A Feasibility Study of Computer Aided Coding of Ground Operations Aerospace Language (GOAL)

Fall 1979

Harvey Glenn Lanier
University of Central Florida
A FEASIBILITY STUDY OF COMPUTER AIDED CODING OF GROUND OPERATIONS AEROSPACE LANGUAGE (GOAL)

BY

HARVEY GLENN LANIER
B.S., North Carolina State University, 1961

RESEARCH REPORT
Submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering in the Graduate Studies Program of the College of Engineering of the University of Central Florida at Orlando, Florida

Fall Quarter
1979
ABSTRACT

The introduction of a new checkout language at Kennedy Space Center has required more effort to create test and operations software than anticipated. The new language is called GOAL, for Ground Operations Aerospace Language. The feasibility of a computer aided GOAL coding system that would reduce the effort required to create GOAL programs is investigated in this report. A background of GOAL, its coding requirements, and the facilities used at present for GOAL coding is presented first. Next, the computer aided GOAL coding concept is presented, and requirements for such a system are developed. After the requirements are presented, a system which has been used to develop some concepts and check their feasibility is described. Finally, some conclusions are drawn on the feasibility of the computer aided GOAL coding concept, and hardware required to implement an operational system are presented.
CONTENTS

GLOSSARY ................................................................. v

I. INTRODUCTION ....................................................... 1

II. BACKGROUND ......................................................... 3

GOAL Definition ...................................................... 3
GOAL Coding Requirements .......................................... 6
Present Coding Facilities ............................................. 6

III. COMPUTER AIDED GOAL CODING ............................... 8

Concept ................................................................. 8
Initial Coding Requirements .......................................... 10
Editing of Existing Code ............................................. 13
Possible System Expansions .......................................... 13
Hardware Requirements ............................................. 14

IV. A COMPUTER AIDED GOAL CODING
DEVELOPMENTAL SYSTEM ........................................... 15

Hardware and Systems Software Used ............................ 16
Syntax Diagram Display ............................................... 17
Data Array ............................................................. 21
Keyword String ......................................................... 23
Choice String .......................................................... 24
Action Code Array ...................................................... 24
CAGC Logic ............................................................. 25
Initialization ............................................................ 25
Statement Type Selection ............................................. 26
Syntax Data Input and Display ....................................... 26
Variables Preset ......................................................... 27
Keyword Logic .......................................................... 27
Manual Input Logic ..................................................... 28
Element Logic .......................................................... 28
Choice Logic ............................................................ 29
Handling Restrictions .................................................. 30
Summary of Developmental System ............................... 31
CONTENTS

V. AN EXAMPLE OF CAGC OPERATION .................. 34
VI. CONCLUSIONS ......................................... 39

Study Results ............................................. 39
Areas Not Implemented .................................. 40
Hardware and Systems Software Requirements .............. 41
Feasibility ................................................. 43

APPENDIXES .................................................. 44

A. GOAL's Character Set .................................. 44
B. CAGC Flow Chart ....................................... 45
C. Listing of Computer Aided GOAL Coding (CAGC) Program ........ 48
D. Action Code Array ....................................... 51
E. CAGC Syntax Diagrams .................................. 52

LIST OF REFERENCES ........................................ 106
# GLOSSARY

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII</td>
<td>American Standard Code for Information Interchange</td>
</tr>
<tr>
<td>BASIC</td>
<td>A computer language</td>
</tr>
<tr>
<td>BAUD</td>
<td>Data transmission rate dimension (baud approximately equals bits/sec)</td>
</tr>
<tr>
<td>BYTE</td>
<td>Eight bits of binary data</td>
</tr>
<tr>
<td>CAGC</td>
<td>Computer Aided GOAL Coding</td>
</tr>
<tr>
<td>CRT</td>
<td>Cathode Ray Tube</td>
</tr>
<tr>
<td>EBCDIC</td>
<td>Extended Binary Coded Decimal Interchange Code</td>
</tr>
<tr>
<td>GOAL</td>
<td>Ground Operations Aerospace Language</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>LPS</td>
<td>Launch Processing System</td>
</tr>
<tr>
<td>LSD</td>
<td>Least Significant Digit</td>
</tr>
<tr>
<td>MODULO</td>
<td>A computer mathematical function which outputs the remainder of an integer divide</td>
</tr>
<tr>
<td>MSD</td>
<td>Most Significant Digit</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
</tr>
<tr>
<td>PROM</td>
<td>Programmable Read Only Memory</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
</tbody>
</table>
I. INTRODUCTION

The introduction of a new checkout language at Kennedy Space Center has required more effort to create test and operations software than anticipated. The new language is called GOAL, for Ground Operations Aerospace Language. The feasibility of a computer aided GOAL coding system that would reduce the effort required to create GOAL programs is investigated in this report.

GOAL is a checkout oriented language used by the National Aeronautics and Space Administration (NASA) and its contractors at Kennedy Space Center, Florida. GOAL is used primarily for automatic test and operation of major aerospace systems. "GOAL is a test engineer oriented language which is designed to be used as a standard procedure terminology and test programming language in performance of ground checkout operations in a Space vehicle test and launch environment. . . . GOAL permits a high degree of readability and retainability by providing the necessary operators required for testing, expressed in a familiar notation"(1). This readability is achieved by a language syntax that
is the same as simple imperative English sentences. However, most GOAL statements are long and wordy by the time the needed modifiers are added. Readability is achieved, but the source code is long and wordy. Originally GOAL programs were to be coded by the system engineer in charge of a system (2). In many cases, however, programmers have been hired to code GOAL programs for the system engineer.

The feasibility of a computer aided GOAL coding system that would eliminate the need for programmers to code GOAL for the engineer is investigated in this report. A background of GOAL, its coding requirements, and the facilities presently used for GOAL coding is presented first. Next, the computer aided GOAL coding concept is presented, and requirements for such a system are developed. After the requirements are presented, a system which has been used to develop some concepts and check their feasibility is described. Finally, some conclusions are drawn on the feasibility of the computer aided GOAL coding concept, and requirements for hardware to implement an operational system are presented.
II. BACKGROUND

Before the need and requirements for a computer aided GOAL coding system can be understood, a better understanding of GOAL and its coding requirements are needed. This section provides more detail on the GOAL language, its coding requirements, and present facilities used for GOAL coding. This discussion is limited primarily to those features of GOAL that affect coding requirements.

**GOAL Definition**

Ground Operations Aerospace Language (GOAL) is a high order language designed by the National Aeronautics and Space Administration (NASA) at Kennedy Space Center, Florida. GOAL is used primarily for ground test and operation of space hardware and its ground support equipment.

GOAL is an interpretive language. GOAL's source code is preprocessed by a language processor (compiler) that generates an intermediate interpretive code. The language processor also checks the input for syntax errors, branching errors, restriction violations, and other explicit errors. The intermediate code is then
executed by the Launch Processing System (LPS) to test and operate Shuttle Transportation System hardware. The GOAL intermediate code is executed by an interpreter under control of a real-time operating system. The interpreter and the real-time operating system allow the real-time control and intervention required to perform test and operations of large aerospace systems such as the Shuttle Orbiter, a Shuttle payload, or other major aerospace system.

