ESD-TR-74-314

MTR-2677, Vol. 10

REMOTE-TERMINAL EMULATOR (DESIGN VERIFICATION MODEL) — USER'S MANUAL

T. Suyemoto

FEBRUARY 1975

Prepared for

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
Hanscom Air Force Base, Bedford, Massachusetts

Project No. 572D

Approved for public release; distribution unlimited.

Prepared by

THE MITRE CORPORATION
Bedford, Massachusetts

Contract No. F19628-75-C-0001

REVIEW AND APPROVAL

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

JAMES S. CAMERON, Maj, USAF	MARVIN E. BROOKING
Project Engineer	Project Officer

FOR THE COMMANDER

ROBERT J. LATINA, Colonel, USAF
Director of ADPE Selection
Deputy for Command and Management Systems

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: ESD-TR-74-314	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): REMOTE-TERMINAL EMULATOR (DESIGN VERIFICATION MODEL) — USER'S MANUAL	5. TYPE OF REPORT & PERIOD COVERED
	6. PERFORMING ORG. REPORT NUMBER: MTR-2677, Vol. 10
7. AUTHOR(s): T. Suyemoto	8. CONTRACT OR GRANT NUMBER(s): F19628-75-C-0001
9. PERFORMING ORGANIZATION NAME AND ADDRESS: The MITRE Corporation, Box 208, Bedford, MA 01730	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Project No. 572D
11. CONTROLLING OFFICE NAME AND ADDRESS: Deputy for Command and Management Systems, Electronic Systems Division, AFSC, Hanscom Air Force Base, Bedford, MA 01731	12. REPORT DATE: February 1975
	13. NUMBER OF PAGES: 215
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

DESIGN VERIFICATION MODEL
REMOTE-TERMINAL EMULATOR

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Remote-Terminal Emulator is a minicomputer-based system which generates message traffic for use in testing and evaluating large-scale, multi-terminal computer systems. This series of reports will describe the two Design Verification Models that were developed on Data General NOVA 800 minicomputers. This volume is a user's manual which contains the information necessary to prepare and run the software portions of the Remote-Terminal Emulator.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

PREFACE

The Remote-Terminal Emulator is a minicomputer-based system which generates message traffic for use in testing and evaluating large-scale, on-line computer systems. In real-time testing, it emulates the actions of a collection of operators, terminals, and, depending upon configuration, modems. In 1972 and early 1973, two Design Verification Models (DVM) of the emulator were developed by The MITRE Corporation under the sponsorship of the Air Force Directorate of Automatic Data Processing Equipment Selection (MCS). The fixed-site system, which is used primarily for program and scenario development, is located at MITRE/Bedford and interfaces with the computer system under test (SUT) through the switched telephone network. The on-site system, which is used primarily for detailed emulator test and evaluation, is representative of the equipment planned for operational use in future computer procurements. This system, which is moved to each SUT site, interfaces through cables directly with the SUT's communication line adapters.

The primary hardware components of each of these systems are a Data General NOVA 800 minicomputer, a fixed-head disk, a magnetic tape unit, a control teletype, and an appropriate emulator/SUT interface unit. Both DVM's have sufficient hardware to emulate up to 16 low-speed interactive terminals. The on-site DVM also has hardware to emulate eight additional terminals or terminal networks by the use of high-speed synchronous line adapters and associated circuitry. The primary software components that have been developed for this project consist of the Macro Preprocessor, the Scenario Assembler, the Real-Time Executive, the Scenario Interpreter and the Data Reduction Program.

The common denominator of remote-terminal emulation is the scenario, which is a program that controls the actions to be taken by the emulator in emulating a given device and mix of devices. The scenario defines the queries (system commands, input data, and control characters) to be sent to the SUT, how SUT responses are to be processed, and other details of the test to be conducted. The Macro Preprocessor is a general purpose support program that provides a basic macro capability to aid in scenario writing and which was also used in emulator program development. In the scenario development process, the Scenario Assembler is used to convert external (symbolic) scenarios to internal (absolute) scenarios which are tailored to a specific terminal type and to specific data communications control procedures. Both the Macro Preprocessor and the Scenario Assembler run under the Data General Disk Operating System (DOS). In real-time testing, internal scenarios are brought into core from disk and are processed by the Scenario Interpreter which runs under the Real-Time Executive. All messages sent to and received from the SUT, as well as messages describing other actions of the emulator, can be time-tagged and logged on magnetic tape. Upon completion of the test, these data are processed in various fashions by the Data Reduction program (which also runs under DOS) to produce scenario trace data and various statistics on the performance and utilization of both the emulator and the SUT.

This document is part of a series of reports which will describe the design, implementation and use of the two Design Verification Models. The titles of the reports in the series are as follows:

Volume	Title
1	Introduction and Summary
2	Scenarios and Data Structures
3	Macro Preprocessor
4	Scenario Assembler

Volume	Title
5	Scenario Interpreter
6	Real-Time Executive
7	Data Reduction Program
8	Hardware
9	Support Software
10	User's Manual

It is suggested that the reader become familiar with the emulator concepts and terminology presented in Volume 1 preparatory to reading other volumes in the series.

TABLE OF CONTENTS

		Page
LIST OF ILLUSTRATIONS		9
LIST OF TABLES		11
SECTION I	INTRODUCTION	12
SECTION II	DOS AND SUPPORT SOFTWARE	14
	44 DOS	14
	Loading DOS	14
	Executing Under DOS	15
	SUPPORT SOFTWARE	16
	Utilities	16
	File Management	18
	Programming Aids	19
SECTION III	MACRO PROCESSOR	20
	INTRODUCTION	20
	PREPARATION AND USE OF MACROS	20
	Macro Names	20
	Macro Body	21
	Macro Definition	21
	Macro Call	21
	Parameter Substitution	22
	Label Generation	23
	Character Set	25
	Features	25
	Special Characters	25
	Quotes	25
	Master Macro Directory	26
	Notes and Restrictions	27
	SYSTEM FLOW	28

TABLE OF CONTENTS (Continued)

		Page
SECTION III (Cont.	OPERATING PROCEDURES	28
	SSUB	28
	Input File	31
	Output File	31
	MACDEF	31
	Input File	32
	Output File	32
	Output Messages	32
SECTION IV	SCENARIO ASSEMBLER	36
	INTRODUCTION	36
	SYSTEM FLOW	36
	OPERATING PROCEDURES	38
	Preparing Files	38
	External Scenario	38
	Program Files	40
	Executing Assembler	40
	OUTPUT	45
	Internal Scenario	45
	Optional Listings	46
	Output Messages	48
SECTION V	EQUIPMENT TABLE	50
	INTRODUCTION	50
	GENERATION	50
	REQUIREMENTS AND CONVENTIONS	51
	FUNCTION	74
SECTION VI	REAL-TIME EMULATOR SYSTEM	
	GENERATION	80
	INTRODUCTION	80
	SSUB	80

TABLE OF CONTENTS (Continued)

		Page
SECTION VI (Cont.)	ASM	81
	RLDR	83
	MKABS	83
	Disk Requirements	85
SECTION VII	REAL-TIME EMULATOR	87
	INTRODUCTION	87
	SYSTEM FLOW	87
	OPERATING INSTRUCTIONS	87
	Startup	89
	Control TTY Inputs	90
	Run ID	90
	Commands	90
	CANCEL Input	90
	BREAK Output	91
	Responses	91
	Shutdown	91
	ERROR MESSAGES	92
	DEVICE STATUS	99
	RING COUNTERS	101
	RESPONSE HANDLING AND LOGGING	104
	DIGITAL I/O	106
	STORAGE REQUIREMENTS	113
	MISCELLANEOUS NOTES	116
	PANIC CODES AND ACTIONS	117
SECTION VIII	DATA REDUCTION PROGRAM	123
	INTRODUCTION	123
	SYSTEM FLOW	123
	OPERATING PROCEDURES	125
	Input Message	125

TABLE OF CONTENTS (Continued)

		Page
SECTION VIII (Cont.)	Command Interpreter	126
	Interactive Mode	126
	Switch Mode	128
	Summaries	131
	Brief Summary	131
	Detailed Summary	135
	Listings	136
	Octal Tape	136
	Actual Times	137
	Time Intervals	137
	Relative Times	138
	ERRORS	138
	SAVING TEST DATA	140
	Program Description	140
	Input Message	141
	Operation	141
	Errors	143
SECTION IX	EXECUTION TIMES	145
	REAL-TIME INSTRUCTIONS	145
	NON-REAL TIME PROGRAMS	159
	SSUB	159
	MACDEF	160
	CVT	160
	DATAR	161
	MASTR	161
REFERENCES		163
APPENDIX I	CONVERSION CODES FOR IBM 2741	164
APPENDIX II	SAMPLE LISTINGS FROM SCENARIO ASSEMBLER	168

TABLE OF CONTENTS (Concluded)

		Page
APPENDIX III	LISTING OF EQUIP. RB	174
APPENDIX IV	DATAR LISTINGS	188
APPENDIX V	EXAMPLE OF TELETYPE LISTING FOR AN EMULATION RUN	196
APPENDIX VI	TIMING SAMPLES FOR NON-REAL TIME PROGRAMS	198

LIST OF ILLUSTRATIONS

Figure Number		Page
1	SSUB System Flow	29
2	MACDEF System Flow	30
3	System Flow of the Scenario Assembler	37
4	External Scenario Format	39
5	Equipment Table Macros	52
6	File EQ of Equipment Table (Macros not Expanded)	56
7	Potion of File EQUIP of Equipment Table (Macros Expanded)	60
8	Portion of File EQUIP.RB, Assembled Equipment Table	65
9	ET Entries for DCM Devices for Lab System	69
10	ET Entries for Asynchronous Devices for 64-line Field Test System	70
11	ET Entries for Asynchronous Devices for 16-Line Field Test System	71
12	Equipment Table Hierarchy	75
13	Example of Device Communication Through Scenarios	78
14	System Flow for Real-Time Emulator	88
15	State Transition Diagram	100
16	Ring Counter Changes	102
17	Digital I/O Connections	109
18	Normal Interface Rack Wiring for Asynchronous Devices	110
19	Normal Asynchronous Correspondence	112
20	Macro Definitions for Digital I/O	112
21	HANDSHAKE Scenario	112

LIST OF ILLUSTRATIONS (Concluded)

Figure Number		Page
22	Example of Panic Message	122
23	General System Flow of Data Reduction Program	124
24	Interactive Tree Diagram for DATAR	129
25	Switch Tree Diagram for DATAR	132
26	Brief Summary Output Format	189
27	Detailed Summary Output Format	190
28	Histogram Output Format	191
29	Octal Tape Output Format	192
30	Actual Time Output Format	193
31	Time Interval Output Format	194
32	Relative Time Output Format	195
33	Fortran Cost Scenario with Macros not Expanded	199
34	Scenarios for Fortran Cost Problem with Macros Expanded	201
35	Macro Libraries for Fortran Cost Problem	212

LIST OF TABLES

Table Number		Page
I	Common Utility Programs	17
II	Common File Management Commands	18
III	Output Messages for Macro Processor	33
IV	Available Codes for Conversion	42
V	Available Codes for SOM/EOM	44
VI	Output Messages for Scenario Assembler	49
VII	Input File Names for Emulator System	82
VIII	Inputs to Relocatable Loader	84
IX	Disk Requirements for Emulator System	86
X	Error Message Classes for Scenario Interpreter	93
XI	Error Messages for Scenario Interpreter	94
XII	Core Storage Requirements for Scenario Interpreter	114
XIII	Core Storage Requirements for Real-Time Exec	115
XIV	RTOS Panic Codes	119
XV	Interactive Requests and Responses for DATAR	127
XVI	Option and Suboption Switches for DATAR	130
XVII	Record Type Switches	133
XVIII	Switch Combinations and Valid Inputs	133
XIX	DATAR Error Message File (ERFILE)	139
XX	MASTR Error Message File	144
XXI	Real-Time Scenario Instruction Execution Times	146
XXII	Control Characters for IBM 2741 Terminal	165
XXIII	Conversion Code Table used for IBM 2741 Terminal	166

SECTION I

INTRODUCTION

The Remote-Terminal Emulator consists of a combination of hardware components and software packages designed to generate message traffic for use in testing and evaluating on-line computer systems. The hardware configurations for both the fixed-site and on-site systems are discussed in Volume 8 of this series. This user's manual presents the user information necessary to prepare and run the software portions of the system. Included here are excerpts from previous volumes as well as additional material required for running the Remote-Terminal Emulator.

The common denominator of remote-terminal emulation is the scenario, which is a program that controls the actions to be taken by the emulator in emulating a given device or mix of devices. A scenario is formed by a series of scenario instructions which determine the queries to be sent to a SUT, how responses are to be handled, and the various control functions of a test. The command is a special instruction which exerts gross control over emulator actions, and is the only means by which the user can exert external control during an emulation run. Both instructions and commands are described in detail in Sections IV and V of Volume 2 of this series.

This paper is organized as a logical presentation of steps needed for preparation, execution, and data reduction of an emulator run. Section II describes both the NOVA Disk Operating System (DOS) as it applies to the Emulator, as well as the system support software which may be applicable in most phases of emulation. The macro processing function is described in Section III and the assembly function is presented in Section IV. These two functions prepare the scenario for the real-time run. Sections V and VI respectively deal with preparing the Equipment Table, and following this, building an emulator system. The operating instructions and other information necessary for execution of a real-time emulation run are presented in Section VII. The final phase of an emulator run, data reduction, is discussed in Section VIII. An example of the on-line teletype output for all processing steps for a single emulation run is given in Appendix V. Section IX contains timing information for both the real-time and non-real time functions of the emulator.

SECTION II

DOS AND SUPPORT SOFTWARE

DOS

All of the non-real time programs included in the Emulator system run under Revision 5 of Data General's Disk Operating System (DOS). The support software described in Volume 9 of this series also operates under control of DOS. A complete description of DOS can be found in Reference 1. Under DOS a carriage return and a line feed are echoed back when the RETURN key is depressed. In this document the symbol ↲ is used to denote the depression of the RETURN key and the echo back of both the carriage return and line feed.

Loading DOS

The DOS system can be loaded into core from tape, or, if it already exists on disk, it can be loaded from there. To load from tape, the following sequence should be performed:

(1) Turn on CPU, disk, tape drive, and system teletype;

(2) Mount the system tape; press LOAD to advance tape to ready position;

(3) Set panel data switches to 100022;

(4) Raise the RESET panel switch and then raise the PROGRAM LOAD panel switch;

(5) The remainder of the process involves the following activity on the system teletype. The underlined portion is what is to be entered by the user. The non-underlined portion is the response of the system.

FULL(Ø) OR PARTIAL(1)? Ø

R

XFER MTØ:1 SYS.SV

R

CHATR SYS.SV SP

R

INSTALL SYS.SV

R

LOAD/A MTØ:2

FILE ALREADY EXISTS, FILE: SYS.DR

FILE ALREADY EXISTS, FILE: MAP.DR

R

To load DOS from disk the following sequence should be performed:

(1) Turn on CPU, disk, and TTY;

(2) Set panel data switches to 100020;

(3) Raise the RESET panel switch and then raise the PROGRAM LOAD panel switch;

(4) The system will respond as follows:

DOS REV Ø5

Press the continue panel switch and DOS responds:

R

There is not enough disk space on the present NOVA to accommodate the complete Disk Operating System plus the emulator system. Therefore, to delete from disk all DOS files which are not essential to preparing or executing an emulator run, the following command line should be typed directly after loading DOS.

@REMAL@

This frees space on the disk to allow for the emulator system and scenarios, which can then be loaded.

Executing Under DOS

Programs which operate under control of DOS are executed in response to a user input request entered at the system teletype. The input message is called a command line and is processed by an executable program called the Command Line Interpreter (CLI). The CLI indicates to the user that it is ready to accept commands by typing the ready message, R↲. The user enters a command by typing a line and depressing the RETURN key. When execution of a program running under DOS is completed, control is returned to the CLI.

When operating under DOS, depressing CTRL and A simultaneously on the system teletype causes an immediate interrupt to the executing program, regardless of the program status. This can be useful, for instance, to discontinue a run when errors have been detected. The word INT is typed by the CLI upon recognition of the CTRL-A break, and control is returned to the CLI which then types R.

SUPPORT SOFTWARE

All support software programs operate under control of DOS. They are described in detail in Volume 9 of this series. A brief presentation of operating instructions for the most commonly needed functions is given here. This section does not include all available programs.

Utilities

The utilities transfer data from one DOS file to another. Note that all peripheral devices are treated as files. Table I below shows some methods for moving data. Where appropriate, filenames for peripherals may be used for input or output files to the utility programs. These names include:

$CDR	card reader input
$TTI	teletype keyboard input
$TTO	teletype printer output
$LPT	line printer output

Table I

Common Utility Programs

Operation	CLI Input Message
Card to disk	XFER/A $CDR filename LXFER $CDR filename
Tape to disk	LOAD MTØ:x [filename1 filename2...]
Disk to line printer	PRINT filename1 ... PRINTL filename1 ...
Disk to tape	DUMP MTØ:x filename1 ...

The switch /A on the XFER command causes the data to be input from the card reader ($CDR) as ASCII data with a carriage return inserted at the end of the text on a card to denote an end of line. Without the switch the input is transferred sequentially without alteration. The LXFER program is MITRE generated and provides the capability to convert Hollerith data to ASCII (the code of the NOVA), including control characters and lower case. It also permits entry of any 8-bit value via card input. A description of the program is given in Volume 9.

Both the LOAD and DUMP commands have an additional option, /V, which causes the names of the files to be verified on the teletype. Also in these commands MTØ signifies transport Ø of the tape drive, and x designates which file on the tape is selected. The brackets indicate optional information; if no filename is specified, all non-permanent files are moved. The PRINT program lists the designated file(s) on the line printer without either a title or line numbers, and truncates a line after 80 characters. The PRINTL program, however, lists the file(s) with both a title and line numbers, and prints lines longer than 76 characters on successive lines without associating a new line number.

File Management

Several DOS programs may be useful in handling files containing scenarios or libraries. Table II shows some of the more common commands.

Table II

Common File Management Commands

Operation	CLI Input Message
Delete file(s) from directory and free space	DELETE filenamel ...↵
Change filename	RENAME oldfilename newfilename↵
Concatenate copies of files to produce a new file	APPEND newfilename filenamel ...↵
List number of disk blocks in use and number available	DISK↵
List names, byte count, and attributes of files in directory	LIST [filenamel ...]↵

The specific command DELETE*.* deletes from disk all files which are not permanent. The LIST command with no parameters causes a listing of the byte count for each file on the teletype. In the option /L is used, the listing is printed on the line printer. If the option /A is used, all permanent files are also listed. If specific files are designated, only those specified are listed.

Programming Aids

The two programs most often employed by an Emulator user are the EDIT and OEDIT (octal edit) programs. The EDIT program is used to build a new source file or edit an existing one. This program is described in full in Reference 2. The octal editor is used to examine and/or modify, in octal, any location in any disk file. A complete description of this program can be found in Reference 1.

SECTION III

MACRO PROCESSOR

INTRODUCTION

The basic function of a macro processor is text substitution, where a name appearing in the source code is replaced by an associated string of characters. A general purpose macro capability, including a macro library generator (MACDEF) and a macro processor (SSUB), was developed on the NOVA 800. One of the main purposes of this software is to facilitate scenario writing by (1) providing a one-to-many statement capability and (2) allowing for substitution of parameter values at the external scenario level. This permits the scenario writer to include common pieces of code in different scenarios and to change subscenario calls to in-line code, or vice-versa. Another use for the macro capability is in writing code in NOVA Assembly language, which is the means used for generating an emulator Equipment Table.

Macros may be created and saved separately in a macro library by using the MACDEF program; or they may be defined in the source file itself during execution of SSUB. Both MACDEF and SSUB are written in Extended ALGOL and operate in 24K core under control of DOS. A description of the design and implementation of the Macro Processor can be found in Volume 3 of this series of reports.

PREPARATION AND USE OF MACROS

The discussion of macros presented here applies to all macros whether they are defined in a library, or directly in the source code.

Macro Names

Macro names are identifiers consisting of ten or less alpha-numeric characters.

Macro Body

In its simplest form a macro body consists of a string of ASCII characters to replace every occurrence of the macro name in the source data. No extra spaces are inserted.

Macro Definition

A macro definition associates an identifier (the macro name) with a string of text (the macro body). Format for a macro definition is as follows:

```
MDEF macroname (number of arguments)
macro body
.
.
.
MEND
```

The literals MDEF and MEND are left-adjusted on separate lines (or cards). The macro body consists of all characters beginning with the next line after MDEF up to, but not including, the carriage return before the MEND. If the macro has no arguments, the initial line may be terminated after the macro name.

Macro Call

A macro call is any reference to a macro name in the source file. Formats for a call are:

macroname (arg 1,arg 2...)	if the macro has arguments.
macroname	if there are no arguments.

Arguments are separated by commas and enclosed in parentheses.

Example 1: Simple Substitution		
Source Data: ALGOL Program		
Macro Definition	Source Code	Output Code
MDEF DIGIT ((CHAR>=60R8) AND (CHAR<=71R8)) MEND	IF DIGIT THEN GO TO EXIT;	IF ((CHAR>=60R8) AND (CHAR<=71R8)) THEN GO TO EXIT;

Parameter Substitution

Macro bodies may contain formal parameters which will be replaced by actual parameters (arguments) in a macro call. Up to 9 formal parameters can be used in a macro definition. Each formal parameter is specified by a $ (dollar sign) followed by a digit n where 0<n<10. When the macro name and its arguments are encountered by SSUB in the source code, the first positional argument will be substituted for the formal parameter $1; the second, for $2, etc. Formal parameters may be passed as macro arguments.

Example 2: Use of Parameters		
Source Data: NOVA Assembly		
Macro Definition	Source Code	Output Code
MDEF LDI (2) JMP .+2;MLDI (R$1,$2) $2 LDA $1,.-1 MEND	LDI (3,50)	JMP .+2;MLDI (R3,50) 50 LDA 3,.-1

Macro calls may be nested within arguments and within macro bodies.

Example 3: Nested Macro Call in Macro Argument		
Source Language:	NOVA Assembly Language	
Macro Definitions	Source Code	Output Code
MDEF LDI (2)	LDI (3, DEC (50))	JMP .+2
JMP .+2		.RDX 10
$2		50
LDA $1,.-1		.RDX 8
MEND		LDA 3,.-1
MDEF DEC (1)		
.RDX 10		
$1		
.RDX 8		
MEND		

<u>Label Generation (The TAIL Function)</u>

To insure that labels appearing within macro bodies will not be multiply defined, a special function $T is provided. Each reference to $T is replaced by a numeric value. This value is unique for each macro call, but remains constant for all $T references within a macro body. $T may be passed one level as a macro argument.

Example 4: Use of $T Function		
Source Data:	Scenario Assembly Code for Login Sequence	
Macro Definitions	Source Code	Output Code
MDEF FINDLIT (1)	ALLOCREGS 10	ALLOGREGS 10
L FL$T	FINDLIT (6000)	L FL3
R ''	QCESDM002	R ''
S FL$T $1	FINDLIT (PASSWORD)	S FL3 6000
MEND	QXXXX	QCESDM002
	FINDLIT (SYSTEM?)	L FL4
		R ''

Example 4: Use of $T Function (Concluded)		
Source Data: Scenario Assembly Code for Login Sequence		
Macro Definitions	Source Code	Output Code
		S FL4 PASSWORD
		QXXXX
		L FL5
		R ''
		S FL5 SYSTEM?

Example 5: Nested Macro Calls in Macro Body		
Source Data: Scenario Assembly Code		
Macro Definitions	Source Code	Output Code
MDEF FINDLIT (1)	LIST	**PRINT FILE**
L FL$T		QB
R ''		L FL40
S FL$T $1		R ''
MEND		S FL40 READY
MDEF BACKUP		QPRINT;*
QB		L FL42
REDY		R ''
MEND		S FL42 FILE
MDEF REDY		
FINDLIT (READY)		
MEND		
MDEF LIST		
PRINT FILE		
BACKUP		
REDY		
QPRINT;*		
EOF		
MEND		
MDEF EOF		
FINDLIT (FILE)		
MEND		

Character Set

Source input to both SSUB and MACDEF normally consists of ASCII characters. The results of using non-ASCII characters are not defined, although in the current version most values are processed correctly. Two known exceptions are the eight-bit values 0 and 1, which are used internally by SSUB and MACDEF and should never be included in source code for either program.

Features

Special Characters

$	The dollar sign is used for three special functions performed by SSUB. It is illegal to use it otherwise in normal source data, other than in a quote string. $T specifies the TAIL function. $Q specifies the quote function. $digit is used to specify formal parameters.
'	A single quote delimits a string not to be scanned by SSUB. The string is passed with quotes.
()	Parentheses are used to enclose arguments in a macro call. Parentheses may appear elsewhere in source data.
,	Commas are used to separate macro arguments. They may also appear elsewhere in source data.

Quotes

When a string of characters is enclosed in single quotes, it is passed on (including quotes) without being scanned.

$Q is a special macro function which can be used to pass a string of characters including commas, leading blanks, etc., in macro arguments. $Q is followed by a string delimited at the beginning and end by a character selected by the user. Delimiter characters may be any ASCII characters except those listed above in the special group and the space character. The expansion of $Q is the string without delimiters. The string itself will be scanned when it is substituted for its corresponding formal parameters.

Example 6: $Q Function		
Source Data: Scenario Assembly Code		
Macro Definition	Source Code	Output Code
MDEF INSTR (1) $1 MEND	INSTR ($Q*LDA 3,A*)	LDA 3,A

Master Macro Directory

As part of its initialization, SSUB creates a master directory which is effectively the sequential concatenation of all library directories in left-to-right order as they appear in the DOS command line. Later, if more definitions are encountered in the source file, they are added to the master directory. During an SSUB run names are never deleted, and no name duplication check is made. The directory is ordered so that if duplicate macro names occur, the text of the macro most recently added to the directory will be used.

Notes and Restrictions

1. Single quote strings are limited to 1000 characters.

2. In an SSUB run the total of all macros in the libraries and all macros defined during the run itself cannot exceed 160.

3. Each macro library is limited to 100 macros.

4. $Q is legal only in macro arguments.

5. The identifiers MDEF and MEND are reserved and cannot be used as macro names, or appear in any source data except in their normal use in macro definitions.

6. The file name TSUB.MB is reserved.

7. The system error message "stack overflow" usually indicates a recursion loop in macro substitution. Example:

```
MDEF OR
COM 1, 1
     AND 1, 2    ;  PERFORMS LOGICAL OR
MEND
```

   When the macro OR is called, infinite recursion will occur because of the "OR" in the comment within the macro body.

9. If an unsuccessful MACDEF run has been made, the .ML file should be deleted before MACDEF is rerun with the same name. Otherwise a new file is not created and the new information is written over the old information. If

this occurs, and if the new file is to be smaller than the old file, whatever has not been overwritten will remain at the end of the file.

SYSTEM FLOW

Overall system flows for SSUB and MACDEF are shown in Figure 1 and Figure 2, respectively. Operations taking place on the NOVA are listed at the bottom of the figures with the required DOS commands.

OPERATING PROCEDURES

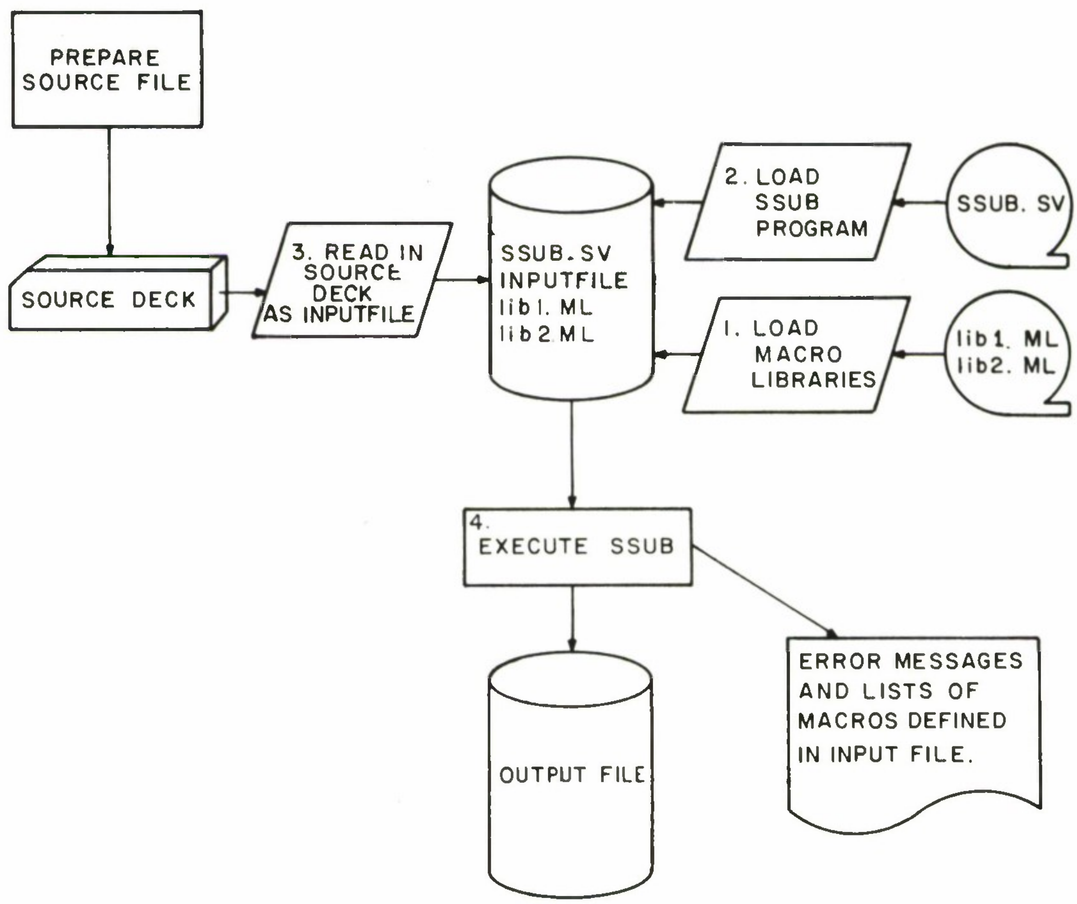
SSUB

SSUB is the actual macro processing program; it performs the macro substitutions. Input to SSUB consists of a source file and up to four macro libraries. SSUB copies the source file into an output file. While copying, it scans the source data for macro definitions and references to macro names (macro calls). When a macro name is detected, the text of the specified macro is copied into the output file replacing the macro name. Macros may have arguments which modify the text of the macro as it is copied. For SSUB, modification consists simply of replacing formal parameter references contained in the macro body by actual parameters supplied as arguments.

To use the SSUB program the following steps should be performed:

1. Load the SSUB save file.
2. Create or load the source file.
3. Load any macro library files to be used.
4. Ready the line printer.
5. Enter the following command at the teletype:

   SSUB input-file output-file library-names↵

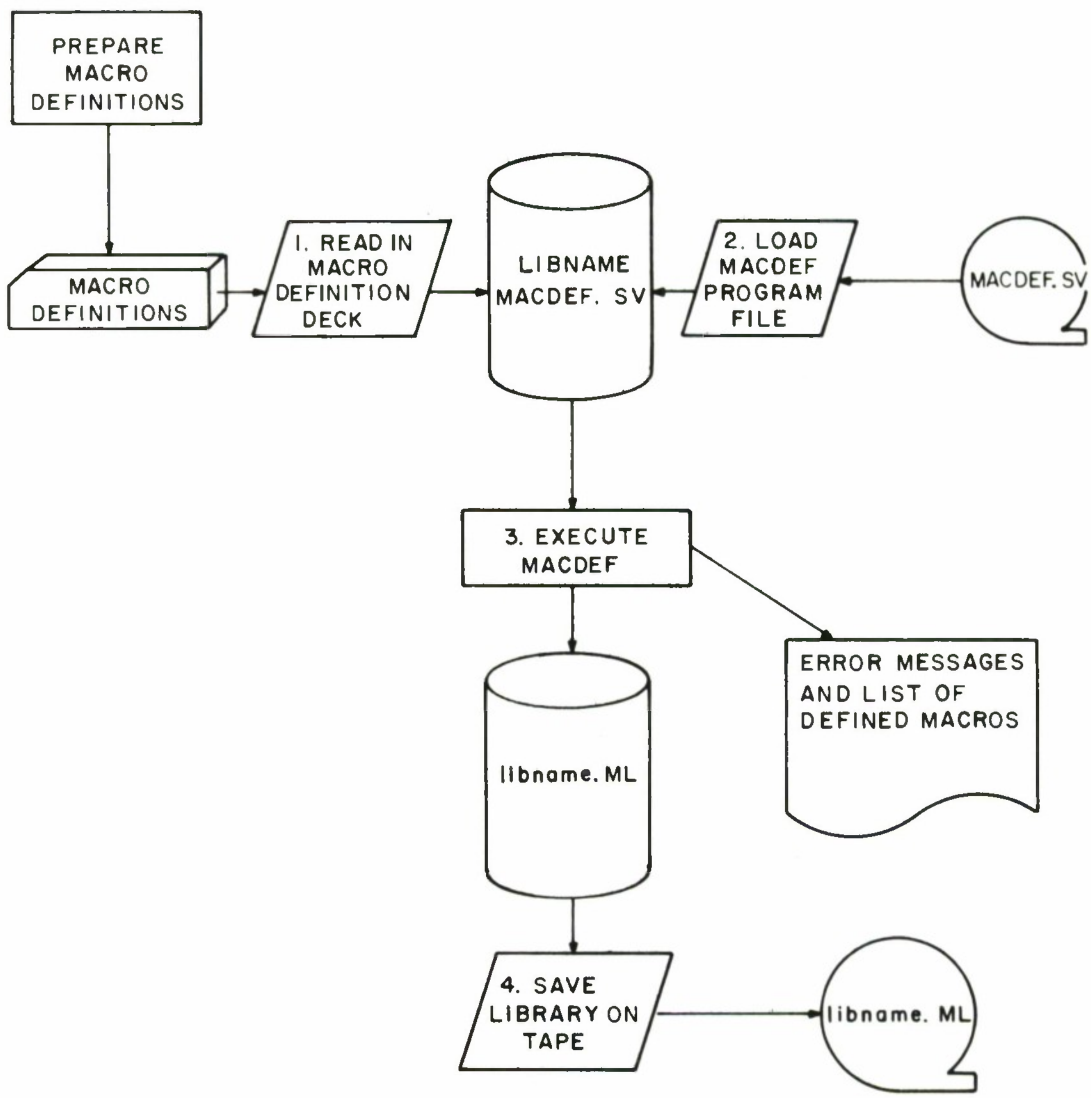


1. LOAD LIBRARY FILE	LOAD MTO:X lib1.ML lib2.ML
2. LOAD PROGRAM FILE	LOAD MTO:X SSUB.SV
3. LOAD INPUT FILE	XFER/A $CDR INPUTFILE
4. EXECUTE SSUB	SSUB INPUTFILE OUTPUTFILE lib1 lib2

UP TO 4 LIBRARIES ARE ALLOWED ALTHOUGH ONLY 2 ARE SHOWN ABOVE.

1A-41,689

Figure 1. SSUB SYSTEM FLOW



1. LOAD MACDEF PROGRAM	LOAD MTO:X MACDEF. SV
2. LOAD MACRO DEFINITION DECK	XFER/A $CDR libname
3. EXECUTE MACDEF	MACDEF libname
4. SAVE MACRO LIBRARY	DUMP MTO:X libname. ML

IA-41,690

Figure 2. MACDEF SYSTEM FLOW

Do not include the .ML after library names. Up to four names may be specified. All libraries must have been processed previously by MACDEF. Error codes will be printed on the line printer. An R ↵ typed out by the CLI indicates that the program is completed.

Input File

The input file contains source data containing macro calls and optionally macro definitions. It should be a normal ASCII file with a legal DOS name. Read-protect attribute must be off.

Output File

File must be new, with a legal DOS file name.

MACDEF

MACDEF is a separate program used to generate macro libraries for later use in SSUB runs. Input to the program is a file containing definitions of commonly used macros. MACDEF produces a file consisting of a library directory and the texts of all macro bodies in the library. This library file is generally saved on magnetic tape by the user for later use with the macro preprocessor program.

To use the MACDEF program the following steps should be performed:

1. Load the MACDEF save file.
2. Create a new file containing the definitions for all macros to be included in the library. The name given to this file is used to form the macro library name.
3. Ready the line printer.
4. Enter the following command at the teletype:

   MACDEF library-name ↵

The names of all defined macros and any error message codes will be printed on the line printer. An R ↲ typed out by the CLI indicates that the program is finished.

