.8/(E21.8,3E16.8))
C    NBGIN=0
C    IF (NOIND) 988,988,987
988  WRITE OUTPUT TAPE 6,1000
1000  FORMAT (64H0***** NO INDEPENDENT VARIABLES SPECIFIED.  ERROR TERM
-INATION.  )
C    CALL ERROR
987  PARAM = 0.
C    IF (NORGRS) 321,321,322
321  NORGRS = 3
C    GRADES(1)=3.0
C    GRADES(2)=3.0
C    GRADES(3)=1.5
322  CALL STDRD(NOFNCT,NOTRYS,NOINTR,IFCNST,NUMCDS,NOTRMS,TOL,PRBIN,PRB
1  OUT,CODTRM,NOPROB,FNAME)

```

```

      NUMCDS=NUMCDS*12
      NORGRS = -NORGRS
      IF (IFMORE) 777,777,778
778   DO 779 III=1,IFMORE
      READ INPUT TAPE 7, 1111
1111  FORMAT(72H
      )
      WRITE OUTPUT TAPE6,1111
779   CONTINUE
777   CONTINUE
307   ALTG=NOTRMS
      ALTP=ALTRMS(NOINTR,NOIND,NOFNCT)
      IF(IFCNST)4260,4261,4261
4260  ALTP=ALTP+1.
4261  WRITE OUTPUT TAPE 6,107,ALTP
107   FORMAT (16H0POSSIBLE TERMS=,F10.0)
      IF (ALTP-ALTG) 4,5,5
4     WRITE OUTPUT TAPE 6,108
108   FORMAT (14H0TERMS REDUCED)
      NOTRMS=ALTP
5     CONTINUE
      ITAPE3=3
      NORCD3(1)=1
      NORCD3(2)=2
      NORCD3(3)=2
      ITAPE4=4
      NORCD4(1)=1
      NOVAR=NOIND+1
      NOZ=NOTRMS+1
      LRAM(1)=0.
      IF(NOINIT)1,1,2
1     NOINIT=0
      GO TO 3
C     IF ORDER OF TERMS INSERTED IS SPECIFIED
      2 READ INPUT TAPE 7,106,(INVAR(I),I=1,NOINIT)
106   FORMAT(14I5)
3     RETURN
      END

```

*

\$ COMPILE FCRTAN,PRINT OBJECT,PUNCH OBJECT

STUDN000 CORE3

```

C
C
C CORE3 -- THE STUDENT.
C   (1) CONTROLS THE LEARNING PROCEDURES.
C   (2) CALLED FROM CORE1, CORE4, AND FROM WINDUP.
C TAPE1 -- TAPE1(I,J) SPACES TAPE I BY J RECORDS.
C GRADER -- GRADES THE INTERACTION ORDERS, POWERS, AND VARIABLES ON
C           THEIR PREFORMANCE IN THE REGRESSION ANALYSIS.
C NRML -- SUMS THE VARIOUS SELECTOR ARRAYS AND REASSIGNS THE NEW
C         WEIGHTED VALUES TO EACH.
C PRINT4 -- PRINTS OUT THE SELECTION ARRAYS AFTER LEARNING.
C
C PKTRM -- PICKS NEW TERMS TO REFILL THE TERMS ARRAY.
C
C TRMCHK -- CHECKS FOR THE INDEPENDENCE OF THE TERMS.
C CORE4 -- THE EDITOR PROGRAM.
C   (1) CHECKS EACH TERM FOR COMPATIBILITY WITH THE MACHINE WORD SIZE.
C   (2) EVALUATES EACH TERM FOR ALL DATA SETS.
C
C

```

SUBROUTINE CORE3

```

COMMON NO,ERASE,IFLRN,ERASS,
1NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
2,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRESO
3,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE
4DIT,NORGRS,ITAFE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN
5DEX,X,Z,NOIN,NOEXIT,
8ARRAY3,ARRAY2,ARRAY1,DMYXXX,TERMS,DMY,NOSPFN
DIMENSION ERASE(9),ERASS(20),
1GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),
2DATA(60),Z(60),TERMS(60,60),DMYXXX(136),
4 ARRAY1(60),ARRAY2(60,2),ARRAY3(60,60),DMY(1060)
EQUIVALENCE (DATA,Z)
EQUIVALENCE (STDEV,COEN,X),(X(61),AVE(1))

```