All phases of GOAL depend on a data bank which defines the external interfaces of GOAL. The data bank gives the name, address, data type, calibration, and other attributes of all inputs and outputs of the system under test. The data bank also defines the system devices (peripherals) of the test system. Names and/or addresses from the data bank are used during coding. During preprocessing, the language processor verifies that Internal Names and addresses are correct and that data types and other attributes are compatible with usage. Data such as calibration data are required from the data bank during execution.

The GOAL language syntax is defined by syntax diagrams. "Syntax diagrams identify legal sequences of items in a GOAL statement including alternate branches,
optional entries, and feedback loops"(1). Syntax diagrams represent repetitive syntax units by boxes which represent subordinate syntax diagrams. The top level syntax diagrams represent GOAL statements, and the subordinate syntax diagrams are called elements.

Element syntax diagrams are in turn made up of smaller elements until the smallest unit of GOAL syntax is reached. The smallest unit is a character in the GOAL character set or a keyword made up of these characters. The GOAL character set is defined in Appendix A. A full set of syntax diagrams for the Solid Rocket Booster (SRB) baseline version of GOAL is shown in Ground Operations Aerospace Language - SRB Baseline Syntax Diagram Handbook (3). The subset of these syntax diagrams implemented in a developmental computer aided GOAL coding system is shown in Appendix E. The SRB baseline version of GOAL has been chosen as the baseline for this study to reduce the impact of GOAL changes on the study effort. Other versions of GOAL, or any other language defined by similar syntax diagrams, can be easily implemented once the techniques are developed.
GOAL Coding Requirements

GOAL statements are generated by creating legal sequences of GOAL code as defined by the syntax diagrams. GOAL code allows an unrestricted free field format for ease of coding and to allow code positioning for better readability.

A GOAL program consists of two major sections: the declarations section and the procedural section. Declaration statements are used to define all internal variable names, data type (numeric, discrete, or text), and data structure (single variable, variable list, or table). The declaration statements are used by the GOAL language processor to allocate memory for internal variable storage. The procedural statements are the test and operations procedure which will be executed at run time. System statements are interspersed through both declaration and procedural sections as required. System statements act as GOAL language processor directives.

Present Coding Facilities

Presently GOAL source code is created using the text editor of a large central computer. Remote terminals are connected to the central computer via acoustic
couplers and 300 baud modems. Code may be generated in real-time and entered directly into the computer, but most users generate the code in advance on coding forms or scratch pads. In both cases the source code must be entered into the computer manually. The text editor can be utilized to make some repetitive entries and to correct errors, but the text editor is not designed to handle the unique problems of GOAL coding.

The productivity of GOAL coding using only a general purpose text editor has proven to be low. Syntax errors are common especially in the more complex statements. Most efficient GOAL test techniques are not being used. Typographical errors are numerous. Many passes with the language processor are required to achieve a program free of syntax and typographical errors. Additional debug runs of the program against hardware simulators or the actual hardware are required to achieve a workable test program. The feasibility of a computer aided GOAL coding system that would alleviate many of these problems is the subject of this report.
III. COMPUTER AIDED GOAL CODING

Some type of system that will increase the productivity of the GOAL user is needed. To achieve the original intent that GOAL be a test engineer oriented language, the test engineer needs to be able to quickly and easily generate an accurate test program for his system. Key entry requirements should be reduced, proper syntax should be insured early in program generation, and tutorial information should be readily available if the engineer is to achieve this. This section presents a concept and requirements for such a system. During the rest of this section and the discussion of a developmental system in Section IV, a Computer Aided GOAL Coding system is referred to by the acronym CAGC for conciseness.

Concept

During an early encounter with GOAL, it became apparent that some type of system was needed to aid the GOAL user in generation of test programs using the GOAL language (4). During that early study of the GOAL language, a system whereby the GOAL programmer would be
aided by a computer began to be conceived. These early concepts have been improved and expanded into the system described below.

As a result of these early encounters with GOAL and continuing as the specific task of this study, a system has been conceived that would greatly increase the productivity of the GOAL programmer. The system would display the GOAL syntax diagrams and allow selection of optional syntax branches by short numeric key entry. The system would automatically enter keywords, phrases, and delimiters. Internal variable names and External Designators would be selected from lists by numeric key entry. The display of syntax diagrams would serve as a tutorial aid to the user. The syntax diagram presentation of GOAL will be familiar to the GOAL user since most documentation of GOAL is based on the syntax diagram presentation of the language. Proper syntax would be enforced by allowing only valid syntax branches to be chosen.

The system described above would require frequent display of full CRT pages. Existing 300 baud lines would not support the required data flow, and data communications equipment fast enough to support higher data rates would be costly. Therefore, this study has concentrated on a stand-alone system. A stand-alone
system has the added advantage of off loading the large scale central computer now being used. The stand-alone system would require a processor to control display of the syntax diagrams, check validity of entries, create GOAL code from stored data and keyboard entries, and perform other processing functions required by the system. Storage would be required for the syntax diagram data, the GOAL code generated, and the system control programs and data. Mass storage for syntax diagrams and data would have to be random access since random repetitive access would be required. The display would have to display complete syntax diagrams and other data necessary for the user to quickly decide on and make valid entries. Display of the completed GOAL code would also be required.

These are the basic concepts and some assumptions upon which this study is based.

**Initial Coding Requirements**

For CAGC to be useful it must assist the user to generate accurate code. It must be relatively fast for both the experienced and the new user. Also the system should help the user to learn and utilize the GOAL language. The requirements for such a system are discussed in the following paragraphs.
CAGC should help the user to generate accurate code free of syntax errors, restriction violations, and typographical errors. CAGC should enforce the language syntax as the statements are formed. Thus any code generated with CAGC would be free of syntax errors. Where practical, restrictions should be enforced in the same way. Typographical errors should be kept to a minimum by minimizing user entry requirements. Manual entries should be easy to proofread and correct before final entry. Backup capability shall be provided such that any part or all of the statement in work may be deleted.

CAGC should be fast. It must assist both the experienced user and the novice to generate GOAL programs relatively fast. The CAGC user should not have to wait on the CAGC system. Any processing, data transfer, and display should be completed such that the user is not aware of any significant system delay. CAGC should display the data needed for the user to make quick decisions on the next entry. Entries required should be short and easy to enter. The system should give enough data and be simple enough to allow the novice GOAL user to generate accurate code relatively fast. On the other hand, CAGC should not impede the progress of
the experienced user. The pertinent data required by the experienced user should be readily recognized and not obscured by tutorial information. No repetitious entries should be required. CAGC should enter all keywords from the syntax. CAGC should not require the repetitious entry of Internal Names or External Designators.

A copy capability should be provided such that manual entries which are the same or similar to previous entries may be copied easily from the previous entry. A search capability should be provided to assist in finding the desired data to copy. Also, buffers should be provided for repetitive manual inputs. Frequently used inputs would be stored in the buffers and entered into the GOAL code by recall of the buffers.