5. To save the library on tape, dump the file created by MACDEF. This file is named "library-name.ML".

6. If any errors are detected by MACDEF, the original file should be corrected, the .ML file deleted, and the program rerun.

Input File

The input file consists of up to 100 macro definitions. Extra cards should not be placed between macro definitions. The file should be a normal ASCII file with a legal DOS name. Read protect attribute should be off.

Output File

The output file is created on disk by MACDEF. The name of this file is the same as the input file with a .ML extension appended.

Output Message

Error messages from SSUB and MACDEF are output to the printer. Error messages have the following format:

"LINE line-number ERROR NO. number"

where "line-number" identifies a line in the input file and "number" identifies the type of error. In Table III errors related to macro definitions are listed under MACDEF although they may also occur in any SSUB run.

Error messages appearing on the teletype are DOS system messages and are described in the DOS User's Manual.

Table III

Output Messages
For Macro Processor

SSUB Errors

Number	Problems	Program Action
6	Input file not specified or not a legal DOS file.	Exit from program.
7	Disk read error.	Processing continues.
8	Output file already exists.	Exit from program.
9	a. Disk write error. b. Disk space exhausted.	Processing continues.
10	End of source data while processing quote string. Source data may be the input file, a macro parameter value, or a macro body.	String is terminated. If source is input file, exit from program. Otherwise processing continues.
11	Quote string greater than 1000 characters.	String terminated. Processing continues.
12	Illegal use of $ in source data.	Processing continues.
13	Illegal number of arguments in macro call.	Macro call is ignored. Processing continues.
14	Illegal delimiter character following $Q.	Processing continues. $Q ignored.
15	Preprocessor storage area exceeded.	No more argument values are accepted. Processing continues but other errors will likely occur.

Table III (Continued)

Output Messages
For Macro Processor

Number	Problems	Program Action
SSUB Errors		
16	Error in macro call argument a. No left parenthesis when arguments expected. b. End of input source before all argument values obtained.	Macro call is ignored. Processing continues.
17	Too many macros. Limit is 160.	Program is terminated.
19	Library file could not be opened.	Program terminates.
MACDEF Errors		
Number	Problems	Program Action
7	Disk read error.	Processing continues.
9	a. Disk write error. b. Disk space exhausted.	Processing continues.
30	Number of arguments on MDEF line not a digit.	Macro is not defined. Scan to next MDEF line.
31	Illegal or missing macro on MDEF line.	Macro is not defined. Scan to next MDEF line.
32	"MDEF" not found where expected.	Continues scan for "MDEF".
33	Unexpected end of input file a. While reading macro body. b. Extra characters follow final MEND line.	 Macro is terminated as if MEND found. Termination of program.

Table III (Concluded)

Output Messages
For Macro Processor

MACDEF Errors

Number	Problems	Program Action
34	Input file cannot be opened.	Termination of program.
35	Attempt to put more than 100 macros in a library.	Program terminates as if end of file read.

MACDEF Informational Message

"MACRO name DEFINED"

SECTION IV

SCENARIO ASSEMBLER

INTRODUCTION

The Scenario Assembler program (CVT) converts external (symbolic) scenarios into internal (absolute) scenarios which are tailored to a specific terminal type and data communications control procedure. This reduces the real-time work of the Scenario Interpreter in the area of scenario processing. To further ease the burden of the Scenario Interpreter, the Scenario Assembler performs character conversions where appropriate and adds start-of-message/end-of-message (SOM/EOM) sequences to queries to be sent to a system under test (SUT). CVT runs under Data General's Disk Operating System (DOS) and its operation must follow the conventions established by DOS. A complete description of the design and implementation of the Scenario Assembler can be found in Volume 4 of this series.

SYSTEM FLOW

The system flow of the assembly process is shown in Figure 3. The external scenarios may be input to the system from a card deck, from magnetic tape, or from the control teletype. The Scenario Assembler program (CVT.SV) and its associated tables, DEVTAB and SUTTAB, must be input from magnetic tape. The external scenarios, the Assembler, and the tables must reside on disk before execution is initiated. The symbol table is a temporary file written to disk during execution of the Assembler and then deleted at the end of the assembly. The listing on the line printer is also a temporary file and can be relisted only by re-executing the Assembler. The internal scenario is written to disk and can remain there or be written on magnetic tape for further use.

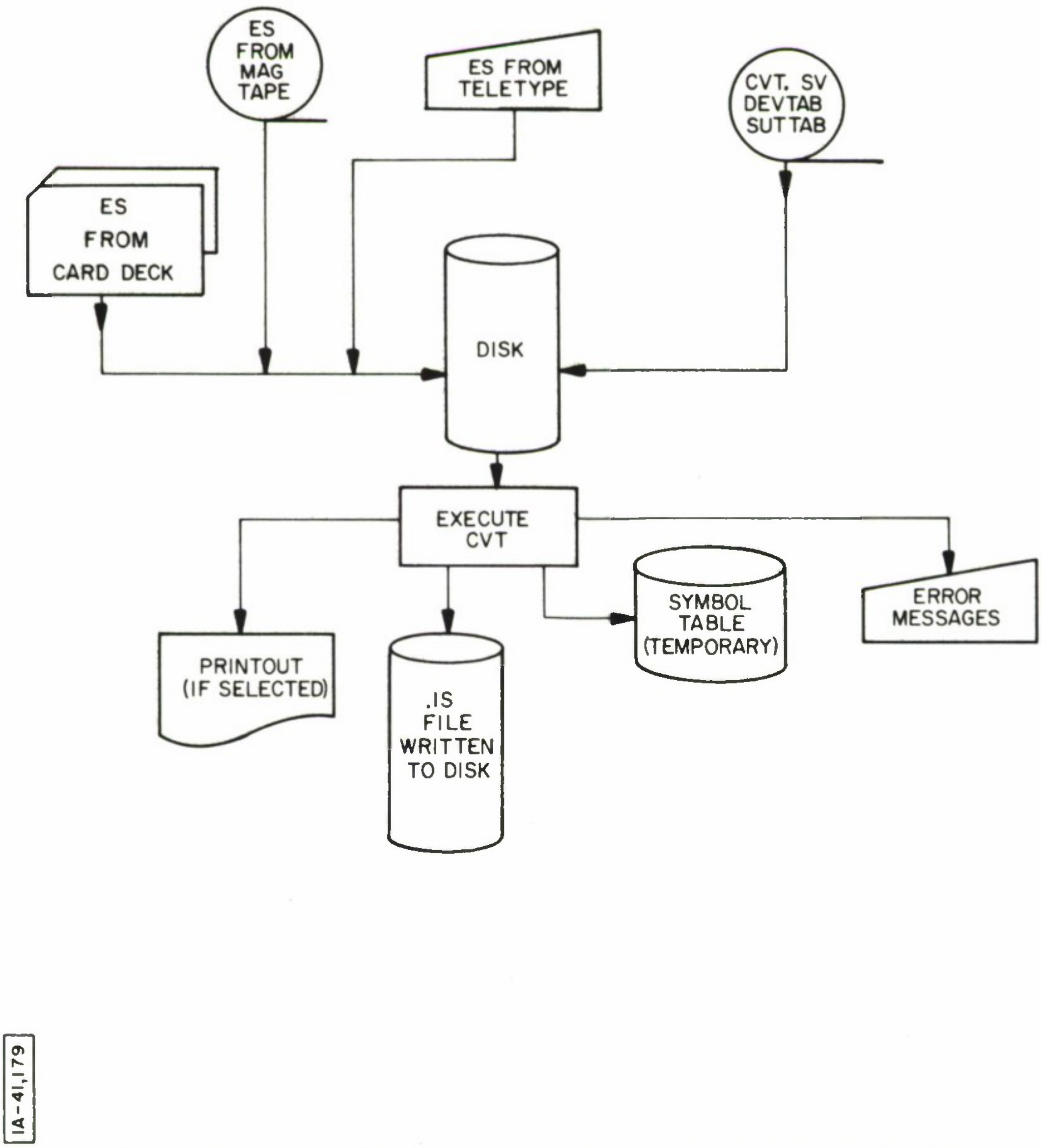


Figure 3 SYSTEM FLOW OF THE SCENARIO ASSEMBLER

OPERATING PROCEDURES

The Scenario Assembler operates with disk files only, and therefore all input files and the program save file itself must reside on disk before execution can begin.

Preparing Files

External Scenario

An external scenario (ES) is a stream of characters containing the scenario instructions to be assembled. The format of the ES is shown in Figure 4. The Assembler processes the ES one instruction at a time, interpreting a carriage return as the end of the instruction. This means that a scenario instruction is not restricted in its length, but must use a carriage return only as an instruction termination character.

The first field of an instruction is the op-code field, which is a single character defining the instruction type. The op-code must always appear as the first character of an instruction with no preceding blanks. If the first character of an instruction is a blank, the instruction is treated as a commend by the Assembler. Following the op-code are 0 to 3 fields, depending upon the requirements of the particular instruction type. These fields are separated by one or more blanks except that a blank between the first field (op-code) and the second field is optional. A detailed list of instruction types and their descriptions may be found in Volume 2 of this series.

Scenarios which are to be assembled may be loaded to disk in several ways, using the Command Line Interpreter (CLI) of the Disk Operating System.

Length in Bytes	Description*
4-6	Allocate instruction to cause a set of Registers to be allocated in core.
1	Instruction type or op code field.
0-j	From 0 to 3 fields (depending on instruction type) which generates fixed length fields in the internal scenario.
0-k	Either Ø or 1 variable length character string field (depending on instruction type). May include control characters.
1	Carriage return character which signals end of a scenario instruction.
	Above 4 fields are repeated for each instruction in the scenario.
	End of scenario signalled by end of DOS file.

*All character data

Figure 4. External Scenario Format

1. Load from tape to disk
   LOAD MTO:x scen

2. Transfer from card reader to disk
   XFER/A $CDR scen or LXFER $CDR scen

3. Created through the DOS Editor

4. Created as an output file of the Macro Preprocessor
   SSUB x scen (lib)

The various DOS commands and programs are fully described in the Data General Software Manuals (References 1 and 2). The Macro Preprocessor is described in Volume 3 of this series.

Program Files

The Assembler program and its associated conversion tables reside on tape as files, and they also must be read to disk. This can be accomplished with the DOS command

LOAD MTØ:x CVT.SV DEVTAB SUTTAB

This loads the Assembler program save file (CVT.SV) as well as the conversion table (DEVTAB) and start/end-of-message table (SUTTAB), from file x of a magnetic tape mounted on the system tape drive selected as transport Ø.

Executing Assembler

The Assembler can be operated in either conversational or non-conversational mode from the control teletype (TTY). In non-conversational mode, all input parameters are included in the initial call. In conversational mode, the Assembler requests the input parameters one at a time. To execute in non-conversational mode, type:

$$\text{CVT} \begin{bmatrix} /P \\ /N \end{bmatrix} \text{scen code1 code2}$$

where:

CVT	is the name of the Assembler program
P	is the optional partial print switch which provides a printout of the ES only
N	is the optional no-print switch
scen	is the name of the external scenario to be assembled
code1	indicates the conversion method and conversion subtable from DEVTAB to be used for string conversions (see Table IV)
code2	indicates the SOM/EOM sequence subtable from SUTTAB to be used (see Table V)

In both conversational and non-conversational modes, the Assembler types the message:

TO CANCEL RUN, TYPE CONTROL-A

which indicates that the assembly process has begun. The Assembler can be interrupted at any time during assembly by depressing the Control and A characters simultaneously.

For conversational mode enter:

$$\text{CVT}\begin{bmatrix}/\text{P}\\/\text{N}\end{bmatrix}$$

and the Assembler responds with:

ENTER EXTERNAL SCENARIO NAME

When a valid external scenario name is entered, followed by a carriage return, the program types:

ENTER CODE FOR CONVERSION

Table IV

Available Codes for Conversion

Code	Comment
1	A one-to-one conversion to 8-bit zero-parity ASCII where the leftmost bit is the parity bit and is always set to zero.
2	A one-to-one conversion to 8-bit even-parity ASCII where the leftmost bit is the parity bit and is set to one only if it is necessary to make the total number of bits in the byte even.
3	A one-to-several conversion to 7-bit 2741 EBCDIC where the parity bit (odd parity) is the rightmost bit, and a zero bit is added at the left to fill the byte. (See Appendix I)
4	A one-to-one conversion to 8-bit one-parity ASCII where the parity bit is the leftmost bit and is always set to 1.
5	A one-to-several conversion to 7-bit 2741 EBCDIC where the seven bits are in the reverse order of those in use for code3 and a zero bit is added at the left to fill the byte.
6	A one-to-several conversion to 7-bit 2741 Correspondence Code reversed for use on the field test system. The parity bit is right most bit and a zero bit is added at the left to fill the byte.

Table IV (Concluded)

Available Codes for Conversion

Code	Comment
7	A one-to-several conversion to 7-bit 2741 Correspondence Code for use in the fixed-site system. The parity bit is the rightmost and a zero bit is added at the left to fill the byte.
8	A one-to-one conversion to 8-bit odd parity ASCII where the leftmost bit is the parity bit.

Table V

Available Codes for SOM/EOM

Code	EOM	SOM
1	15_8 = CR	none
2	223_8	none
3	176_8 = ~	none
4	133_8 37_8 = CR C	26_8 = D
5	215_8 = CR	none
6	15_8 12_8 = CR LF	none
7	none	none
8	155_8 174_8 = CR C	64_8 = D
9	none	26_8 26_8 26_8 2_8 26_8 = SYN SYN SYN STX SYN
10	04 = EOT	none

An integer, from Table IV, should be entered, followed by a carriage return. The Assembler then asks:

ENTER CODE FOR END-OF-MESSAGE SEQUENCE

and a value from Table V should be entered. This completes the conversational mode of input.

If an assembly error occurs, the number of the line which caused it and the error message are printed on the teletype. This happens regardless of the print option selected. At the end of the run, or if Control-A is used, control is returned to the NOVA disk operating system (DOS) and an "R" is typed.

OUTPUT

Output of the Assembler is an internal scenario written to disk with the same name as the external scenario but with the extension .IS appended. If an internal scenario already exists for a particular scenario, the old one is automatically deleted and a new one is created for the new Assembly run. Other output of the Assembler includes optional printed listings on the line printer and messages printed to the teletype.

Internal Scenario

The internal scenario consists of 3 initial bytes of information, followed by processed scenario instructions, and ended by a 2-byte null word. The first information byte is an 8-bit error indicator, each bit being set only if a specific error occurred during assembly. The Scenario Interpreter will accept an internal scenario only if its first byte is zero, i.e., no errors have occurred.

The second byte of the internal scenario identifies the equipment type for which the scenario was assembled. It contains the conversion parameters used to assemble the scenario and make it specific to a given SUT and terminal. The first four bits are the conversion code (first input parameter) and the second four bits are the SOM/EOM code (second input parameter). If the internal scenario is completely independent of any conversion parameters (i.e., no queries are sent to or received from the SUT), the scenario is called universal, the equipment type is set to zero, and the Scenario Interpreter will accept it to run on any device because it has not been tailored for a particular SUT or terminal.

The third byte indicates the number of registers to be allocated for each use of this scenario. This number may vary from 3 to 127. The Assembler determines this number, not from input parameters as with byte two, but from an Assembler Directive instruction included within the scenario itself, preferably the first instruction. This instruction (op-code = a) should appear only once per scenario; if the instruction is missing, byte three contains the default value of 8.

The scenario instructions themselves follow these three initial bytes. Each instruction begins with a 2-byte length field, giving the length in bytes of the instruction, including the length field. The 1-byte op-code field is next. Depending upon the particular instruction requirements, there may follow 0 to 3 fixed length fields, 0 or 1 variable-length-string field, or no additional fields. The instructions immediately follow one another, with no intervening delimiters. The end of the internal scenario is signalled by a 2-byte null word.

Optional Listings

When running the Scenario Assembler, three print options are available for printing on a line printer.

1. full printing
2. partial printing
3. no printing

A sample output listing is given in Appendix II. Full printing is selected when invoking the Assembler by typing CVT without either the P or N options in either the conversational or non-conversational mode. This produces first a listing of the external scenario. Each line contains the external line number, the starting byte address of the corresponding instruction in the internal scenario, and then up to 58 more characters of the instruction. If the instruction is longer than 59 characters, it is truncated. Interspersed in this listing are error messages listed beneath the instructions which caused them.

The listing of the internal scenario appears after the external scenario. This begins with the printing of the error indicator, equipment type, and the Register allocation bytes. Each instruction of the ES is printed, followed by the corresponding internal scenario instruction if one exists (assembler directives are never written in the internal scenario). The internal scenario instruction is printed, 2 bytes on a line, preceded by the byte address, in decimal, of the first of the two bytes. Following the two bytes is the ASCII translation of the bytes with control characters printed as blanks. Two bytes are always printed. Therefore, if the instruction has an odd number of bytes, the first byte of the next instruction is printed and is also repeated as the first byte of the next IS instruction.

The symbol table is printed after the internal scenario. Each entry of the symbol table is represented by a line of print which gives the length of the label, the label, the internal scenario byte address associated with the label, and the line number of the external scenario instruction which first referenced the label. Also printed is the number of entries in the table. An example of the full printout is given in Appendix II.

The partial print option is selected by typing CVT/P in either the conversational or non-conversational mode. This option produces the listing of the external scenario as described above plus a printout of the name, indicator byte, equipment byte, and Register allocation byte of the internal scenario. The rest of the listing of the internal scenario and the listing of the symbol table are omitted.

The no-print option produces no listing to the line printer. As in the case of the other two options, if any errors occur, the error messages are printed on the teletype.

Output Messages

Messages are printed to the teletype for two reasons:

1. to request an input in conversational mode; and
2. to report an error.

Both types are self-explanatory. To correct errors in input parameters, input corrections must be typed in. For other messages, no immediate action is needed, except when it may be desirable to interrupt the assembly with a Control-A command. If assembly errors occur, they need to be corrected in the external scenario, and the external scenario needs to be reassembled. Otherwise, the error indicator byte will not be zero, and the internal scenario will not be accepted by the Scenario Interpreter. Table VI includes all possible output messages. The error message designates the number of the line which caused it, except for the LABEL UNDEFINED message which indicates the line number of the first reference to the label.

Table VI

Output Messages
For Scenario Assembler

<u>Messages Requiring Responses</u>

TO CANCEL RUN, USE CONTROL-A.

ENTER EXTERNAL SCENARIO NAME.

SCENARIO NAME NOT FOUND, RE-ENTER OR CANCEL RUN.

ENTER CODE FOR CONVERSION.

CONVERSION CODE NOT IN TABLE. ENTER NEW CODE OR CANCEL RUN.

ENTER CODE FOR END-OF-MESSAGE SEQUENCE.

END-OF-MESSAGE CODE NOT IN TABLE. ENTER NEW CODE OR CANCEL RUN.

TABLE NOT FOUND. CANCEL RUN.

<u>Messages Requiring No Responses</u>

TOO MANY FIELDS.

LABEL --- IS UNDEFINED.

ALLOCATE IS TOO SMALL.

UNDEFINED OP CODE = ---.

LITERAL MISSING.

OUT-OF-RANGE NUMBER.

WARNING, SHOULD DOUBLE QUOTE BE TWO SINGLE QUOTES.

LABEL MULTIPLY DEFINED.

FIELD MISSING.

ILLEGAL FIELD.

SECTION V

EQUIPMENT TABLE

INTRODUCTION

The Equipment Table (ET) is not considered part of the Scenario Interpreter, but is a separate entity to be created by the user to reflect the characteristics of the equipment to be emulated. The Equipment Table consists of a set of ET entries which describe the SUT remote-terminal equipment to be emulated (as well as the control TTY), and relate it to the emulator I/O ports. Each entry (25_8 words long) describes one equipment component of the SUT. The format of an ET entry is given in Table V of Volume 2 of this series.

GENERATION

The Equipment Table must be generated by the user to depict the particular equipment configuration to be emulated. A source file (EQUIP) of the ET is normally created and then assembled with the NOVA assembler. The assembled file (EQUIP.RB) must be included when generating an emulator system, as described in Section VI.

(The EQUIP file contains several items in addition to the ET. The ET history record (ETREC), which is the second record written on the log tape during a run, is a proper subset of EQUIP. The ET itself is a proper subset of ETREC. The requirements and conventions of EQUIP, ETREC, and ET will be clarified in the next subsection.)

The ET source file, EQUIP, is normally written in NOVA assembly language, with each entry correctly formatted. This can be accomplished by creating the file line by line as needed, or by using macros and the macro processor to ease the burden of repetition. Most often macros will be used. The macros used to create an Equipment Table for the present field-test system (including digital I/O facilities) are described in Figure 5.

An ET entry is generated by a sequence of four ordered macro calls: either ETENTRY1, ETENTRY5, ETENTRY3, ETENTRY6 or ETENTRY1, ETENTRY2, ETENTRY3, ETENTRY6. The only difference between the two sequences is that the former (ETENTRY5) allows ETEOM to be specified as a parameter whereas the latter (ETENTRY2) generates an ETEOM value of EOM1. For ease of reading the assembly listing, the ETENTRY1 card should start in column 1 and the others in column 10.*

An input file, EQ, for an Equipment Table with macros not yet expanded is shown in Figure 6. The six macro definitions used to create the EQUIP file from the EQ file appear at the beginning of the EQ file. A seventh macro definition occurs later in lines 89-91 but is is not essential to the proper formatting of the file. Figure 7 shows a portion of the EQUIP file after execution of the macro processor. In this form, the file is acceptable to the NOVA assembler. Figure 8 shows a portion of the ET after it has been assembled. Appendix III contains a complete listing of EQUIP.RB, the assembled Equipment Table file.

REQUIREMENTS AND CONVENTIONS

The following mandatory requirements must be met by EQUIP, ETREC, and the ET. Line numbers referenced below are those of Figure 6.

1. The following (defined below) must be declared as entry points (external/global variables) as shown at lines 44 to 48: EØØØØ, EØ, E1, ETREC, ETEND, ETENT, E2, ETLEN.

---

*For the lab system (one with no digital I/O), ETENTRY6 can be eliminated and ETENTRY3 modified to generate zero values for words 22_8 - 24_8.

MACRO NAME	PURPOSE	PARAMETERS
ETENTRY1	Generates words 0-5 of ET entry.	$1 = NOVA assembler label for ET Entry $2 = ETRO. Should be initialized to zero. $3 = first ASCII character of ETYPE. $4 = second ASCII character of ETYPE. $5 = ETID in decimal. $6 = CHILD. The NOVA assembler label for some other ET entry or zero. $7 = LINK. The NOVA assembler label for some other ET entry or zero. $8 = PARNT. The NOVA assembler label for some other ET entry or zero.
ETENTRY2	Generates words $6\text{-}17_8$ of ET entry with help of ETENTRY4	$1 = ETRAT in octal.* $2 = TERMT in octal. $3 = STATI. Enter I or U. $4 = PORTO in octal. $5 = PORTI in octal. $6 = SPRTO in octal. $7 = SPRTI in octal.

*To enter a decimal value, follow it with a decimal point.

Figure 5. Equipment Table Macros

MACRO NAME	PURPOSE	PARAMETERS
ETENTRY3	Generates words $2\emptyset_8$ - 21_8 of ET entry	\$1 = SUTAD in octal. \$2 = ETIND in octal. Bits 1, 2, and 3 should be initialized to zero, the others as desired. Bit Ø must be set to 1 for the control TTY. \$3 = BYTEL in octal. \$4 = PARTY in octal.
ETENTRY6	Generates words 22_8 - 24_8 of ET entry	\$1 = CCC+1 in ETDID. Number of digital inputs in decimal. \$2 = BSSSS in ETDID. First input in octal. \$3 = DDDDDD in ETDID. Device number in octal. \$4 = CCC+1 in ETDOD. Number of digital outputs in decimal. \$5 = BSSSS in ETDOD. First output in octal. \$6 = DDDDDD in ETDOD. Device number in octal. \$7 = ETDOA.

Figure 5. Equipment Table Macros (Continued)

MACRO NAME	PURPOSE	PARAMETERS
ETENTRY5	Generates words $6\text{-}17_8$ of ET entry with help of ETENTRY4	\$1 = ETRAT in octal*. \$2 = TERMT in octal. \$3 = STATI. Enter I or U. \$4 = PORTO in octal. \$5 = PORTI in octal. \$6 = SPRTO in octal. \$7 = SPRTI in octal. \$8 = ETEOM.
ETENTRY4	Generates words 14_8 - 17_8 when called by ETENTRY2 or ETENTRY5	\$1 = unused. Enter zero. \$2 = TERMT in octal. \$3 = STATI. Enter I or U. \$4 = PORTO in octal. \$5 = PORTI in octal. \$6 = SPRTO in octal. \$7 = SPRTI in octal.
* To enter a decimal value, follow it with a decimal point		

Figure 5. Equipment Table Macros (Concluded)

The following labels must be used for particular ET entries (although the user may also assign labels of his own choice to the same entries):

2. The label EØØØØ must be used for the first ET entry which must be the control TTY (see line 83).

3. The label EØ must be used for the ET entry for the control TTY. Therefore, EØ is equivalent to EØØØØ. See line 84; the first parameter of the macro ETENTRY1 is the label EØ.

4. The label E1 must be used for the ET entry for the single asynchronous device in the lab system (see line 89).

5. The label E2 must be used for the ET entry for the first asynchronous device in the field-test system and for the first DCM device in the lab system. The Exec assumes that the ET entries for the asynchronous devices in the 64-line field-test system are ordered as shown in Figure 9 and that those for the DCM in the lab system are ordered as shown in Figure 10. (Figure 11 shows the ordering and device numbers used for the 16-line field-test system which are those of Figure 6, lines 96-159.)

ETREC must be defined so that:

6. It includes the entire ET, preceded by four words as shown in Table XII of Volume 2 of this series (see lines 79-191).

EQUIP must include the following definitions:

7. ETEND must contain the length of an ET entry (see line 78).

8. ETLEN is equivalent to ETEND (see line 195).

9. ETENT must contain the number of ET entries (see line 194).

10. One or more EOM lists must be established as in lines 196-225. An EOM list is of variable length, terminated by -1

```
EQ
            .TITL EQUIP
MDEF ETENTRY1(8)
$1:
            $2                        ;ETR0
            "$3*256.+"$4              ;ETYPE
            $5.                       ;ETID
            $6                        ;CHILD
            $7                        ;LINK
            $8                        ;PARNT
MEND
MDEF ETENTRY2(7)
$1                                    ;ETRAT
            0                         ;ETQBP
            EDM1                      ;ETEDM
            0                         ;ETRSP
            0                         ;ETPAD
            0                         ;RRING,PRING
ETENTRY4(0,$2,$3,$4,$5,$6,$7)
MEND
MDEF ETENTRY4(7)
            0*256.+$7                 ;ETLGA, ETLGN
            $2*256.+$3                ;TERMT, STATI
            $4*256.+$5                ;PORTO, PORTI
            $6*256.+$7                ;SPRTO, SPRTI
MEND
MDEF ETENTRY5(8)
$1                                    ;ETRAT
            0                         ;ETQBP
            $8                        ;ETEDM
            0                         ;ETRSP
            0                         ;ETPAD
            0                         ;RRING,PRING
ETENTRY4(0,$2,$3,$4,$5,$6,$7)
MEND
MDEF ETENTRY3(4)
$1*256.+$2B8                          ;SUTAD, ETIND
            $3*256.+$4                ;BYTEL, PARTY
MEND
MDEF ETENTRY6(7)
$1.-1*1B2+$2B7+$3                     ;ETDID
            $4.-1*1B2+$5B7+$6         ;ETDUD
            $7                        ;ETDOA
MEND
            .ENT E0000,E0,E1,ETREC
            .ENT ETEND
            .ENT ETENT
            .ENT E2
            .ENT ETLEN
            .DUSR A=101
            .DUSR I=111
            .DUSR S=123
            .DUSR T=124
            .DUSR U=125
            .DUSR W=127
            .DUSR E=105
            .DUSR Z=132
            .DUSR N=116
            .DUSR O=117
            .DUSR RT1=135.
            .DUSR BL1=7.
            .DUSR BL2=8.
            .DUSR PT1=0
```

Figure 6. File EQ of Equipment Table (Macros not Expanded)

```
                .DUSM PT2=N
                .DUSM ODDLINE=3*16.+4
                .DUSM IBM2848=3*16.+4
                .DUSM IBM2260=3*15.+4
                .DUSM IBM1053=5
                .DUSM D2000=6
                .DUSM IBM2741=7
                .DUSM I2741=3*15.+4
                .DUSM ZASC1=1*15.+1
                .DUSM ZASC6=1*15.+6
                .DUSM EASC2=2*16.+2
                .DUSM EASC5=2*15.+5
                .TXTM E
                .ZREL
                .NREL
ETEND:          E0END-E0000
ETREC:          E0077+"E                ;USED TO WRITE ET ON TAPE
                E9999-E0000+4
                "M
                .+1
E0000:
ETENTRY1(E0,0,C,T,0,0,E1,0)
                ETENTRY5(110.,ZASC6,I,11,10,0,0,EOM2)
                ETENTRY3(0,1,9.,Z)
                ETENTRY5(1,0,0,1,0,0,0)
E0END:
ETENTRY1(E1,0,C,S,14,0,E2,0)
                ETENTRY5(0110.,EASC2,I,51,50,1,1,EOM3)
                ETENTRY3(15.,0,9.,E)
                ETENTRY5(1,0,0,1,0,0,0)
MDEF TTY33
I2741
MEND
ETENTRY1(E2,0,T,Y,1,0,E3,0)
                ETENTRY5(RT1,TTY33,I,43,42,1,1,EOM4)
                ETENTRY3(30.,0,BL1,PT1)
                ETENTRY6(2,00.,71,4,00.,66,D066A)
ETENTRY1(E3,0,T,Y,2,0,E4,0)
                ETENTRY5(RT1,TTY33,I,43,42,2,2,EOM5)
                ETENTRY3(31.,0,BL1,PT1)
                ETENTRY5(2,02.,71,4,04.,66,D066A)
ETENTRY1(E4,0,T,Y,3,0,E4A,0)
                ETENTRY2(RT1,TTY33,I,43,42,3,3)
                ETENTRY3(32.,0,BL1,PT1)
                ETENTRY6(2,04.,71,4,08.,66,D066A)
ETENTRY1(E4A,0,T,Y,4,0,E13,0)
                ETENTRY2(RT1,TTY33,I,43,42,4,4)
                ETENTRY3(29.,0,BL1,PT1)
                ETENTRY6(2,06.,71,4,12.,66,D066A)
ETENTRY1(E13,0,T,Y,5,0,E14,0)
                ETENTRY2(RT1,TTY33,I,43,42,5,5)
                ETENTRY3(33.,0,BL1,PT1)
                ETENTRY6(2,08.,71,4,16.,66,D066B)
ETENTRY1(E14,0,T,Y,6,0,E15,0)
                ETENTRY2(RT1,TTY33,I,43,42,6,6)
                ETENTRY3(34.,0,BL1,PT1)
                ETENTRY6(2,10.,71,4,20.,66,D066B)
ETENTRY1(E15,0,T,Y,7,0,E16,0)
                ETENTRY2(RT1,TTY33,I,43,42,7,7)
                ETENTRY3(35.,0,BL1,PT1)
                ETENTRY5(2,12.,71,4,24.,66,D066B)
ETENTRY1(E16,0,T,Y,8,0,E17,0)
                ETENTRY2(RT1,TTY33,I,43,42,8.,8.)
```

Figure 6. File EQ of Equipment Table (Macros not Expanded) (continued)

```
            ETENTRY3(36.,0,BL1,PT1)
            ETENTRY6(2,14.,71,4,28.,66,D060B)
ETENTRY1(E17,0,T,Y,9,0,E18,0)
            ETENTRY2(RT1,TTY33,I,45,44,1,1)
            ETENTRY3(37.,0,BL1,PT1)
            ETENTRY6(2,16.,71,4,00.,67,D067A)
ETENTRY1(E18,0,T,Y,10,0,E19,0)
            ETENTRY2(RT1,TTY33,I,45,44,2,2)
            ETENTRY3(38.,0,BL1,PT1)
            ETENTRY6(2,18.,71,4,04.,67,D067A)
ETENTRY1(E19,0,T,Y,11,0,E20,0)
            ETENTRY2(RT1,TTY33,I,45,44,3,3)
            ETENTRY3(39.,0,BL1,PT1)
            ETENTRY6(2,20.,71,4,08.,67,D067A)
ETENTRY1(E20,0,T,Y,12,0,E21,0)
            ETENTRY2(RT1,TTY33,I,45,44,4,4)
            ETENTRY3(40.,0,BL1,PT1)
            ETENTRY6(2,22.,71,4,12.,67,D067A)
ETENTRY1(E21,0,T,Y,13,0,E22,0)
            ETENTRY2(RT1,TTY33,I,45,44,5,5)
            ETENTRY3(41.,0,BL1,PT1)
            ETENTRY6(2,24.,71,4,16.,67,D067B)
ETENTRY1(E22,0,T,Y,14,0,E23,0)
            ETENTRY2(RT1,TTY33,I,45,44,6,6)
            ETENTRY3(42.,0,BL1,PT1)
            ETENTRY6(2,26.,71,4,20.,67,D067B)
ETENTRY1(E23,0,T,Y,15,0,E24,0)
            ETENTRY2(RT1,TTY33,I,45,44,7,7)
            ETENTRY3(43.,0,BL1,PT1)
            ETENTRY6(2,28.,71,4,24.,67,D067B)
ETENTRY1(E24,0,T,Y,16,0,E5,0)
            ETENTRY2(RT1,TTY33,I,45,44,8.,8.)
            ETENTRY3(44.,0,BL1,PT1)
            ETENTRY6(2,30.,71,4,28.,67,D067B)
ETENTRY1(E5,0,L,N,5,E6,0,0)
            ETENTRY2(2400.,DDDLINE,I,32,31,0,0)
            ETENTRY3(43.,0,BL2,PT2)
            ETENTRY6(1,0,0,1,0,0,0)
ETENTRY1(E6,0,C,N,6,E8,E7,E5)
            ETENTRY2(2400.,IBM2848,I,32,31,0,0)
            ETENTRY3(116,0,BL2,PT2)
            ETENTRY6(1,0,0,1,0,0,0)
ETENTRY1(E7,0,C,N,7,E11,0,E5)
            ETENTRY2(2400.,IBM2848,U,32,31,0,0)
            ETENTRY3(250,0,BL2,PT2)
            ETENTRY6(1,0,0,1,0,0,0)
ETENTRY1(E8,0,D,S,8,0,E9,E6)
            ETENTRY2(2400.,IBM2260,I,32,31,0,0)
            ETENTRY3(240,0,BL2,PT2)
            ETENTRY6(1,0,0,1,0,0,0)
ETENTRY1(E9,0,D,S,9,0,E10,E6)
            ETENTRY2(2400.,IBM2260,I,32,31,0,0)
            ETENTRY3(241,0,BL2,PT2)
            ETENTRY6(1,0,0,1,0,0,0)
ETENTRY1(E10,0,P,T,10,0,0,E6)
            ETENTRY2(150.,IBM1053,U,32,0,0,0)
            ETENTRY3(242,0,BL2,PT2)
            ETENTRY6(1,0,0,1,0,0,0)
ETENTRY1(E11,0,D,S,11,0,E12,E7)
            ETENTRY2(2400.,IBM2260,U,32,31,0,0)
            ETENTRY3(244,0,BL2,PT2)
            ETENTRY6(1,0,0,1,0,0,0)
ETENTRY1(E12,0,D,S,12,0,0,E7)
```

Figure 6. File EQ of Equipment Table (Macros not Expanded) (continued)

```
                ETENTRY2(2400.,IBM2260,0,32,31,0,0)
                ETENTRY3(245,0,BL2,PT2)
                ETENTRY6(1,0,0,1,0,0,0)
    E9999:
    LEN         =ETEND-E0002
    ETENT:      E9999-E0002/LEN
    ETLEN:      LEN
    EQM1:       37
                -1
                -1
                -1
                -1
                -1
    EQM2:       12
                5
                3
                -1
                -1
                -1
    EQM3:       -1
                -1
                -1
                -1
                -1
                -1
    EQM4:       37
                -1
                -1
                -1
                -1
                -1
    EQM5:       37
                43
                -1
                -1
                -1
                -1
    D066A:      0
    D066B:      0
    D067A:      0
    D067B:      0
                .END
```