```

C
C
IF (NOPASS) 76,40,40
40 CALL TAPE1(ITAPE3,NORCD3(1))
READ TAPE ITAPE3,ARRAY1,ARRAY2,ARRAY3
CALL TAPE1(ITAPE3,NORCD3(3))
IF(IFCNST)1,2,2
1 IF(INDEX(1)) 800,800,111
11 1 I1=2
NOCNT=1
GO TO 3
800 IFCNST = 1
I1=2
GO TO 801
2 I1=1
801 NOCNT=0
3 NIN = XABSF(NOIN)
FIN = NOFNCT - NOSPFN
DO 4 I=I1,NOTRMS
NO=I
N=2
IF(NOIN)30,5,6
C NOIN IS NEGATIVE IF RETURNING FROM CORE4,POSITIVE IF COMING
C FROM WINDUP, AND ZERO IF COMING FROM CORE1.

```

```

30 N=3
6 IF (INDEX(I)) 8,5,8
8 NOCNT=NOCNT+1
IF(NOCNT-1)17,9,9
17 DO 171 J=1,NOIND
171 TERMS(NOCNT,J)=TERMS(I,J)
TERMS(NOCNT,60)=TERMS(I,60)
GO TO 9
5 N=1
C AT THIS POINT, N=1 IF TERM UNSUCCESSFUL, N=2 IF TERM SUCCESSFUL
C N=3 IF TERM RETAINED WITHOUT GRADING
9 GO TO (31,31,4),N
31 CALL GRADER(ARRAY1,ARRAY2,ARRAY3,TERMS,NOTRMS,NOIND,FIN ,N,GRADE
-S,NOPROB,NORGRS)
4 CONTINUE
CALL NRML
IF(IFLRN) 32,32,11
32 CALL PRINT4
11 IF (NOCNT)13,13,14
14 IF (NOCNT+1-NOTRMS)13,13,16
76 NOCNT = 0
CALL CMTRWT
NOPASS = -NOPASS
13 J1=NOCNT+1
20 DO 18 I=J1,NOTRMS
NO = I
19 CALL PKTRM(TERMS,LRAM,ARRAY1,ARRAY2,ARRAY3,NOIND,NOFNCT)
CALL TRMCHK(N1,NOCNT,TERMS,NOIND,NOSTRT)
GO TO (19,18),N1
18 CONTINUE
NOEXIT=NOEXIT
GO TO (21,22),NOEXIT
22 CALL TAPE1(ITAPE3,NORCD3(1))
WRITE TAPE ITAPE3,ARRAY1,ARRAY2,ARRAY3
CALL CORE4
16 WRITE OUTPUT TAPE 6,100
100 FORMAT (58H0TERM MATRIX FILLED WITH OLD TERMS. FRESH SELECTION MA
-DE.)
NOCNT = I1-1
GO TO 13
21 IF (IFPCH) 12,12,10
10 IF(IFLRN)12,12,900
900 CALL PRINT4
12 CALL CORE7
RETURN
101 FORMAT(5E14.7)
102 FORMAT(14F5.0)
END
$ COMPILE FORTRAN, PRINT OBJECT, PUNCH OBJECT VARCHK00
C VARCHK COMBINED WITH VINCHK. ALLOWS S.O. OR
C TERM INSERTION,AS DESIRED.
SUBROUTINE VARCHK
COMMON NO,NSW,I,ERASE,ICOUNT,ERA,
1NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
2,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRES
3,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE
4DIT,NCRGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN
5DEX,X,Z,NOIN,NOEXIT,
6ARRAY5,DEFR,SIGMA,SIGMCO,NINIT,N,SIGY,RSQ,R,VMAX,VMIN,VAR,NOMIN,NO

```