CAGC should aid the GOAL novice to learn and use the GOAL language. CAGC should also prompt the experienced user on infrequently used features of GOAL. CAGC should have tutorial features that make it easy to learn and easy to use. CAGC should require no special skills or training to use. The system should require minimum setup, and CAGC should be transparent to the user during operation.
Editing of Existing Code

The process of generating error free code that meets the users requirements is usually a reiterative process. Requirements change, errors must be corrected, and sometimes the engineer just thinks of a better way. Therefore, any computer aided GOAL coding system must support editing of existing code. Although full edit of existing code is a goal; it would be difficult to achieve. Full edit would require lexical and syntactical analysis of the existing code to identify the syntax. Then the system would have to control insertions and deletions and verify the changed code to be syntactically correct. It appears to be a better use of resources to just recreate the changed statements from scratch except that capability should be provided to copy long manual user inputs from the old statements.

Possible System Expansions

Two areas of possible system expansion appear to be easily obtainable with a minimum of effort. Requirements for these two possible system expansions should be kept in mind when implementing CAGC.

One possible expansion would be to implement a complete GOAL training course in conjunction with CAGC. This feature would allow GOAL training to be conducted
on an individual basis at any time. The using organizations would not be dependent on training class schedules to train new GOAL users.

The second possible expansion of CAGC would be to perform some of the functions of the GOAL language processor during the coding operation. In fact, much of the data needed by the language processor is available to the CAGC processor during GOAL coding. This data could be collected and retained for use by the processor.

**Hardware Requirements**

The primary hardware restrictions are related to cost. Maximum utilization should be made of existing equipment. Existing equipment at Kennedy Space Center is CRTs with attached printers. This equipment is connected to the central computer by 300 baud dialup lines with acoustic couplers. At present, the text editor and the GOAL language processor in this computer are utilized for GOAL coding. GOAL code from CAGC will still have to be input to this computer for processing by the GOAL language processor.

At any given location, a big factor on selection of hardware would be the requirements for other uses of the same hardware and/or compatibility with existing systems.
IV. A COMPUTER AIDED GOAL CODING DEVELOPMENTAL SYSTEM

After first proposing a computer aided GOAL coding system, an attempt was made to demonstrate the feasibility of such a system. The equipment needed was only available for two weeks, but many of the concepts of the present system were developed and demonstrated. Time did not permit enough data to be collected to size the system; however, it was felt that CAGC could be implemented on an 8-bit microcomputer with a floppy disk. A CRT and printer would also be required to complete the system.

A hardware system, except for a printer, that meets the above requirements was obtained; and a developmental computer aided GOAL coding system has been programmed. This system meets most of the requirements in Section III of this report. This system is called CAGC for conciseness. Emphasis has been placed on developing some of the unique and basic concepts of the system. Features like the standard text editor routines for search, copy, and buffered phrases can be added after the basic techniques are developed.
This section is a description of the system that has been developed. First, the hardware and systems software used is described. Next, CAGC's logic approach is described. An assessment is then made of the work performed on CAGC to date.

Hardware and Systems Software Used

The CAGC system has been implemented on an 8080 based microcomputer with floppy disk and a CRT terminal. A printer is needed to provide hardcopy, but it is not mandatory for developmental work. The microcomputer has 29K bytes of RAM memory and a 2K byte PROM board. A disk boot loader and a monitor are available in PROM. A two-channel serial I/O board is used to interface the CRT to the processor. The CRT interface is operating at 9600 baud. The single disk drive has a 300K byte capacity. The disk controller can handle 16 disk drives.

The systems software is an extended Basic interpreter with disk file management, edit, and diagnostic features. Extensive string variable capability is available in the Basic interpreter; this capability is required by CAGC.
Syntax Diagram Display

The need for CAGC was fostered by the wordiness of GOAL source code. The need for a system to improve user efficiency was apparent. One approach would be some sort of menu-display-select or question-answer routine. The fact that the GOAL syntax was originally defined by syntax diagrams prompted the technique of direct display of these diagrams. This subsection describes the use of syntax diagram display, its limitations, and special cases.

GOAL syntax diagrams are ideally suited for a menu like display for CAGC. Almost all documentation of GOAL is based on the syntax diagram presentation of the language. Therefore, anyone that has studied or read about GOAL will be familiar with the presentation. Also the size of the GOAL syntax diagrams is readily adaptable to display on an alpha-numeric CRT terminal. Alpha-numeric CRT terminals are relatively economical and are readily available.

The largest syntax diagram in the SRB baseline GOAL (3) can be displayed on less than 20 eighty-character lines without any abbreviations or shortcuts. However, some minor abbreviations and syntax restrictions are recommended for system efficiency and
clarity. For instance in the Time Value element (E103) syntax, it would greatly simplify the syntax and its use if only a single dimension was used after each decimal number. For instance, a single dimension Hr or HR(s) could be used instead of a choice of four dimensions, Hr, Hrs, Hour, or Hours. However, only a few minor abbreviations are implemented in the present system.

Prior to describing CAGC handling of syntax diagrams, more discussion of GOAL syntax diagrams is needed. There are two types of syntax diagrams in GOAL: statement and element. The top level syntax diagrams are the GOAL statement diagrams. GOAL statements are broken into three classifications: the declaration statements, the procedural statements, and the system statements.

Declaration statements are non-test action statements. They are used at the beginning of a GOAL program to define all internal variables to be used by the program. The declaration statement defines the variable name, its type, and size and may set the variable to an initial value. Even though declaration statements must all be at the beginning of a GOAL program, the approach used by CAGC makes it advantageous to create the declaration statements last.
Procedural statements are GOAL test action statements that control the testing sequence to be performed. System statements primarily direct the GOAL language processor and are interspersed with the other statements as required.

The second type of syntax diagram is the element syntax diagram. The element is a subordinate unit of syntax used to define repetitive syntax. Element syntax diagrams are in turn made up of other smaller elements until the lowest syntax elements are reached. The lowest syntax elements are the characters of the GOAL character set.

The concept of syntax display and optional branch selection by the user works well for the statements and larger elements. But two conditions limit this technique. First, it is not practical to extend the numeric-key menu selection down to the lowest element syntax. In the extreme, this would result in the selection of each character in a text entry by a numeric key entry. This would be very cumbersome. Secondly, External Designators are predefined according to syntax rules and stored in a data bank. The data bank is unique for each combination of checkout computer system and group of systems under test. Therefore when the
element "External Designator" is encountered in CAGC, lists of previously defined External Designators are displayed, and the desired External Designator is selected by line number. Different lists are provided for each combination of data type and command/response designation.

All the External Designators in a large data bank are neither needed for a given program nor is it desirable to have to select from such a long list for each entry. Therefore, lists are made in advance by manual entry or selection from the data bank list. Alternatively each External Designator may be added manually to the list the first time it is used. In a small system, storage probably would not be available to keep a large data bank on line continuously to allow selection of new External Designators from the data bank. Internal Names are handled in a similar manner.

The GOAL syntax diagrams lend themselves to easy display on an alpha-numeric CRT terminal. All the required symbols are easily portrayed with the ASCII 96 character set. All the syntax diagrams of the SRB baseline version of GOAL required for CAGC implementation were input to CAGC; the result is shown in Appendix E. The Time Value syntax (E103) was the most
difficult to adapt to CRT display. Some of the guidelines used in generating these displays were:

1. No change of the syntax.
2. Syntax diagram clarity must be maintained.
3. Only minor abbreviations allowed where clarity is not impaired.