Figure 6. File EQ of Equipment Table (Macros not Expanded) (concluded)

```
EQUIP
                .TITL EQUIP
                .ENT E0000,EM,E1,ETREC
                .ENT ETEND
                .ENT ETLNT
                .ENT E2
                .ENT ETLEN
                .DUSR A=101
                .DUSR I=111
                .DUSR S=123
                .DUSR T=124
                .DUSR U=125
                .DUSR W=127
                .DUSR E=105
                .DUSR Z=132
                .DUSR N=116
                .DUSR O=117
                .DUSR RT1=135.
                .DUSR BL1=7.
                .DUSR BL2=8.
                .DUSR PT1=O
                .DUSR PT2=N
                .DUSR UPDLINE=3*16.+4
                .DUSR IBM2848=3*16.+4
                .DUSR IBM2260=3*16.+4
                .DUSR IBM1053=5
                .DUSR U2000=6
                .DUSR IBM2741=7
                .DUSR I2741=3*16.+4
                .DUSR ZASC1=1*16.+1
                .DUSR ZASC6=1*16.+6
                .DUSR EASC2=2*16.+2
                .DUSR EASC5=2*16.+5
                .TXTM 5
                .ZREL
                .NREL
    ETEND:      E0END-E0000
    ETREC:      2000+"E                 ;USED TO WRITE ET ON TAPE
                E9999-E0000+4
                "H
                .+1
    E0000:
    E0:
                0                     ;ETR0
                "C*256.+"T           ;ETYPE
                0.                    ;ETID
                0                     ;CHILD
                E1                     ;LINK
                0                     ;PARNT
                110.                                ;ETRAT
                0                      ;ETQBP
                EOM2                      ;ETEOM
                0                      ;ETRSP
                0                      ;ETPAD
                0                      ;RRING,PRING
                0*256.+37              ;ETLGA, ETLGN
                ZASC6*256.+I              ;TERMT, STATI
                11*256.+10             ;PORTO, PORTI
                0*256.+0             ;SPRTO, SPRTI
                0*256.+188                     ;SUTAD, ETIND
                6.*256.+Z            ;BYTEL, PARTY
                1.-1*182+087+0              ;ETDID
              . 1.-1*182+087+0    ;ETDOD
```

Figure 7. Portion of File EQUIP of Equipment Table (Macros Expanded)

```
         0                     ;ETDDA
    E0END:
    E1:
         0                     ;ETR0
         "D*256.+"S            ;ETYPE
         14.                   ;ETID
         0                     ;CHILD
         E2                    ;LINK
         0                     ;PARNT
         0110.                 ;ETRAT
         0                     ;ETQBP
         EOM3                  ;ETEOM
         0                     ;ETRSP
         0                     ;ETPAD
         0                     ;RRING,PRING
         0*256.+37             ;ETLGA, ETLGN
         EASC2*256.+I          ;TERMT, STATI
         51*256.+50            ;PORTO, PORTI
         1*256.+1              ;SPRTO, SPRTI
         15.*256.+0B8          ;SUTAD, ETIND
         8.*256.+E             ;BYTEL, PARTY
         1.-1*182+0B7+0        ;ETDID
         1.-1*182+0B7+0        ;ETDOD
         0                     ;ETDDA
    E2:
         0                     ;ETR0
         "T*256.+"Y            ;ETYPE
         1.                    ;ETID
         0                     ;CHILD
         E3                    ;LINK
         0                     ;PARNT
         RT1                   ;ETRAT
         0                     ;ETQBP
         EOM4                  ;ETEOM
         0                     ;ETRSP
         0                     ;ETPAD
         0                     ;RRING,PRING
         0*256.+37             ;ETLGA, ETLGN
         12741*256.+I          ;TERMT, STATI
         43*256.+42            ;PORTO, PORTI
         1*256.+1              ;SPRTO, SPRTI
         30.*256.+0B8          ;SUTAD, ETIND
         BL1*256.+PT1          ;BYTEL, PARTY
         2.-1*182+00.B7+71     ;ETDID
         4.-1*182+00.B7+56     ;ETDOD
         D086A                 ;ETDDA
    E3:
         0                     ;ETR0
         "T*256.+"Y            ;ETYPE
         2.                    ;ETID
         0                     ;CHILD
         E4                    ;LINK
         0                     ;PARNT
         RT1                   ;ETRAT
         0                     ;ETQBP
         EOM5                  ;ETEOM
         0                     ;ETRSP
         0                     ;ETPAD
         0                     ;RRING,PRING
         0*256.+37             ;ETLGA, ETLGN
         12741*256.+I          ;TERMT, STATI
         43*256.+42            ;PORTO, PORTI
         2*256.+2              ;SPRTO, SPRTI
```

Figure 7. Portion of File EQUIP of Equipment Table (Continued)

```
        31.*256.+0B8                ;SUTAD, ETIND
        5L1*256.+PT1             ;BYTEL, PARTY
        2.-1*182+02.B7+71               ;ETDID
        4.-1*182+04.B7+65    ;ETDOD
        DD06A                      ;ETDOA
E4:
        0                       ;ETR0
        "T*256.+"Y             ;ETYPE
        3.                      ;ETID
        0                       ;CHILD
        E4A                       ;LINK
        0                       ;PARNT
        RT1                                 ;ETRAT
        0                         ;ETQBP
        EOM1                      ;ETEOM
        0                         ;ETRSP
        0                         ;ETPAD
        0                         ;RRING,PRING
        0*256.+37                 ;ETLGA, ETLGN
        I2741*256.+I                 ;TERMT, STATI
        43*256.+42               ;PORTO, PORTI
        3*256.+3               ;SPRTO, SPRTI
        32.*256.+0B8                ;SUTAD, ETIND
        5L1*256.+PT1             ;BYTEL, PARTY
        2.-1*182+04.B7+71               ;ETDID
        4.-1*182+08.B7+65    ;ETDOD
        DD06A                      ;ETDOA
E4A:
        0                       ;ETR0
        "T*256.+"Y             ;ETYPE
        4.                      ;ETID
        0                       ;CHILD
        E13                       ;LINK
        0                       ;PARNT
        RT1                                 ;ETRAT
        0                         ;ETQBP
        EOM1                      ;ETEOM
        0                         ;ETRSP
        0                         ;ETPAD
        0                         ;RRING,PRING
        0*256.+37                 ;ETLGA, ETLGN
        I2741*256.+I                 ;TERMT, STATI
        43*256.+42               ;PORTO, PORTI
        4*256.+4               ;SPRTO, SPRTI
        29.*256.+0B8                ;SUTAD, ETIND
        5L1*256.+PT1             ;BYTEL, PARTY
        2.-1*182+06.B7+71               ;ETDID
        4.-1*182+12.B7+66    ;ETDOD
        DD06A                      ;ETDOA
E13:
        0                       ;ETR0
        "T*256.+"Y             ;ETYPE
        5.                      ;ETID
        0                       ;CHILD
        E14                       ;LINK
        0                       ;PARNT
        RT1                                 ;ETRAT
        0                         ;ETQBP
        EOM1                      ;ETEOM
        0                         ;ETRSP
        0                         ;ETPAD
        0                         ;RRING,PRING
        0*256.+37                 ;ETLGA, ETLGN
```

Figure 7. Portion of File EQUIP of Equipment Table (Continued)

```
          8L2*256.+PT2          ;BYTEL, PARTY
          1.-1*182+0B7+0            ;ETDID
          1.-1*182+0B7+0     ;ETDDD
          0                   ;ETDOA
    E11:
          0                   ;ETR0
          "D*256.+"S         ;ETYPE
          11.                  ;ETID
          0                   ;CHILD
          E12                   ;LINK
          E7                   ;PARNT
          2400.                              ;ETRAT
          0                    ;ETQBP
          EDM1                 ;ETEDM
          0                    ;ETRSP
          0                    ;ETPAD
          0                    ;RRING,PRING
          0*256.+37            ;ETLGA, ETLGN
          IBM2260*256.+U           ;TERMT, STATI
          32*256.+31           ;PORTO, PORTI
          0*256.+0           ;SPRTO, SPRTI
          244*256.+088                   ;SUTAD, ETIND
          8L2*256.+PT2          ;BYTEL, PARTY
          1.-1*182+0B7+0            ;ETDID
          1.-1*182+0B7+0     ;ETDDD
          0                   ;ETDOA
    E12:
          0                   ;ETR0
          "D*256.+"S         ;ETYPE
          12.                  ;ETID
          0                   ;CHILD
          0                   ;LINK
          E7                   ;PARNT
          2400.                              ;ETRAT
          0                    ;ETQBP
          EDM1                 ;ETEDM
          0                    ;ETRSP
          0                    ;ETPAD
          0                    ;RRING,PRING
          0*256.+37            ;ETLGA, ETLGN
          IBM2260*256.+U           ;TERMT, STATI
          32*256.+31           ;PORTO, PORTI
          0*256.+0           ;SPRTO, SPRTI
          245*256.+088                   ;SUTAD, ETIND
          8L2*256.+PT2          ;BYTEL, PARTY
          1.-1*182+0B7+0            ;ETDID
          1.-1*182+0B7+0     ;ETDDD
          0                   ;ETDOA
    E9999:
    LEN   =E0END-E0000
    ETENT: E9999-E0000/LEN
    ETLEN: LEN
    EDM1: 37
          -1
          -1
          -1
          -1
          -1
    EDM2: 12
          5
          30
          -1
          -1
```

Figure 7. Portion of File EQUIP of Equipment Table (Continued)

```
                -1
EOM3:           -1
                -1
                -1
                -1
                -1
                -1
EOM4:           37
                -1
                -1
                -1
                -1
                -1
EOM5:           37
                43
                -1
                -1
                -1
                -1
D066A:          0
D066B:          0
D067A:          0
D067B:          0
                .END
```

Figure 7. Portion of File EQUIP of Equipment Table
(Concluded)

```
0001  EQUIP
                         .TITL EQUIP
                         .ENT E0000,E0,E1,ETREC
                         .ENT ETEND
                         .ENT ETENT
                         .ENT E2
                         .ENT ETLEN
        000101           .DUSR A=101
        000111           .DUSR I=111
        000123           .DUSR S=123
        000124           .DUSR T=124
        000125           .DUSR U=125
        000127           .DUSR W=127
        000105           .DUSR E=105
        000132           .DUSR Z=132
        000116           .DUSR N=116
        000117           .DUSR O=117
        000207           .DUSR RT1=135.
        000007           .DUSR BL1=7.
        000010           .DUSR BL2=8.
        000117           .DUSR PT1=O
        000116           .DUSR PT2=N
        000064           .DUSR DDDLINE=3*16.+4
        000064           .DUSR IBM2848=3*16.+4
        000064           .DUSR IBM2260=3*16.+4
        000005           .DUSR IBM1053=5
        000006           .DUSR D2000=6
        000007           .DUSR IBM2741=7
        000064           .DUSR I2741=3*16.+4
        000021           .DUSR ZASC1=1*16.+1
        000026           .DUSR ZASC6=1*16.+6
        000042           .DUSR EASC2=2*16.+2
        000045           .DUSR EASC5=2*16.+5
        000005           .TXTM 5
                         .ZREL
                         .NREL
 00000'000025 ETEND:     E0END-E0000
 00001'020105 ETREC:     20000+"E              ;USED TO WRITE ET ON TAPE
 00002'001046            E9999-E0000+4
 00003'000110            "H
 00004'000005'           .+1
              E0000:
              E0:
 00005'000000            0                    ;ETR0
 00006'041524            "C*256.+"T          ;ETYPE
 00007'000000            0.                   ;ETID
 00010'000000            0                    ;CHILD
 00011'000032'           E1                   ;LINK
 00012'000000            0                    ;PARNT
 00013'000156            110.                                  ;ETRAT
 00014'000000            0                     ;ETQBP
 00015'001057'           EOM2                    ;ETEOM
 00016'000000            0                     ;ETRSP
 00017'000000            0                     ;ETPAD
 00020'000000            0                     ;RRING,PRING
 00021'000037            0*256.+37             ;ETLGA, ETLGN
 00022'013111            ZASC6*256.+I            ;TERMT, STATI
 00023'004410            11*256.+10            ;PORTO, PORTI
 00024'000000            0*256.+0             ;SPRTO, SPRTI
 00025'000200            0*256.+1B8                     ;SUTAD, ETIND
```

Figure 8. Portion of File EQUIP.RB, Assembled Equipment Table

```
0002  EQUIP
 00026'004132          8.*256.+Z           ;BYTEL, PARTY
 00027'000000          1.-1*1B2+0B7+0              ;ETDIO
 00030'000000          1.-1*1B2+0B7+0     ;ETDOO
 00031'000000          0                   ;ETDOA
                 EQEND:
                 E1:
 00032'000000          0                   ;ETR0
 00033'042123          "D*256.+"S         ;ETYPE
 00034'000016          14.                  ;ETID
 00035'000000          0                   ;CHILD
 00036'000057'         E2                   ;LINK
 00037'000000          0                   ;PARNT
 00040'000155          0110.                               ;ETRAT
 00041'000000          0                    ;ETQBP
 00042'001065'         EOM3                  ;ETEOM
 00043'000000          0                    ;ETRSP
 00044'000000          0                    ;ETPAD
 00045'000000          0                    ;RRING,PRING
 00046'000037          0*256.+37            ;ETLGA, ETLGN
 00047'021111          EASC2*256.+I           ;TERMT, STATI
 00050'024450          51*256.+50           ;PORTO, PORTI
 00051'000401          1*256.+1            ;SPRTO, SPRTI
 00052'007400          15.*256.+0B8                   ;SUTAD, ETINO
 00053'004105          8.*256.+E           ;BYTEL, PARTY
 00054'000000          1.-1*1B2+0B7+0              ;ETDIO
 00055'000000          1.-1*1B2+0B7+0     ;ETDOO
 00056'000000          0                   ;ETDOA
                 E2:
 00057'000000          0                   ;ETR0
 00060'052131          "T*256.+"Y         ;ETYPE
 00061'000001          1.                  ;ETID
 00062'000000          0                   ;CHILD
 00063'000104'         E3                   ;LINK
 00064'000000          0                   ;PARNT
 00065'000207          RT1                              ;ETRAT
 00066'000000          0                    ;ETQBP
 00067'001073'         EOM4                  ;ETEOM
 00070'000000          0                    ;ETRSP
 00071'000000          0                    ;ETPAD
 00072'000000          0                    ;RRING,PRING
 00073'000037          0*256.+37            ;ETLGA, ETLGN
 00074'032111          I2741*256.+I           ;TERMT, STATI
 00075'021442          43*256.+42           ;PORTO, PORTI
 00076'000401          1*256.+1            ;SPRTO, SPRTI
 00077'017000          30.*256.+0B8                   ;SUTAD, ETINO
 00100'003517          BL1*256.+PT1              ;BYTEL, PARTY
 00101'020071          2.-1*1B2+00.B7+71               ;ETDIO
 00102'060066          4.-1*1B2+00.B7+66    ;ETDOO
 00103'001107'         0066A                     ;ETDOA
                 E3:
 00104'000000          0                   ;ETR0
 00105'052131          "T*256.+"Y         ;ETYPE
 00106'000002          2.                  ;ETID
 00107'000000          0                   ;CHILD
 00110'000131'         E4                   ;LINK
 00111'000000          0                   ;PARNT
 00112'000207          RT1                              ;ETRAT
 00113'000000          0                    ;ETQBP
 00114'001101'         EOM5                  ;ETEOM
```

Figure 8. Portion of File EQUIP.RB, Assembled Equipment Table (Continued)

```
0011  EQUIP
 01020'000000           1.-1*1B2+0B7+0   ;ETOOD
 01021'000000           0                  ;ETDOA
               E12:
 01022'000000           0                  ;ETR0
 01023'042123           "O*256.+"S       ;ETYPE
 01024'000014           12.                 ;ETID
 01025'000000           0                  ;CHILD
 01026'000000           0                  ;LINK
 01027'000651'          E7                  ;PARNT
 01030'004540           2400.                              ;ETRAT
 01031'000000           0                   ;ETQBP
 01032'001051'          EOM1                ;ETEOM
 01033'000000           0                   ;ETRSP
 01034'000000           0                   ;ETPAD
 01035'000000           0                   ;RRING,PRING
 01036'000037           0*256.+37           ;ETLGA, ETLGN
 01037'032125           IBM2260*256.+U          ;TERMT, STATI
 01040'015031           32*256.+31          ;PORTO, PORTI
 01041'000000           0*256.+0          ;SPRTO, SPRTI
 01042'122400           245*256.+0B8                  ;SUTAD, ETIND
 01043'004116           BL2*256.+PT2          ;BYTEL, PARTY
 01044'000000           1.-1*1B2+0B7+0            ;ETOID
 01045'000000           1.-1*1B2+0B7+0   ;ETDOD
 01046'000000           0                  ;ETDOA
               E9999:
       000025 LEN       =E0ENO-E0000
 01047'000032 ETENT:    E9999-E0000/LEN
 01050'000025 ETLEN:    LEN
 01051'000037 EOM1:     37
 01052'177777           -1
 01053'177777           -1
 01054'177777           -1
 01055'177777           -1
 01056'177777           -1
 01057'000012 EOM2:     12
 01060'000005           5
 01061'000030           30
 01062'177777           -1
 01063'177777           -1
 01064'177777           -1
 01065'177777 EOM3:     -1
 01066'177777           -1
 01067'177777           -1
 01070'177777           -1
 01071'177777           -1
 01072'177777           -1
 01073'000037 EOM4:     37
 01074'177777           -1
 01075'177777           -1
 01076'177777           -1
 01077'177777           -1
 01100'177777           -1
 01101'000037 EOM5:     37
 01102'000043           43
 01103'177777           -1
 01104'177777           -1
 01105'177777           -1
 01106'177777           -1
 01107'000000 DO66A:    0
```

Figure 8. Portion of File EQUIP.RB, Assembled Equipment Table (Continued)

```
0012  EQUIP
 01110'000000 D0066B:    0
 01111'000000 D0067A:    0
 01112'000000 D0067B:    0
                         .END
```

Figure 8. Portion of File EQUIP.RB, Assembled Equipment Table (Concluded)

INTERFACE ADAPTER	ASYNCHRONOUS LINE ADAPTERS			
	Port		Subport	
	Output	Input	Output	Input
	24	24	1	1
	24	24	2	2
	24	24	3	3
	24	24	4	4
	24	24	5	5
	24	24	6	6
	24	24	7	7
	24	24	8.	8.
	24	24	9.	9.
	24	24	10.	10.
	24	24	11.	11.
	24	24	12.	12.
	24	24	13.	13.
	24	24	14.	14.
	24	24	15.	15.
	24	24	16.	16.

Figure 9. ET Entries for DCM Devices for Lab System

INTERFACE ADAPTER	ASYNCHRONOUS LINE ADAPTERS: Port Output	ASYNCHRONOUS LINE ADAPTERS: Port Input	ASYNCHRONOUS LINE ADAPTERS: Subport Output	ASYNCHRONOUS LINE ADAPTERS: Subport Input	DIGITAL I/O: Inputs (ETDID) First Input (BSSSS)	DIGITAL I/O: Inputs (ETDID) Device (DDDDDD)	DIGITAL I/O: Outputs (ETDOD) First Output (BSSSS)	DIGITAL I/O: Outputs (ETDOD) Device (DDDDDD)
	41	40	0	0	0	73	0	62
	41	40	1	1	2	73	4	62
	41	40	2	2	4	73	8.	62
	41	40	3	3	6	73	12.	62
	41	40	4	4	8.	73	16.	62
	41	40	5	5	10.	73	20.	62
	41	40	6	6	12.	73	24.	62
	41	40	7	7	14.	73	28.	62
	43	42	0	0	16.	73	0	63
	43	42	1	1	18.	73	4	63
	43	42	2	2	20.	73	8.	63
	43	42	3	3	22.	73	12.	63
	43	42	4	4	24.	73	16.	63
	43	42	5	5	26.	73	20.	63
	43	42	6	6	28.	73	24.	63
	43	42	7	7	30.	73	28.	63
	45	44	0-7	0-7	0-14.	74	0-28.	64
	47	46	0-7	0-7	16.-30.	74	0-28.	65
	51	50	0-7	0-7	0-14.	75	0-28.	66
	53	52	0-7	0-7	16.-30.	75	0-28.	67
	55	54	0-7	0-7	0-14.	76	0-28.	70
	57	56	0-7	0-7	16.-30.	76	0-28.	71

Figure 10. ET Entries for Asynchronous Devices for 64-Line Field-Test System

INTERFACE ADAPTER	ASYNCHRONOUS LINE ADAPTERS				DIGITAL I/O			
					Inputs (ETDID)		Outputs (ETDOD)	
	Port		Subport		First Input (BSSSS)	Device (DDDDDD)	First Output (BSSSS)	Device (DDDDDD)
	Output	Input	Output	Input				
	43	42	1	1	0	71	0	66
	43	42	2	2	2	71	4	66
	43	42	3	3	4	71	8.	66
	43	42	4	4	6	71	12.	66
	43	42	5	5	8.	71	16.	66
	43	42	6	6	10.	71	20.	66
	43	42	7	7	12.	71	24.	66
	43	42	8	8	14.	71	28.	66
	45	44	1	1	16.	71	0	67
	45	44	2	2	18.	71	4	67
	45	44	3	3	20.	71	8.	67
	45	44	4	4	22.	71	12.	67
	45	44	5	5	24.	71	16.	67
	45	44	6	6	26.	71	20.	67
	45	44	7	7	28.	71	24.	67
	45	44	8	8	30.	71	28.	67

Figure 11. ET Entries for Asynchronous Devices for 16-Line Field-Test System

(177777 octal). The lists are pointed to by ETEOM in each ET entry. If no EOM checking is to be done, ETEOM must point to a location containing -1. Figure 6 presently contains duplicate lists (EOM1 and EOM4). The lists are longer than needed so that additional EOM character codes can be added octally if needed. The 30 words in lines 196-225 are equivalent to the following seven words (except that the order of list EOM5 is changed):

```
EOM5:    43
EOM1:
EOM4:    37
EOM3:    -1
EOM2:    12
          5
         30
         -1
```

11. One word of storage must be provided for each group of 16 contiguous digital outputs which are to be used in the test, as shown in lines 226-229 as DO66A, DO66B, DO67A, and DO67B. The words are pointed to by ETDOA in each ET entry which uses digital outputs. The storage must be initialized to zero.

A number of conventions were observed in generating the file in Figure 6. The Macro Processor was used to perform certain substitutions and the NOVA assembler pseudo-op .DUSR (see lines 49-74) was used to perform others. The Macro Processor performs its substitutions prior to the assembly. The differences can be seen between the file EQ and the EQUIP (symbolic) portion of the assembly listing. The macro TTY33 defined at lines 93-95 of Figure 6 changes TTY33 in line 97, for instance, to I2741. The pseudo-op .DUSR causes the substitution to be made internally by the assembler. Therefore, the symbolic portion of the assembly listing gives the symbol and the assembled code shows the substituted value. For instance, on line 8 of page 1 of Figure 8, the name I is assigned the value 111_8. On line 56 of the same page, the I is shown in the symbolic code and the 111 is the rightmost portion of the assembled value of 13111_8.

The labels E3, E4, etc., (as well as EØ, E1, and E2) for each ET entry are needed to provide values for the cross-reference fields CHILD, LINK, and PARNT. A better tactic than using the arbitrary labels, however, would be to use the device names for labels, to use TY2 as a label rather than E3 at line 100 of Figure 6. The field ETYPE should be used to group like devices and to distinguish unlike devices, for instance: TT for TTY's, TY for IBM 2741's, CT for the control TTY, DS for displays, LN for communications lines, CN for multiplexor device-controllers, PT for printers, etc. Several combinations should be used to distinguish displays with different characteristics, for instance.

The label EØEND (line 88) is used to define the end of entry EØ and in defining ETEND (line 78). The label E9999 (line 192) is used to define the end of the last ET entry and in defining ETENT (line 194) and the length of ETREC (line 80). The symbol LEN (line 193) has the <u>value</u> of the length of an ET entry and is used in defining ETENT and ETLEN.

The equivalences for A through W at lines 49-54 are provided for use in giving values to the field STATI although only I and U should normally be used for initial values. The equivalences for W through O at lines 54-58 are for use in defining parity type (PARTY). The meanings are:

W = one (parity bit set to a constant 1)

E = even parity

Z = zero (parity bit set to a constant Ø)

N = no parity bit

O = odd parity

Only the values E and O are used by emulator programs.

The equivalences at lines 64-74 are used to define terminal type (TERMT). Those at lines 67-69 are of the earlier, arbitrary type which have not been updated.

The equivalences at lines 59-63 are used so that the fields ETRAT, BYTEL and PARTY in the ET entries may be given symbolic values rather than absolute values. Only the equivalence statement has to be changed to assign a new value rather than changing each ET entry.

FUNCTION

Each Equipment Table entry defines one equipment component of the SUT. In the simplest case, one ET entry is used to describe a point-to-point communications channel, possibly a pair of modems, and the single device attached to the channel. In a more complicated case, one entry describes the channel (and possibly modems), one is used to describe each controller or terminal (in a multipoint configuration), and one is used to describe each device at each terminal.

In the latter case, cross references (CHILD, LINK, and PARNT) are used to describe the hierarchical structure. As an example, the hierarchical ET structure described in Figures 6 through 8 is shown in Figure 12. Since each ET entry can reflect only one of each relationship, the arrows and labels indicate which relationship is expressed in the ET. Using this method of cross-referencing most configurations of equipment can be easily described. The number of levels and the number of entries at each level are limited only by core memory.

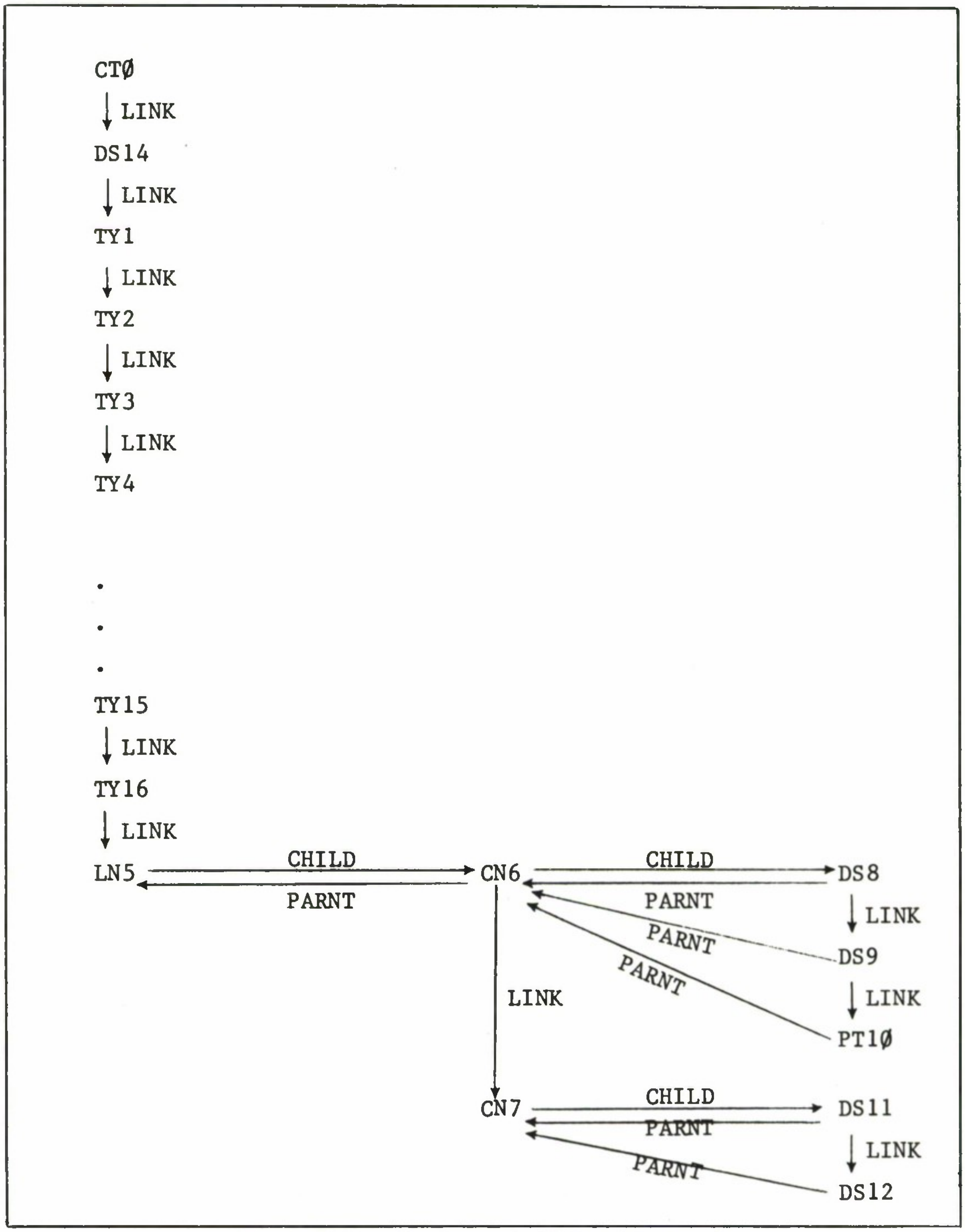


Figure 12. Equipment Table Hierarchy

Information in the Equipment Table is used by the Scenario Interpreter and by the Exec and is available to a scenario by means of certain scenario instruction types. The scenario may examine information, or in limited cases, change information in an ET entry. A scenario may access its own ET entry, or, through the relationships described above, access the ET entry of a relative, a relative's relative, and so on (in the direction of the arrows only). This capability of a scenario becomes increasingly useful as the equipment being emulated becomes increasingly complex.

The set of Registers of the current scenario associated with a particular device is pointed to by the first word (ETRØ) in the ET entry for that device. The first word of an ET entry is pointed to by the relationship pointers described above. Using instruction types h and then g and p (as defined in Volume 2 Table XVII), a scenario A running on device LN5 (as shown in Figure 12) can access the ET entry and Registers of scenario B running on device CN6, and then gain access to the ET entry and Registers of scenario C running on DS8, and so on. An example of this method of communication among devices is shown in the scenario segments in Figure 13.* In this case LN5 running with scenario A establishes the linkage to DS8 running with scenario C. Scenario A checks Register 9 of scenario C to determine when DS8 is ready to send a query. When scenario A senses that R9 = 1, it then performs a specified function (function 1) and resets R9 to zero. This zero indicator is put into R9 of scenario C, which senses the indicator and proceeds to send the query. Meanwhile CN6 running with scenario B is engaged in performing function 2, which may or may not be involved in communication with LN 5 or DS8.

---

* The scenario library SCENLIB, shown in Figure 35 in Appendix VI, establishes the macros used in this example.

Also, using instruction type h, and then instruction types Y or n, scenario A can examine the bit indicators (ETIND) of the ET entry of device CN6 and then DS8, etc. There are other scenario instructions which access the Equipment Table contents, and can be used in numerous ways to enhance scenario abilities and efficiency. A complete presentation of scenario instructions is given in Volume 2, Table XVII.

The technique of utilizing the Equipment Table to examine or pass information among devices can be useful, for example, when emulating a polled network. Assume, for instance, that CN6 was a controller and DS8 and DS9 were polled terminals. Then by making use of the cross references in the Equipment Table, the scenario for CN6 could poll the scenarios for DS8 and DS9 by examining indicator bytes or Registers to determine which devices were active, ready to send, or ready to receive. The individual terminal scenarios could send their queries and examine responses when indicated by the controller scenario.

LN5 SCENARIO A (SCA)	CN6 SCENARIO B (SCB)	DS8 SCENARIO C (SCC)
ALLOCREGS 15		
C[START CN6 SCB	C[START DS8 SCC	ALLOCREGS 15
ETOREG Ø Ø R1Ø	function 2	A 12
R1Ø CONTAINS ADDRESS	⋮	ALLOCATE 12 BYTE QUERY
TO ET ENTRY OF SCA		BUFFER
ETOREG R1Ø 3 R11		5BUILD QUERY
R11 CONTAINS CHILD		+ Ø 13 R11
POINTER (WORD 3) OF		PUT ASCII CR INTO R11
R1Ø WHICH IS ADDRESS		¬ R11
OF ET ENTRY OF CN6		ADD CONTENTS OF R11
ETOREG R11 3 R12		TO QUERY BUFFER
R12 CONTAINS CHILD		+ Ø 1 R9
POINTER OF R11 WHICH		R9 SET TO 1 INDICATES
IS ADDRESS OF ET		THAT QUERY IS READY
ENTRY OF DS8		L LAB1
L LAB1		B CONT Ø R9
GTR 9 R12 R9		IF R9=Ø THEN GO TO CONT
THE CONTENTS OF R9		D 1
OF SCC IS PUT INTO		J LAB1
R9 OF THIS SCENARIO		OTHERWISE DELAY 1 SEC
B CONT 1 R9		AND JUMP TO LAB1
IF R9=1 THEN GO TO CONT		L CONT
D 1		JUMP HERE WHEN R9 RESET
J LAB1		TO ZERO BY SCA
OTHERWISE, DELAY 1 SEC.		0
AND JUMP TO LAB1		SEND THE QUERY
		⋮

Figure 13. Example of Device Communication Through Scenarios

LN5 SCENARIO A (SCA)	CN6 SCENARIO B (SCB)	DS8 SCENARIO C (SCC)
L CONT function 1 ⋮ LDR Ø R9 R9 SET TO ZERO PTR R9 9 R12 PUT CONTENTS OF CURRENT R9 INTO R9 OF SET OF REGISTERS POINTED TO BY R12 (DS8) ⋮		

Figure 13. Example of Device Communication Through Scenarios (Concluded)

SECTION VI

REAL-TIME EMULATOR SYSTEM GENERATION

INTRODUCTION

The generation of the real-time emulator system is a four-step process which can be represented as follows:

$$\text{input files} \xrightarrow{\text{SSUB}} \text{source files} \xrightarrow{\text{ASM}} \left.\begin{array}{c}\text{.RB files}\\ \\ \text{EQUIP.RB}\\ \\ \text{Exec .RB files}\end{array}\right\} \xrightarrow{\text{RLDR}} \text{RTOS.SV} \xrightarrow{\text{MKABS}} \text{SCINT.BN}$$

The four steps are execution of the Macro Processor (SSUB), execution of the NOVA assembler (ASM), execution of the NOVA relocatable loader (RLDR), and execution of the DOS command MKABS. The first two steps must be performed separately for each assembly module which is to be changed (including the Equipment Table which is not considered a part of the Scenario Interpreter). The last two steps must be performed once each whenever one or more assembly modules (including those of the Exec) have been changed. In creating the Executive from the various source files, there is some flexibility available in defining buffer sizes, storage requirements, and parity checking on SUT terminals. These options are described in detail in Volume 6 of this series in the User Information Section.

SSUB

For purposes of this discussion the general form of the command to execute the Macro Processor is assumed to be:

SSUB   input-file   source-file   macro-libraries

The input-file names, source-file names, and the macro libraries needed for the Scenario Interpreter are given in Table VII. The implementation uses of the Macro Processor are also discussed in Section III.

To execute the Macro Processor, type on the control TTY;

SSUB II ININT RTOSLIB LIB LIB1↵

or

SSUB EQ EQUIP↵

where II and EQ are the input files; ININT and EQUIP are the output files; RTOSLIB, LIB and LIB1 are libraries; and ↵ represents the carriage-return key. The macro libraries must be in the form of the output files produced by the macro library generator (MACDEF), the file LIB.ML, for instance. Unlike the last three steps, the output file (ININT or EQUIP, above) must be absent from the DOS file directory before executing SSUB.

If one of the three macro libraries must be changed, it must be read into the NOVA using LIB, for instance, as the input file name. Typing

MACDEF LIB↵

on the control TTY will execute the macro library generator which will generate the macro library LIB.ML.

<u>ASM</u>

An output file from the Macro Processor (Source File) must next be processed by the Data General assembler by typing, for instance:

ASM/L/X $LPT/L ININT↵

The output file produced is a relocatable, binary file, ININT.RB in this case. Because the switches /L/X and the line printer $LPT are specified, an assembly listing including the source file and cross reference list will be produced on the line printer.