*

```

7MAX,K,NOENT,FLEVEL,CNST,FLEVL
  DIMENSION ERASE(6),ERA(21),
1GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),
2DATA(60),Z(60)
6ARRAY5(61,61),AVE(60),SIGMA(60),COEN(60),SIGMCO(60),FLEVL(60)
  EQUIVALENCE (DATA,Z)
  EQUIVALENCE(STDEV,COEN,X),(X(61),AVE(1))
30 IF(NOSTEP-NOINIT-1)1,2,2
  1 NSW=2
  3 IF(NOSTEP-1)20, 4,4
  4 IF (IFCNST)6,7,6
  6 CNST=0.
  8 IF (IFSTEP)9,10,9
  9 GO TO (11,12),NSW
  2 NSW=1
  GO TO 3
20 WRITE OUTPUT TAPE6,100,SIGY
100 FORMAT (1H021HSTANDARD ERROR OF Y =E17.9)
  IF (IFCNST) 462,12,12
462  N=4
  INDEX(1)=1
  K=1
  NOENT=1
  GO TO 17
  7 CNST=AVE(NOZ)
  J = 0
  DO 41 I = 1, NOTRMS
  IF (INDEX(I)) 14,41,14
14  J = J + 1
  CNST = CNST - AVE(I) * COEN(J)
41  CONTINUE
  GO TO 8
10 CALL PRINT1(NSW)
  IF (IFPRDI)9,15,9
15 CALL PREDCT(0)
  GO TO 9
11 N=1
  IF(NOIN)26,26,16
26  N=2
  GO TO 16
163  K=NOMAX
  NOENT=K
  N=4
  INDEX(NOMAX)=1
  GO TO 188
16  IF(ICOUNT)1169,1169,350
350  IF (NO) 1168,163,1168
1168 IF (NOSTEP) 1169,1169,2268
2268 NO = 0
1169 CALL TSTLVL
188  GOTO (17,18),NSW
  18 NINIT = XMAXOF(0,NINIT-1)
  17 RETURN
  12 N=3
  GO TO 16
  END

```

*

```

$ COMPIL FORTRAN,PUNCH OBJECT,PRINT OBJECT
  FUNCTION FCT(NUM)
    FN=NUM
    IF (FN-30.) 6,6,7
7    FCT=1.E30
    RETURN
6    CONTINUE
    FCT=1.
1    IF (FN-1.) 3,3,2
2    FCT=FCT*FN
    FN=FN-1.
    GO TO 1
3    RETURN
    END

```

FCT00000

*

\$ COMPILE FORTRAN,PRINT OBJECT,PUNCH OBJECT

FLVL0000

```
FUNCTION FLVL(P,D)
  IF(P)2,2,1
1 IF(P-1.)8,9,9
8 IF(D)2,2,3
3 FLVL=-3.48265507+((-0.151687965E03*D**(-5))+(.680817276E01*D**(-1)
X))*P**(-.5)+(.518849328E01-.378849201E01*D**(-.33333333)-.27726566
XE01*D**(-1)+.100921422E03*D**(-5))*P**(-.2)+(.753659189E01*D**(-4)
X-.335350573E-02+.378218263E02*D**(-6))*P**(-1)-.1288222518E00*D**(-
X.33333333)-.605177797E-13)*P**(-5)*D**(-1)
  IF(FLVL)9,5,5
2 FLVL=1.E10
  GO TO 5
9 FLVL=0.
5 RETURN
END
```

*

SCOMPILE FORTRAN,PRINT OBJECT,PUNCH OBJECT

SAVER000

```

SUBROUTINE SAVER(ISWITCH)
C   SUBROUTINE TO SAVE REGRSSN MATRIX AND STATISTICS
COMMON NO,ERASE,ITAPE9,DMY,
1  NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
2  ,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRES
3  ,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE
4  ,DIT,NORGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN
5  ,DEX,X,Z,NOIN,NCEXIT,
6  ,ARRAY5,DEFR,SIGMA,SIGMCO,NINIT,N,SIGY,RSQ,R,VMAX,VMIN,VAR,NOMIN,NO
7  ,MAX,K,NOENT,FLEVE,CNST,FLEVL,TERMS,DDMY,SAVRSQ,SAVSIG,SVSGMO,
8  ,SVCOEN,ISVTRM
  DIMENSION DDMY(1061)
  DIMENSION FLEVL(60)
  DIMENSION SVSGMO(60),SVCOEN(60),ISVTRM(60)
  DIMENSION ERASE(26),COEN(60),DMY(3),
1  GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),
2  DATA(60),Z(60),TERMS(60,60),ARRAY5(61,61),SIGMA(60),SIGMCO(60)
  EQUIVALENCE (DATA,Z)
  EQUIVALENCE (STDEV,COEN,X),(X(61),AVE(1))
  IF (ISWITCH) 1,1,2
1  NOIN = NOINSV
  DO 3 ITER=1,NOIN
    SIGMCO(ITER)=SVSGMO(ITER)
    COEN(ITER)=SVCOEN(ITER)
3  CONTINUE
  DO 4 ITER=1,NOTRMS
4  INDEX(ITER)=ISVTRM(ITER)
    RSQ=SAVRSQ
    SIGY=SAVSIG
    CNST = SVCNST
    READ TAPE ITAPE9,(DEFR,((ARRAY5(IT1,IT2),IT1=1,NOZ),IT2=1,NOZ))
    REWIND ITAPE9
12  RETURN
2  WRITE TAPE ITAPE9,(DEFR,((ARRAY5(IT1,IT2),IT1=1,NOZ),IT2=1,NOZ))
    REWIND ITAPE9
    DO 5 ITER=1,NOIN
      SVSGMO(ITER)=SIGMCO(ITER)
      SVCOEN(ITER)=COEN(ITER)
5  CONTINUE
    DO 6 ITER=1,NOTRMS
6  ISVTRM(ITER)=INDEX(ITER)
      SAVRSQ=RSQ
      SAVSIG=SIGY
      NOINSV = NOIN
      SVCNST = CNST
      GO TO 12
    END
```