Once each syntax diagram display format has been worked out, the data is stored in a sequential file on disk; each line is represented by a string variable.

**Data Array**

The syntax diagram as displayed on a CRT is formatted for human clarity and is not in the best form for use as data by the computer during GOAL coding. Data required by the computer is stored in a data array (A), a keyword string (K$), and a choice string (C$) for each syntax. The data array contains the data which determine what sequence of action is taken when each syntax branch is chosen and which branches are legal at any given time.

The data array has four columns and a row for each optional branch on the syntax diagrams plus continuation rows when required. The data in the array is stored as five digit intergers, and each integer represents two pieces of data. The three most significant
digits (MSD) represent one piece of data and are accessed by an integer divide by 100. The two least significant digits (LSD) represent the other piece of data and is accessed by a Modulo 100 function.

Each branch on the syntax diagram has a numeric key, and the data in the corresponding row of the data array controls what action is taken when that branch is selected. Row zero controls the action on initial entry into the syntax.

In each row of the data array, the action to be taken with the data in columns 1-3 is specified by data in the corresponding columns of a row in the action code array. The controlling row of the action code array is specified by the three most significant digits (MSD) in column zero of the data array. The two least significant digits (LSD) of data array column zero contains the row number of a continuation row in the data array if needed.

Each five digit integer in columns 1-3 of the data array represents one of the following as determined by action codes from the referenced action code array row:
1. If the action code signifies keyword, the data represents the position of the first character and the length of the desired keyword in the keyword string (K$).
2. If the action code signifies element, the data represents the file number of the element syntax data and any restrictions applicable to this use of the element.

3. If the action code signifies choice, the data represents the position of the first character and the length of the desired group of valid numeric keys in the choice string (C$). If the data array entry is greater than 29000, an exit from the present syntax is indicated.

4. If the action code signifies manual entry, the data represents restrictions on the characters allowed.

**Keyword String**

The keywords from a syntax diagram are grouped together in a string variable (K$) and stored in the sequential file for that syntax. A keyword is any word, phrase, delimiter, or other character string required by the syntax. Keywords are placed in the string such that they can be retrieved by starting address and length. Some words or character sequences can be used for more than one keyword as long as each can be retrieved as a contiguous keyword by starting position and length. Spaces are retrieved with keywords as required to provide proper spacing in the finished code.
Choice String

Groups of valid choices are stored and handled in a string variable (C$) similar to the keywords. Groups of numeric key numbers (choices) are stored in the string variable separated by commas such that all possible combinations of valid groups of choices can be retrieved from contiguous locations by starting location and length. These groups are displayed for the operator to choose from, and the operator entry is checked for validity against the list.

Action Code Array

The action code array (L), contains the data that control the sequences of action performed by CAGC. Each active action code array location contains a digit to represent a type of action as follows: 1 = keyword, 2 = Element, 3 = Choice, or 4 = Manual Input. All sequences of these actions required by the GOAL syntax is represented by the action code array rows. The three most significant digits of column zero of each row of the data array points to a row in the action code array. The entries in the referenced row of the action code array specify the action to be taken with the data in the other three columns of the data array. This technique saves having to store the action code with
each piece of data in the data array. The complete action code array is shown in Appendix D.

**CAGC Logic**

The logic that converts syntax data and user inputs into GOAL code is a program written in Basic. The functions of this program are described in the following paragraphs. The discussion is based on a top level flow chart of this program shown in Appendix B; a listing of the program is shown in Appendix C. References to variable names and values in this subsection refer to those used in the listing. Comments have been kept to a minimum; because in Basic, the source code, including comments, resides in memory during execution.

**Initialization**

Initialization of CAGC consists of (1) defining all CAGC variables as integers to conserve storage, (2) clearing string variable space, (3) dimensioning array variables and string variable lists, and (4) entering data in the action code array. The program presently assumes that either each External Designator and each Internal Name will be entered from the keyboard when first used or that variable lists have been previously created and stored on the system disk.
After initialization, the user is requested to enter a file name for the source code of the GOAL program to be created, and the user is warned if the name selected is a duplicate of any other file on the disk.

Statement Type Selection

Once the program is initialized and again after the completion of each statement, CAGC will display a list of all the GOAL procedural and system statements and allow choice by numeric key entry of the next GOAL statement to be written.

Syntax Data Input and Display

This section of logic inputs all data having to do with the syntax being used from disk and displays the syntax diagram. Entry into this section can occur in several ways as shown in Appendix B. Entry can occur by new statement choice, by encountering an element, by return from an element, or on return from manual entry. To conserve time syntax display is not performed when next action is manual entry or element. In these cases no user action based on the display is required before another syntax diagram will require display.
Variables Preset

Once the syntax data array has been retrieved from disk, CAGC program variables and pointers can be set based on this data, data in the action array, and data retained from previous actions. These variables and pointers determine which of the four sections of action logic will be entered: keyword, manual entry, element, or choice. These four logic sections are discussed next.

Keyword Logic

If action code is keyword, the keyword logic is entered. If the syntax being processed is a statement or element syntax diagram, a keyword is extracted from the keyboard string (K$) and inserted in the code for the GOAL statement being created. The location of the keyword in K$ is determined by data from the data array. The three most significant digits (MSD) represent starting point and the two least significant digits (LSD) represent the length. Spaces are included in keywords, including a leading space on most keywords, to maintain source code readability.

If the syntax being processed is a variable list (External Designator or Internal Name), the appropriate External Designator or Internal Name is added to the
The appropriate display data line B$(N1) less the line number is taken as the GOAL code entry.

Manual Input Logic

If the action code is manual input, the open keyboard logic is entered. Before the open keyboard logic is entered, the syntax diagram being displayed will have given instructions as to what is to be entered. Four types of keyboard entry is required by CAGC: (1) specific numeric data, (2) specific text data (data used by GOAL as a value for a text variable), (3) text used for display messages and comments, (4) text used as a new External Designator, Internal Name or other name required by GOAL. In all four cases the user entry is added to the GOAL statement being created. In the fourth case the new name is also added as a new line of the list being processed by adding it to the sequential file for that list. In the present system an input of "ZZ" is required to indicate the end of a manual input.

Element Logic

If the action code is element, the element logic is entered. When the element logic is entered, the present syntax number and the present row and the next column of
the present data array are pushed onto the return stack (M). The new element syntax number is then retrieved from the data array. Program control is then transferred to the syntax data entry and display logic, and the new element is then the syntax controlling the GOAL programming process.

Choice Logic

If the action code is choice, the choice logic is entered. The choice logic first checks to see if the present syntax diagram is complete. For convenience, syntax diagram complete is handled by the choice logic. The choice logic automatically terminates action on the present syntax if the choice data in the data array is greater than 29000.

If the present syntax diagram is complete, the program next checks to see if the present GOAL statement is complete. GOAL statement complete is indicated by absence of any further return data in the return stack (M). If the statement is complete, the completed statement code is processed, and control is transferred to the statement type selection logic.

If the present element is complete, but the statement is not complete; the system returns to the next action of the syntax diagram which called the
syntax just completed. The data saved when the completed element was called is popped from the return stack, and program control is transferred to the syntax data entry and display logic. The GOAL programming process continues under control of the syntax whose data was just popped from the return stack.