TABLE VII

Input File Names for Emulator System

Input File Name	Source File Name	Macro Library Names
EQ	EQUIP	--
SI	SCINT	RTOSLIB, LIB, LIB1
CI	CMINT	RTOSLIB, LIB, LIB1
II	ININT	RTOSLIB, LIB, LIB1
FC	FETCH	RTOSLIB, LIB, LIB1
TP	TESTP	RTOSLIB, LIB, LIB1
S1	SUBR1	RTOSLIB, LIB, LIB1
S2	SUBR2	RTOSLIB, LIB, LIB1
AF	ALF	RTOSLIB, LIB, LIB1
ERROR	ERMSG	RTOSLIB, LIB, LIB1
FTC	FTCHG	RTOSLIB, LIB, LIB1
DW	DUMPW	RTOSLIB, LIB, LIB1
DH	DUMPH	RTOSLIB, LIB, LIB1
IS	ISCEN	--

RLDR

Table VIII lists the assembly modules needed by the Data General relocatable loader to generate the real-time software for each of the two versions of the emulator. The files used by RLDR are those with the .RB suffixes. A list of the module names (excluding the suffix) must be given to RLDR. These can be typed from the list in Table VIII, if desired; however, the system tapes for each of the emulator versions contain a file called LOADLIST which is a list of the file names needed for each version. To execute RLDR, type on the control TTY:

RLDR/Z MAP/L @LOADLIST@↲

The output file produced by RLDR is in a form suitable for execution under control of DOS. Although the real-time emulator cannot be executed under DOS, the step is a necessary preliminary to producing the required file. The output file is named RTOS.SV since RTOS is the first file in the list in LOADLIST. Since MAP/L is specified the core map produced by RLDR will be placed in a DOS disk file called MAP. It can be listed by typing:

PRINT MAP↲ or PRINTL MAP↲

The MAP file should be saved on tape with the other files for future reference. The file RTOS.SV should also be saved since octal patches, if needed, can be made to it, with the MAP file for guidance. The fourth step must then be performed with a new or patched RTOS.SV.

MKABS

The DOS command MKABS produces a file which can be executed independently of DOS. The command is executed by typing:

MKABS/Z RTOS SCINT.BN INIT/S↲

The octal equivalent of INIT (obtained from the MAP file) is the

TABLE VIII

Inputs to Relocatable Loader

	Assembly Module Name	Lab System	Field-Test System
Exec	*RTOS	X	X
	*RTIN	X	X
	LPT	X	X
	MTA	X	X
	TTY1	X	
	DCM	X	
	DCMT	X	
	ASYNC		X
	SCMGT	X	X
	PAGE	X	X
	DSK	X	X
	DMP	X	X
ET	*EQUIP	X	X
Scenario Interpreter	SCINT	X	X
	CMINT	X	X
	ININT	X	X
	FETCH	X	X
	TESTP	X	X
	SUBR1	X	X
	SUBR2	X	X
	ALF	X	X
	ERMSG	X	X
	*FTCHG	X	X
	DUMPW	X	X
	DUMPH	X	X
	ISCEN	X	X

* Different versions needed

value to be used in the command. MKABS uses RTOS.SV as the input file and produces SCINT.BN as the output file. SCINT.BN is the real-time emulator program, containing the Exec, the Equipment Table, and the Scenario Interpreter. It may be executed, by means of the DOS program EXEC, by typing:

EXEC SCINT↵

A more convenient method of executing SCINT.BN, however, is discussed under Operating Instructions for the Scenario Interpreter.

Disk Requirements

After a system is generated, it is not necessary to maintain all the binary and source program files on disk. These files should be saved on tape, and disk space freed to allow space for additional macro libraries and scenarios. Table IX indicates the disk requirements of the files which should be retained on disk during emulator operation.

Table IX

Disk Requirements for Emulator System

File	Size Bytes/Pages	Comments
DOS,etc	101221/210	Includes basic support software after @REMAL@ has been executed. Includes SYS.DR, MAP.DR, EDIT.SV,XFER.SV,SYS.LB,RLDR.SV,OEDIT.SV PRINTL.SV,REMAL,BLDR.SV,EXEC.SV,ASM.SV
MACDEF.SV	14976/30	Macro Processor. See MTR 2677 Volume 3.
SSUB.SV	20736/41	Macro Processor. See MTR 2677 Volume 3.
SCENLIB.ML	242/1	Lower-case scenario instruction op-codes. See MTR 2677, Volume 2, Table XIV and related text.
CVT.SV	31488/62	Scenario Assembler. See MTR 2677, Volume 4.
SUTTAB	384/1	Scenario Assembler. See MTR 2677, Volume 4.
DEVTAB	1792/4	Scenario Assembler. See MTR 2677, Volume 4.
RTOS.SV	32512/64	Real-Time Emulator. See MTR 2677, Volumes 5 and 6.
SCINT.BN	33514/66	Real-Time Emulator. See MTR 2677, Volumes 5 and 6.
P	30/1	Real-Time Emulator. See MTR 2677, Volumes 5 and 6.
C	3/1	Real-Time Emulator. See MTR 2677, Volumes 5 and 6.
LOADLIST*	130/1	Real-Time Emulator. See MTR 2677, Volumes 5 and 6.
DATAR.SV	29056/57	Data Reduction Program. See MTR 2677, Volume 7.
SUMRY.SV	27904/55	Data Reduction Program. See MTR 2677, Volume 7.
TLIST.SV	27264/54	Data Reduction Program. See MTR 2677, Volume 7.
CTABS	1664/4	Data Reduction Program. See MTR 2677, Volume 7.
ERFILE	420/1	Data Reduction Program. See MTR 2677, Volume 7.
TREL.SV	26240/52	Data Reduction Program. See MTR 2677, Volume 7.
MASTR.SV	17024/34	Data Reduction Program. See MTR 2677, Volume 7.
MAP	3752/8	Core map of RTOS.SV and, thus, of SCINT.BN
NOTES	1926/4	Text description of system. Should be updated when changes made in public or private copy.
FILECH.BN	4806/10	Verifies file validity on disk. See Reference 3.
MTLIST.BN	3606/8	Physical tape dump for MT1. See Reference 4.
Total	380690/758	

* When used, also need EQUIP.RB and .RB files for Scenario Interpreter and Exec.

SECTION VII

REAL-TIME EMULATOR

INTRODUCTION

The Scenario Interpreter is the real-time, emulator application program which operates in conjunction with the Real-Time Exec, a multi-tasking, application-oriented executive program. The Scenario Interpreter executes commands used to exert gross control over the run, executes scenarios which describe the actions to be taken in emulating terminal and operator functions, and records real-time events on a log tape. The Scenario Interpreter and the Real-Time Exec perform all the functions of the real-time emulator run.

SYSTEM FLOW

As shown in Figure 14 the real-time emulator system as well as the internal scenarios to be used must reside on disk before a run can be initiated. The Scenario Interpreter program (running under the Real-Time Executive) is then started by input from the control teletype. Once the emulation has begun, the teletype may be used for both output messages and input commands for the run. The events of the emulation are recorded on the log tape during the run, and this tape is used at the completion of the run for analytical purposes. If any dumps of the emulator system are requested during the real-time run, they will be printed on the line printer during the run.

OPERATING INSTRUCTIONS

External control over a real-time emulator run is exerted primarily through the control TTY. The run is started under DOS conventions. Once started, emulator conventions apply. In existing Equipment Tables, the control TTY is defined as device CTO. Device CTO is made to look

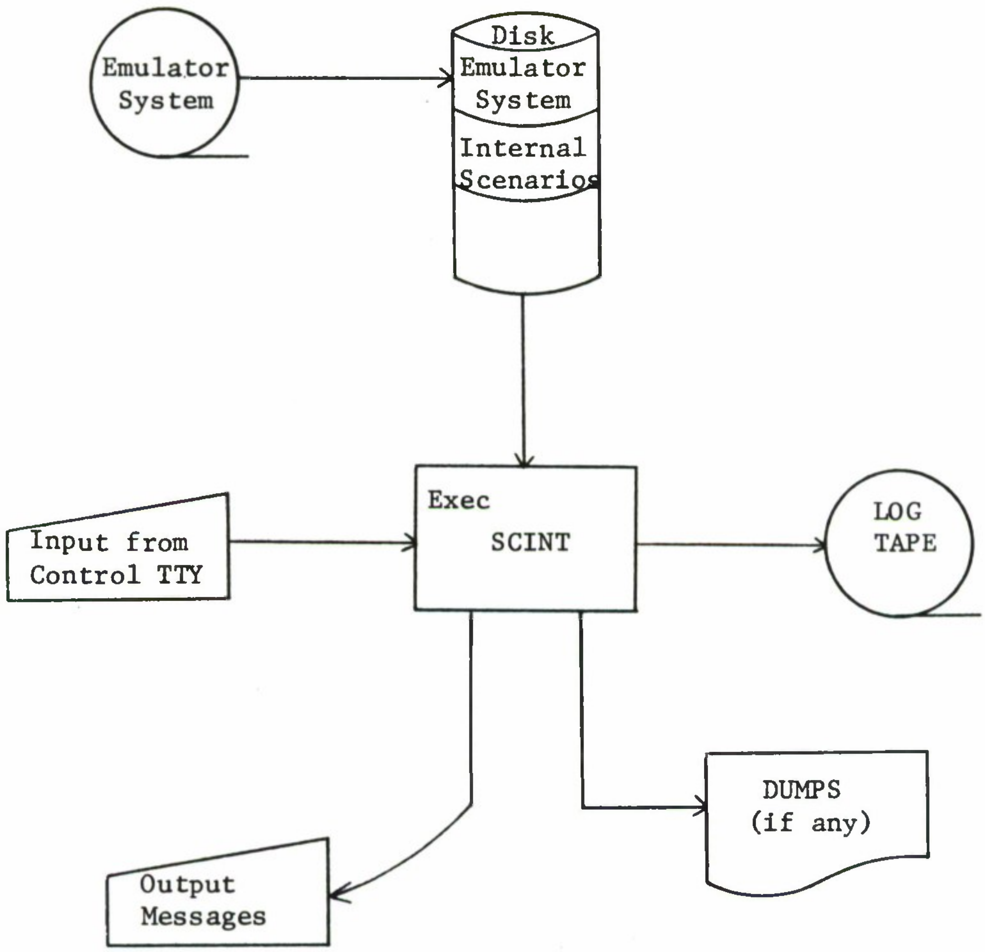


Figure 14. System Flow for Real-Time Emulator

as much like other (emulated) devices as possible. CTO can be used as an emulated device if desired although responses must be supplied by the user, of course. Unlike other devices, CTO is operated in echo-plex mode so that keystrokes will cause printing on the TTY. Unlike DOS, the Exec does not echo back a carriage-return and a line feed when the carriage return key is depressed. Therefore, the symbol ↲ is used to denote depression of the carriage-return key and echo back of both carriage-return and line-feed under DOS. Under Exec control, both keys must be depressed and they are represented below by CR/LF. It is assumed that the list of EOM characters pointed to by ETEOM in the ET entry for CTO includes LF (12_8), CANCEL (30_8), and BREAK (5).

Startup

If not already mounted, a scratch tape is needed on tape drive 0. The real-time run is most conveniently started by typing on the control TTY:

@P@↲

This input causes DOS to treat the file P as a list of DOS commands.

The file P contains:

```
RELEASE MTO;
TYPE C;
EXEC SCINT;
```

This set of commands causes the contents of file C (containing WAIT↲) to be typed on the control TTY. The log tape is then rewound if it was left other than at the load point by a previous real-time run or by an aborted Data Reduction run. While the tape is rewinding, the real-time emulator program (SCINT.BN) is called and initialization is begun. All further control TTY inputs must follow Exec conventions.

Control TTY Inputs

Run ID

After the word 'WAIT' is typed on CTO, the user must wait for the message 'ENTER RUN ID' to be typed on CTO before taking any further action. The user must then enter a character string, terminated by CR/LF, which will be written on the log tape as the run identification. The run identification consists of all characters typed up to but not including the first control character (those with octal values less than 40) or the first 20_{10} non-control characters. If an error is made in entering the run ID, simultaneous depression of the control and X keys (Control-X) will cancel the input and the user can start again. Almost immediately after entry of the run ID, the emulator will write the run ID and the other two history records on the log tape in one burst and then type "READY" on CTO. The emulator is now ready to accept commands so as to start emulation.

Commands

The emulator will remain in the idle state until a command is entered from CTO or from another emulator module or until an (unsolicited) response is received from the SUT, from CTO, or from another emulator module. Even then, the emulator will return to the idle state until one or more START commands are executed by the emulator. Commands are described in Volume 2 of this series. Commands from CTO (or another emulator module) must be preceded by an ASCII left-bracket character (Control-K). With a single START command, the user can execute a control scenario, if he desires, which can automatically START other devices and execute other commands (by means of the type-C scenario instruction) and any of the scenario instructions defined in Volume 2.

CANCEL Input

Any CTO input can be cancelled by depressing Control-X. The input will not be logged, and CR/LF will be typed as an acknowledgement.

BREAK Output

If the first (or any odd) character of a CTO input message is a BREAK character (input by depressing Control-E), the input is considered a BREAK input whose purpose is to BREAK or stop output on CTO of error messages (see ERROR command) and the monitor output of queries and responses (see MONITOR command). Error messages, queries, and responses already queued for typing, will be typed, but no more will be queued until another ERROR or MONITOR command causes them to be queued again.

Responses

A CTO input not in any of the above classes is considered a response. If no scenario is operating for CTO, they will be treated as unsolicited. If a scenario is operating and is waiting for a solicited response, the response will be processed immediately. Otherwise, the response will be queued until the scenario requests it or until the scenario terminates.

Shutdown

The real-time run is terminated by execution of a QUIT command from CTO, another emulator module, or a scenario. If the run does not terminate immediately, the emulator is so busy that the QUIT command (which is purposely given the lowest possible priority) is never executed because of a continuing string of higher priority tasks. One or more devices must be STOPped for the QUIT command to be executed. When the QUIT command is executed, two lines of emulator statistics are typed on CTO and the NOVA halts. By depressing Continue on the panel, DOS will be brought back in core and executed. DOS will type 'DOS REV XX.' and it will halt. Depressing Continue again will cause 'R' to be typed, and DOS is again in control. If desired, the Data Reduction program can be executed for the run just completed or any DOS function can be performed.

ERROR MESSAGES

The high-order digit of the printed error message number has been used to classify the error messages generated by the Scenario Interpreter as to seriousness. The most serious errors correspond to the highest digit. The ten error message classes are given in Table X. General comments are also included as to the kinds of errors associated with each class and the system action following detection of the error.

Table XI lists and explains all the error messages generated by the Scenario Interpreter. Each three-digit number shown is a part of the message. The message itself represents the only use in the Scenario Interpreter of the three-digit numbers. Elsewhere, error messages are referenced only by the two low-order digits, and the table is in order based on these digits. The convention (6)40 has been used to indicate the internal and external message numbers. The table gives the meaning and cause of each error message as well as the subroutines and modules which generate the message.

Table X

Error Message Classes for Scenario Interpreter

Class	Meaning
9	Not used. Reserved for severe errors which would abort real-time run.
8	System errors. Bring to attention of system programmer. Action terminated for device and device made inactive. (Same as if end of top-level scenario reached).
7	Relatively serious problem. May be system error or user error. Action terminated as for class 8.
6	Relatively serious user error, probably in a scenario. Action terminated as for class 8 unless able to proceed.
5	Error encountered in attempt to free a block of allocable core memory. Probably a system error although improper use of a type-F scenario instruction or previous improper action with Registers could cause it. System attempts to continue with emulation of device.
4	User error. Improper use of a command. Command not executed. Action continues as for class 5.
3	Unable to execute command. May be a problem of synchronization between devices. Action continues as for class 5.
2	Unable to execute command. Erroneous command operator or operand. Action continues as for class 5.
1	Usually an indication of an action taken although an error may be present also.
0	Not an error. Indication of action taken.

Table XI

Error Messages for Scenario Interpreter

Message	Meaning
800 STACK OVERFLOW	System error. Attempt to PUSH a value into stack portion of RS when stack full. (Subroutine POSHØ, POSH1, POSH2, or POSH3).
801 STACK UNDERFLOW	System error. Attempt to POP a value from stack portion of RS when stack empty (Subroutine PUPØ, PUP1, PUP2, or PUP3).
502 NO RS TO FREE	System error. Attempt to free RS when STACK=Ø. (Subroutine FRRS).
503 ILLEGAL FREE ADDRESS	Probably a system error. Attempt to free RS or buffer whose address not in allocable core. (Subroutine FRRS or FRBF).
504 NO BUFFER TO FREE	Probably a system error. Attempt to free a non-existent buffer, i.e., pointer = Ø (Subroutine FRBF).
406 TOO FEW REGS FOR SUBSCENARIO CALL	Register RGCAL not allocated in current set so that execution of a SUB command is ruled to be invalid. (Subroutines CMINT or ALRG).
507 NO REGS TO FREE	The set of Registers pointed to may have been freed previously or the contents of the Register may have been altered erroneously by a scenario. Otherwise, a system error. (Subroutine FRRG).
210 COMMAND NOT IMPLEMENTED	Specified command (MOD or TRANSFER) has not been implemented. (Subroutine CMINT).
211 INCORRECT COMMAND OPERATOR	Erroneous command operator. (Subroutine CMINT).
312 EQUIPMENT UNAVAILABLE	Attempt to START a device whose status is other than 'I' or 'S'. (Subroutine CMINT).

Table XI (Continued)

Error Messages for Scenario Interpreter

Message	Meaning
613 OUT-OF-RANGE REG #	Attempt to access Register not allocated in current set (module FETCH) or in another set (module ININT - type g or p scenario instruction).
114 DEVICE STOPPED	End of top-level scenario reached by normal operation or simulated due to serious error. (Module FETCH).
215 VALUE NEEDED FOR COMMAND	Numeric (decimal) value missing from SCALE command or numeric portion of equipment name missing from MONITOR, RESTART, START, STATUS, or STOP command. (Subroutine CMINT or FNENT).
216 UNKNOWN DEVICE NAME IN COMMAND	Unable to find equipment name specified in MONITOR, RESTART, START, STATUS, or STOP command in Equipment Table. (Subroutine FNENT).
217 INCORRECT SCENARIO NAME	Unable to find scenario name specified in START or SUB command in Scenario Directory (Subroutine CMINT).
020 ACTION TAKEN	Indicates successful execution of DUMP, ERROR, MONITOR, RESTART, SCALE, START, STOP, or SUB command (Subroutine CMINT).
121 SUB COMMAND LEGAL ONLY FROM SCENARIO	No rational way to execute a SUB command from one device for another since they operate asynchronously (Subroutine CMINT).
422 INVALID SUB-SCENARIO COMMAND REFERENCE	Attempt to execute a SUB command with no scenario specified when no uncompleted subscenario exists for device (RGCAL = Ø) or when Register RGCAL does not point to a valid set of Registers (C(RGRØ) ≠ RGRØ). (Subroutine CMINT).
223 ONLY "ON" OR "OFF" LEGAL	First operand of LOG command specifies 'ALL' and second operand specifies neither 'ON' nor 'OFF' (Subroutine CMINT).

Table XI (Continued)

Error Messages for Scenario Interpreter

Message	Meaning
224 ONLY "A", "N", OR "U" LEGAL	First operand of LOG command specifies 'THIS' OR equipment name and second operand specifies none of 'A', 'N', or 'U'. (Subroutine CMINT).
125 LOG ACTION COMPLETE	LOG command has processed as much as it can of the third operand. Each component of this operand is processed separately and program has reached illegal component or end of command. Rather than attempting in an iterative program to separate the cases of missing third operand, error in nth component but first n-1 of them were processed, or all components were correct, a combination message is used which is intended to cause the user to verify that there was no error in the third operand. Note that for this type of SUB command, the SUBSCENARIO form is invalid and no character (such as a blank) may follow 'SUB' in the command instruction or the program will assume a scenario is specified.
826 STATI INCORRECT	System error. Instruction Interpreter attempting to emulate device whose status (STATI) is neither 'A' nor 'T'. (Module FETCH.)
327 DEVICE INACTIVE OR STOPPED	Attempt to STOP a device whose status (STATI) is 'I', 'T', 'S', or 'U'. (Subroutine CMINT.)
330 DEVICE NOT STOPPED	Attempt to RESTART a device whose STATUS (STATI) is neither 'T' nor 'S'. (Subroutine CMINT.)
631 QUERY BUFFER OVERFILL	Attempt to fill query buffer beyond end by scenario instruction of type 5, \, or @. Note that if an error message intervenes after generation of query buffer,but before filling it, the error message buffer will displace the query buffer and the error message buffer will then be filled by the instruction. (Module ININT.)

Table XI (Continued)

Error Messages for Scenario Interpreter

Message	Meaning
732 NO QUERY BUFFER TO FILL	This message will only appear if there is no query buffer (or error message buffer) associated with the device and a scenario instruction of type 5, \, or @ is executed. This condition will only occur prior to generation of the first buffer or following execution of a type-E scenario instruction and before generation of next query buffer or of next error message buffer which is not the result of a type-E instruction.
333 DEVICE STOPPED BY TYPE-7 INSTR	A RESTART command is not legal for the device since it was STOPped by a type-7 scenario instruction rather than by a STOP command so that there is no current task which can be RESTARTed. See Miscellaneous Notes section. (Subroutine CMINT).
634 OTHER REG SET DOES NOT EXIST	Attempt to execute a scenario instruction of type g or p when the other set of Registers does not exist (pointer = Ø). (Module ININT).
035 TTY OUTPUT SUPPRESSED	A BREAK input was recognized and executed. (Module SCINT.)
336 ASSEMBLY ERROR IN SCEN	First byte of internal scenario is non-zero. Scenario needs to be reassembled after correction of errors before it will be acceptable for use with START or SUB command. (Subroutine CMINT).
337 EQUIPMENT TYPE MISMATCH	Scenario may not be used with specified device (START command) or with current device (SUB command) because scenario is not a universal scenario and the second byte of the internal scenario fails to match TERMT in the ET entry for the device. (Subroutine CMINT).

Table XI (Concluded)

Error Messages for Scenario Interpreter

Message	Meaning
640 BEHIND SCHEDULE	A type-W scenario instruction was executed after the specified time had passed. The amount of time by which the task is behind schedule, in milliseconds, is contained in the start transmission time fields of the buffer. Processing continues for device. (Module ININT.)
641 WAIT INSTR IGNORED	Type-W scenario instruction may not specify a time in excess of approximately 4.62 hours because of conversion problems. Instruction ignored and processing continues for the device. (Module ININT.)

DEVICE STATUS

Figure 15 shows all the valid state (STATI) transitions which can occur for a device. These transitions occur as the following functions are performed:

- I→A occurs when a START command is successfully executed for the device.
- A→I occurs when the end of the top-level scenario (RGRET = 0 for the current set of Registers) is reached for the device.
- A→T occurs when a STOP command is successfully executed for the device.
- A→W occurs when a time delay type of scenario instruction (type D, W, or d) is executed.
- A→S occurs for the current device when a type-7 scenario instruction transfers control of the task to another device.
- W→A occurs upon the expiration of a time delay caused by execution of a scenario instruction of type D, W, or d.
- W→T occurs when a device is STOPped while executing a scenario instruction of type D, W, or d.
- S→A occurs when a STOPped device is STARTed or RESTARTed after the transition from T to S has taken place or for the new device during execution of a type-7 scenario instruction.
- T→S occurs for a STOPped device after completion of execution of the current scenario instruction or upon receipt of a response following execution of a scenario instruction of type R or I
- T→A occurs when a RESTART command is executed for a STOPped device before the T to S transition has taken place.

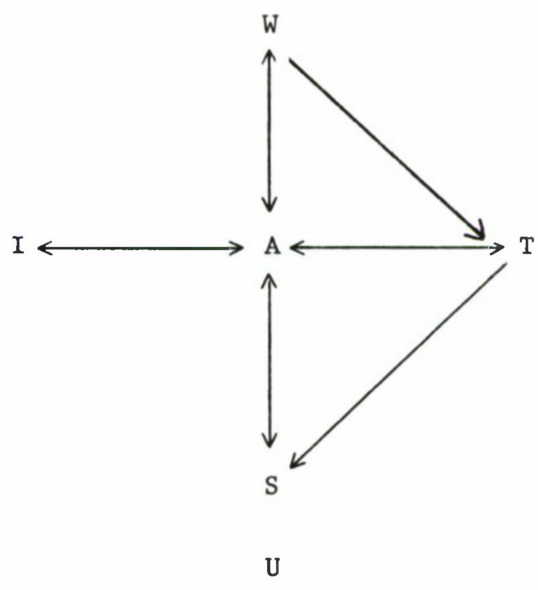


Figure 15. State Transition Diagram

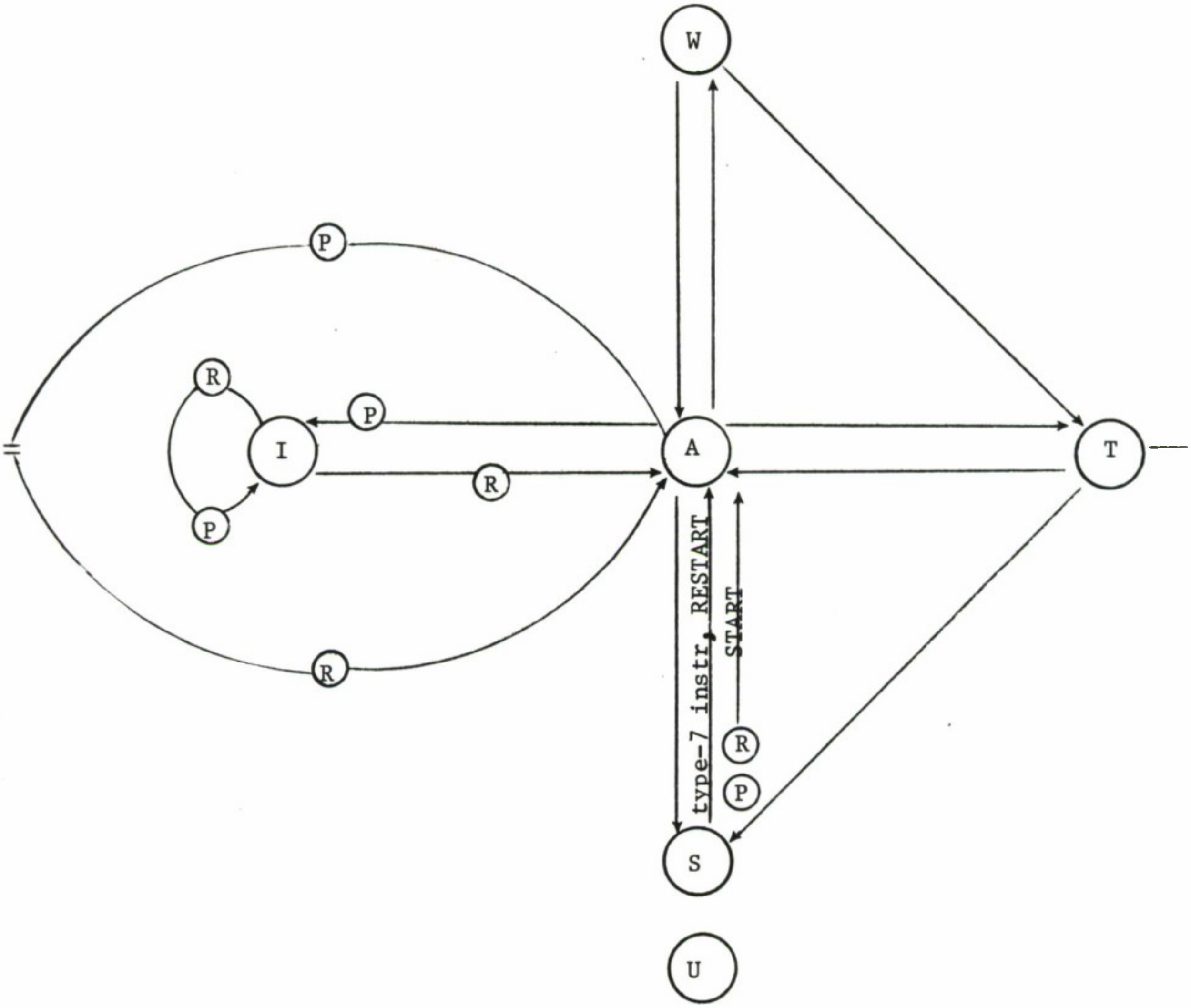
U device is unavailable and status cannot be changed by the emulator (can only be changed in non-real-time by reassembly of the ET or with the octal editor).

RING COUNTERS

There is a pair of ring counters in each ET entry (for each emulated device). They are used to sequence number tasks of types 6 (unsolicited responses), 7 (solicited responses), and 8a (newly STARTed devices), so that only one task of these types at a time (per device) can proceed past a certain point in the Input Processor (types 6 and 7) or the Instruction Interpreter (type 8a) so as to preserve reentrancy. The ring counter RRING (the response ring counter) is used to count and sequence number such tasks. The subroutine CHEKR is used to maintain RRING. CHEKR fetches RRING and uses it to sequence number the task (by setting RSEQU), steps RRING, and stores the updated value. CHEKR then compares RSEQU with PRING (the processing ring counter). If they are equal, the task is allowed to proceed. Otherwise, the task remains in CHEKR until PRING equals RSEQU. Thus, a queue of such tasks is maintained for each device, when necessary, and the tasks are released one at a time in the order in which they reached CHEKR.

The processing ring counter (PRING) is maintained by the subroutine STEPR. STEPR is called when task types 6 (unsolicited responses), 7b (type R or I scenario instruction executed), or 7c (end top-level scenario) terminate and when certain tasks of type 8a are generated (when a STOPped device is STARTed, the STOPped task must first be terminated). The only function performed by STEPR is to step PRING so that the next sequence numbered task may proceed.

These steps are shown in Figure 16 which is a modification of the state diagram in Figure 15. In Figure 16, when a device is STARTed, its status (STATI) changes from I to A. RRING is also



R RRING used and stepped by CHEKR

P PRING stepped by STEPR

Figure 16. Ring Counter Changes

stepped and the new task may be queued. When the end of a top-level scenario is reached for a device, its status changes from A to I and PRING is stepped.

If a STOPped device is STARTed (not RESTARTed), its status changes from S to A. When the STOPped task is terminated, PRING is stepped for the old task. RRING is then stepped for the new task (which may be queued.)

The loop around the I status indicates no change in status but the fact that if the device is inactive, receipt of an unsolicited response first causes RRING to be stepped and then PRING. Unsolicited responses are queued since a change in device status while the response is queued may cause a change in the type of response. The final determination as to the type of response is made when the response leaves the queue.

Similarly the loop around the A status indicates no change in status but the execution of a scenario instruction of type R or I which causes PRING to be stepped followed by a new task which steps RRING. Had one or more responses already been queued for the device, the stepping of PRING would allow the first of these to advance.

The discussion also indicates possible problems regarding use of the type-7 scenario instruction. For a type-7 instruction to be valid, the device to which control of the task is transferred must be STOPped. Thus, for this new device there already exists a suspended Scenario Interpreter task. If a task which has been generated for one device is allowed to terminate for a second device, PRING will not get stepped at the end of the task for the old device but for the new device. Thus, since the ring counters provide for 256_{10} sequence numbers, the old device would have to accumulate a total of 255 queued responses (which would tie up 255 Exec clock blocks) before any further activity could occur for the old device. The new device should be able to resume activity when a new task is generated for it, but the original STOPped task would be destroyed without its allocable core being freed when the task which executed the type-7 instruction terminated. The first problem is the more serious one, of course, but the latter ties up system resources for the duration of the run. Therefore, a task which is started for one device should be terminated for the same device to avoid these problems.

RESPONSE HANDLING AND LOGGING

The determination of whether logging is enabled or not for a particular device and a particular buffer type is made at the time the buffer is allocated. Changing the setting of the logging indicators, with the LOG command, has no affect on logging of buffers which have already been allocated. In the present implementation, if logging is enabled in a given case, a long buffer (one with a long header) is allocated and all long buffers are logged. For all long buffers, the log processing bit in BFIND is set at time of allocation. For either long or short buffers, one of the other five processing bits is set (based on buffer type) at time of allocation. When a task is done with a buffer or when it needs the buffer pointer space in the RS for a new buffer to be allocated, it resets the appropriate processing bit and attempts to free the buffer. If all six processing bits are reset, the free attempt is successful.

Unlike the other four types of buffers, response buffers are not automatically logged in all cases. Every long response buffer must be logged by one means or another or it will not be freed and the space will not be available for reallocation during the rest of the run. A separate response queue is maintained for each emulated device so that only one main task can be active at a time to process a single response. When a response and its associated task leave the queue, the determination is made as to whether the response is solicited (or unsolicited) depending essentially on whether the device is active (or inactive). If the device is inactive when the response leaves the queue, the response will be logged automatically as unsolicited, if logging is enabled, and the task is terminated.

If the device is active at the time the response leaves the queue, the response will be logged automatically as solicited if Response Indicator 2 in ETIND is set and logging is enabled. The indicator must be set by executing a scenario instruction of the form = 2 prior to the time the response leaves the queue. A long response buffer can also be logged by executing a type-8 scenario instruction, which specifies whether the response is solicited or unsolicited. The use of both techniques will cause the buffer to be logged two or more times, once automatically and once for each type-8 instruction executed. Since there is no apparent advantage in logging a response more than once and the solicited response indicator (bit 0 in BFIND) is initially reset, execution of a type-8 instruction to log a response as unsolicited does not reset the solicited response indicator. Therefore, once the indicator has been set by either means, any further type-8 instructions will cause logging as solicited regardless of the value of the first operand.

When a device is active, all responses received will be queued until one is requested by the scenario by means of executing a scenario instruction of type R or I. When such an instruction is executed, the main task for the device is terminated at the end of execution of that instruction. Further execution of the scenario is done by the task associated with the next queued response which starts execution with the scenario instruction following the R or I instruction. If any responses are queued for a device when the end of the top-level scenario is reached or when a STOPped device is STARTed (<u>not</u> RESTARTed), the responses will be logged automatically as unsolicited. In addition, when either event occurs, all indicators in ETIND are reset except for the Command Indicator and the Monitor Indicator. Therefore, if responses are to be logged automatically as solicited, each scenario STARTed (not RESTARTed or executed by a SUB command) must set Response Indicator 2.

DIGITAL I/O

Digital I/O devices are installed on the field-test system but not on the lab system. With the field-test system connected directly to a SUT (without use of modems), the emulator must emulate the actions of modems as well as devices and operators. For each device, the SUT must believe it is communicating with the modem at its end of a communications channel. To provide more direct and complete control over the modem control lines (those not used for data transfer) than that provided by most line adapters, the emulator uses digital input devices to read the control signals set and reset by the SUT and digital output devices to set and reset the control signals read by the SUT.

The field-test system to be discussed is that containing 16 asynchronous communications channels and 8 synchronous channels. The discussion is largely concerned with emulation of asynchronous devices, with comments as to the extensions for synchronous devices.

The digital I/O design was done by Data General. The intended software design had to be modified to interface with the hardware as delivered.

A digital output device contains the capability of setting 32_{10} digital outputs. Since a single NOVA instruction can set only 16_{10} outputs, the outputs associated with one device address are separated into A and B groups. Since the outputs must be continuous rather than momentary, a register is associated with each of the two groups of an output device. Thus a NOVA digital output instruction loads either the A or the B register and the SUT reads (senses) the bits in those registers. Loading a register corresponds to the simultaneous setting of some outputs to 1 and resetting of others to 0. Since Data General provided no means of reading an output register, the emulator software has to maintain a record of the status of each set of 16 outputs, in the word pointed to by ETDOA. (Each such word contains the current settings of outputs associated with 2 to 16 emulated devices, as should be clear later.) When one or more digital outputs must be set or reset for an emulated device, the software has to fetch the word pointed to by ETDOA and either reset the appropriate bits by masking or set them by ORing. The updated word then has to be stored back in memory and loaded into the appropriate register.

The system contains four digital output devices with (octal) addresses of 64, 65, 66, and 67. The outputs for a single digital output device are numbered from 0 to 31 decimal (0 to 15 in the A register, 16 to 31 in the B register). The system contains 128 digital outputs. Devices 64 and 65 are reserved for synchronous emulation, and 66 and 67 are used for asynchronous emulation.

The digital input hardware is similar to that for digital output but simpler. A digital input device allows reading (sensing) 32_{10} inputs. The inputs are grouped in A and B groups although a group is simply a group of lines in the emulator hardware since, in this case, the inputs read are in registers in the SUT. When one or more digital inputs must be read and tested for an emulated device, the appropriate digital input device and group (containing inputs associated with 2 to 16 emulated devices) must be read, and the appropriate inputs tested.