*

```

$COMPILE FORTRAN, PRINT OBJECT, PUNCH OBJECT, EXECUTE, DUMP      CORE6000
C
C
C CORE6 --
C   (1) SETS UP THE TAPES FOR RANDOM TRIALS.
C   (2) STARTS THE REGRESSION ANALYSIS.
C   (3) CALLED FROM CORE 4.
C SUMSQ -- COMPUTES THE RAW SUM OF SQUARES AND THE CROSS PRODUCTS.
C
C RSDSUM -- CALCULATES THE RESIDUAL SUMS OF SQUARES AND CROSS PRODUCTS.
C
C PRCRCN -- CALCULATES THE PARTIAL CORRELATION COEFFICIENTS OF THE DATA
C
C RGRSSN -- THE SUBROUTINE WHICH CONTROLS THE REGRESSION ANALYSIS.
C
C PRINT1 -- PRINTS OUT THE FINAL STEP OF THE REGRESSION FOR EACH
C           RANDOM TRIAL.
C WINDUP --
C   (1) CALLS ON THE ROUTINES WHICH CONTROL THE RANDOM SELECTION
C           MECHANISMS.
C   (2) TEST RESULTS OF EACH REGRESSION ANALYSIS TO DETERMINE WHETHER
C           OR NOT THE GIVEN CRITERION HAVE BEEN MET.
C   (3) TESTS THE RESULTS OF EACH PASS AND SAVES THE BEST PASS.
C
C
C   SUBROUTINE CORE6
C     COMMON NO,J1,ERASE,ICOUNT,
C     1DMMY,IFOUT,ERAS,NOPASK,NOINKP,EERAS,ITAPE9,DM,NORAND,FNAME,
C     1NOPROB,TOL,PRBIN,PRBOUT,NOIND,NOVAR,NOFNCT,NOTRMS,NOZ,NODATA,NOINT
C     2,NOINIT,IFCNST,IFWT,CODTRM,SIGMAY,NOTRYS,IFTRWT,IFRAW,IFAVE,IFRES
C     3,IFSIG,IFCOEN,IFSTEP,IFPRDI,IFPRED,IFPRNT,GRADES,NOSTRT,NSTDNT,NOE
C     4DIT,NCRGRS,ITAPE3,NORCD3,ITAPE4,NORCD4,LRAM,INVAR,NOSTEP,NOPASS,IN
C     5DEX,X,Z,NOIN,NOEXIT,
C     6ARRAY5,DEFR,SIGMA,SIGMCO,NINIT,N,SIGY,RSQ,R,VMAX,VMIN,VAR,NOMIN,NO
C     7 MAX,K,NOENT,FLEVEL,CNST,FLEVL,TERMS,DMY,FUNC,NOSPFN
C     DIMENSION ERASE(7),ERAS( 5),EERAS(8),TERMS(60,60),
C     1GRADES(12),NORCD3(5),NORCD4(5),LRAM(3),INVAR(60),X(120),INDEX(60),
C     2 DATA(60),Z(60),FUNC(60),DMY(1000),
C     6ARRAY5(61,61),AVE(60),SIGMA(60),COEN(60),SIGMCO(60),FLEVL(60)
C     EQUIVALENCE(DATA,Z)
C     EQUIVALENCE(STDEV,COEN,X),(X(61),AVE(1))
C
C
C 256 PRIN =PRBIN*100.
C     PROUT = PRBOUT*100.
C     NO = 0
C     WRITE OUTPUT TAPE 6,100,NOPROB,NODATA,NOTRMS,PRIN,PROUT
C 100 FORMAT(21H1STEPWISE REGRESSION//12H PROBLEM NO.15//19H NO. OF DAT
C -A SETS =15//22H NO. OF TERM CHOICES =15//15H PROBABILITY OF/32H
C - 1) ERROR IN ENTERING TERM =F8.4,4H 0/0/32H      2) ERROR IN DELET
C -ING TERM =F8.4,4H 0/0//)
C     CALL SUMSQ
C     IF (IFCNST)111,222,111
C 222 CALL RSDSUM
C 111 CALL PRCRCN
C CORE 6 ENABLES 'N' RANDOM TERM SELECTION PASSES,RETAINING BEST PASS
C     ICOUNT =0
C     M=1
C     NINIT=NOINIT

```