If the present syntax is not complete, CAGC choice logic displays a list of valid numeric keys (syntax branches or variable list lines). The system then allows the user to enter the desired numeric key and checks the validity of the entry made. If the entry is not valid, the user is notified and allowed to make another entry. If the entry is valid, program execution continues on the syntax branch or list line indicated by the entry. The entry indicates a branch on the syntax diagram or a variable line number. The entry also specifies the row in the data array that relates to that syntax branch or variable line.

Handling Restrictions

Some restrictions on forming GOAL code from syntax diagrams have been expressed in the form of restrictive notes on the syntax diagrams or in accompanying text. These restrictions apply to things such as data types allowed for Internal Names and/or External Designators,
and size restrictions on data items. The concept of automatically enforcing as many of these restrictions as practical has been recently added to CAGC, but little coding has been done in this area. The concept includes: (1) flags to be stored with each syntax diagram which represent restrictions related to that syntax, (2) flags to be stored as the two least significant digits in the data array with an element number (this data represents restrictions related to that particular use of that element), and (3) user transparent changes to the syntax data to enforce restrictions.

Summary of Developmental System

The system described in this section was programmed as a feasibility study and development tool for a Computer Aided GOAL Coding (CAGC) system. It is a functional system and GOAL code can be created using the system. The following paragraphs summarize system operation.

After system initialization, a statement is chosen and the data file related to that statement is input to the computer. The statement syntax diagram is displayed with numeric keys on optional branches. CAGC displays a list of valid numeric keys, and the user selects an option by entering the appropriate numeric
key. When a valid numeric key is entered, CAGC performs a series of functions until the next optional syntax branch is reached. The series of functions performed after each entry is determined by data stored in the data array. Each function performed falls into one of four of the following types:

1. **Keyword** - If the function type is keyword, CAGC selects a keyword out of the keyword string. CAGC then inserts the keyword chosen in the GOAL statement being created. Variable names selected from a list are also treated as keywords by CAGC.

2. **Manual Entry** - If the function type is manual entry, the CRT keyboard is activated for user entry. The three types of user entry required are: (a) Enter a numerical value such as step number, program constant, or data for a program variable. (b) Enter text for comment, display message, or text data for a program variable. (c) Enter new internal variable name or External Designator address and name.

3. **Element** - If the function type is element, data required to return to the present syntax is pushed on a stack. Processing then continues on the new element syntax in the same manner.
4. Choice - If the function type is choice, a list of valid choices (numeric keys) are displayed and the user is allowed to select the desired syntax branch or variable line number. If the user entry is valid, CAGC performs the functions associated with the selected numeric key.

Internal Names and External Designators are selected from lists using the same logic. The lists are treated as syntax diagrams with line numbers as the optional branches. The logic for manual entry allows new items to be added to the lists by storing them in the appropriate disk file.
V. AN EXAMPLE OF CAGC OPERATION

The following is an example of how CAGC works. The example is broken down into CAGC actions and user actions. Each line of user action is a keyboard input which must be terminated with a carriage return. Data associated with each syntax (syntax diagram, choice string (C$), keyword string (K$), and data array (A)) is shown in Appendix E. The action code array (L) is shown in Appendix D. The example starts with the disk Basic operating system loaded, CAGC is not loaded or running.

USER - RUN "CAGC"
CAGC - Displays "INPUT GOAL FILE NAME"
USER - TESTPROG
CAGC - Opens new disk file "TESTPROG" for input.
    - Displays list of statements and prompts user to select a statement.
USER - 11 (to choose S 11 APPLY ANALOG STMT)
CAGC - Inputs data from disk file S 11 and displays syntax diagram.
- Displays ",1,2,3," as valid choices per data array (A) and choice string (C$). Also displays last 2 lines of completed GOAL code.

USER - 3 (to select branch 3 of syntax S 11)

CAGC - Inserts the keyword "APPLY" into the GOAL code per data array (A) and keyword string (K$).

- Displays ",4,5,6," as valid choices per data array (A) and choice string (C$). Also displays last 2 lines of completed GOAL code. GOAL code is now "APPLY". (S 11 syntax is still displayed).

USER - 4 (to select branch 4 of syntax S 11).

CAGC - Pushes S 11 data onto return stack (M) and inputs data from disk file E107 (QUANTITY) and displays syntax diagram and last two lines of completed GOAL code.

- Displays ",1,2,3,5," as valid choices per data array (A) and choice string (C$).

USER - 1 (to select branch 1 of syntax E107).

CAGC - Pushes E107 data onto return stack (M). Inputs data from disk file E108 (DECIMAL NUMBER) and displays syntax diagram and last two lines of completed GOAL code. The syntax diagram in this case contains instructions on how to enter a decimal number.
USER - 28.0 (A value as required by the system under test)
- ZZ (terminates manual input)

CAGC - Pops El07 data from return stack. Inputs data from disk file El07 and displays syntax diagram and last two lines of completed GOAL code. GOAL code is now "APPLY 28.0".
- Displays ",3,4," as valid choices per data array (A) and choice string (C$).

USER - 3 (to select branch 3 on syntax El07)

CAGC - Pushes El07 data onto return stack (M). Inputs data from disk file El90 (DIMENSION) and displays syntax diagram and last two lines of completed GOAL code.

USER - VOLTS (a dimension as required by the system under test)
- ZZ (terminates manual input)

CAGC - Pops El07 data from return stack. Inputs data from disk file El07 and sees that El07 is complete.
- Pops S 11 data from return stack. Inputs data from disk file S 11 and displays syntax diagram and last two lines of completed GOAL code. GOAL code is now "APPLY 28.0 VOLTS".
- Displays ",7,8," as valid choices per data array (A) and choice string (C$).

**USER - 8** (to select branch 8 on syntax S 11).

**CAGC** - Inserts the keyword "TO" into the GOAL code per data array (A) and keyword string (K$).

- Pushes S 11 data onto return stack (M). Inputs data from disk file E230 (EXTERNAL DESIGNATOR (ANALOG LOAD)) and displays the syntax diagram and last two lines of completed GOAL code. GOAL code is now "APPLY 28.0 VOLTS TO".

- Displays ",1,2,3,4 ...,17,18,19,20," as valid choices per data array (A) and choice string (C$).

**USER - 1** (to select External Designator displayed on line 1).

**CAGC** - Inserts the External Designator displayed on line 1, <ANALOG LOAD>, in the GOAL code. The External Designator is extracted from the syntax display data (B$).

- Displays ",1,2,3,4, ...,17,18,19,20," as valid choices per data array (A) and choice string (C$). Also displays last two lines of completed GOAL code. GOAL is now "APPLY 28.0 VOLTS TO <ANALOG LOAD>".
USER - 19 (to select no other External Designators and return).

CAGC - Pops S 11 data from return stack.
- Inserts the keyword ";" into the GOAL code per data array (A) and keyword string (K$). The GOAL statement being created is now complete. The GOAL statement created is "APPLY 28.0 VOLTS TO <ANALOG LOAD>;".