The system contains two digital input devices with (octal) addresses of 70 and 71. The inputs for a single digital input device are numbered from 0 to 31 decimal (0 to 15 in the A group, 16 to 31 in the B group). The system contains 64 digital inputs. Device 70 is reserved for synchronous emulation, and 71 is used for asynchronous emulation.

Figure 17 shows the types of connections between the NOVA rack and the SUT, by way of the interface rack. On the left are the connection points, and on the right is shown the type of path connecting each pair of adjacent points. The jumpers between the A and B barrier strips are intended to be the primary means of changing configurations. For asynchronous devices, there are 16 A barrier strips and 16 B strips, one of each per device. Up to 10 separate connections can be made from an A barrier strip to 10 or less of the 24 connection points on a B barrier strip.

The relationships within the interface rack should be clarified by Figure 18. A single cable carries all 32 inputs or outputs (both A and B groups) of a single digital I/O device between the NOVA rack and the interface rack. A single section of the interface rack accommodates 16 emulated asynchronous devices. The normal wiring needed for emulating Bell 103A modems is shown in the figure. Only the digital I/O wiring is shown. For each emulated device, two digital inputs and four digital outputs are shown although only one of the inputs is used. Digital input device 71 is adequate for the needs of all 16 emulated devices. Digital output devices 66 and 67 are needed to provide four outputs per emulated device. In the diagram, the outputs are labeled from 0 through 3 and the inputs from 0 through 1. These are the addresses to be used by scenarios.

The purpose of the ETDID and ETDOD fields in an ET entry is to describe the relationship between the fixed digital I/O addresses used by scenarios (the same for all emulated devices) and the hardware addresses which are different for each emulated device. ETDID and ETDOD as well as the four types of digital I/O scenario instructions allow up to eight digital inputs and eight digital outputs to be associated with each emulated device. Since only one NOVA instruction is used to read digital inputs or to set and reset digital outputs and to conserve space in the ET, all the inputs (or outputs) for an

	CONNECTION POINTS	PATH TYPES
	device, A or B	
NOVA		hardwired
	NOVA connector	
		cable
	IR back connector	
		one jumpered end
	A Barrier Strip	
IR		jumper
	B Barrier Strip	
		one jumpered end
	IR front connector	
		cable
SUT	SUT modem connector	

IR = interface rack

Figure 17. Digital I/O Connections

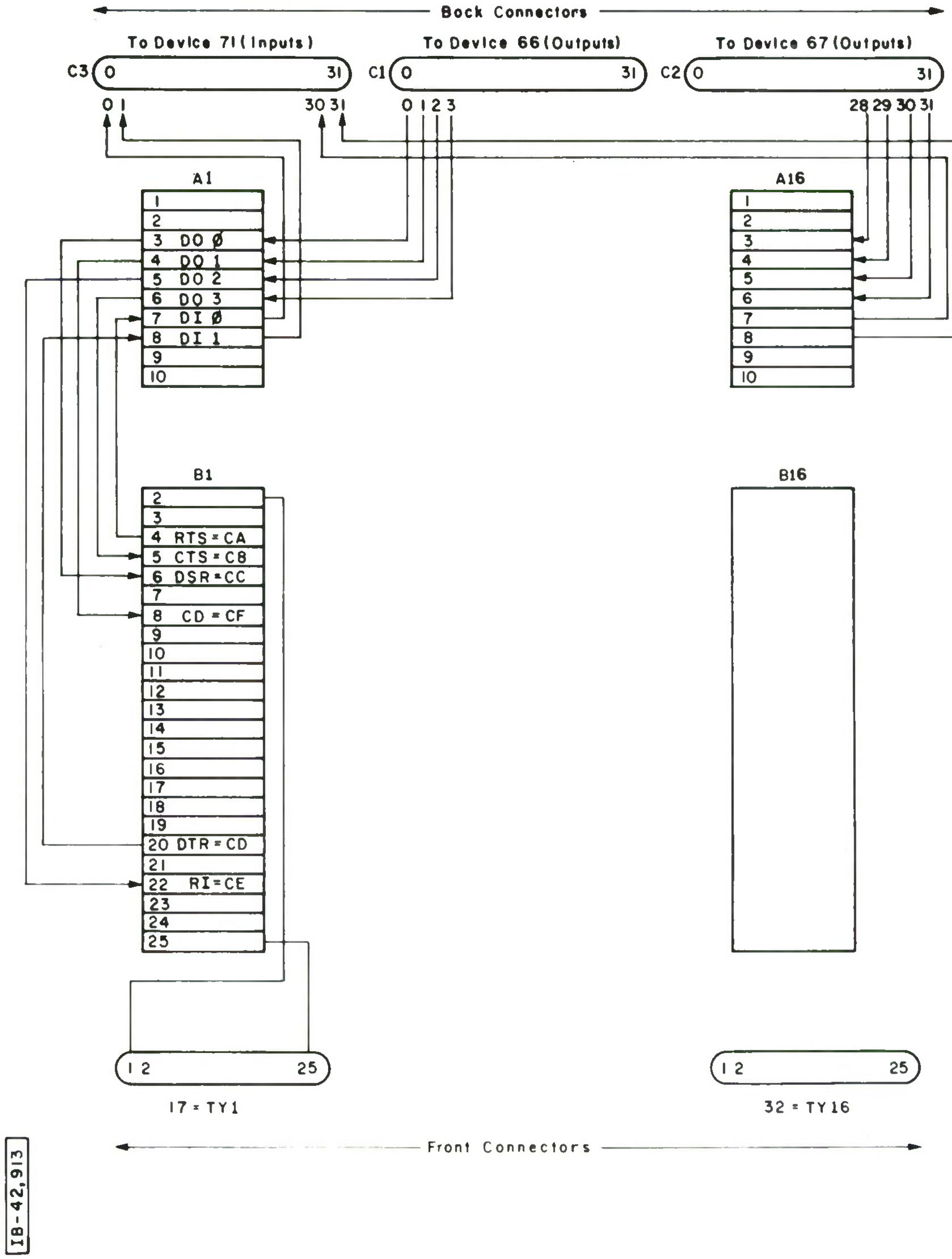


Figure 18 NORMAL INTERFACE RACK WIRING FOR ASYNCHRONOUS DEVICES

emulated device must have the same digital I/O device address, be in the same group (A or B), and be adjacent to one another.

ETDID and ETDOD have the same format (CCCBSSSSOODDDDDD in binary) and specify the digital I/O device address (DDDDDD), the number of the left-most input or output (BSSSS, where the value of the high-order (B) bit separates the A group from the B group), and the number of consecutive inputs or outputs minus one (CCC). If ETDID (or ETDOD) is zero, there are no inputs (or outputs) associated with the emulated device. From Figure 18 it can be seen that:

for device TY1:

ETDID: CCC = 1, BSSSS = 0, DDDDDD = 71

ETDOD: CCC = 3, BSSSS = 0, DDDDDD = 66

for device TY2:

ETDID: CCC = 1, BSSSS = 2, DDDDDD = 71

ETDOD: CCC = 3, BSSSS = 4, DDDDDD = 66

for device TY16:

ETDID: CCC = 1, BSSSS = 30., DDDDDD = 71

ETDOD: CCC = 3, BSSSS = 28., DDDDDD = 67

where a decimal point following a number indicates a decimal number, otherwise octal.

In Figure 18, the six digital input and output connections on an A barrier strip are connected to six points on a B barrier strip which in turn are connected to six pins on a front connector which is cabled to the SUT. These correspondences are shown in Figure 19. The codes are standard pin or signal codes. Figure 20 contains synonyms for the five scenario instruction op-codes used for digital I/O as well as correspondences between the digital I/O addresses used by a scenario and the two-letter signal codes. These equivalences can be made by use of the Macro Processor.

Scenario I/O Address	Pin Number	Code	Function
DO-0	6	CC	Data Set Ready (DSR)
DO-1	8	CF	Carrier Detect (CD)
DO-2	22	CE	Ring Indicator (RI)
DO-3	5	CB	Clear to Send (CTS)
DI-0	4	CA	Request to Send (RTS)
DI-1	20	CD	Data Terminal Ready (DTR)

Figure 19. Normal Asynchronous Correspondence

```
DON = ;
DOF = :
BDN = 9
BDF = q
ADY = d
CC  = 0
CF  = 1
CE  = 2
CB  = 3
CA  = 0
CD  = 1
```

Figure 20. Macro Definitions for Digital I/O

```
DON CE
L CDLOOP
BDN CDON CD
ADY 250
J CDLOOP
L CDON
ADY 500
DON CC
DOF CE
ADY 4500
DON CB CF
```

Figure 21. HANDSHAKE Scenario

Figure 21 contains the HANDSHAKE scenario which causes the emulator to exchange the modem control signals necessary prior to data transmission. The scenario first turns on (sets) the Ring Indicator (CE). At the label CDLOOP, a branch is taken to the label CDON if Data Terminal Ready (CD) is on. Otherwise, a 250-ms delay is taken followed by a branch to CDLOOP to test CD again. When CD has been turned on by the SUT (at CDON), a 500-ms delay is taken, Data Set Ready (CC) is turned on, and Ring Indicator is turned back off. A 4½-second delay is then taken and Clear to Send (CB) and Carrier Detect (CF) are both turned on.

In Figure 18, connection points 1, 2, 9, and 10 are not used for digital I/O. Points 1 and 2 are received and transmitted data, and 9 and 10 are for clock signals for synchronous emulation. If more than two digital inputs or four digital outputs are needed for an emulated device or if secondary data transmission paths are needed, two A barrier strips must be connected to the same B barrier strip. This technique is necessary for synchronous emulation. From the standpoint of digital I/O, two adjacent A barrier strips will have to be used so that the digital inputs and digital outputs for the emulated device form consecutive sets. ETDID can then be changed to describe up to four inputs, and ETDOD can be changed to describe up to eight outputs.

STORAGE REQUIREMENTS

The core storage requirements for both the Scenario Interpreter and the Real-Time Exec are presented in Tables XII and XIII respectively. The data for the Real-Time Exec are based on the 64-line field test system, while the information for the Scenario Interpreter applies to both lab and field test systems.

Table XII

Core Storage Requirements for Scenario Interpreter

Assembly Module	Program, Words	Major Tables, Words	Total, Words
SCINT	417	-	417
CMINT	668	-	668
ININT	996	64	1060
FETCH	464	64	528
TESTP	162	-	162
SUBR1	310	-	310
SUBR2	292	-	292
ALF	317	45	362
ERMSG	195	491	686
FTCHG	192*	-	192*
DUMPW	171	-	171
DUMPH	185	-	185
ISCEN	-	7	7
	4369	671	5040

* For field test system

Table XIII

Core Storage Requirements for Real-Time Exec

Name		Words
RTOS		2686
RTIN		672
MTA		758
LPT		98
SCMGT		442
PAGE		385
DSK		64
DMP		164
ASYNC		2916
	TOTAL	8185

MISCELLANEOUS NOTES

(1) Assume devices A and B are both STARTed and then device B is STOPped by a STOP command. Further assume that the scenario for device A executes a type-7 scenario instruction to transfer control to device B at time T and that the scenario for device B transfers control back to A at time T'. An attempt to RESTART device A between times T and T' is not legal since device A has no task associated with it (its original task is associated with device B) even though its status (STATI) is 'S'. Error message #33 is generated in this case. Device B may not be RESTARTed during the interval since its status is not 'S', although it may be RESTARTed after the STOP command and prior to T, and after T'.

(2) The Scenario Directory is ordered the same as the DOS file directory. (LIST/L *.IS) will produce a list on the printer of internal scenarios and their order in the DOS file directory.) By design, the Scenario Interpreter will find the first entry in the Scenario Directory whose n-character name matches the first n-characters of a scenario name in a command. Thus, if TEST precedes TESTA in the directory, a command specifying TESTA will find TEST in the directory. Similarily, M can prevent access to M1, MATCH, etc. Implementation was done in this manner since there is no guarantee as to which of many characters may follow the last character of a scenario name. In particular, a user may declare any ASCII character as an EOM character, which would follow a scenario name.

To avoid problems of selection of an unintended scenario because of such subset names, various techniques are available. No subsetting will occur if all scenario names contain the same number of characters. In particular, if all scenario names are ten characters or more in length, no problems will occur because the DOS file directory contains only the first ten characters of a file name. Another solution is to end each scenario name with a character which is used nowhere else in a scenario name (the ASCII $ sign appears a likely candidate). If subset names occur, they will cause no problems if the longer names precede the shorter ones in the DOS file directory. The final solution, of course, is not to form scenario names by appending one or more characters to previous scenario names.

(3) Commands entered at the control TTY must be preceded by a left bracket (control-K):

[START DS14 Y

Command instructions punched in cards should be in the form:

C¢START DS14 Y

The cents sign is the keypunch equivalent of the left bracket. (In the case of the scenario instruction, the cents sign is not needed for identification, but the first character in the literal is skipped over.)

(4) Partial core dumps on the printer will result from:

a. use of the DUMP command

b. use of the Structure Dump (?) instruction

The dump routines used to implement these functions are not reentrant since interleaved usage by several tasks of the same printer seems unuseful. The continuity of the dump is necessary to identify the device (and scenario) causing it. The dump functions are for diagnostic purposes and should be used with care to avoid reentrancy violations.

PANIC CODES AND ACTIONS

If during the normal operation of the emulator, certain abnormal conditions occur, the Real-Time Exec will abort the run. Before aborting the run, however, the system saves the contents of accumulators AC0-AC3 in locations 12, 13, 14, and 15, respectively, disables interrupts, prints out a panic code on the control teletype, and halts. The panic codes are described in Table XIV.

The user can obtain a full core dump of the system at this point by depressing the "CONTINUE" switch on the NOVA console. If only a partial dump is desired, the word count and starting address of the desired area can be entered into accumulators 0 and 1, respectively, before depressing the "CONTINUE" switch. When the dump is completed, the system will automatically try to write the magnetic tape buffers to tape, write an end-of-file on the tape and then try to make a normal emulator exit, printing out the run statistics. An example of a panic message and termination is given in Figure 22.

The run statistics that are printed on the control teletype at the end of an emulator run are: the maximum number of task control blocks that were in use at any one time (TCB MAX XXX), the maximum number of tasks that existed on the task pending queue at any one time, the number of available core blocks that exist at exit time, and the total number of core words available at exit.

Table XIV

RTOS Panic Codes

Error Code	Meaning
1	System error. Two tasks are illegally trying to remove core space from the free chain at the same time.
2	System error. Two tasks are illegally trying to return core space to the free chain at the same time.
3	System error. A task issuing a .FREE supervisor call has illegally given a block size of zero length. Usually means the core chain or Scenario Interpreter data structures are in error.
5	System error. A task issuing a .FREE supervisor call has illegally tried to free a block with a starting address the same as a block already in the free chain. Usually means Scenario Interpreter data structures are in error.
6	System error. A task issuing a .FREE supervisor call has illegally tried to free a block which overlaps the front part of a block already in the free chain. Usually means core chain or Scenario Interpreter data structures are in error.
7	System error. A task issuing a .FREE supervisor call has illegally tried to free a block which overlaps the end part of a block already in the free chain. Usually means core chain or Scenario Interpreter data structures are in error.
8	System error. A task exiting from either a .ALOC or .FREE supervisor call has found the core chain busy indicator illegally set.
9	System error. A task exiting from either a .ALOC or .FREE supervisor call has found that the link word of its TCB is illegally set. Usually means that the queue stack is in error.
10	System error. A task issuing a .FORK supervisor call has illegally given a value of zero for the new task's stack address. Usually Scenario Interpreter error.
11	System error. A task issuing any supervisor call other than .ALOC or .FREE has a zero value for its stack address. Usually Scenario Interpreter error.

Table XIV (Continued)

RTOS Panic Codes

Error Code	Meaning
12	System error. The number of clock blocks reserved at system generation have been used up by tasks issuing .WAIT supervisor calls. User is either trying to emulate too many lines with space for clock blocks or is running in loopback mode at a high baud rate.
13	Hardware error. An undefined device has caused an interrupt. Location 14 (accumulator 2) contains the device number of the offending device.
14	System error. A response having an odd number of characters has been terminated without padding out the right byte of the last word. Usually indicates response handling logic is in error when adding a new device to system.
15	System error. The word count in a query buffer is greater than 32,768, which is outside the address space of the NOVA 800. Usually means the Scenario Interpreter data structures are in error.
17	System error. The interrupt dismissal routine was called with an illegal interrupt data block address. Usually means an executive error.
18	System error. The interrupt data block address was equal to zero for a device that was trying to perform an end of operation at the non-interrupt level because the queue for the device was not available at time of interrupt.
19	System error. The initial word count for the text portion of a query buffer is equal to zero. Usually means scenario is in error or Scenario Interpreter data structures are in error.
20	System error. Lab system only. On exiting from the DCM handler the bit time indicator had been reset illegally. This panic condition was part of original Data General software.
21	System error. Lab system only. The system was unable to service all DCM lines in 5 bit times. Usually means core chain became too long. Part of original Data General software.
25	Hardware error. The magnetic tape controller indicated an error when a status instruction was executed upon a

Table XIV (Concluded)

RTOS Panic Codes

Error Code	Meaning
	tape interrupt. Location 12 (accumulator Ø) contains the status of the tape drive. The explanation of the status is given in Reference 5.
26	System error. The magnetic tape handler received a non-error interrupt and did not have a record of having written a tape buffer. Usually means the tape device unit control block has been destroyed.
27	Hardware error. In reading the magnetic tape status before writing, either bit 1, 2, 3, or 5 has been set indicating some type of tape unit trouble. From experience panic code 25 usually occurs before this condition.
28	System error. A task issuing a .FTCH supervisor call has passed a scenario program counter which is larger than the scenario itself. Usually means that the internal scenario on disk has been destroyed or the scenario management routine has an error.
29	Hardware error. The disk controller indicated an error when a status instruction was executed upon a disk interrupt. Location 12 (accumulator Ø) contains the status of the disk controller. The explanation of the disk status is given in Reference 5.

Note: The above panic conditions were inserted during the debugging and development phase of the emulator software. From experience the only ones that a user may usually encounter are 12, 13, 21, and 25. Any of the others occurring usually means a new problem uncovered and should be reported to the system programmers.

```
@P@
WAIT
ENTER RUN ID

1
READY


PANIC: ERRØR CØDE=21
HIT CØNTINUE FØR FULL CØRE DUMP

TCB MAX 000003  TPQ MAX 000003
CØRE LINKS  000002  CØRE AVAIL  027363
DØS REV 05.

R
```

Figure 22. Example of Panic Message

SECTION VIII

DATA REDUCTION PROGRAM

INTRODUCTION

The Data Reduction program (DATAR) processes log tape data gathered during an emulator test run. The program produces scenario trace data and various statistics on the performance and utilization of both the emulator and the SUT. A complete description of the design and implementation of the program can be found in Volume 7 of this series. DATAR runs under Data General Corporation's standard Disk Operating System (DOS), Revision 5.

DATAR may be used to produce several kinds of summary and detailed listings from the log tape, and thus it allows the user to obtain a quick summary of activity during the run on an individual basis or as an entire system. DATAR also gives detailed information in the form of record-by-record listings that include information such as readable real-time clock (RRTC) times, various timing calculations, and the text message.

After the tape file is processed by DATAR, the user may save the test data on master log tapes (to consolidate tapes or to put similar runs on one tape). The master (or original) log tape may be used for later analysis on the NOVA 800 or on a larger machine with more sophisticated data reduction and analysis capabilities.

SYSTEM FLOW

Figure 23 depicts the system flow of DATAR programs. The log tape, with data gathered from a single emulation run or a series of runs, is mounted and readied on the system tape drive, transport Ø, prior to any user input requests. The log tape provides the input to DATAR.

IA-41,733

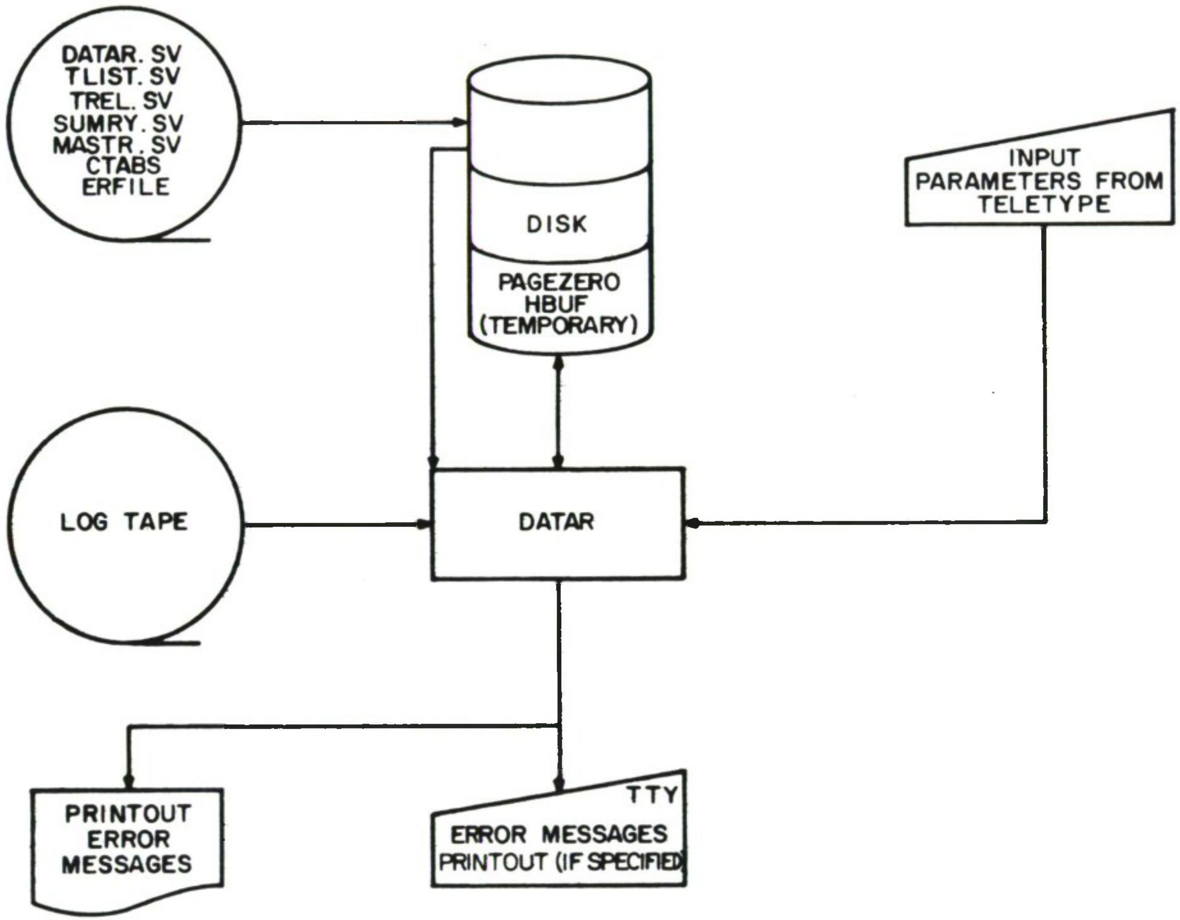


Figure 23 GENERAL SYSTEM FLOW OF DATA REDUCTION PROGRAM

DATAR is called by entering an input message on the system teletype. There are two forms of input messages which result in two modes of operation, interactive (conversational) or switch. The interactive mode requires the user to specify input arguments by responding to a series of interactive requests output by DATAR. The switch mode, where a switch is the character / (slash) followed immediately by an alphabetic character, uses switches to modify input groups and specify input arguments.

On entry, the Command Interpreter (CI) residing in DATAR.SV is loaded from disk and uses the input arguments to determine the type of output to be produced. The user may obtain a brief summary, a detailed summary, an octal tape listing, or a listing with actual RRTC times, with time differences (intervals), or with relative times. The output device, (line printer or teletype) is also determined from the input message. DATAR output is printed at the specified device, and error messages are output to the teletype and, if in use, the line printer.

DATAR requires the conversion tables (CTABS) and the error message file (ERFILE) to be disk resident for all types of output. If an octal listing is desired, the CI begins printout on the output device. However, if a summary or another type of listing is desired, the CI saves some information on disk in two temporary files, PAGEZERO and HBUF, and calls one of the save files (SUMRY.SV, TLIST.SV, or TREL.SV) into execution to do the processing. Error messages are directed to the teletype and the output device. Note that a CONTROL-A interrupt stops all programs and returns to DOS without deleting the temporary disk files, PAGEZERO and HBUF.

OPERATING PROCEDURES

Input Message

DATAR is called by entry of a user input request starting with the program name DATAR. The two valid messages are:

1. DATAR [ƀout-device] ↵

2. DATAR/ $\begin{pmatrix} B \\ D \\ L \end{pmatrix}$ [/sub-options] [ƀid] [ƀRECORDS/types]

   [ƀout-device] ↵

where ƀ indicates a space.

Both messages result in the disk iperating system (DOS) loading the save file DATAR.SV and passing control to the CI portion of DATAR. The ordering of the input groups is important and should be adhered to as illustrated above.

<u>Command Interpreter</u>

The CI operates in two modes, interactive (conversational) and switch. The interactive mode is invoked by message type 1 above. The switch mode requires a more complex input message (type-2) but minimal user interaction. Also, the switch mode is easier to enter and is processed by the CI in less time.

<u>Interactive Mode</u>

The interactive mode operates in the following manner. DATAR types an interactive request that includes all valid responses as shown in Table XV. The user must reply with either the full word response or the corresponding integer. Based upon the user response, DATAR either types another request or determines that the required input parameters have been obtained and passes control to processing. A user reply of COMBINATION (or 7) to request number 5 or of COMBINATION (or 8) to request 7 causes the CI to type requests 6 or 8, respectively. In either case, a 1 to 5 or 6 digit integer must be entered using the specified digits from the preceding request. Also, a user reply of LIST to request 9 causes the CI to type a list of the numbers and names of all devices defined in the Equipment Table. Following the list, the CI reissues request 9. The user may respond with numbers or names, but repetitions are ignored. A list of requested devices

Table XV

Interactive Requests and Responses for DATAR

Request Number	Text
1	ENTER SUT RUN NAME.
2	ENTER OPTION: BRIEF(1), DETAILED(2), OR LIST(3).
3	ENTER YES(1),OR NO(Ø) FOR PLOT.
4	ENTER SUB-OPTION: INTERVAL(1), SPECIFIC(2), ORDERED(3).
5	ENTER SUB-OPTION: INTERVAL(1), SPECIFIC(2), ORDERED(3), ACTUAL(4), OCTAL(5), RELATIVE(6), OR COMBINATION(7).
6	ENTER COMBINATION AS 1 TO 5 DIGIT INTEGER USING 2 TO 6 ABOVE.
7	ENTER RECORD KEY: ALL(1), HISTORY(2), SCENARIO(3), QUERY(4), RESPONSE(5), COMMAND(6), ERROR(7), OR COMBINATION(8).
8	ENTER COMBINATION AS 1 TO 6 DIGIT INTEGER USING 2 TO 7 ABOVE.
9	ENTER DESIRED DEVICE NUMBERS OR NAMES SEPARATED BY BLANKS OR LIST.
10	ENTER YES(1), OR NO(Ø), FOR START, STOP SPECIFICATION.
11	TO TERMINATE, ENTER END.
	ENTER LOGICAL OR PHYSICAL RECORD START, STOP PRECEDED BY L OR P.

is printed in the order defined by the Equipment Table. Figure 24 illustrates the various interactive paths to obtain the desired output.

The output device to be used must be specified in the original message. The optional input group, Out-device, has a value of $TTO for the system teletype or $LPT for the system line printer (the default output device).

Switch Mode

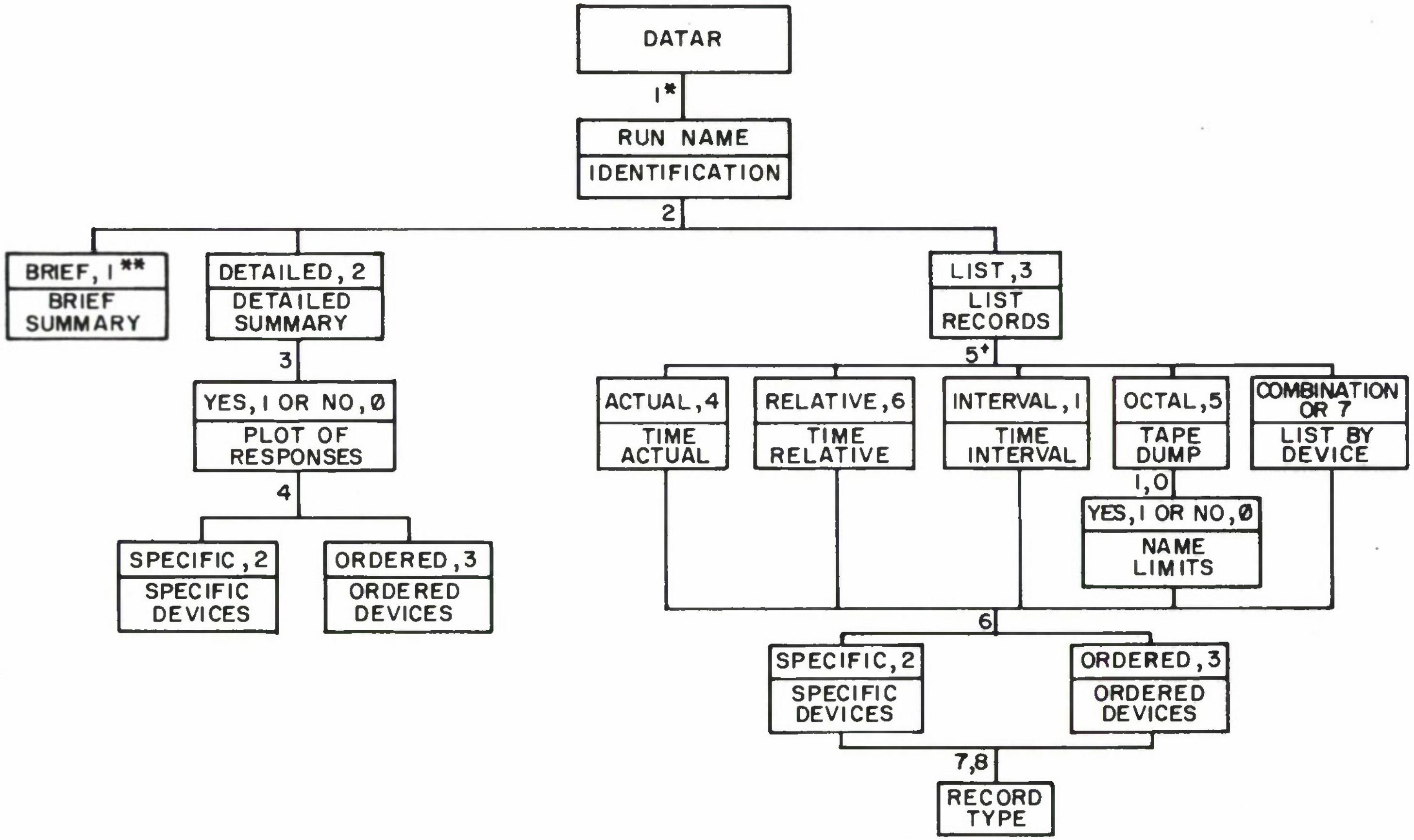
The message which invokes the switch mode is given in general form by message type 2 above. One of the three switches (/B, /D, or /L) must accompany the program name DATAR, otherwise the interactive mode is entered. All switch letters were chosen to relate to the function performed and to simplify mnemonic identification.

The input group DATAR/$\begin{matrix}B\\D\\L\end{matrix}$[suboption(s)] allows various combinations of option and suboption switches. One of the option switches B, D or L is required; if more than one is given, precedence is given first to B, then D. The option switches, listed in Table XVI, determine the type of output to be generated: brief summary, detailed summary, or listing.

The suboption switches are also listed in Table XVI. The suboption switches are meaningless for the B option. For option D, only O, S, and P are meaningful. For the L option, all are meaningful except P. The suboption switches specify the type of data to be included in the output option. They also determine if the data are to be given sequentially or on an individual device basis. If the data are to be given by device, the suboption switches tell DATAR whether all or user specified devices are to be examined.

The optional input group <u>id</u> specifies the run identification. It is the first n ($1 \leq n \leq 20$) characters of the run identification given at the start of a real-time emulator run. If <u>id</u> is not given, DATAR uses data from the first run on the tape.

IA-41,734



* THE INTEGER n SPECIFIES REQUEST NUMBER FROM TABLE XV
** THE GROUPS (RESPONSE WORD, NUMBER) ARE TAKEN FROM TABLE XV
\+ A RESPONSE OF 2 OR 3 GIVES ACTUAL TIMES BY DEVICE

Figure 24 INTERACTIVE TREE DIAGRAM FOR DATAR

Table XVI

Option and Suboption Switches for DATAR

Switch	Function
<u>Options</u>	
/B	Brief Summary
/D	Detailed Summary
/L	Listing
<u>Suboptions</u>	
/S	Examine only user specified devices
/O	Examined all devices in order of E.T.
/P	Histograms of Response Distributions
/N	Name records for octal dump
/I	Print time intervals rather than actual times
/R	Print Relative times rather than actual times
/T	Octal format tape dump

The optional input group RECORDS/type(s) specifies the type(s) of logical records to be included in the output. Valid switches are given in Table XVII. Any combination of values is allowed. Omission of this group implies all types. The B and D options ignore this group.

The optional input group Out-device is defined above under interactive mode.

The option and suboption switches may be combined as shown in Table XVIII and Figure 25. Table XVIII presents all meaningful input requests with a brief description of the output. (The optional input groups are not listed.) Figure 25 also illustrates the meaningful switch combinations.

Summaries

There are two types of summaries, brief and detailed. The brief summary examines all records for all devices, listing error messages and gathering general statistics. The detailed summary gives similar statistics but does so by device. Note that the input group RECORDS is meaningless since both summaries examine all types of records.

Brief Summary

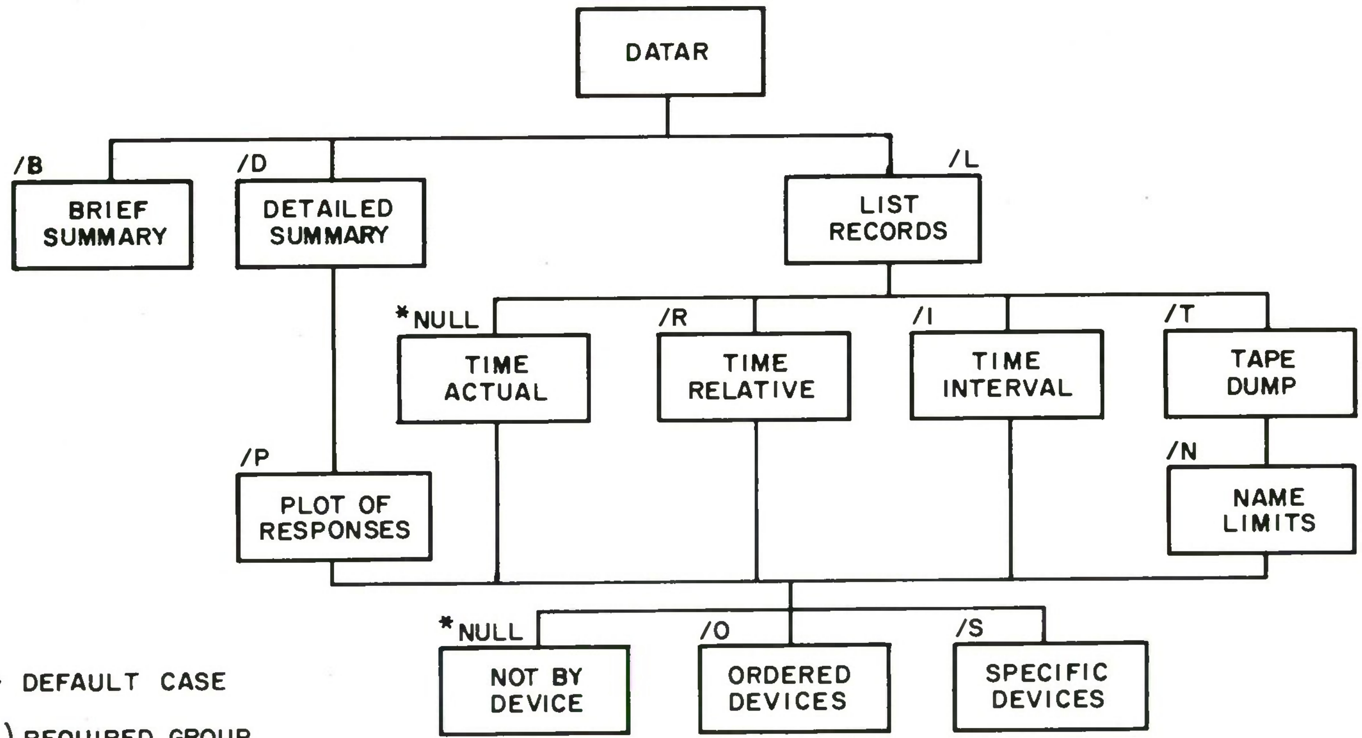
The brief summary ignores suboption selections. The format of the brief summary output is illustrated in Appendix IV, Figure 26. The summary data in this figure is taken from the file with the run identification of "RUN FT7". A list of all error messages with associated device names precedes the summary data.