```

212 IF (NORAND)232,232,212
NINIT = 0
M=3
WRITE TAPE ITAPE9,(DEFR,((ARRAY5(IT1,IT2),IT1=1,NOZ),IT2=1,NOZ))
C SAVES ARRAY5 AND DEFR FOR USE IN NRAND.
WRITE OUTPUT TAPE 6,1001,NOPASS
1001 FORMAT(9HOPASS NO.,I5,20H STANDARD TRIAL. )
232 NOSTEP=-1
DO 542 I=1,NOTRMS
542 INDEX(I)=0
CALL RGRSSN(N,ICOUNT)
C M=1 IF THERE ARE NO RANDOM TRIALS OR IF EACHSTEP OF REGRESSION
C IS TO BE PRINTED. M=3 IF THE STANDARD TRIAL WAS JUST PREFORMED
C M=2 IF THE LAST STEP OF THE RANDOM TRIAL IS TO BE PRINTED.
GO TO (601,600,501),M
501 ICOUNT = 1
IF (IFSTEP) 700,700,701
700 M = 1
GO TO 601
701 M = 2
GO TO 601
600 CALL PRINT1(1)
601 CALL WINDUP
C WINDUP DOES NOT RETURN IF THE SPECIFIED NUMBER OF RANDOM TRIALS
C HAVE BEEN EXECUTED.
25 ICOUNT=ICOUNT+1
IF (ICOUNT) 232,1,232
1 RETURN
END

```

*

```

$COMPILE MAD,PRINT OBJECT,PUNCH OBJECT
                                DATARD00
                                EXTERNAL FUNCTION(NP,VEC)
                                ENTRY TO DATARD.
                                NORMAL MODE IS INTEGER
                                DIMENSION V(72)
                                K=0
                                NOSPFN = (NP .RS. 18)
                                I=1
READ      READ FORMAT $72C1*$,V...V(71)
                                J=0
                                THROUGH LOOP1,FOR I=0,1,I.G.71
                                WHENEVER V(I).NE.$$
                                V(J)=V(I)
                                J=J+1
LOOP1     END OF CONDITIONAL
                                THROUGH LOOP2,FOR I=J,1,I.G.71
LOOP2     V(I)=$$
                                J1=0
                                THROUGH L1, FOR K=K,1,K.G.NOSPFN
                                THROUGH L2, FOR L=J1,1,V(L).E.$.$
                                WHENEVER V(L).E.$$
                                WHENEVER L.NE.J1,TRANSFER TO ERR
                                W'R K .E. NOSPFN, F'N 0
                                TRANSFER TO READ
L2        END OF CONDITIONAL
                                GATHER.(VEC(K),$6C1*$,V(J1)...V(L-1) )
                                WHENEVER V(L+1).E.$,$,L=L+1
L1        J1=L+1
                                F'N 0
                                VECTOR VALUES F=$6C1*$
ERR       PRINT COMMENT$6***ERROR...ILLEGAL FORMATION IN SPECIAL FUNCTI
                                ION SPECIFICATIONS
                                F'N 1.
                                END OF FUNCTION

```

*

\$ASSEMBLE	PUNCH	OBJECT
	SST	PROOFF, PROTON
	ENTRY	SET8
	ENTRY	RSET8
SET8	CLA*	1,4
BACK	SXA	OUT,4
	LDQ	8
	STQ	SAVE8
	TSX	PROOFF,4
	STO	8
	TSX	PROTON,4
OUT	AXT	**,4
	TRA	2,4
RSET8	CLA	SAVE8
	TRA	BACK
SAVE8	PZE	
	END	

R+SET800

*