CAGC returns to the statement type selection logic and the process begins again on another statement. In the above process, the user inputs are short, and the process happens almost as fast as the user can make the inputs. In this way, GOAL code creation is fast and accurate.
VI. CONCLUSIONS

This feasibility study of computer aided coding of GOAL has been very successful. At the same time, it has uncovered difficulties which were not considered before. Some of these difficulties have been solved while others will have to be solved later. This study forms a good background for further study.

Study Results

This study has proven the feasibility of computer aided GOAL coding using an eight-bit microcomputer with CRT, printer and floppy disk storage. The concept of a data array driven system has proven very successful. This concept was refined to use an integer array for storage efficiency. Even so, the ability to add restriction flags into the array with the element numbers was added. System timing has proven more than adequate. The 8080 microprocessor with a clock rate of 2MHz, the disk with a transfer rate of 15KHz and the CRT with a transfer rate of 9600 baud executes the CAGC program at a rate such that the user is not aware of any significant delay.
Areas Not Implemented

Due to the time involved, several areas have not been attempted. Concepts have been worked on, and no insurmountable problems are expected. A brief discussion of some of these areas follow.

The use of Tables in GOAL can be complex. Using the techniques of CAGC the logic of tables is relatively easy to implement, but the primary problem of tables in CAGC will be to develop the best user presentation of table data. It is impossible to display on a single CRT page the row and column data of any but the simplest table.

As mentioned in the section on CAGC requirements, full edit of GOAL will be difficult. The plan initially is to recreate GOAL statements from scratch during edit except for a copy capability for long manual entries. This copy capability has not been implemented.

Another area that was not implemented is the capability to backup and correct errors in real-time during GOAL programming. Correction capability is basically a function of memory storage available for the GOAL code buffer. If a record is kept of syntax branches taken and if the code related to each numeric key is retained in memory, it will be easy to backup to any syntax
branch point and delete the related code. However once the code is committed to a sequential disk file, backup capability is lost.

Creation of the declaration section of a GOAL program has not been attempted as yet. No major difficulty is expected. As mentioned earlier, the declaration section will use the Internal Name lists as data.

Hardware and Systems Software Requirements

One of the objectives of this study was to determine if a computer aided GOAL coding system could be implemented on commonly available microcomputer equipment with floppy disk storage. The developmental system has performed very well on the system described in Section IV. The following is preliminary hardware and system software requirements for a system to implement an operational CAGC system.

1. Processor - Any of the commonly available 8-bit microcomputers that will support the peripherals below.

2. Memory - At least 16K above that required for systems software.
3. Display - A CRT display with the ASCII 96 character set, 24 line by 80 column screen size, addressable cursor, 9600 baud transmission rate minimum, RS232 serial interface.

4. Printer - A printer with a minimum of 64 ASCII character set and 300 baud RS 232 serial interface. (Even at 300 baud the printer may slow the system.)

5. Mass Storage - A dual drive floppy disk system or equivalent is required. The dual drive system is required to allow the system software to be on one disk while user storage is on another. Storage of a minimum of 250K bytes is required on each disk.

6. Systems Software - The system shall include a systems software package that includes the operating system and file management system. Also, the system shall include a High Order Language capability that includes two dimensional data array and extensive string variable capability.

Many relatively economical commercially available systems meet these requirements. Many of the available systems have all these components, except maybe the printer, in one table top unit.
Feasibility

The results of this study are that it is both technically and economically feasible to implement a computer aided GOAL coding system on stand-alone microcomputer equipment.
APPENDIX A

GOAL'S CHARACTER SET

CAPITAL LETTERS A-Z

NUMERALS 0-9

SPECIAL CHARACTERS:

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>BLANK</td>
<td>Blank</td>
</tr>
<tr>
<td>,</td>
<td>Comma</td>
</tr>
<tr>
<td>$</td>
<td>Currency</td>
</tr>
<tr>
<td>=</td>
<td>Equal</td>
</tr>
<tr>
<td>&lt;</td>
<td>Left angle bracket</td>
</tr>
<tr>
<td>&gt;</td>
<td>Right angle bracket</td>
</tr>
<tr>
<td>(</td>
<td>Left parenthesis</td>
</tr>
<tr>
<td>)</td>
<td>Right parenthesis</td>
</tr>
<tr>
<td>MINUS</td>
<td>Minus</td>
</tr>
<tr>
<td>SLASH</td>
<td>Slash</td>
</tr>
<tr>
<td>SEMICOLON</td>
<td>Semicolon</td>
</tr>
<tr>
<td>DECIMAL POINT</td>
<td>Decimal point</td>
</tr>
<tr>
<td>PLUS</td>
<td>Plus</td>
</tr>
</tbody>
</table>

NOTE: All characters are compatible with both the American Standard Code for Information Interchange (ASCII) and the Extended Binary Coded Decimal Interchange Code (EBCDIC).
APPENDIX B

CAGC FLOW CHART

NOTE: FLOW DIRECTION IS DOWN AND TO THE RIGHT.

CAGC

INITIALIZE CAGC

INITIALIZE GOAL VARIABLE LIST

OPEN GOAL SOURCE CODE FILE

CHOOSE GOAL STATEMENT TYPE

INPUT VARIABLES

INPUT FILE NAME

INPUT STMT. NO.
INPUT SYNTAX DATA AND DISPLAY SYNTAX DIAGRAM.