Various RRTC times are given in the following units: elapsed time is expressed in seconds to the nearest 100,000 th, response times in seconds to the nearest hundredth, total emulator CPU time in tens of microseconds, and percent emulator CPU to the nearest hundredth.

IA-41,735

GENERAL SWITCH-MODE INPUT MESSAGE

DATAR (/B /D /L) [/I /R /T /P /N /S /O] b̸ [id] b̸ [RECORDS [/H /S /Q /R /C /E]] b̸ [$LPT $TTO]



* DEFAULT CASE

( ) REQUIRED GROUP

[ ] OPTIONAL GROUP

Figure 25 SWITCH TREE DIAGRAM FOR DATAR

Table XVII

Record Type Switches

Logical Record	Symbol	Switch
HISTORY	H	/H
RESPONSE	R	/R
QUERY	Q	/Q
SCENARIO INSTRUCTION	S	/S
COMMAND	C	/C
ERROR	E	/E

Table XVIII

Switch Combinations and Valid Inputs

Input Message	Action Taken
1. DATAR/B	BRIEF summary of all data preceded by a list of error messages.
2. DATAR/D [/P] or DATAR/D/O [/P]	DETAILED summary for each active device in the order established by the Equipment Table. A plot of response times is available as an option (/P).
3. DATAR/D/S [/P]	Same as above except only those devices specified by the user (upon request) are examined.
4. DATAR/L/I	LIST all records in sequence written. Include transmission time intervals, processing (task) time intervals, and response times.
5. DATAR/L/I/O	Same as above except list separately for each active device.
6. DATAR/L/I/S	Same as above except devices must be specified by user.

Table XVIII (Concluded)

Switch Combinations and Valid Inputs

Input Message	Action Taken
7. DATAR/L	LIST all records in sequence written. Include internal scenario address and actual clock times for start transmission and start/end task.
8. DATAR/L/O	Same as above except list separately for each active device.
9. DATAR/L/S	Same as above except devices must be specified by user.
10. DATAR/L/R/O	LIST separately for each active device all records in sequence written. Include internal scenario address and start/end transmission times relative to LOGON and test start time.
11. DATAR/L/R/S or DATAR/L/R	Same as above except devices must be specified by user.
12. DATAR/L/T [/N]	LIST all records in sequence written in octal tape dump format. Naming of starting and stopping logical (or physical) record numbers is available as an option (/N).
13. DATAR/L/T/O [/N]	Same as above except list separately for each active device.
14. DATAR/L/T/S [/N]	Same as above except devices must be specified by user.

The logical and physical record counts are given by the counts following the headings MESSAGES and RECORDS, respectively. The headings UN-R and UNSOLICITED specify unsolicited responses. The TERMINAL-MAX heading is used to name the terminal associated with the maximum response. The asterisk (*) following a scenario instruction type denotes a lower case character or a non-printable special character.

Detailed Summary

The detailed summary allows a device specification suboption as well as a special histogram output. The format of the detailed summary is illustrated in Appendix IV Figure 27. A list of all requested devices to be examined is printed prior to summary data, and consists of either all devices defined in the Equipment Table or only those devices specified by the user.

The name of the file used in Figure 27 is "RUN FT7". A detailed summary is given for each active, requested device, and the name of the device is given as a terminal identification. Unsolicited responses are counted as record types. Also, the average and maximum RRTC response times are given in seconds, to the nearest hundredth. As in the brief summary, an asterisk (*) is used to identify non-printable lower case and special characters which are used as scenario instruction types.

If requested, a histogram of response distribution is printed for each active device following the summary data. Figure 28, Appendix IV illustrates the format of the histogram. As can be seen, the name of the device is given at the top of each page and is followed, on the first page, by a list of all quarter-second response intervals which have a positive count and percentage. The count gives the actual number of responses which fall within the specified interval. The percentage is calculated by dividing the count by the total number of responses. All responses less than Ø.25 seconds are included in the first interval, while all responses greater than 15.ØØ seconds are shown in the 15.ØØ second interval. If there are no intervals with a positive count, then a histogram is not generated.

Following the summary data (and histogram if requested) for the last active device, the program lists all requested devices which were found to be inactive during the run.

Listings

There are basically four types of listings: octal tape, actual times, time intervals, and relative times. All these suboptions allow record selection based on device and/or record type. If the user decides to obtain the listing by device, then all devices defined in the Equipment Table must be requested or the desired device names and/or numbers must be specified in response to the interactive request number 9. A list of all requested devices will precede any data and a list of requested but inactive devices will terminate the listing.

The types of logical records to be listed may be selected by using the RECORDS input group. In the switch mode, all records are listed if the RECORDS group is omitted. The heading MESSAGE on each listing page refers to the logical record number of the first non-history record on the page.

Octal Tape

The octal tape dump listing is used to print the contents of each logical record in octal byte format. The user may name the starting and stopping logical (or physical) record number by using the /N option. If starting and stopping numbers are given, the program skips all logical (or physical) records up to the start. It produces its octal output in logical record format and stops at the given logical (or physical) record number. Figure 29 in Appendix IV illustrates an octal tape listing output format. The user requested that all devices in the Equipment Table be examined and named the starting and stopping logical record numbers as 101 to 110.

As can be seen, the output for each active device gives the range limits and device name prior to the data. After a range is completed, the user may specify another range of limits or continue to the next active device. Note that there may not be records within the range associated with the given device (CTØ in Figure 29). The character P represents the physical record boundary.

Actual Times

The actual time listing contains the actual RRTC start of transmission and the start and end of task processing times. The values are taken directly from the record and listed in tens of microseconds. The actual time listing is the default suboption in the switch mode. Figure 30 in Appendix IV illustrates the format of the actual time listing.

As shown by the example in Figure 30, the user requests an actual time listing of Query and Response records ordered sequentially and output on the system teletype. The name of the run is "6-14 4:30 PM." For each record, the type of record is given followed by transmission start, task start, and task end times. The heading SCEN ADDR gives the location of the start of the scenario instruction relative to the beginning of the scenario, if any.

Time Intervals

The time interval listing contains differences between the RRTC times. This listing also calculates response times as the difference between the start of transmission for a solicited Response and the end of transmission from the preceding query associated with the same device. Figure 31 in Appendix IV presents the format of the time interval listing.

The example in Figure 31 shows a time interval listing of "RUN2" in which the user chooses to specify the devices to be examined. For each active device, the terminal identification is given, followed by all the data associated with the particular device. For each record, the record type is given as well as the difference between the end and start of transmission time, the difference between the end and start of task processing time, and the cumulative emulator CPU time, all in tens of microseconds. The response times are given in seconds to the nearest hundredth.

Relative Times

The relative time listings are by device with user specification of devices being the default case. Figure 32 in Appendix IV illustrates the format of a relative time listing.

In the example shown in Figure 32, the user requests that all devices defined in the Equipment Table be examined. Both the run-start time and the user start time (UST) are given in tens of micro-seconds. The run-start time is the start of transmission of the first non-history record in the file. The UST is the start of transmission of the first Query or solicited Response associated with the device. A value of BELOW is given for UST if a Query or solicited Response is not the first record type in the file for the particular device. For each record, the record type is given in addition to the start and end of transmission minus the UST, the start and end of transmission minus the end of transmission time of the previous Query, and the location of the scenario instruction (as SCEN ADDR) relative to the beginning of the scenario, if any.

ERRORS

There are several error conditions recognized by the various programs. Table XIX lists all error conditions and messages that may

Table XIX

DATAR Error Message File (ERFILE)

Number	Message	Cause or Corrective Action
1	Invalid option	Submit valid option.
2	Invalid termination option	Submit valid option.
3	Invalid sub-option or key (record)	Submit valid option.
4	Invalid device specification	Submit valid device name or bad device address logged.
5	Disk file accessing error (read/write)	Error from DOS, disk file may be missing.
6	End-of-file (on tape)	End of run.
7	Invalid tape identification	Log tape file incorrectly logged.
8	Unrecognizable message type	Bad record type logged.
9	Zero length record found	Two successive records with zero word length.
10	Illegal program call	Overlay problem, maybe disk file is missing.
11	Command instruction missing	C-type record with null text.
12	DISK SPACE exhausted	Not enough disk for temporary files or overlay.
13	Invalid device table format	Equipment Table not second record in file.
14	Tape read error	Tape drive problems, may not be mounted properly.

occur, during execution of DATAR.

The general format of the error message is:

RECORD m, WORD n: error message text

where m specifies the physical record that contains the erroneous logical record and n specifies the first word of the logical record relative to the start of the in-core buffer containing the record. Many of the conditions allow the user to start over or submit another choice. However, some (such as tape and disk errors) are unrecoverable. The cause of error condition and/or corrective action for each error is also given in Table XIX.

SAVING TEST DATA

After analyzing the test data with DATAR, the user may wish to save the data for future analysis on the NOVA 800 or some larger computer. A program (MASTR) has been written to transfer data from a log tape to a master log tape (to consolidate tapes or to get comparable runs on one tape). The master tape (or original tape) may then be used as input to DATAR to analyze the run again or compare a series of runs manually. In addition, more sophisticated statistical methods may be employed to produce more meaningful statistics for comparing and evaluating an SUT.

Program Description

In general, the MASTR program (written for a one tape drive system) reads the data from the input log tape, temporarily stores it in a file on disk, waits for the output (master) tape to be mounted, writes the data from disk onto the tape as the last sequential file, and terminates the file with two end-of-file (EOF) marks. If disk storage is insufficient to complete the transfer in one pass, the program continues through as many passes as necessary, each time notifying the user that an additional pass is required. Obviously, a multiple pass transfer requires input and output tapes to be mounted and dismounted several times.

Input Message

MASTR requires two user supplied input parameters: a run identification (used to locate the test run) and the amount of available disk space (used as temporary storage). The two commands that activate the tape transfer program are:

1. MASTR↲
2. MASTR id ds↲

(Although message 1 appears more concise, note that requests to supply values for the input groups id and ds will be issued by the program.) The first input group, id, specifies the first n ($1 \leq n \leq 20$) characters of the run identification as found in the Identification-History record, the first logical record logged. The run identification, which was entered at the start of emulation, is required to allow access to different runs on multiple run tapes.

The second group, ds, is the number of unused disk blocks available for temporary storage. The program uses ds-2 blocks to protect the used portions of disk. The number of unused blocks is given by the DOS command DISK. This number can be increased by deleting disk files no longer in use. A good approximation for the number of blocks required for a single pass transfer is the number of physical records used for the run (obtained from the record count in a brief (/B) summary) plus five (two for the unused blocks and three for disk file linkage words). This number must be multiplied by the ratio of physical record size to disk block size, which presently is 1.

Operation

The MASTR program is called by one of the input messages described above. It obtains the run identification and disk size from the input message (#2), or as responses to the program commands ENTER RUN

IDENTIFICATION and ENTER AMOUNT DISK LEFT. The program then issues the command:

MOUNT INPUT TAPE, STRIKE CARRIAGE RETURN

and waits for a carriage return. Upon receipt of the carriage return, MASTR locates the first file (on the input tape) that contains the specified run identification as the first n characters in the History-Identification record.

The program uses the disk size and physical record size to calculate the number of tape records that can be written in the temporary disk file, MITCHTEMP. MASTR reads the tape until disk space is exhausted or an EOF mark is encountered. If disk space is insufficient the message:

NOT ENOUGH DISK.
REMOUNT INPUT TAPE AFTER OUTPUT TAPE IS WRITTEN.

notifies the user that one or more additional passes are necessary to complete the transfer. This implies remounting the input tape after the first segment is transferred to the master tape.

After the disk file is written, MASTR issues the command:

MOUNT OUTPUT TAPE, STRIKE CARRIAGE RETURN

and waits for the carriage return. Upon receipt, the program locates the double EOF mark on the master and writes all the data from the disk file onto the output tape, overwriting the second EOF of the preceding run. If an additional pass is necessary, the program requests that the input tape be mounted and continues the loop until the transfer is completed. Upon completion, the message:

LOG TAPE TRANSFER COMPLETE

is output and two EOF marks are written. The first EOF terminates the file while the second indicates that the file is the last one on the tape. Note that a tape intended to be a master must be initialized by the DOS command INIT/F MTØ prior to the transfer operation. The command writes two EOF marks at the beginning of the tape.

The program does not check the run identification of each file on the output file. Therefore, files may be written with duplicate file names. However, only the first file with a duplicate file name is accessible.

Errors

The MASTR program checks for various error conditions. If an error exists, a message is output and the transfer terminates by returning to DOS. Table XX lists the error conditions, messages, and suggested corrective action. Remember that files on a master tape are only as unique as the run identification given at the start of the emulation test.

Table XX

MASTR Error Message File

ERROR MESSAGE	ERROR CONDITION	CORRECTIVE ACTION
1. NOT ENOUGH DISK	Space too small for one physical tape record.	Delete some files and specify larger number.
2. ERROR LOCATING INPUT FILE	Invalid run id, illegal format, tape read error.	Check id, format, read errors by using DATAR/B with and without run id.
3. DISK ERROR	Trouble writing/reading file MITCHTEMP.	Ensure disk accessibility. Try again.
4. INPUT TAPE READ ERROR	Tape equipment or parity problem.	Check channel number unit ready, etc. Otherwise, fatal parity error.
5. OUTPUT TAPE WRITE ERROR	Tape equipment or parity problem.	No double EOF or check channel number unit ready, write lockout, etc. Otherwise, fatal parity error.
6. EOF WRITE ERROR	Tape equipment or parity problem.	No double EOF or check channel number unit ready, write lockout, etc. Otherwise, fatal parity error.
7. ERROR LOCATING OUTPUT FILE	No second EOF, tape equipment or parity problem.	Check equipment, initialize tape if never done before.

SECTION IX

EXECUTION TIMES

REAL-TIME INSTRUCTIONS

Because of the variety of scenario instructions available to the user, it may be possible in some instances to accomplish the same task using more than one method, or combination of scenario instructions. In these cases, execution timing for scenario instructions may be a consideration in determining maximum scenario efficiency.

Table XXI gives the current best estimates of real-time emulator execution times. The times given represent the total cost (Scenario Interpreter as well as Real-Time Exec execution time) in microseconds of emulator CPU time for executing each function once. The functions timed include two miscellaneous functions (logging and the receipt of an unsolicited response) followed by the scenario instruction types given in the same order as Table XVI of Volume 2 of this series followed by the command types in alphabetical order.

The data were obtained by making a very large number of short runs on the field-test emulator. In most cases, a run consisted of executing a single scenario for a single device. After performing its task, the scenario executed a QUIT command. The data reduction brief summary operation was used to obtain the CPU time.

The general technique used was to execute the desired function 1000 times in a loop, as in the case below of one of the scenarios used to test the add instruction:

```
1     3    A 12
2     3    C [ LOG ALL OFF ALL
3    22    1 1000 R9
4    28    L LOOP
```

Table XXI
Execution Times
Real-Time Scenario Instruction
(in microseconds)

Function	Execution Time	Footnotes
Miscellaneous Functions		
Logging	1778 + 6.6b	
Receipt of Unsolicited Response	2325 + 231r	10, 26
Control Instructions		
R	4397 + 477i	10, 26, 27
R''	3312 + 220r	10, 26, 27
Q	1721 + 255q	10, 26, 27
I	3845 + 440i	10, 26, 27
O	953 + 213q	26, 27
;	679	27
:	679	13, 27
C	–	10, 14, 27
E	3137 + 54e	10, 15, 27
E	1333	10, 16, 27
D	1305	1, 3
W	1458	1, 3, 13
d	1236	1, 3
e	1136	5, 27
X	7490	7, 27
7	761	3
8	2703	1, 17, 27

Table XXI (Continued)
Real-Time Scenario Instruction Execution Times

Function	Execution Time	Footnotes
Arithmetic and Logical Instructions		
+	691	1, 27
-	691	1, 13, 27
*	768	1, 27
/	788	1, 6, 27
&	708	3
Assembler Directive Instructions		
L	—	18
a	—	18
blank	—	18
t	—	18
i	—	18
Branch and Comparison Instructions		
J	631	3
B	689	1, 2, 3, 13
U	689	1, 2, 3, 13
>	689	1, 2, 3, 13
<	689	1, 2, 3
G	689	1, 2, 3
H	689	1, 2, 3
M	657 + 39m	3, 19
S	677 + 39m + 48n	1, 9, 27
Y	719	2, 3

Table XXI (Continued)
Real-Time Scenario Instruction Execution Times

Function	Execution Time	Footnotes
Branch and Comparison Instructions (continued)		
n	719	2, 3, 13
9	721	2, 3
q	722	2, 3, 13
K	682	3, 20
3	682	3, 4, 20
P	1016 + 43.5p	3
Core Memory Allocation Instructions		
A	1361	3, 10
F	7858	10, 21, 27
Move Instructions		
l	668	3
g	687	27
p	687	1, 27
5	621 + 43.6t	27
T	640	27
=	717	27
Z	715	13, 27
V	663 + 21.6c	13, 27
6	647 + 21.6c	27
\	708	27
@	748	27
r	680	27
c	666	4, 27
h	684	27

Table XXI (Continued)
Real-Time Scenario Instruction Execution Times

Function	Execution Time	Footnotes
Diagnostic Instruction		
?	–	
Commands		
DUMP	–	
ERROR	3950	
LOG	5355	12, 22
MONITOR	7204	12
QUIT	2060	23
RESTART	12,571	12, 24
SCALE	5046	25
START	14,540	8, 10, 11, 12
STATUS	4681	12
STOP	12,571	12, 24
SUB	7490	7, 10, 11
SUB	7858	10, 11, 21

Table XXI (Continued)
Real-Time Scenario Instruction Execution Times

Nomenclature

b = length of MESBF (variable or text) portion of buffer to be logged, in bytes

c = number of bytes for which longitudinal redundancy check (LRC) byte is calculated

e = length of error message in type-E scenario instruction, in bytes

i = length of query and length of response, in bytes

m = number of bytes successfully matched

n = number of bytes unsuccessfully matched

p = number of bytes parity checked

q = length of query transmitted, in bytes (those up to, but not including, the first NULL (zero) byte in a query buffer)

r = length of response received, in bytes

t = number of bytes transferred from (contained in) a type-5 scenario instruction to a query buffer

Table XXI (Continued)
Real-Time Scenario Instruction Execution Times

Footnotes

(1) Add 6.4 microseconds for each field of type 10 or 11 which contains a Register number.

(2) Add 6.4 microseconds if the branch is not taken.

(3) Add 14 microseconds if instruction starts at an even byte.

(4) Add 26.8 microseconds if initial value of RGRPT points to an odd byte.

(5) The time includes the time for the type-e instruction plus the additional time for the following (executed) instruction over what it would be if executed normally rather than by the execute instruction. The normal execution time of the following instruction is excluded. Increased time over most other instructions is spent in scenario management code in the Exec. The type-e instruction causes two changes in the scenario associated with the device. The time given includes the time to free each core page when control passes to the other but no time to read pages from disk since the core pages were not overlaid. If one or both core pages were in use by other devices, the freeing time would be less, but if all core pages were in use, the type-e instruction could require disk reads to be done.

(6) Execution time varies by 11.4 microseconds from minimum to maximum, depending upon values used.

(7) Includes time to execute the SUB command with scenario specified and the type-X scenario instruction. Includes time to allocate set of Registers but not the time to free them.

(8) Includes the time to start the scenario for the named device and to terminate that scenario by execution of end-of-scenario.

(9) Add 13 microseconds if branch not taken. If substrings of the instruction string occur in the response, the number of comparisons may be relatively large. For instance, if the response ABCABACABABCABABACABABABC is searched for the string ABABABA, then $m = 27$ and $n = 19$ and the execution time is 2642 microseconds.

Table XXI (Continued)
Real-Time Scenario Instruction Execution Times

Footnotes

(10) Execution time will vary depending upon the number of blocks in the free chain which have to be examined to find a large enough block to allocate and/or to find the proper place in the chain to place a freed block.

(11) Execution time will vary depending upon the number of Scenario Directory entries which have to be examined before the named one is found.

(12) Execution time will vary depending upon the number of Equipment Table entries which have to be examined before the named one is found, and this number may be different depending upon whether hierarchical equipment names are used in the command or not.

(13) Time estimated based on measured time for a similar instruction.

(14) Time varies widely with command type. The time to execute the type-C instruction is included in the command execution time, except for the QUIT command.

(15) Time with error-message logging enabled. Time includes logging time.

(16) Time with error-message logging disabled.

(17) Includes time to log the response. If response logging is disabled, the instruction is equivalent to a NOP.

(18) Assembler directives are not executed in real-time and are not even included in the internal scenario.

(19) Add 24 microseconds if the branch is taken. If an m-character compare is made, the first four characters match, but the fifth one does not, the execution time should be 657 + 4 (39) + 24 = 837 microseconds.

(20) Add 10 microseconds if branch not taken.

(21) Includes time to execute the SUB command with scenario specified and the type-F scenario instruction.

Table XXI (Concluded)

Real-Time Scenario Instruction Execution Times

Footnotes	
(22)	Time for LOG ALL OFF ALL
(23)	Time through the time the record is logged. Certain termination activities are performed after logging.
(24)	Includes time to execute RESTART of a named device and time to execute STOP THIS for the named device.
(25)	Time will vary depending upon number of digits in scale factor to be converted. Conversion time is 31 microseconds per decimal digit.
(26)	Using an asynchronous line adapter at 10 characters per second.
(27)	Add 12.8 microseconds if instruction starts at an odd byte.

5	28	+ R9 34 R11
6	34	+ R8 1 R8
7	40	U LOOP R9 R8
8	47	C [LOG ALL ON C
9	63	C [QUIT

Instruction 5 is the one being timed. Instructions 6 and 7 are for loop control and instruction 3 controls the iteration count. Instructions 2 and 8 turn logging off and then back on to capture the final CPU time value. Such scenarios were run two or more times each to check the degree of reproducibility.

A second, base scenario was then prepared, identical to the above except that instruction 5 was eliminated. The second scenario was then run two or more times, and the most representative CPU time value was chosen for each of the two scenarios. The difference between these values divided by the iteration count gives the function execution time.

The contents of the two scenarios were varied depending upon the function being timed. In the case of several of the commands, more than one scenario had to be run concurrently. The iteration count was reduced to 100 for the miscellaneous functions, for some of the commands, and for the query instructions. In any case, an appropriate base scenario was always constructed and run so that the difference in CPU times would isolate the function or functions being timed (a few of the functions cannot be executed multiple times independently of other functions).

The measured results were given general reasonableness checks and were also evaluated by comparing differences between measured results for different functions (primarily the scenario instructions) and differences obtained from NOVA instruction counts for the same functions. No attempt was made to verify the absolute values given in Table XXI because of the complexity of the emulator system. The relative comparisons checked reasonably well, although certain differences have not yet been explained. The data in Table XXI cannot be regarded as precise. The presence of a zero in the units position cannot be regarded as indicating low precision nor can the presence of a decimal place be regarded as indicating high precision in all cases. In the latter case, the increments given in the table proper for those functions whose execution times vary with string length, the increments given were obtained by computations on the measured results, although these increments checked rather well in those cases in which instruction counts were made. The increments given in the footnotes are generally more precise since most of them are based on instruction counts (assuming that the CPU clock is accurate).

The relative comparisons made for approximately 15 scenario instruction types indicate precision varying from 0 to 35 microseconds. No formal comparisons were made for the commands although it appears possible that much larger discrepancies may be present. In particular, from scanning the code for the LOG command and the MONITOR command, it does not seem reasonable that the latter should require nearly 2 milliseconds more than the former. It should also be noted that a typical command generally provides many more options than a typical instruction and, therefore, will result in a much greater range of execution times. It was not possible to time and report each option of each function. In addition, as the footnotes show, a number of run-dependent factors can significantly affect the timing results.

Several factors are present which would make it very costly to attempt to resolve the discrepancies noted above. At least 700 runs were made to obtain the current data. Most of these lasted several seconds in real-time, but some lasted a minute or two. The results had to be listed, recorded, and analyzed. Most of the scenarios were run two or more times each since the results frequently showed some variation in total emulator CPU time. It was felt that replicated runs should agree within possibly 10 to 30 microseconds based on early experience with the simpler instructions. In the case of some of the commands and query instructions, the total variation was sometimes 200 or 300 microseconds. In the case of a common base scenario run a number of times over a two-month period, the total variation was 3200 microseconds (for a 1½ second run - 0.2%). It seems likely that these variations are the result of clock frequency variations, possibly the result of temperature differences. The clocks involved were the CPU clock, the Readable Real-Time Clock used for timing measurements, the "Real-Time Clock" used for response timeouts and which places a continuous overhead on the DVM, and the line-adapter clocks in the case of query instructions. In addition, it is known that the timing characteristics of the magnetic tape drive have a rather coarse control, and the tape drive had to be used in all runs to record at least the first and last event of the run.

A cause of greater variation in execution times is the fact that the NOVA computer has only very superficial byte-manipulation ability. The Exec uses 12.8 microseconds more to fetch the two-byte scenario instruction length field (for any instruction) from two adjacent words (when the instruction starts at an odd byte) than when it starts at an even byte. The Scenario Interpreter uses 26.8 microseconds more to fetch a two-byte operand (contained in certain instructions) when the instruction starts at an even byte than when it starts at an odd byte. The effect of these differences is that to achieve the best results one needs to examine the starting byte of each instruction in a scenario (or at least those within the loop) and make adjustments in case of differences between a base scenario and a timing scenario. One may also need to modify both scenarios by adding one or more instructions or changing their positions to cause cancellation of the even-odd effects. The nature and magnitude of this problem were only realized after a number of runs were made, instruction counts were made for portions of certain instructions, and relative comparisons were made. Making such even-odd corrections for a large number of scenarios would be quite time consuming.

In the case of the START and SUB commands, the present implementation reads at least the first two bytes of the Scenario name from the command for each Scenario Directory (SD) entry encountered. If 30 entries have to be compared and the scenario name starts at an odd byte in the command, the execution time for the command is 800 microseconds more than if the scenario name started at an even byte. (To force the even byte case, an odd number of blanks must occur after "START" and before the scenario name if the device name contains one digit.) To control this situation, the length and content of the SD as well as the location of the scenario name within a command must be controlled.

In the case of commands which contain device names, a further variation can arise. A total of 26.8 to 80.4 microseconds more will be used if the device name or "THIS" starts at an odd byte in a command. A total of 31 microseconds is used to convert each digit (after the two initial characters) in a device name. The execution time will further vary depending upon the number of Equipment Table (ET) entries which have to be searched. The number of entries searched will depend upon the ordering and linking of the ET entries and whether or not hierarchical equipment names are used in commands.

If the above factors are handled properly, one may be able to obtain relatively accurate results for the tests run. Certain additional factors need to be considered before applying the results. Of necessity, the tests were run under conditions whereby there was little competition for resources within the emulator. As the number of active, emulated devices increases, allocable core memory becomes splintered and those functions which must allocate and/or free core memory will use up more emulator CPU time. When a block is to be freed, each link in the free chain which must be examined, uses up 7 microseconds of CPU time, and approximately the same amount of time is needed during allocation. If an average of 25 links needs to be examined, the cost is 175 microseconds for each allocate or free operation. Every command executed requires the allocation of a command buffer, freeing of the command buffer, and allocation of an error-message buffer for the response to the command and may also require the freeing of a previous error-message or query buffer. In addition, 6 or 7 instructions (see footnote 10) and one of the miscellaneous functions allocate and/or free core memory. There is no dynamic measure of the length of the free chain, but the timing tests probably only caused a free chain of five or ten links. Very little logging was done (from 2 to 6 records per run), but each record logged requires one allocation and one free operation (for a Register Stack).

Scenario management can also have a significant effect on individual execution times. If an instruction spans a scenario page boundary, it must be buffered and a new scenario page becomes the active page for the device. The cost of the latter operation is in the vicinity of 200 microseconds. If the new page must be read from disk, the cost is greater. When the number of active scenario pages in core approaches the number of core pages allocated, a disk read may be required for each scenario instruction fetched from a new page. The emulator is designed to cope with this situation to handle peak loading problems. If an emulator module operates in this mode more than a relatively small fraction of the time, it is overloaded and its load should be reduced.

The data given in Table XXI ignores the effect of any error conditions. The only error messages allowed for are the normal responses to commands.

The "Real-Time Clock", used for response timeouts, provides a continuous overhead estimated at between 0.2% (2000 microseconds per second of elapsed time) and 0.3%. The effect of this overhead has been ignored in Table XXI in those cases in which emulator %CPU time was near 100% since the effect on a 700-microsecond instruction is only 1 or 2 microseconds. In those cases in which the % CPU time was lower (primarily some of the commands, the query instructions, the delay and wait instructions, and the miscellaneous functions), the emulator CPU times for the base scenario and the timing scenario were corrected for this overhead based on elapsed time, generally using a conservative 0.2% factor. This overhead is present throughout the elapsed time of a run, regardless of the amount of emulator activity.

NON-REAL TIME PROGRAMS

It is difficult to give anything but intelligent estimates as to the running times of the non-real-time programs. This is because of the many variables involved which determine execution times for each of the programs. Presented here is a sample problem for each program, with key characteristics defined, and approximate running times given. The times are based on an average derived from several runs of each program, and may vary within a 5 second range.

SSUB

The example shown in Appendix VI, Figure 33 shows a scenario called 34FORTN with macro calls not yet expanded. Figure 34 in Appendix VI shows the same scenario, now called FORTN, with macros expanded. The libraries which contain the macro definitions are given in Figure 35. The table below summarizes the key characteristics pertinent to the macro processing of this example. In this case, the macro processor takes about 20 seconds to complete execution.

number of libraries	2
number of macros in libraries	16
length of file without macros expanded	1172

length of file with macros expanded	3956
number of macro substitutions	185

MACDEF

The program used to generate macro libraries is MACDEF. The execution time of this program depends on characteristics summarized in the table below for the example shown in Appendix VI, Figure 35, the KAPLIB library.

number of definitions	3
length of input file	205
length of output file (.ML)	203

Execution time to create KAPLIB.ML from KAPLIB is 4 seconds.

CVT

The scenario assembler program may convert the FORTN scenario (Figure 34) into an internal scenario by using any of its three printing options. Average times for execution are 35 seconds for assembly with no listings (CVT/N option), 55 seconds for assembly with partial listings (CVT/P option), and 3 minutes 10 seconds for assembly with complete listings (CVT option). These times, of course, reflect to some degree, the speed of the printer. The table below summarizes the key characteristics pertinent to the Assembly of the example.

label definitions	22	
other label references	24	
queries	25	
arithmetic instructions	133	
search instructions	22	
commands	3	
assembler directives	3	
other instructions	159	
Total instructions	391	
length, in bytes, of internal scenario		2435

DATAR

The data reduction program processes the log tape written during an emulation run and can produce many combinations of listings and summaries. Execution times for all combinations are too cumbersome to be presented here. The table below describes the key characteristics pertinent to the data reduction of a log tape from a sample emulation of the Fortran Cost scenarios presented in Figures 33-35.

number of physical records	43
number of logical records	376
number of internal scenarios in directory	70
number of devices in ET	26
number of active devices	2
number of queries	123
number of responses	226
number of scenario instructions	1
number of lines of output for relative-time listing	660

Using such a log tape, the data reduction program produces a brief summary in 20 seconds and a relative-time listing for a single emulated device in an average of 4 minutes. These times are for processing of a file which is the first file on a log tape. If more than one emulation file is on a tape (perhaps a tape created by the MASTR program) the DATAR program rewinds to the beginning of tape and re-searches for the correct run every time it begins a new device listing for the run. This, of course, may consume considerably more time.

MASTR

The execution time of the MASTR program depends on several factors, as described in the table below:

number of physical records in run
disk space available
file number of MASTR tape

Also included in the complete execution time is the length of time it takes the user to dismount the original log tape and mount the MASTR tape, for as many times as is needed to complete the transfer. Therefore it is unrealistic to give any meaningful timing estimates.

REFERENCES

1. Data General Corporation, Disk Operating System User's Manual, 093-000048-03, Southboro, Massachusetts, 1971.

2. Data General Corporation, NOVA Editing Routines, 093-000018-02, Southboro, Massachusetts, 1971.

3. Data General Corporation, File Check Program, 093-000071-00, Southboro, Massachusetts, 1971.

4. Data General Corporation, Tape Dump Program, 093-000059-01, Southboro, Massachusetts, 1971.

5. Data General Corporation, How to Use the NOVA Computers, Southboro, Massachusetts, 1971.

APPENDIX I

Conversion Codes for IBM 2741

Because some of the 2741 control characters do not have a direct counterpart in the ASCII character set, an exact mapping was not possible. Table XXII is a list of the 2741 control characters, and their position in the ASCII table. This same mapping was used in the 2741 conversion code tables used for the on-site model of the emulator.

Table XXIII represents the conversion codes used by the Scenario Assembler for 2741 EBCDIC odd parity code, with the parity bit as the right-most bit. The "lab" conversion is used on the fixed-site model of the Emulator when emulating an IBM 2741 terminal using Data General's software driven data communications multiplexor. The "field" conversion reverses the order of the bits, and is used on the on-site model of the Emulator when emulating an IBM 2741 terminal using Digital Computer Controls asynchronous line adapters.