PRESET VARIABLES

ACTION CODE SELECT

ACTION = KEYWORD

ACTION = ELEMENT

ACTION = CHOICE

ACTION = MANUAL INPUT

INSERT KEYWORD IN GOAL CODE

SAVE DATA OF STACK AND CALL ELEMENT

INPUT DATA PER CRT DIRECTIONS—ADD NEW VARIABLES TO DISK FILES

INPUT DATA
SUBROUTINES

DISPLAY GOAL
  -- DISPLAY GOAL CODE ON CRT
  -- RETURN

GOAL CODE
  -- COLLECT GOAL CODE
  -- RETURN

DISK WRITE
  -- WRITE GOAL CODE TO DISK
  -- RETURN

RETURN

DISPLAY VALID CHOICES CHECK INPUT VALIDITY

IS SYNTAX COMPLETE

INPUT CHOICE

IS STATEMENT COMPLETE

POP DATA FROM RETURN STACK
APPENDIX G

LISTING OF COMPUTER AIDED GOAL CODING (CAGC) PROGRAM

100 'CAGC 11-21-79
110 'INITIALIZATION
120 REMINT A-Z
130 WIDTH=0
140 CLEAR 12*256
150 DIM A(3,30),L(3,17),M(2,10)
160 DIM X$(16),B$(22)
170 'LOAD ACTION CODE ARRAY
180 DATA 0,0,3,1,2,3,2,1,3,2,2,3,1,1,3,0,0,1,4,3,
0,0,0,4,1,3,4,2,3,0,0,1,2,2,2,1,2,2,2,1,
1,2,1,2,2,1,1,2,1
190 FOR K=1TO3:FOR J=1TO3:READ L(J,K):NEXTJ:NEXTK
200 'OPEN DISK FILE FOR GOAL CODE
210 GOTO 280 'TEMPORARY
220 INPUT "INPUT GOAL CODE FILE NAME";G$ 
230 ONERRORGOTO260:OPEN"I",2,G$:ONERRORGOTO0:CLOSE:PRINT$ 
240 INPUT "DUPLICATE FILE NAME-ENTER 1 IF OK (DESTROYS 
EXISTING FILE) OR ENTER 0 TO CHOOSE NEW NAME";X
250 IF X<>1 THEN 220 ELSE 270
260 RESUME
270 'OPEN "O",2,G$
280 'STATEMENT TYPE SELECTION
290 OPEN "I",1,"STMTS"
300 IF EOF(1) THEN CLOSE ELSE INPUT#1,AS:PRINT$;GOTO300
310 INPUT "CHOOSE STATEMENT NUMBER";N5
320 IF N5=0 THEN 1070
330 'SYNTAX DATA INPUT AND DISPLAY
340 N1=0:N2=1
350 IF N5<100 THEN SS="S"+STR$(N5)
ELSE SS="E"+RIGHT$(STR$(N5),3)
360 ONERROR GOTO370:OPEN"I",1,SS:ONERROR GOTO0:GOTO390
370 CLOSE:PRINT"ERROR CODE";ERR:"OCCURED IN LINE";ERL
380 IF ERL=53 THEN PRINT"FILE";SS;"DOES NOT EXIST";RESUME280
ELSE PRINT "CONTROL C TO STOP";X:RESUME 280
390 INPUT#1,I;FORJ=1TO1;FORK=1TO3;INPUT#1,A(K,J):NEXTK: 
NEXTJ
400 INPUT "+1,C$,K$
410 IF N2>3 THEN N2=1:N1=A(0,N1)MOD100
420 N3=A(0,N1)\100:IF A(N2,N1)=0 THEN N2=N2+1:GOTO 410
430 N4=L(N2,N3):IF N4<3 THEN 470
440 PRINT CHR$(27);"* ":PRINT:PRINTF 'Clears screen'
450 J=0
460 IF EOF(1) THEN 470
ELSE INPUT#1,B$(J):PRINTB$(J):J=J+1:GOTO 460
470 CLOSE 1
480 'Variables preset'
490 IF N2>3 THEN N2=1:N1=A(0,N1)MOD100:N3=A(0,N1)\100
500 IF N3=0 THEN N2=3
510 N4=L(N2,N3)
520 'Action logic selection'
530 ON N4 GOTO 560,740,770,590
540 INPUT "Invalid action code";X:GOTO 1070
550 'Keyw ord logic'
560 IF A(N2,N1)>0 THEN IF N5>199 THEN W$=MID$(B$(N1),4):GOSUB 970
ELSE W$=MID$(K$,A(N2,N1)\100,A(N2,N1)MOD100):GOSUB 970
570 N2=N2+1:GOTO 490
580 'Manual input logic'
590 INPUT W$
600 IF W$="ZZ" THEN N2=N2+1:GOTO 490
610 IF N5<299 THEN GOSUB 970:GOTO 590
620 IF N5>299 THEN W$="('+W$+'")":GOTO 640
630 W$="<"+W$+">
640 PRINT CHR$(27);"=";W$;
650 INPUT "Input line #; 0 to redo";X:IF X=0 THEN 590
660 IF X<10 THEN B$(X)=""+STR$(X)+"."+W$:
ELSE B$(X)=STR$(X)+"."+W$:
670 INPUT "Ready to change file?";D
680 OPEN "O",1,S$:PRINT #1,I
690 FOR J=0TOI:FOR K=0TO3:PRINT #1,A(K,J):NEXTK:NEXTJ
700 PRINT #1,CHR$(34);C$:CHR$(34);CHR$(34);K$:CHR$(34)
710 FOR J=0TOI:PRINT #1,CHR$(34);B$(J);CHR$(34):NEXTJ
720 CLOSE 1:GOSUB 970:GOTO 590
730 'Element logic'
740 IF A(N2,N1)=0 THEN N2=N2+1:GOTO 490
750 IF N5>199 THEN N5=N5:GOTO 330
ELSE N6=N6+1:GOTO 330
760 N5=A(N2,N1):GOTO 330
### APPENDIX D

#### ACTION CODE ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COLUMN 0</th>
<th>COLUMN 1</th>
<th>COLUMN 2</th>
<th>COLUMN 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**NOTES:**

1. **Action Codes:** 1=Keyword, 2=Element, 3=Choice, and 4=Manual Input.

2. Column zero was left blank so that the codes in the columns 1-3 would apply to columns 1-3 of the syntax data array columns 1-3 respectively.
APPENDIX E

CAGC SYNTAX DIAGRAMS

"STMTS" FILE DISPLAYED DURING STATEMENT SELECTION

*******COMMAND STMTS
S 11 APPLY ANALOG STMT
S 12 ISSUE DIGITAL PATTERN STMT
S 13 SET DISCRETE STMT
S 14 RECORD DATA STMT
*******RESPONSE STMTS
S 15 AVERAGE STMT
S 16 READ STMT
S 17 REQUEST KEYBOARD STMT
*******INTERNAL SEQUENCE CONTROL
S 31 GO TO/CONTINUE STMT
S 32 STOP STMT
S 33 DELAY STMT
S 34 REPEAT STMT
S 35 TERMINATE STMT
========BOUNDARY STMTS
S 71 BEGIN MACRO STMT
S 72 BEGIN PROGRAM STMT
S 73 BEGIN SUBROUTINE STMT
S 74 END STMT

*******ARITHMETIC/LOGICAL STMTS
S 41 ASSIGN STMT
S 42 LET EQUAL STMT
*******EXECUTION CONTROL STMTS
S 51 CONCURRENT STMT
S 52 RELEASE CONCURRENT STMT
S 53 PERFORM PROGRAM STMT
S 54 PERFORM SUBROUTINE STMT
*******EXECUTION CONTROL STMTS
S 61 WHEN INTERRUPT STMT
S 62 DISABLE INTERRUPT STMT
S 63 SUSPEND INTERRUPT STMT
S 64 ENABLE INTERRUPT STMT
S 65 ACTIVATE TABLE STMT
S 66 INHIBIT TABLE STMT
========LANGUAGE PROCESSOR DIRECTIVES
S 81 COMMENT STMT
S 82 EXPAND MACRO STMT
S 83 REPLACE STMT
S 84 USE DATA BANK STMT
S 85 DEFINE STMT
S 86 FORCED TEST END STMT
SYNTAX DATA FOR S 11

S 11

*PROCEPURAL*

*STATEMENT *

1 *PREFIX *

*APPLY *

*ANALOG *

*STATEMENT*
S 11 CONTINUED

CHOICE STRING (C$)
,1,2,3,4,5,6,7,8,;

KEYWORD STRING (K$)
SEND APPLY PRESENT VALUE OF TO , ;

<table>
<thead>
<tr>
<th>DATA ARRAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 12

S 12

*PROCT_STMT*

<table>
<thead>
<tr>
<th>ISSUE DIGITAL</th>
<th><em>PREP</em></th>
</tr>
</thead>
</table>

*PATTERN* |

*STATEMENT* |

<table>
<thead>
<tr>
<th>NUMBER</th>
<th></th>
</tr>
</thead>
</table>

A/------------------< 

| *EXTERNAL* |

\-------------------> 

| *DESIGNATOR* |

5

CHOICE STRING (C$)