Table XXII

Control Characters for IBM 2741 Terminal

2741		ASCII	
Octal	Character	Octal	Character
∅37	EOT = control D	∅∅4	EOT = end-of-transmission
135	BS = backspace	∅1∅	BS = backspace
172	HT = horizontal tab	∅11	HT = horizontal tab
∅73	LF = line feed	∅12	LF = line feed
13∅	RES = restore	∅14	FF = form feed
133	NL = new line	∅15	CR = carriage return
∅34	UC = upper case	∅16	SO = shift out
174	LC = lower case	∅17	SI = shift in
∅31	PN = punch on	∅22	DC2 = device control 2
∅32	RS = reader stop	∅23	DC3 = device control 3
171	PF = punch off	∅24	DC4 = device control 4
136	IL = idle	∅26	SYN = synchronous idle
∅75	EOB = end-of-block	∅27	ETB = end-of-block
∅76	PRE = prefix	∅33	ESC = escape

Table XXIII

Conversion Code Table used for IBM 2741 Terminal

ASCII CHARACTER	ASCII CODE	2741* LAB CODE	2741 FIELD CODE	ASCII CHARACTER	ASCII CODE	2741* LAB CODE	2741 FIELD CODE
NUL	000			SP	040	U 001	100
SOH	001			!	041	U 127	165
STX	002			"	042	U 026	064
ETX	003			#	043	L 026	064
EOT	004	C 037	174	$	044	L 127	165
ENQ	005			%	045	U 013	150
ACK	006			&	046	L 141	103
BEL	007			'	047	U 015	130
BS	010	C 135	135	(	050	U 023	144
HT	011	C 172	057	)	051	U 025	124
LF	012	C 073	156	*	052	U 020	004
VT	013			+	053	U 141	103
FF	014	C 130	015	,	054	L 067	166
CR	015	C 133	155	-	055	L 100	001
SO	016	C 034	034	.	056	L 166	067
SI	017	C 174	037	/	057	L 043	142
DLE	020			ø	060	L 025	124
DC1	021			1	061	L 002	040
DC2	022	C 031	114	2	062	L 004	020
DC3	023	C 032	054	3	063	L 007	160
DC4	024	C 171	117	4	064	L 010	010
NAK	025			5	065	L 013	150
SYN	026	C 136	075	6	066	L 015	130
ETB	027	C 075	136	7	067	L 016	070
CAN	030			8	070	L 020	004
EM	031			9	071	L 023	144
SUB	032			:	072	U 010	010
ESC	033	C 076	076	;	073	U 007	160
FS	034			<	074	U 004	020
GS	035			=	075	U 002	040
RS	036			>	076	U 016	070
VS	037			?	077	U 043	142
				@	100	L 040	002

* C = control
U = upper case
L = lower case

Table XXIII

Conversion Code Table used for IBM 2741 Terminal (Concluded)

ASCII CHARACTER	ASCII CODE	2741 LAB CODE	2741 FIELD CODE	ASCII CHARACTER	ASCII CODE	2741 LAB CODE	2741 FIELD CODE
A	101	U 142	043	a	141	L 142	043
B	102	U 144	023	b	142	L 144	023
C	103	U 147	163	c	143	L 147	163
D	104	U 150	013	d	144	L 150	013
E	105	U 153	153	e	145	L 153	153
F	106	U 155	133	f	146	L 155	133
G	107	U 156	073	g	147	L 156	073
H	110	U 160	007	h	150	L 160	007
I	111	U 163	147	i	151	L 163	147
J	112	U 103	141	j	152	L 103	141
K	113	U 105	121	k	153	L 105	121
L	114	U 106	061	l	154	L 106	061
M	115	U 111	111	m	155	L 111	111
N	116	U 112	051	n	156	L 112	051
O	117	U 114	031	o	157	L 114	031
P	120	U 117	171	p	160	L 117	171
Q	121	U 121	105	q	161	L 121	105
R	122	U 122	045	r	162	L 122	045
S	123	U 045	122	s	163	L 045	122
T	124	U 046	062	t	164	L 046	062
U	125	U 051	112	u	165	L 051	112
V	126	U 052	052	v	166	L 052	052
W	127	U 054	032	w	167	L 054	032
X	130	U 057	172	x	170	L 057	172
Y	131	U 061	106	y	171	L 061	106
Z	132	U 062	046	z	172	L 062	046
[	133	U 040	002	{	173		
\	134	U 166	067	|	174		
]	135			}	175		
↑	136	U 067	166	~	176		
←	137	U 100	001	DEL	177	L 177	177
`	140						

APPENDIX II

Sample Listings from Scenario Assembler

TEST

```
                    CONVERSION CODE = 1
                    END-OF-MESSAGE CODE =  1

 1       3    A    9
 2       3    L   FL3
 3       3    R   ''
 4       6    S   FL3 CANDE
 5      16    Q MITRE/EMULATE
 6      33    L   FL4
 7      33    R   ''
 8      36    S   FL4 LOGGED
 9      47    A    6
10      52    +    13   0   R8
11      58    5 FILES
12      66    ◊    R8
13      70    O
14      73    L   FL5
15      73    R   ''
16      76    S   FL5 #
17      82    Q REMOVE
18      92    L   FL7
19      92    R   ''
20      95    S   FL7 #
21     101    Q BYE
22     108    L   FL8
23     108    R   ''
24     111    S   FL8 ET
25     118    C [QUIT
```

```
TEST.IS              INDICATOR 0
                          CONVERSION CODE = 1
                          END-OF-MESSAGE CODE =  1

                     ALLOCATE 9

      1 A   9

      2 L   FL3

      3 R   ''

          3 '   0   3
          5 ' 122   0              R

      4 S   FL3 CANDE

          6 '   0  12
          8 ' 123   0              S
         10 '   3 103               C
         12 ' 101 116              AN
         14 ' 104 105              DE

      5 U MITRE/EMULATE

         16 '   0  21
         18 ' 121 115              QM
         20 ' 111 124              IT
         22 ' 122 105              RE
         24 '  57 105              /E
         26 ' 115 125              MU
         28 ' 114 101              LA
         30 ' 124 105              TE
         32 '  15   0

      6 L   FL4

      7 R   ''

         33 '   0   3
         35 ' 122   0              R

      8 S   FL4 LOGGED

         36 '   0  13
         38 ' 123   0              S
         40 '  41 114              !L
         42 ' 117 107              OG
         44 ' 107 105              GE
         46 ' 104   0              D

      9 A   6

         47 '   0   5
         49 ' 101   0              A
         51 '   6   0

     10 +   .13   0   R8
```

```
    52 !   0   6
    54 !  53  15          +
    56 !   0  10

11 5 FILES

    58 !   0  10
    60 !  65 106          5F
    62 ! 111 114          IL
    64 ! 105 123          ES

12 0   R8

    66 !   0   4
    68 ! 134  10          0

13 O

    70 !   0   3
    72 ! 117   0          O

14 L  FL5


15 R  ''

    73 !   0   3
    75 ! 122   0          R

16 S  FL5 #

    76 !   0   6
    78 ! 123   0          S
    80 ! 111  43          I#

17 Q REMOVE

    82 !   0  12
    84 ! 121 122          QR
    86 ! 105 115          EM
    88 ! 117 126          OV
    90 ! 105  15          E

18 L  FL7


19 R  ''

    92 !   0   3
    94 ! 122   0          R

20 S  FL7 #

    95 !   0   6
    97 ! 123   0          S
    99 ! 134  43          0#

21 Q BYE

   101 !   0   7
   103 ! 121 102          QB
   105 ! 131 105          YE
```

```
   107 '  15   0

22 L  FL8


23 R  ''

   108 '   0   3
   110 ' 122   0          R

24 S  FL8 ET

   111 '   0   7
   113 ' 123   0          S
   115 ' 154 105          LE
   117 ' 124   0          T

25 C [QUIT

   118 '   0  10
   120 ' 103 133          C[
   122 ' 121 125          QU
   124 ' 111 124          IT
```

SYMBOL TABLE
NUMBER OF ENTRIES 5

LENGTH	LABEL	IS ADDRESS	ES LINE NO.
3	FL3	3	2
3	FL4	33	6
3	FL5	73	14
3	FL7	92	18
3	FL8	108	22

APPENDIX III

Listing of EQUIP.RB

```
0001  EQUIP
                          .TITL EQUIP
                          .ENT E0000,E0,E1,ETREC
                          .ENT ETEND
                          .ENT ETENT
                          .ENT E2
                          .ENT ETLEN
     000101               .DUSR A=101
     000111               .DUSR I=111
     000123               .DUSR S=123
     000124               .DUSR T=124
     000125               .DUSR U=125
     000127               .DUSR W=127
     000105               .DUSR E=105
     000132               .DUSR Z=132
     000116               .DUSR N=116
     000117               .DUSR O=117
     000207               .DUSR RT1=135.
     000007               .DUSR BL1=7.
     000010               .DUSR BL2=6.
     000117               .DUSR PT1=O
     000116               .DUSR PT2=N
     000064               .DUSR DDDLINE=3*16.+4
     000064               .DUSR IBM2848=3*16.+4
     000064               .DUSR IBM2260=3*16.+4
     000005               .DUSR IBM1053=5
     000006               .DUSR D2000=6
     000007               .DUSR IBM2741=7
     000064               .DUSR I2741=3*16.+4
     000021               .DUSR ZASC1=1*16.+1
     000026               .DUSR ZASC6=1*16.+6
     000042               .DUSR EASC2=2*16.+2
     000045               .DUSR EASC5=2*16.+5
     000005               .TXTM 5
                          .ZREL
                          .NREL
 00000'000025 ETEND:      E0END-E0000
 00001'020105 ETREC:      20000+"E             ;USED TO WRITE ET ON TAPE
 00002'001046             E9999-E0000+4
 00003'000110             "H
 00004'000005'            .+1
              E0000:
              E0:
 00005'000000             0                    ;ETR0
 00006'041524             "C*256.+"T          ;ETYPE
 00007'000000             0.                   ;ETIO
 00010'000000             0                    ;CHILD
 00011'000032'            E1                    ;LINK
 00012'000000             0                    ;PARNT
 00013'000156             110.                             ;ETRAT
 00014'000000             0                     ;ETQBP
 00015'001057'            EDM2                    ;ETEOM
 00016'000000             0                     ;ETRSP
 00017'000000             0                     ;ETPAD
 00020'000000             0                     ;RRING,PRING
 00021'000037             0*256.+37             ;ETLGA, ETLGN
 00022'013111             ZASC6*256.+I            ;TERMT, STATI
 00023'004410             11*256.+10            ;PORTO, PORTI
 00024'000000             0*256.+0            ;SPRTO, SPRTI
 00025'000200             0*256.+186                   ;SUTAD, ETIND
```

```
0002  EQUIP
 00026'004132          8,*256,+Z             ;BYTEL, PARTY
 00027'000000          1,-1*1B2+0B7+0              ;ETDIO
 00030'000000          1,-1*1B2+0B7+0      ;ETDOD
 00031'000000          0                     ;ETDOA
               E0END:
               E1:
 00032'000000          0                     ;ETR0
 00033'042123          "D*256,+"S           ;ETYPE
 00034'000016          14.                   ;ETID
 00035'000000          0                     ;CHILD
 00036'000057'         E2                    ;LINK
 00037'000000          0                     ;PARNT
 00040'000156          0110.                               ;ETRAT
 00041'000000          0                     ;ETQBP
 00042'001065'         EOM3                    ;ETEOM
 00043'000000          0                     ;ETRSP
 00044'000000          0                     ;ETPAD
 00045'000000          0                     ;RRING,PRING
 00046'000037          0*256,+37             ;ETLGA, ETLGN
 00047'021111          EASC2*256,+I            ;TERMT, STATI
 00050'024450          51*256,+50            ;PORTO, PORTI
 00051'000401          1*256,+1             ;SPRTO, SPRTI
 00052'007400          15,*256,+0B8                  ;SUTAD, ETIND
 00053'004105          8,*256,+E             ;BYTEL, PARTY
 00054'000000          1,-1*1B2+0B7+0              ;ETDIO
 00055'000000          1,-1*1B2+0B7+0      ;ETDOD
 00056'000000          0                     ;ETDOA
               E2:
 00057'000000          0                     ;ETR0
 00060'052131          "T*256,+"Y           ;ETYPE
 00061'000001          1.                    ;ETID
 00062'000000          0                     ;CHILD
 00063'000104'         E3                    ;LINK
 00064'000000          0                     ;PARNT
 00065'000207          RT1                                ;ETRAT
 00066'000000          0                     ;ETQBP
 00067'001073'         EOM4                    ;ETEOM
 00070'000000          0                     ;ETRSP
 00071'000000          0                     ;ETPAD
 00072'000000          0                     ;RRING,PRING
 00073'000037          0*256,+37             ;ETLGA, ETLGN
 00074'032111          I2741*256,+I            ;TERMT, STATI
 00075'021442          43*256,+42            ;PORTO, PORTI
 00076'000401          1*256,+1             ;SPRTO, SPRTI
 00077'017000          30,*256,+0B8                  ;SUTAD, ETIND
 00100'003517          BL1*256,+PT1            ;BYTEL, PARTY
 00101'020071          2,-1*1B2+00,B7+71            ;ETDID
 00102'060066          4,-1*1B2+00,B7+66     ;ETDOD
 00103'001107'         DO66A                     ;ETDOA
               E3:
 00104'000000          0                     ;ETR0
 00105'052131          "T*256,+"Y           ;ETYPE
 00106'000002          2.                    ;ETID
 00107'000000          0                     ;CHILD
 00110'000131'         E4                    ;LINK
 00111'000000          0                     ;PARNT
 00112'000207          RT1                                ;ETRAT
 00113'000000          0                     ;ETQBP
 00114'001101'         EOM5                    ;ETEOM
```

```
0003  EQUIP
00115'000000        0                      ;ETRSP
00116'000000        0                      ;ETPAD
00117'000000        0                      ;RRING,PRING
00120'000037        0*256,+37              ;ETLGA, ETLGN
00121'032111        I2741*256,+I             ;TERMT, STATI
00122'021442        43*256,+42             ;PORTO, PORTI
00123'001002        2*256,+2             ;SPRTO, SPRTI
00124'017400        31,*256,+0B8                     ;SUTAD, ETIND
00125'003517        BL1*256,+PT1             ;BYTEL, PARTY
00126'021071        2,-1*1B2+02,B7+71                ;ETDIO
00127'062066        4,-1*1B2+04,B7+66     ;ETDOO
00130'001107'       DO66A                      ;ETDOA
          E4:
00131'000000        0                     ;ETR0
00132'052131        "T*256,+"Y          ;ETYPE
00133'000003        3,                    ;ETID
00134'000000        0                     ;CHILO
00135'000156'       E4A                     ;LINK
00136'000000        0                     ;PARNT
00137'000207        RT1                              ;ETRAT
00140'000000        0                      ;ETQBP
00141'001051'       EOM1                   ;ETEOM
00142'000000        0                      ;ETRSP
00143'000000        0                      ;ETPAD
00144'000000        0                      ;RRING,PRING
00145'000037        0*256,+37              ;ETLGA, ETLGN
00146'032111        I2741*256,+I             ;TERMT, STATI
00147'021442        43*256,+42             ;PORTO, PORTI
00150'001403        3*256,+3             ;SPRTO, SPRTI
00151'020000        32,*256,+0B8                     ;SUTAD, ETIND
00152'003517        BL1*256,+PT1             ;BYTEL, PARTY
00153'022071        2,-1*1B2+04,B7+71                ;ETDIO
00154'064066        4,-1*1B2+08,B7+66     ;ETDOO
00155'001107'       DO66A                      ;ETDOA
          E4A:
00156'000000        0                     ;ETR0
00157'052131        "T*256,+"Y          ;ETYPE
00160'000004        4,                    ;ETIO
00161'000000        0                     ;CHILO
00162'000203'       E13                     ;LINK
00163'000000        0                     ;PARNT
00164'000207        RT1                              ;ETRAT
00165'000000        0                      ;ETQBP
00166'001051'       EOM1                   ;ETEOM
00167'000000        0                      ;ETRSP
00170'000000        0                      ;ETPAD
00171'000000        0                      ;RRING,PRING
00172'000037        0*256,+37              ;ETLGA, ETLGN
00173'032111        I2741*256,+I             ;TERMT, STATI
00174'021442        43*256,+42             ;PORTO, PORTI
00175'002004        4*256,+4             ;SPRTO, SPRTI
00176'016400        29,*256,+0B8                     ;SUTAD, ETIND
00177'003517        BL1*256,+PT1             ;BYTEL, PARTY
00200'023071        2,-1*1B2+06,B7+71                ;ETDIO
00201'066066        4,-1*1B2+12,B7+66     ;ETDOO
00202'001107'       DO66A                      ;ETDOA
          E13:
00203'000000        0                     ;ETR0
00204'052131        "T*256,+"Y          ;ETYPE
```

```
0004  EQUIP
 00205'000005          5.                  ;ETIO
 00206'000000          0                   ;CHILD
 00207'000230'         E14                   ;LINK
 00210'000000          0                   ;PARNT
 00211'000207          RT1                                ;ETRAT
 00212'000000          0                    ;ETQBP
 00213'001051'         EOM1                 ;ETEOM
 00214'000000          0                    ;ETRSP
 00215'000000          0                    ;ETPAD
 00216'000000          0                    ;RRING,PRING
 00217'000037          0*256,+37            ;ETLGA, ETLGN
 00220'032111          I2741*256,+I           ;TERMT, STATI
 00221'021442          43*256,+42           ;PORTO, PORTI
 00222'002405          5*256,+5           ;SPRTO, SPRTI
 00223'020400          33,*256,+0B8                    ;SUTAD, ETINO
 00224'003517          BL1*256,+PT1           ;BYTEL, PARTY
 00225'024071          2,-1*1B2+08,B7+71               ;ETDIO
 00226'070066          4,-1*1B2+16,B7+66    ;ETOOD
 00227'001110'         DO66B                    ;ETDOA
              E14:
 00230'000000          0                   ;ETR0
 00231'052131          "T*256,+"Y         ;ETYPE
 00232'000006          6,                  ;ETIO
 00233'000000          0                   ;CHILD
 00234'000255'         E15                   ;LINK
 00235'000000          0                   ;PARNT
 00236'000207          RT1                                ;ETRAT
 00237'000000          0                    ;ETQBP
 00240'001051'         EOM1                 ;ETEOM
 00241'000000          0                    ;ETRSP
 00242'000000          0                    ;ETPAD
 00243'000000          0                    ;RRING,PRING
 00244'000037          0*256,+37            ;ETLGA, ETLGN
 00245'032111          I2741*256.+I           ;TERMT, STATI
 00246'021442          43*256,+42           ;PORTO, PORTI
 00247'003006          6*256,+6           ;SPRTO, SPRTI
 00250'021000          34,*256,+0B8                    ;SUTAD, ETINO
 00251'003517          BL1*256,+PT1           ;BYTEL, PARTY
 00252'025071          2,-1*1B2+10,B7+71               ;ETOID
 00253'072066          4,-1*1B2+20,B7+66    ;ETOOD
 00254'001110'         DO66B                    ;ETDOA
              E15:
 00255'000000          0                   ;ETR0
 00256'052131          "T*256,+"Y         ;ETYPE
 00257'000007          7,                  ;ETIO
 00260'000000          0                   ;CHILD
 00261'000302'         E16                   ;LINK
 00262'000000          0                   ;PARNT
 00263'000207          RT1                                ;ETRAT
 00264'000000          0                    ;ETQBP
 00265'001051'         EOM1                 ;ETEOM
 00266'000000          0                    ;ETRSP
 00267'000000          0                    ;ETPAD
 00270'000000          0                    ;RRING,PRING
 00271'000037          0*256,+37            ;ETLGA, ETLGN
 00272'032111          I2741*256,+I           ;TERMT, STATI
 00273'021442          43*256,+42           ;PORTO, PORTI
 00274'003407          7*256,+7           ;SPRTO, SPRTI
 00275'021400          35,*256,+0B8                    ;SUTAD, ETIND
```

```
0005  EQUIP
 00276'003517          BL1*256,+PT1          ;BYTEL, PARTY
 00277'026071          2,-1*1B2+12,B7+71          ;ETDID
 00300'074066          4,-1*1B2+24,B7+66     ;ETDOD
 00301'001110'         DD66B                  ;ETDDA
              E16:
 00302'000000          0                     ;ETR0
 00303'052131          "T*256,+"Y           ;ETYPE
 00304'000010          8,                    ;ETID
 00305'000000          0                     ;CHILD
 00306'000327'         E17                     ;LINK
 00307'000000          0                     ;PARNT
 00310'000207          RT1                              ;ETRAT
 00311'000000          0                      ;ETQBP
 00312'001051'         EDM1                   ;ETEDM
 00313'000000          0                      ;ETRSP
 00314'000000          0                      ;ETPAD
 00315'000000          0                      ;RRING,PRING
 00316'000037          0*256,+37              ;ETLGA, ETLGN
 00317'032111          I2741*256,+I             ;TERMT, STATI
 00320'021442          43*256,+42             ;PDRTO, PDRTI
 00321'004010          8,*256,+8,             ;SPRTO, SPRTI
 00322'022000          36,*256,+0BB                   ;SUTAD, ETIND
 00323'003517          BL1*256,+PT1             ;BYTEL, PARTY
 00324'027071          2,-1*1B2+14,B7+71             ;ETDID
 00325'076066          4,-1*1B2+28,B7+66      ;ETDOD
 00326'001110'         DD66B                    ;ETDDA
              E17:
 00327'000000          0                      ;ETR0
 00330'052131          "T*256,+"Y            ;ETYPE
 00331'000011          9,                     ;ETID
 00332'000000          0                      ;CHILD
 00333'000354'         E18                      ;LINK
 00334'000000          0                      ;PARNT
 00335'000207          RT1                               ;ETRAT
 00336'000000          0                       ;ETQBP
 00337'001051'         EOM1                    ;ETEOM
 00340'000000          0                       ;ETRSP
 00341'000000          0                       ;ETPAD
 00342'000000          0                       ;RRING,PRING
 00343'000037          0*256,+37               ;ETLGA, ETLGN
 00344'032111          I2741*256,+I              ;TERMT, STATI
 00345'022444          45*256,+44              ;PDRTO, PDRTI
 00346'000401          1*256,+1              ;SPRTD, SPRTI
 00347'022400          37,*256,+0BB                   ;SUTAD, ETIND
 00350'003517          BL1*256,+PT1              ;BYTEL, PARTY
 00351'030071          2,-1*1B2+16,B7+71              ;ETDID
 00352'060067          4,-1*1B2+00,B7+67       ;ETDDD
 00353'001111'         DD67A                     ;ETDDA
              E18:
 00354'000000          0                       ;ETR0
 00355'052131          "T*256,+"Y             ;ETYPE
 00356'000012          10,                       ;ETID
 00357'000000          0                       ;CHILD
 00360'000401'         E19                       ;LINK
 00361'000000          0                       ;PARNT
 00362'000207          RT1                                ;ETRAT
 00363'000000          0                        ;ETQBP
 00364'001051'         EOM1                     ;ETEDM
 00365'000000          0                        ;ETRSP
```

```
0006  EQUIP
 00366'000000          0                        ;ETPAD
 00367'000000          0                        ;RRING,PRING
 00370'000037          0*256,+37                ;ETLGA, ETLGN
 00371'032111          I2741*256,+I               ;TERMT, STATI
 00372'022444          45*256,+44               ;PORTO, PORTI
 00373'001002          2*256,+2                ;SPRTO, SPRTI
 00374'023000          38,*256,+0B8                      ;SUTAD, ETIND
 00375'003517          BL1*256,+PT1                 ;BYTEL, PARTY
 00376'031071          2,-1*1B2+18,B7+71                 ;ETOID
 00377'062067          4,-1*1B2+04,B7+67        ;ETODD
 00400'001111'         DD67A                       ;ETDDA
                 E19:
 00401'000000          0                       ;ETR0
 00402'052131          "T*256,+"Y              ;ETYPE
 00403'000013          11.                       ;ETID
 00404'000000          0                       ;CHILD
 00405'000426'         E20                        ;LINK
 00406'000000          0                       ;PARNT
 00407'000207          RT1                                ;ETRAT
 00410'000000          0                        ;ETQBP
 00411'001051'         EDM1                     ;ETEOM
 00412'000000          0                        ;ETRSP
 00413'000000          0                        ;ETPAD
 00414'000000          0                        ;RRING,PRING
 00415'000037          0*256,+37                ;ETLGA, ETLGN
 00416'032111          I2741*256,+I               ;TERMT, STATI
 00417'022444          45*256,+44               ;PORTO, PORTI
 00420'001403          3*256,+3                ;SPRTO, SPRTI
 00421'023400          39,*256,+0B8                      ;SUTAD, ETIND
 00422'003517          BL1*256,+PT1                 ;BYTEL, PARTY
 00423'032071          2,-1*1B2+20,B7+71                 ;ETOID
 00424'064067          4,-1*1B2+08,B7+67        ;ETDDD
 00425'001111'         DD67A                       ;ETDDA
                 E20:
 00426'000000          0                       ;ETR0
 00427'052131          "T*256,+"Y              ;ETYPE
 00430'000014          12.                       ;ETID
 00431'000000          0                       ;CHILD
 00432'000453'         E21                        ;LINK
 00433'000000          0                       ;PARNT
 00434'000207          RT1                                ;ETRAT
 00435'000000          0                        ;ETQBP
 00436'001051'         EDM1                     ;ETEOM
 00437'000000          0                        ;ETRSP
 00440'000000          0                        ;ETPAD
 00441'000000          0                        ;RRING,PRING
 00442'000037          0*256,+37                ;ETLGA, ETLGN
 00443'032111          I2741*256,+I               ;TERMT, STATI
 00444'022444          45*256,+44               ;PORTO, PORTI
 00445'002004          4*256,+4                ;SPRTO, SPRTI
 00446'024000          40,*256,+0B8                      ;SUTAD, ETIND
 00447'003517          BL1*256,+PT1                 ;BYTEL, PARTY
 00450'033071          2,-1*1B2+22,B7+71                 ;ETOID
 00451'066067          4,-1*1B2+12,B7+67        ;ETDDD
 00452'001111'         DD67A                       ;ETDDA
                 E21:
 00453'000000          0                       ;ETR0
 00454'052131          "T*256,+"Y              ;ETYPE
 00455'000015          13.                       ;ETID
```

```
0007  EQUIP
 00456'000000          0                    ;CHILD
 00457'000500'         E22                    ;LINK
 00460'000000          0                    ;PARNT
 00461'000207          RT1                                  ;ETRAT
 00462'000000          0                     ;ETQBP
 00463'001051'         EOM1                  ;ETEOM
 00464'000000          0                     ;ETRSP
 00465'000000          0                     ;ETPAD
 00466'000000          0                     ;RRING,PRING
 00467'000037          0*256,+37             ;ETLGA, ETLGN
 00470'032111          I2741*256,+I            ;TERMT, STATI
 00471'022444          45*256,+44            ;PORTO, PORTI
 00472'002405          5*256,+5            ;SPRTO, SPRTI
 00473'024400          41,*256,+0B8                       ;SUTAO, ETIND
 00474'003517          BL1*256,+PT1              ;BYTEL, PARTY
 00475'034071          2,-1*1B2+24,B7+71                  ;ETDID
 00476'070067          4,-1*1B2+16,B7+67     ;ETDDD
 00477'001112'         DO67B                    ;ETODA
                E22:
 00500'000000          0                    ;ETR0
 00501'052131          "T*256,+"Y          ;ETYPE
 00502'000016          14,                    ;ETID
 00503'000000          0                    ;CHILD
 00504'000525'         E23                    ;LINK
 00505'000000          0                    ;PARNT
 00506'000207          RT1                                  ;ETRAT
 00507'000000          0                     ;ETQBP
 00510'001051'         EOM1                  ;ETEOM
 00511'000000          0                     ;ETRSP
 00512'000000          0                     ;ETPAD
 00513'000000          0                     ;RRING,PRING
 00514'000037          0*256,+37             ;ETLGA, ETLGN
 00515'032111          I2741*256,+I            ;TERMT, STATI
 00516'022444          45*256,+44            ;PORTD, PORTI
 00517'003006          6*256,+6            ;SPRTD, SPRTI
 00520'025000          42,*256,+0B8                       ;SUTAD, ETIND
 00521'003517          BL1*256,+PT1              ;BYTEL, PARTY
 00522'035071          2,-1*1B2+26,B7+71                  ;ETOID
 00523'072067          4,-1*1B2+20,B7+67     ;ETDOD
 00524'001112'         DO67B                    ;ETDDA
                E23:
 00525'000000          0                    ;ETR0
 00526'052131          "T*256,+"Y          ;ETYPE
 00527'000017          15,                    ;ETID
 00530'000000          0                    ;CHILD
 00531'000552'         E24                    ;LINK
 00532'000000          0                    ;PARNT
 00533'000207          RT1                                  ;ETRAT
 00534'000000          0                     ;ETQBP
 00535'001051'         EOM1                  ;ETEOM
 00536'000000          0                     ;ETRSP
 00537'000000          0                     ;ETPAD
 00540'000000          0                     ;RRING,PRING
 00541'000037          0*256,+37             ;ETLGA, ETLGN
 00542'032111          I2741*256,+I            ;TERMT, STATI
 00543'022444          45*256,+44            ;PORTD, PORTI
 00544'003407          7*256,+7            ;SPRTD, SPRTI
 00545'025400          43,*256,+0B8                       ;SUTAO, ETIND
 00546'003517          BL1*256,+PT1              ;BYTEL, PARTY
```

```
0008  EQUIP
 00547'036071          2,-1*1B2+28,B7+71             ;ETDID
 00550'074067          4,-1*1B2+24,B7+67    ;ETDOD
 00551'001112'         DD678                   ;ETDDA
             E24:
 00552'000000          0                    ;ETR0
 00553'052131          "T*256,+"Y          ;ETYPE
 00554'000020          16.                    ;ETID
 00555'000000          0                     ;CHILD
 00556'000577'         E5                     ;LINK
 00557'000000          0                     ;PARNT
 00560'000207          RT1                              ;ETRAT
 00561'000000          0                      ;ETQBP
 00562'001051'         EOM1                   ;ETEOM
 00563'000000          0                      ;ETRSP
 00564'000000          0                      ;ETPAD
 00565'000000          0                      ;RRING,PRING
 00566'000037          0*256,+37              ;ETLGA, ETLGN
 00567'032111          I2741*256,+I             ;TERMT, STATI
 00570'022444          45*256,+44             ;PDRTO, PORTI
 00571'004010          8,*256,+8,             ;SPRTD, SPRTI
 00572'026000          44,*256,+0B8                     ;SUTAD, ETIND
 00573'003517          BL1*256,+PT1              ;BYTEL, PARTY
 00574'037071          2,-1*1B2+30,B7+71             ;ETDID
 00575'076067          4,-1*1B2+28,B7+67    ;ETDOD
 00576'001112'         DD678                   ;ETDDA
             E5:
 00577'000000          0                     ;ETR0
 00600'046116          "L*256,+"N          ;ETYPE
 00601'000005          5.                    ;ETID
 00602'000624'         E6                    ;CHILD
 00603'000000          0                    ;LINK
 00604'000000          0                    ;PARNT
 00605'004540          2400.                              ;ETRAT
 00606'000000          0                      ;ETQBP
 00607'001051'         EOM1                   ;ETEOM
 00610'000000          0                      ;ETRSP
 00611'000000          0                      ;ETPAD
 00612'000000          0                      ;RRING,PRING
 00613'000037          0*256,+37              ;ETLGA, ETLGN
 00614'032111          DDDLINE*256,+I            ;TERMT, STATI
 00615'015031          32*256,+31           ;PDRTD, PDRTI
 00616'000000          0*256,+0            ;SPRTD, SPRTI
 00617'025400          43,*256,+0B8                     ;SUTAD, ETIND
 00620'004116          BL2*256,+PT2              ;BYTEL, PARTY
 00621'000000          1,-1*1B2+0B7+0             ;ETDID
 00622'000000          1,-1*1B2+0B7+0    ;ETDOD
 00623'000000          0                    ;ETDDA
             E6:
 00624'000000          0                     ;ETR0
 00625'041516          "C*256,+"N          ;ETYPE
 00626'000006          6.                    ;ETID
 00627'000676'         E8                     ;CHILD
 00630'000651'         E7                     ;LINK
 00631'000577'         E5                     ;PARNT
 00632'004540          2400.                              ;ETRAT
 00633'000000          0                      ;ETQBP
 00634'001051'         EOM1                   ;ETEOM
 00635'000000          0                      ;ETRSP
 00636'000000          0                      ;ETPAD
```

```
0009  EQUIP
 00637'000000        0                    ;RRING,PRING
 00640'000037        0*256.+37            ;ETLGA, ETLGN
 00641'032111        IBM2848*256.+I          ;TERMT, STATI
 00642'015031        32*256.+31           ;PORTO, PORTI
 00643'000000        0*256.+0            ;SPRTO, SPRTI
 00644'047000        116*256.+0B8                 ;SUTAO, ETIND
 00645'004116        BL2*256.+PT2           ;BYTEL, PARTY
 00646'000000        1,-1*1B2+0B7+0            ;ETOIO
 00647'000000        1,-1*1B2+0B7+0    ;ETODO
 00650'000000        0                   ;ETDOA
              E7:
 00651'000000        0                   ;ETR0
 00652'041516        "C*256.+"N         ;ETYPE
 00653'000007        7.                  ;ETID
 00654'000775'       E11                   ;CHILD
 00655'000000        0                   ;LINK
 00656'000577'       E5                   ;PARNT
 00657'004540        2400.                             ;ETRAT
 00660'000000        0                    ;ETQBP
 00661'001051'       EDM1                 ;ETEOM
 00662'000000        0                    ;ETRSP
 00663'000000        0                    ;ETPAD
 00664'000000        0                    ;RRING,PRING
 00665'000037        0*256.+37            ;ETLGA, ETLGN
 00666'032125        IBM2848*256.+U          ;TERMT, STATI
 00667'015031        32*256.+31           ;PORTO, PORTI
 00670'000000        0*256.+0            ;SPRTO, SPRTI
 00671'124000        250*256.+0B8                 ;SUTAO, ETIND
 00672'004116        BL2*256.+PT2           ;BYTEL, PARTY
 00673'000000        1,-1*1B2+0B7+0            ;ETDID
 00674'000000        1,-1*1B2+0B7+0    ;ETDDO
 00675'000000        0                   ;ETDDA
              E8:
 00676'000000        0                   ;ETR0
 00677'042123        "D*256.+"S         ;ETYPE
 00700'000010        8.                  ;ETID
 00701'000000        0                   ;CHILD
 00702'000723'       E9                    ;LINK
 00703'000624'       E6                    ;PARNT
 00704'004540        2400.                             ;ETRAT
 00705'000000        0                    ;ETQBP
 00706'001051'       EDM1                 ;ETEOM
 00707'000000        0                    ;ETRSP
 00710'000000        0                    ;ETPAD
 00711'000000        0                    ;RRING,PRING
 00712'000037        0*256.+37            ;ETLGA, ETLGN
 00713'032111        IBM2260*256.+I          ;TERMT, STATI
 00714'015031        32*256.+31           ;PORTO, PORTI
 00715'000000        0*256.+0            ;SPRTO, SPRTI
 00716'120000        240*256.+0B8                 ;SUTAD, ETIND
 00717'004116        BL2*256.+PT2           ;BYTEL, PARTY
 00720'000000        1,-1*1B2+0B7+0            ;ETOID
 00721'000000        1,-1*1B2+0B7+0    ;ETDOD
 00722'000000        0                   ;ETOOA
              E9:
 00723'000000        0                   ;ETR0
 00724'042123        "D*256.+"S         ;ETYPE
 00725'000011        9.                  ;ETIO
 00726'000000        0                   ;CHILD
```

```
0010  EQUIP
 00727'000750'          E10                 ;LINK
 00730'000624'          E6                  ;PARNT
 00731'004540           2400.                           ;ETRAT
 00732'000000           0                   ;ETQBP
 00733'001051'          EDM1                ;ETEOM
 00734'000000           0                   ;ETRSP
 00735'000000           0                   ;ETPAD
 00736'000000           0                   ;RRING,PRING
 00737'000037           0*256.+37           ;ETLGA, ETLGN
 00740'032111           IBM2260*256.+I          ;TERMT, STATI
 00741'015031           32*256.+31          ;PORTO, PORTI
 00742'000000           0*256.+0           ;SPRTO, SPRTI
 00743'120400           241*256.+0B8                ;SUTAD, ETIND
 00744'004116           BL2*256.+PT2          ;BYTEL, PARTY
 00745'000000           1.-1*1B2+0B7+0            ;ETDID
 00746'000000           1.-1*1B2+0B7+0     ;ETDOD
 00747'000000           0                   ;ETDOA
                 E10:
 00750'000000           0                   ;ETR0
 00751'050124           "P*256.+"T         ;ETYPE
 00752'000012           10.                  ;ETID
 00753'000000           0                   ;CHILD
 00754'000000           0                   ;LINK
 00755'000624'          E6                   ;PARNT
 00756'000226           150.                            ;ETRAT
 00757'000000           0                    ;ETQBP
 00760'001051'          EDM1                 ;ETEOM
 00761'000000           0                    ;ETRSP
 00762'000000           0                    ;ETPAD
 00763'000000           0                    ;RRING,PRING
 00764'000037           0*256.+37            ;ETLGA, ETLGN
 00765'002525           IBM1053*256.+U           ;TERMT, STATI
 00766'015000           32*256.+0           ;PORTO, PORTI
 00767'000000           0*256.+0           ;SPRTO, SPRTI
 00770'121000           242*256.+0B8                ;SUTAD, ETIND
 00771'004116           BL2*256.+PT2          ;BYTEL, PARTY
 00772'000000           1.-1*1B2+0B7+0            ;ETDID
 00773'000000           1.-1*1B2+0B7+0     ;ETDOD
 00774'000000           0                   ;ETDOA
                 E11:
 00775'000000           0                   ;ETR0
 00776'042123           "D*256.+"S         ;ETYPE
 00777'000013           11.                  ;ETID
 01000'000000           0                   ;CHILD
 01001'001022'          E12                  ;LINK
 01002'000651'          E7                   ;PARNT
 01003'004540           2400.                           ;ETRAT
 01004'000000           0                    ;ETQBP
 01005'001051'          EDM1                 ;ETEOM
 01006'000000           0                    ;ETRSP
 01007'000000           0                    ;ETPAD
 01010'000000           0                    ;RRING,PRING
 01011'000037           0*256.+37            ;ETLGA, ETLGN
 01012'032125           IBM2260*256.+U           ;TERMT, STATI
 01013'015031           32*256.+31          ;PORTO, PORTI
 01014'000000           0*256.+0           ;SPRTO, SPRTI
 01015'122000           244*256.+0B8                ;SUTAD, ETIND
 01016'004116           BL2*256.+PT2          ;BYTEL, PARTY
 01017'000000           1.-1*1B2+0B7+0            ;ETDID
```

```
0011  EQUIP
 01020'000000            1,-1*1B2+0B7+0    ;ETDOD
 01021'000000            0                   ;ETDOA
               E12:
 01022'000000            0                   ;ETR0
 01023'042123            "D*256,+"S         ;ETYPE
 01024'000014            12.                  ;ETID
 01025'000000            0                   ;CHILD
 01026'000000            0                   ;LINK
 01027'000651'           E7                   ;PARNT
 01030'004540            2400.                              ;ETRAT
 01031'000000            0                    ;ETQBP
 01032'001051'           EOM1                 ;ETEOM
 01033'000000            0                    ;ETRSP
 01034'000000            0                    ;ETPAD
 01035'000000            0                    ;RRING,PRING
 01036'000037            0*256,+37            ;ETLGA, ETLGN
 01037'032125            IBM2260*256,+U          ;TERMT, STATI
 01040'015031            32*256,+31          ;PORTO, PORTI
 01041'000000            0*256,+0          ;SPRTO, SPRTI
 01042'122400            245*256,+0B8                 ;SUTAD, ETIND
 01043'004116            BL2*256,+PT2           ;BYTEL, PARTY
 01044'000000            1,-1*1B2+0B7+0             ;ETDID
 01045'000000            1,-1*1B2+0B7+0    ;ETDOD
 01046'000000            0                   ;ETDOA
               E9999:
        000025 LEN       =E0END-E0000
 01047'000032 ETENT:     E9999-E0000/LEN
 01050'000025 ETLEN:     LEN
 01051'000037 EOM1:      37
 01052'177777            -1
 01053'177777            -1
 01054'177777            -1
 01055'177777            -1
 01056'177777            -1
 01057'000012 EOM2:      12
 01060'000005            5
 01061'000030            30
 01062'177777            -1
 01063'177777            -1
 01064'177777            -1
 01065'177777 EOM3:      -1
 01066'177777            -1
 01067'177777            -1
 01070'177777            -1
 01071'177777            -1
 01072'177777            -1
 01073'000037 EOM4:      37
 01074'177777            -1
 01075'177777            -1
 01076'177777            -1
 01077'177777            -1
 01100'177777            -1
 01101'000037 EOM5:      37
 01102'000043            43
 01103'177777            -1
 01104'177777            -1
 01105'177777            -1
 01106'177777            -1
 01107'000000 DO66A:     0
```

```
0012  EQUIP
 01110'000000 D066B:   0
 01111'000000 D067A:   0
 01112'000000 D067B:   0
                       .END
```

```
  0013  EQUIP
0066A   001107'
0066B   001110'
0067A   001111'
0067B   001112'
E0      000005'
E0000   000005'
E0END   000032'
E1      000032'
E10     000750'
E11     000775'
E12     001022'
E13     000203'
E14     000230'
E15     000255'
E16     000302'
E17     000327'
E18     000354'
E19     000401'
E2      000057'
E20     000426'
E21     000453'
E22     000500'
E23     000525'
E24     000552'
E3      000104'
E4      000131'
E4A     000156'
E5      000577'
E6      000624'
E7      000651'
E8      000676'
E9      000723'
E9999   001047'
EOM1    001051'
EOM2    001057'
EOM3    001065'
EOM4    001073'
EOM5    001101'
ETEND   000000'
ETENT   001047'
ETLEN   001050'
ETREC   000001'
LEN     000025
```

APPENDIX IV

DATAR Listings

```
                         USER INPUT:     DATAR/B)

BRIEF SUMMARY OF RUN FT7                     PAGE   1

DEV  ERROR MESSAGE:
---------------------

CT0  020 ACTION TAKEN
TY15 020 ACTION TAKEN
TY10 114 DEVICE STOPPED
TY15 020 ACTION TAKEN
CT0  020 ACTION TAKEN
TY10 020 ACTION TAKEN
TY10 020 ACTION TAKEN
TY10 040 BEHIND SCHEDULE

-------------------------------------------------------------------------

BRIEF SUMMARY OF RUN FT7                     PAGE   2

TERMINALS:    3

ELAPSED TIME:  29.79358     MESSAGES:   135    RECORDS:    14

CHARACTERS:   TOTAL:   172    R:    86    Q:    86    UN-R:      0

RECORD TYPES:   M:    3    S:  110    Q:    5
                R:    5    C:    7    E:    0    UNSOLICITED:    0

TIMES:  AVG RESP:  6,06     MAX RESP: 30,62     TERMINAL-MAX: TY10
        PERCENT CPU:  1,89     TOTAL CPU:     56569

SCENARIO INSTRUCTIONS USED:

& :     4   B :     2   9 :    23   I :     3   ! :     4
* :     1   C :     4   D :    22   J :    21   Q :     2
R :     6   W :     1   D*:     8   L*:     2   Q*:     3
R*:     4

COMMANDS ISSUED:

QUIT        :     1   START      :     3   SUB        :     3
END-OF-FILE
```

Figure 26. Brief Summary Output Format

```
                    USER INPUT:     DATAR/D/S

DEVICES REQUESTED ARE :                               PAGE    1

  1  CT4
  3  TY1
  5  TY3
 17  TY15
 18  TY16
 19  [illegible]
------------------------------------------------------------------

  DETAILED SUMMARY OF RUN FT7              PAGE   2

  TERMINAL IDENTIFICATION:   CT4

  RECORD TYPES:   H:    3     S:    [illegible]   O:    0
                  R:    0     C:    3     E:    2    UNSOLICITED:    0

  TIMES:  AVG RESP:  [illegible]     MAX RESP:  [illegible]

  COMMANDS ISSUED:

  QUIT       1      1    START      1      2
------------------------------------------------------------------

  DETAILED SUMMARY OF RUN FT7              PAGE   3

  TERMINAL IDENTIFICATION:   TY15

  RECORD TYPES:   H:    3     S:   76     O:    2
                  R:    3     C:    2     E:    2    UNSOLICITED:    0

  TIMES:  AVG RESP: 1[illegible].90    MAX RESP: 30.02

  SCENARIO INSTRUCTIONS USED:

  A :    4   9 :    17   T :     1   J :     1   * :     1
  C :    2   U :    2[illegible]  J :    15   O :     2   # :     3
  D*:    1   L*:     2   Q*:     3   N*:     4

  COMMANDS ISSUED:

  START      1      1    SUB        1      1
------------------------------------------------------------------

  DETAILED SUMMARY OF RUN FT7              PAGE   4

  TERMINAL IDENTIFICATION:   TY16

  RECORD TYPES:   H:    3     S:   34     O:    3
                  R:    2     C:    4     E:    4    UNSOLICITED:    0

  TIMES:  AVG RESP:  [illegible].17    MAX RESP:  [illegible].17

  SCENARIO INSTRUCTIONS USED:

  B :    2   W :    [illegible]   I :     2   J :     3   C :     2
  U :    2   J :     6   N :     3   * :     1   O*:     7

  COMMANDS ISSUED:

  SUB        1      2
------------------------------------------------------------------

  FOLLOWING DEVICES ARE INACTIVE:                     PAGE    5

   3  TY1
   5  TY3
  19  [illegible]
  END-OF-FILE
```

Figure 27. Detailed Summary Output Format

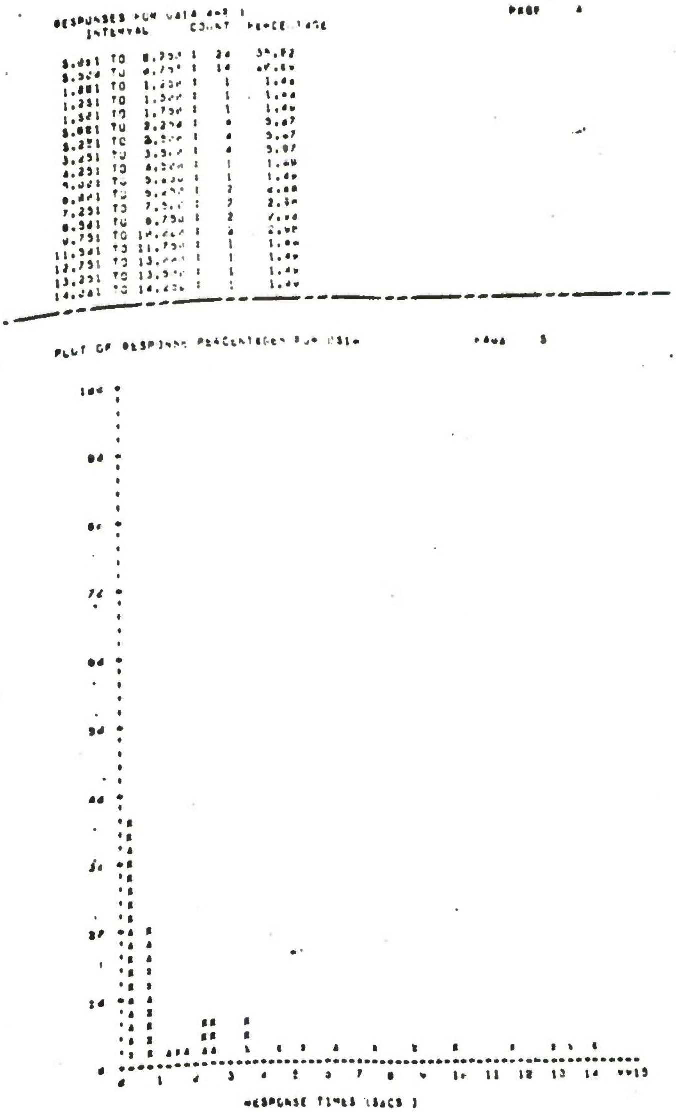


Figure 28. Histogram Output Format

```
                    USER INPUT:     DATAR/L/T/O/N

DEVICES REQUESTED ARE :                                  PAGE    1

1  CT0
2  DS14
3  TY15
4  TY16

LOGICAL RECORDS 101 TO 110

TERMINAL:  CT0
0                                  (None for CT0)

LOGICAL RECORDS 101 TO 110

TERMINAL:  TY15

103
    242 002 001 027 000 122 000 056 272 144 000 000 071 300 000 000
    072 037 000 000 072 300 000 240 020 200 052 263 055 070 000 230
    020 034 121 051 153 122 001 001 174 220 013 133 037 000
104
    040 001 001 022 000 123 200 000 020 000 000 000 000 300 000 000
    073 057 000 000 074 031 000 000 000 000 052 263 055 270 000 230
    000 004 162 011
105
    040 001 001 024 202 123 000 000 002 000 000 000 000 000 000 002
    074 122 000 050 075 151 000 000 000 000 052 263 055 070 000 242
    000 007 040 000 003 011 012 000
106
    040 001 000 023 000 123 000 000 000 000 000 002 000 000 000 000
    075 244 000 000 076 330 000 000 000 000 052 263 055 070 002 251
    000 005 104 200 012 000
107
    044 004 000 033 000 121 000 000 100 053 000 002 277 363 000 000
    077 260 000 002 300 104 000 000 000 020 052 263 255 270 000 250
    020 034 045 040 142 112 150 142 122 150 001 122 153 045 117 114
    112 045 153 133 037 000
100
    040 001 000 034 000 123 000 000 200 000 000 000 020 000 000 000
    077 032 000 062 301 105 000 000 000 000 052 263 055 070 000 250
    000 032 121 020 034 045 040 142 112 150 142 122 150 001 122 153
    045 117 114 112 045 153 133 037P

LOGICAL RECORDS 101 TO 110

TERMINAL:  TY16

101
    044 004 000 027 000 121 000 056 233 343 000 000 033 335 000 050
    233 104 000 060 334 127 000 000 000 000 052 310 055 027 000 010
    020 034 121 051 153 122 001 001 174 020 013 133 037 000
102
    040 001 000 030 000 123 000 000 000 000 000 000 000 000 000 050
    232 250 000 000 035 023 200 000 000 000 052 310 055 027 000 010
    000 020 122 020 034 121 051 153 122 001 001 174 020 013 133 037P
100
    242 002 000 033 000 122 000 000 136 311 000 002 330 007 000 002
    330 000 000 002 337 145 000 000 000 000 052 310 055 027 000 030
    020 034 045 040 142 112 150 142 122 150 001 122 153 045 117 114
    112 045 153 133 037 000
110
    040 001 002 023 000 123 000 000 000 000 000 000 000 000 000 062
    330 303 000 002 340 070 000 000 000 200 052 310 055 027 000 030
    000 005 070 001 000 000
--------------------------------------------------------------------
FOLLOWING DEVICES ARE INACTIVE:                        PAGE    2

2  DS14

END-OF-FILE
```

Figure 29. Octal Tape Output Format

USER INPUT: DATAR/L RECORDS/Q/R STTO

```
TIME-ACTUAL LIST OF 6-14 4:30PM  MESSAGE    36  PAGE   1

REC  TRANS      TASK       TASK      SCEN  SCENARIO    DEV
TYP. START      START      END       ADDR  NAME        ID    TEXT
------------------------------------------------------------------------

O    6056430    6156291    6063437    12 FW2         TY1   #X
R    6111634    6512453    6512662    23 FW2         TY1   #OW+43929P USCU
                                                           N  YOAOS 1K.153
                                                           4294 ENTER  LOGON

O    6614104    6613882    6906780    38 FW2         TY1   #LOGON TS0217 W
                                                           (DESK) T(99) NA(
                                                           WAGNER)
R    6937672    6925646    6925856    89 FW2         TY1   #
R    7055898    7285700    7285911    89 FW2         TY1   ENTER PASSWORD FO
                                                           R TS0217
R    7313864    7375718    7375928    89 FW2         TY1
O    7718954    7718885    7745043    12 MAINTSO     TY2   #X
R    7774123    7807581    7807792    23 MAINTSO     TY2   #OW
R    7411269    7815801    7816013    89 FW2         TY1   AAAAAAAA
                                                           00000000
                                                           XXXXXXXX
R    7814228    7928037    7928250    23 MAINTSO     TY2   43929P USCUN  YOA
R    7934684    7948115    7948325    23 MAINTSO     TY2   OS
O    7917768    7917791    7971545   107 FW2         TY1   #PASSWORD
R    7842483    7858880    7975185   102 FW2         TY1
R    7954760    8004625    8004835    23 MAINTSO     TY2    IKJ
R    8003990    8045847    8046056   122 FW2         TY1   #
R    8011270    8051469    8051680    23 MAINTSO     TY2   530284
R    8080420    8125862    8126071   122 FW2         TY1
R    8058113    8145154    8145364    23 MAINTSO     TY2    ENTER  LOGON
R    8162211    8675966    8676176   122 FW2         TY1   TS0217 LOGON IN
                                                            PROGRESS AT 16:
                                                           31:32 ON JUNE
                                                           14, 1973
O    8448814    8448567    8783646    38 MAINTSO     TY2   #LOGON TS0328
                                                           W(DESK) T(99)
                                                            NA(WAGNER)
        ↓          ↓          ↓             ↓          ↓

O    9867966    9867898    9928434   132 FW2         TY1   #LOGOFF
R    9958493   10006219   10006428   148 FW2         TY1   #
R   10085988   11526517   11526716   148 FW2         TY1    86K CORE USED
                                                               01 TGETS
                                                            02 TPUTS      0
                                                           1 DISK EXCPS

R   11558477   12086613   12086828   148 FW2         TY1   JOB LOG 73165594
                                                           9589     2.45 CP
                                                           U SECS   22.76 E
                                                           LAPSED SECS

R   12117581   12626716   12626925   148 FW2         TY1   TS0217 LOGGED O
                                                           FF TSO AT 16:32
                                                           :08 ON JUNE 14
                                                           , 1973+

END-OF-FILE
R
```

Figure 30. Actual Time Output Format

```
                    USER INPUT:     DATAR/L/I/S RUN2

DEVICES REQUESTED ARE :                                  PAGE    1

 1   CT0
17   TY1
18   TY2
23   OS9
------------------------------------------------------------------------
TIME-INTERVAL LIST OF RUN2        MESSAGE     4  PAGE   3

TERMINAL IDENTIFICATION:  CT0

REC TRANS     TASK     CPU  RESP  SCENARIO
TYP TIME      TIME     TIME TIME  NAME          TEXT
------------------------------------------------------------------

M                                               MOD  0  TABLE  I
M                                               MOD  0  TABLE  E
M                                               MOD  0  TABLE  S
C  393863      371       0                   (START CT0 ONE
E  -----       391    1325         ONE       020 ACTION TAKEN
S      0       489       0         ONE       0 PAW
C      0       224       0         ONE       (ERROR Y
E      0       276    2452         ONE       020 ACTION TAKEN
S      0      1142       0         ONE       C(ERROR Y
C      0       393       0         ONE       (START TY2 SC2
E      0       397    4449         ONE       020 ACTION TAKEN
------------------------------------------------------------------------
TIME-INTERVAL LIST OF RUN2        MESSAGE    10  PAGE   3

TERMINAL IDENTIFICATION:  TY1

REC TRANS     TASK     CPU  RESP  SCENARIO
TYP TIME      TIME     TIME TIME  NAME          TEXT
------------------------------------------------------------------

M                                               MOD  0  TABLE  I
M                                               MOD  0  TABLE  E
M                                               MOD  0  TABLE  S
S      0       360       0        SC1        0 040
Q  16700     10963       0        SC1
S      0     11351       0        SC1        D
S      0       256       0        SC1        0
S      0       231       0  0.10  SC1
S      0       200       0        SC1        0 X.
S      0       350       0        SC1        0
R  20000       230       0  0.20  SC1         X.
S      0       291       0        SC1        0 X.
S      0       351       0        SC1        R
S  00119       215       0  0.50  SC1        SIGN ON
S      0       274       0        SC1        S)SIGN ON
------------------------------------------------------------------------
TIME-INTERVAL LIST OF RUN2        MESSAGE    12  PAGE  12

TERMINAL IDENTIFICATION:  TY2

REC TRANS     TASK     CPU  RESP  SCENARIO
TYP TIME      TIME     TIME TIME  NAME          TEXT
------------------------------------------------------------------

M                                               MOD  0  TABLE  I
M                                               MOD  0  TABLE  E
M                                               MOD  0  TABLE  S
S      0       404       0        SC2        0 040
Q  10307     10500       0        SC2
S      0     11223       0        SC2        0
S      0       233       0        SC2        0
S      0       213       0  0.00  SC2
S      0       270       0        SC2        S X.
------------------------------------------------------------------------
FOLLOWING DEVICES ARE INACTIVE:                         PAGE   20

23  OS9
END-OF-FILE
```

Figure 31. Time Interval Output Format

```
USER INPUT:     DATAR/L/R/O

DEVICES REQUESTED ARE :                                   PAGE    1

 1  CT0
 2  DS14
 3  TY1
 4  TY2
 5  TY3
------------------------------------------------------------------------

TIME-RELATIVE LIST OF Z  MESSAGE     4  PAGE   2

TERMINAL IDENTIFICATION: CT0              RUN START TIME:   0.67953
USER START TIME: BELOW
REC  START      END        START       END      SCENARIO
TYP     REL  UST                 REL   Q         ADDR NAME         TEXT

------------------------------------------------------------------

                                                 MOD  w  TABLE  I
M                                                MOD  W  TABLE  E
M                                                MOD  W  TABLE  S
M
C    0.0020     6.5879     0.0020     6.5879                  (START TY1 ED2741
E    6.6296     6.6354     6.6298     6.6354                  020 ACTION TAKEN
USER START TIME: NONE
------------------------------------------------------------------------

TIME-RELATIVE LIST OF Z  MESSAGE     6  PAGE   4

TERMINAL IDENTIFICATION:  TY1              RUN START TIME:   0.87950
USER START TIME:    7.31451
REC  START      END        START       END      SCENARIC
TYP     REL  UST                 REL   Q         ADDR NAME         TEXT

------------------------------------------------------------------

                                                 MOD  0  TABLE  I
M                                                MOD  0  TABLE  E
M                                                MOD  6  TABLE  S
M
Q    0.0000     3.0191    -3.0191     0.0000     3 ED2741     WLOGON TS0217 W(DE
                                                              SK) T(99) NA(WAGNER
                                                              )
S   -0.0067     3.0233    -3.0250     0.0041     3 ED2741     DWLOGON TS0217 W(D
                                                              ESK) T(99) NA(WAGNE
                                                              R)
S    3.0242     3.0272     0.2030     0.2001    51 ED2741     R
R    3.2647    12.3997     0.2456     9.3806    54 ED2741     WENTER PASSWORD FOR
                                                              TS0217- AAAAAAAA
                                                                      OOGOOOCO
                                                                     XXXXXXXX

S   12.4040    12.4114     9.3849     9.3922    54 ED2741     S ZXXXXXXXX
Q   12.4159    13.0245     9.3967    10.0053    67 ED2741     #PASSWORD
S   12.4122    13.0283    -0.6123     0.0037    67 ED2741     Q#PASSWORD
S   13.2290    13.2325     0.0045     0.2200    79 ED2741     R
R   13.2627    25.7621     0.2380    12.7355    82 ED2741     W TS0217 LOGON IN
                                                              PROGRESS AT 14:03
         ↓          ↓          ↓          ↓        ↓               ↓
------------------------------------------------------------------------
FOLLOWING DEVICES ARE INACTIVE                   PAGE     8

 2  DS14
 4  TY2
 5  TY3
END-OF-FILE
```

Figure 32. Relative Time Output Format

APPENDIX V

Example of teletype on-line listing for preparation of a single scenario, a real-time emulation, and a single data reduction listing.

```
XFER/A SCDR KAP
LOAD SCDR, STRIKE ANY KEY.
R


SSUB KAP TYPE SCENLIB
R


CVT TYPE 3 4

TO CANCEL RUN, USE CONTROL-A

R


9P9
WAIT
ENTER RUN ID

TYPE
READY

[START] TA32 TYPE
020 ACTION TAKEN
  TCB MAX 000006  TPQ MAX 000004
CORE LINKS  000034  CORE AVAIL  027554


DOS REV 05.

R


DATAR/L

END-OF-FILE

DATAR TERMINATED
R
```

APPENDIX VI

Timing Samples for Non-Real Time Programs

In Figure 34 where macros are expanded, lower case op-codes and some special characters do not print. These instructions can be referenced from Figure 33 in conjunction with the SCENLIB library macro substitutions.

```
34FORTN
ALLOCREGS 13
[illegible]
ETOREG R9 5 R10
    ***  R10 IS TRANSMISSION RATE
LDR 3 R9
     ***  R9 IS TYPING RATE
-R10 R9 R11
*R10 R9 R13
TYPE(8)
QEDITOR.
FIND(..)
TYPE(9)
QFORMAT,F
FIND(..)
C(SUB COST
TYPE(13)
QCREATE 10 10
LLAB1
R!!
SLAB2 =
EXECUTE
*R9 R11 R12
/R12 R10 R12
LDR 1000 R9
*R9 R12 R12
ADY R12
EXECUTE
JLAB1
LLAB2
SLAB1 ..
TYPE(6)
QRUN,F
FIND(..)
TYPE(19)
R240=15      CONTINUE
TYPE(24)
R330=        WRITE (6,100)
TYPE(62)
R390=104     FORMAT (1H0,3X,*EQUIPMENT COSTS*/7X,*SUBSYSTEM 1*,
TYPE(10)
QLIST 240
FIND(..)
TYPE(10)
QLIST 330
FIND(..)
TYPE(10)
QLIST 420
FIND(..)
TYPE(13)
QSAVE,TEST,O
FIND(..)
C(SUB INFO
TYPE(15)
QCREATE 100 10
LLAB3
R!!
SLAB4 =
EXECUTE
*R9 R11 R12
/R12 R10 R12
LDR 1000 R9
*R9 R12 R12
```

Figure 33. Fortran Cost Scenario with Macros not Expanded

```
ADY R12
EXECUTE
JLAB3
[illegible]
SLAB3 ..
TYPE(17)
QSAVE,INFO,NOSEQ
FIND(..)
TYPE(10)
QEDIT,TEST
FIND(..)
TYPE(6)
QRUN,F
FIND(..)
TYPE(12)
QSAVE,TEST,O
FIND(..)
TYPE(12)
QEDIT,INFO,S
FIND(..)
TYPE(36)
R144=201000         PTR WITH CONTROLLER
TYPE(25)
R490=001000         PROGRAMMING
TYPE(7)
QLIST,A
FIND(..)
TYPE(14)
QSAVE,INFO,O,N
FIND(..)
TYPE(10)
QEDIT,TEST
FIND(..)
TYPE(6)
QRUN,F
FIND(..)
TYPE(8)
QBYE,BYE
FIND(COMMAND)
TYPE(8)
QLOGOUT.
FIND(AT)
C(SUB OOFT1111
```

Figure 33. Fortran Cost Scenario with Macros not Expanded (Concluded)

```
FORTN
      13
      R0 0 R9
      R9 6 R10
      ***  R10 IS TRANSMISSION RATE
      3 R9
       ***  R9 IS TYPING RATE
   -R10 R9 R11
   *R10 R9 R10
      R0 0 R9
      R9 6 R10
      3 R9
   -R10 R9 R11
   *R9 R10 R10
      8 R9
   *R9 R11 R11
   /R11 R10 R10
      1000 R9
   *R9 R10 R10
      R10
   QEDITOR.
   L LL12
   R''
   S LL12 ..
      R0 0 R9
      R9 6 R10
      3 R9
   -R10 R9 R11
   *R9 R10 R10
      9 R9
   *R9 R11 R11
   /R11 R10 R10
      1000 R9
   *R9 R10 R10
      R10
   QFORMAT,F
   L LL20
   R''
   S LL20 ..
   CISUB COST
      R0 0 R9
      R9 6 R10
      3 R9
   -R10 R9 R11
   *R9 R10 R10
      13 R9
   *R9 R11 R11
   /R11 R10 R10
      1000 R9
   *R9 R10 R10
      R10
   UCREATE 10 10
   LLAB1
   R''
   SLAB2 =

   *R9 R11 R12
   /R12 R10 R12
      1000 R9
   *R9 R12 R12
      R12

   JLAB1
```

Figure 34. Scenarios for Fortran Cost Problem with Macros Expanded

```
LLAB2
SLAB1 ..
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  6 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
QRUN,F
L LL39
R11
S LL39 ..
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  19 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
R240=15     CONTINUE
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  24 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
R330=       WRITE (6,100)
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  62 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
R390=104    FORMAT (1H0,3X,*EQUIPMENT COSTS*/7X,*SUBSYSTEM 1*,
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  10 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
```

Figure 34. Scenarios for Fortran Cost Problem with Macros Expanded (Continued)

```
  R10
QLIST 240
L LL68
R''
S LL68 ..
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  10 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
QLIST 330
L LL76
R''
S LL76 ..
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  10 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
QLIST 420
L LL84
R''
S LL84 ..
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  13 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
QSAVE,TEST,O
L LL92
R''
S LL92 ..
C[SUB INFO
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  15 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
QCREATE 100 10
```

Figure 34. Scenarios for Fortran Cost Problem with Macros Expanded (Continued)

```
LLAB3
R''
SLAB4 *

*R9 R11 R12
/R12 R10 R12
  1000 R9
*R9 R12 R12
  R12

JLAB3
LLAB4
SLAB3 ..
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  17 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
@SAVE,INFO,NOSEQ
L LL111
R''
S LL111 ..
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  10 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
@EDIT,TEST
L LL119
R''
S LL119 ..
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  6 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
@RUN,F
L LL127
R''
S LL127 ..
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
```

85

Figure 34. Scenarios for Fortran Cost Problem with Macros Expanded (Continued)

```
  12 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
QSAVE,TEST,O
L LL135
R''
S LL135 ..
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  12 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
QEDIT,INFO,S
L LL143
R''
S LL143 ..
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  36 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
R140=001000            WITH CONTROLLER
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  28 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
R490=001000          PROGRAMMING
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  7 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
QLIST,A
L LL166
R''
```

Figure 34. Scenarios for Fortran Cost Problem with Macros Expanded (Continued)

```
S LL166 ..
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  14 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
QSAVE,INFO,O,N
L LL174
R''
S LL174 ..
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  10 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
QEDIT,TEST
L LL182
R''
S LL182 ..
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  6 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
QRUN,F
L LL190
R''
S LL190 ..
  R0 0 R9
  R9 6 R10
  3 R9
-R10 R9 R11
*R9 R10 R10
  8 R9
*R9 R11 R11
/R11 R10 R10
  1000 R9
*R9 R10 R10
  R10
QBYE,BYE
L LL198
R''
S LL198 COMMAND
  R0 0 R9
  R9 6 R10
```

Figure 34. Scenarios for Fortran Cost Problem with Macros Expanded (Continued)

```
   3 R9
-R10 R9 R11
*R9 R10 R10
   8 R9
*R9 R11 R11
/R11 R10 R10
   1000 R9
*R9 R10 R10
   R10
QLOGOUT.
L LL206
R''
S LL206 AT
C(SUB DUFT1111
```

Figure 34. Scenarios for Fortran Cost Problem with Macros Expanded (Continued)

```
COST
X
   51 9 R6
Q/PROGRAM COST(OUTPUT,INFO,TAPE5=INFO,TAPE6=OUTPUT)
   45 9 R6
QC ****PROGRAM TO COMPUTE COST ESTIMATES ****
   39 9 R6
Q/DIMENSION IM(20),IN(20),IP(20),IQ(20)
   18 9 R6
Q/DO 75 ICOUNT=1,2
   9 9 R6
Q/ISUMM=0
   17 9 R6
Q/READ (5,1) NUMM
   14 9 R6
Q1/FORMAT (I6)
   15 9 R6
Q/DO 5 I=1,NUMM
   18 9 R6
Q/READ (5,1) IM(1)
   19 9 R6
Q/ISUMM=IM(I)+ISUMM
   11 9 R6
Q5/CONTINUE
   9 9 R6
Q/ISUMN=0
   17 9 R6
Q/READ (5,1) NUMN
   16 9 R6
Q/DO 10 I=1,NUMN
   18 9 R6
Q/READ (5,1) IN(I)
   19 9 R6
Q/ISUMN=IN(1)+ISUMN
   12 9 R6
Q10/CONTINUE
   20 9 R6
Q/IEQSUM=ISUMN+ISUMM
   9 9 R6
Q/ISUMP=0
   17 9 R6
Q/READ (5,1) NUMP
   16 9 R6
Q/DO 15 I=1,NUMP
   18 9 R6
Q/READ (5,1) IP(I)
   19 9 R6
Q/ISUMP=IP(I)+ISUMP
   11 9 R6
Q15/CONINUE
   9 9 R6
Q/ISUMQ=0
   17 9 R6
Q/READ (5,1) NUMQ
   16 9 R6
Q/DO 20 I=1,NUMQ
   18 9 R6
Q/READ (5,1) IQ(I)
   19 9 R6
Q/ISUMQ=IQ(I)+ISUMQ
   12 9 R6
Q20/CONTINUE
   27 9 R6
```

Figure 34. Scenarios for Fortran Cost Problem with Macros Expanded (Continued)

```
Q;ITOTAL=IEQSUM+ISUMP+ISUMQ
   29 9 R6
Q;IF (ICOUNT .EQ. 2) GO TO 50
   14 9 R6
Q;WRITE ),100)
   64 9 R6
Q100;FORMAT (1H0//6X,*PRELIMINARY COST ESTIMATE*//1X,*SYSTEM A*)
   10 9 R6
Q;GO TO 55
   17 9 R6
Q50;WRITE (6,102)
   32 9 R6
Q102;FORMAT (1H0//1X,*SYSTEM B*)
   36 9 R6
Q55;WRITE (6,104) ISUMM,ISUMN,IEQSUM
   55 9 R6
Q104;FORMAT (1H0,3X,*EQUIPMENT COSTS*/7X,*SUBSYSTEM 1*,
   55 9 R6
Q      +5X,I8/7X,*SUBSYSTEM 2*,5X,I8/19X,*TOTAL*,2X,I10)
   34 9 R6
Q;WRITE (6,106) ISUMP,ISUMQ,ITOTAL
   57 9 R6
Q106;FORMAT (1H0,3X,*DEVELOPMENT COSTS*,5X,I10/4X,*O & M*
   44 9 R6
Q      +* COSTS*,11X,I10//19X,*TOTAL*,2X,I10)
   12 9 R6
Q75;CONTINUE
   6 9 R6
Q;STOP
   5 9 R6
Q;END
   2 9 R6
Q*
```

Figure 34. Scenarios for Fortran Cost Problem with Macros Expanded (Continued)

```
INFO
X
   29 9 R6
0000008        NUMBER IN LIST M
   29 9 R6
0001000        CPU WITH 24K MEM
   36 9 R5
0001000        FH DISC WITH CONTROLLER
   37 9 R6
0001000        MAG TAPE WITH CONTROLLER
   32 9 R6
0001000          WITH CONTROLLER
   32 9 R6
0001000        TTY WITH CONTROLLER
   38 9 R6
0001000        LINE PRINTER & CONTROLLER
   39 9 R6
0001000        16 ASYNCHRONOUS LINE ADAPT
   46 9 R6
0001000        1 HISPEED ASYNCHRONOUS LINE ADAPT
   29 9 R6
0000002        NUMBER IN LIST N
   22 9 R6
0001000        16 MODEMS
   23 9 R6
0001000        MODEM RACK
   29 9 R6
0000005        NUMBER IN LIST P
   23 9 R6
0001000        ELEC ENGIN
   23 9 R6
0001000        MECH ENGIN
   24 9 R6
0001000        PROGRAMMING
   26 9 R6
0001000        DOCUMENTATION
   18 9 R6
0001000        T & E
   29 9 R6
0000004        NUMBER IN LIST Q
   33 9 R6
0001000        OPERATIONS PERSONNEL
   29 9 R6
0001000        SERVICE CONTRACT
   34 9 R5
0001000        TELEPHONE & DAA LEASE
   28 9 R6
0001000        TELEPHONE USAGE
   27 9 R6
0000006        NUM IN LIST M*
   28 9 R6
0001000        CPU WITH 8K MEM
   32 9 R6
0001000          WITH CONTROLLER
   32 9 R6
0001000        TTY WITH CONTROLLER
   30 9 R6
0001000        0 LJ-LINE DIG I/O
   42 9 R6
0001000        16 ASYNCHRONOUS LINE ADAPTERS
   40 9 R6
0001000        8 SYNCHRONOUS LINE ADAPTERS
   27 9 R6
```

Figure 34. Scenarios for Fortran Cost Problem with Macros Expanded (Continued)

```
Q0000005      NUM IN LIST N*
  29 9 R6
Q0001000      2 HISPEED MODEMS
  24 9 R6
Q0001000      MODEM CLOCK
  17 9 R6
Q0001000      RACK
  33 9 R6
Q0001000      PANEL & SPECIAL CKTS
  45 9 R6
Q0001000      HISPEED SYNCHRONOUS LINE ADAPTER
  26 9 R6
Q0000005      NUM IN LIST P*
  23 9 R6
Q0001000      ELEC ENGIN
  23 9 R6
Q0001000      MECH ENGIN
  24 9 R6
Q0001000      PROGRAMMING
  26 9 R6
Q0001000      DOCUMENTATION
  18 9 R6
Q0001000      T & E
  25 9 R6
Q0000004      NUM IN LIST Q*
  33 9 R6
Q0001000      OPERATIONS PERSONNEL
  29 9 R6
Q0001000      SERVICE CONTRACT
  35 9 R6
Q0001000      TELEPHONE & DATA LEASE
  28 9 R6
Q0001000      TELEPHONE USAGE
  2 9 R6
Q*
```

Figure 34. Scenarios for Fortran Cost Problem with Macros Expanded (Concluded)

Contents of SCENLIB Macro Library

Name	Value
ALLOCREGS	a
BESPTOREG	c
ADY	d
EXECUTE	e
FREEBUFF	f
GTR	g
ETOREG	h
INPUTPARAM	i
LDR	l
BROFF	n
PTR	p
BDF	q
RANDOM	r
TYPEOUT	t

Figure 35. Macro Libraries for Fortran Cost Problem

```
KAPLIB
MDEF FIND(1)
L LLST
R!!
S LLST $1
MEND
MDEF NDEV
1
MEND
MDEF TYPE (1)
ETOREG R0 0 R9
ETOREG R9 6 R10
LDR 3 R9
-R10 R9 R11
*R9 R10 R10
LDR $1 R9
*R9 R11 R11
/R11 R10 R10
LDR 1000 R9
*R9 R10 R10
ADY R10
MEND
```

Figure 35. Macro Libraries for Fortran Cost Problem (Concluded)