,1,2,3,4,5,6,

KEYWORD STRING (K$)

ISSUE TO , ;

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>1202</td>
<td>101</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>107</td>
<td>0</td>
<td>505</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>115</td>
<td>0</td>
<td>905</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>400</td>
<td>0</td>
<td>905</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1101</td>
<td>0</td>
<td>505</td>
</tr>
<tr>
<td>6</td>
<td>1407</td>
<td>703</td>
<td>400</td>
<td>1301</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 13

S13
*PROCEDURAL*

*STATEMENT*

2-OPEN--

*EXTERNAL*

3-CLOSE--

*EXTERNAL*

7-FOR-*VALUE*

*DISCRETE*-

4-TURN-OFF--

8-TURN-ON/

5-TURN-ON/

6-/A

*EXTERNAL* V

11-*STATE*---

A/SET--<9-*DESIGNATOR*-TO--<

10

*EXTERNAL* V

12-*INTERNAL*/

*NAME*

-PRESENT-VALUE-OF-*EXTERNAL* *TO-*EXTERNAL *--;

*DESIGNATOR* *DESIGNATOR*
S 13 CONTINUED

**CHOICE STRING (C$)**
1, 2, 3, 4, 5, 6, 7, 8, 13, 9, 10, 11, 12.

**KEYWORD STRING (K3)**
OPEN CLOSE TURN ON TURN OFF FOR SET TO PRESENT VALUE OF .

**DATA ARRAY**

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>113</td>
</tr>
<tr>
<td>1</td>
<td>200</td>
<td>101</td>
<td>0</td>
<td>311</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>105</td>
<td>210</td>
<td>1305</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>606</td>
<td>210</td>
<td>1305</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>1208</td>
<td>210</td>
<td>1305</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>2009</td>
<td>210</td>
<td>1305</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>3304</td>
<td>0</td>
<td>2006</td>
</tr>
<tr>
<td>7</td>
<td>1408</td>
<td>2904</td>
<td>103</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>6001</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>9</td>
<td>200</td>
<td>210</td>
<td>3103</td>
<td>2507</td>
</tr>
<tr>
<td>10</td>
<td>1414</td>
<td>4017</td>
<td>220</td>
<td>3703</td>
</tr>
<tr>
<td>11</td>
<td>200</td>
<td>102</td>
<td>0</td>
<td>1308</td>
</tr>
<tr>
<td>12</td>
<td>200</td>
<td>400</td>
<td>0</td>
<td>1308</td>
</tr>
<tr>
<td>13</td>
<td>100</td>
<td>5801</td>
<td>0</td>
<td>2507</td>
</tr>
<tr>
<td>14</td>
<td>200</td>
<td>210</td>
<td>6001</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 14

S 14

*PROC STMT*
--2--DISPLAY--
*RECORD*
--3--PRINT--
*DATA*
--4--RECORD--
*STATEMENT*
--5--WRITE-->----/A

<table>
<thead>
<tr>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
</tr>
</tbody>
</table>

|--6--TEXT--*CONSTANT*-->

|--7--*INTERNAL*--->

|--8--*

|--9--*PRESENT-VALUE-OF-*EXTERNAL--*EXTERNAL--*

|--10--*COMMAND-STATUS-OF-*DESIGNATOR*--*

|--11--TO-*EXTERNAL--*

|--12--*DESIGNATOR*
CHOICE STRING (C$)
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 6, 7, 8, 11, 12,

KEYWORD STRING (K$)
DISPLAY PRINT RECORD WRITE TEXT PRESENT VALUE OF COMMAND STATUS OF TO , ;

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>111</td>
</tr>
<tr>
<td>1</td>
<td>.280</td>
<td>101</td>
<td>0</td>
<td>309</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>108</td>
<td>0</td>
<td>1112</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>906</td>
<td>0</td>
<td>1112</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>1507</td>
<td>0</td>
<td>1112</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>2206</td>
<td>0</td>
<td>1112</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>2805</td>
<td>111</td>
<td>2213</td>
</tr>
<tr>
<td>7</td>
<td>200</td>
<td>400</td>
<td>0</td>
<td>2213</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>7201</td>
<td>0</td>
<td>2213</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>3317</td>
<td>200</td>
<td>2807</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>5016</td>
<td>200</td>
<td>2807</td>
</tr>
<tr>
<td>11</td>
<td>1413</td>
<td>6803</td>
<td>260</td>
<td>7401</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>7401</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 15

S 15

*PROC STMT*

/*PREFIX*/

*AVERAGE* 1

*STATEMENT*<----------------->--2--AVERAGE--*NUMBER*--*READINGS-OF--*/

*(2TO100)*

*EXTERNAL*  

*INTERNAL*  

A/---*DESIGNATOR*--AND-SAVE-AS--*NAME*  

*------------------------------;

CHOICE STRING (C$)

',1,2,';

KEYWORD STRING (K$)

AVERAGE READINGS OF AND SAVE AS ;

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>1402</td>
<td>109</td>
<td>113</td>
<td>912</td>
</tr>
<tr>
<td>2</td>
<td>1203</td>
<td>200</td>
<td>2112</td>
<td>400</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3401</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 16

S 16
  *PROC STMT*
  
  *READ 1* /-*PREFIX -*
  */-2-READ-----/
  *STATEMENT*--<-----------------------<---3-MEASURE-->-------------*/
  */

  */-4-COMMAND-STATUS-OF-*DESIGNATOR*--/
  A/---<
  */-5-*EXTERNAL *------------------/
  A/---<
  */-5-*EXTERNAL *------------------/
  A/---<

CHOICE STRING (C$)
,1,2,3,4,5,

KEYWORD STRING (K$)
READ MEASURE COMMAND STATUS OF AND SAVE AS ;

DATA ARRAY
ROW  COL 0  COL 1  COL 2  COL 3
0    0    0    0    107
1    2    101   0    305
2    1    105   0    705
3    1    608   0    705
4    1106  1418  200  0
5    1206  200   0    0
6    1407  3212  400  4501
7    0    0    0    29999
SYNTAX DATA FOR 3 17

S 17

*TEXT
| 2--TEXT--*CONSTANT*

*PROC *

*REQUEST* 1

*INTERNAL*

*NAME*

*FUNCTION *

*INTERNAL*

A/---------FROM--*DESIGNATOR*--AND-SAVE-AS--*NAME*

CHOICE STRING (C$)
,1,2,3,4,5;

KEYWORD STRING (K$)
TEXT FROM AND SAVE AS , ;

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>111</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>101</td>
<td>0</td>
<td>309</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>105</td>
<td>111</td>
<td>309</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>400</td>
<td>0</td>
<td>309</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2401</td>
<td>0</td>
<td>309</td>
</tr>
<tr>
<td>5</td>
<td>1406</td>
<td>605</td>
<td>2406</td>
<td>1109</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>400</td>
<td>2601</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 31

S31

*PROC STMT*

*GO TO/ *

*CONTINUE *----<------------------->-<-2-GO-TO--*NUMBER*--------;

*STEP *

*STATEMENT* \

\-3--CONTINUE----------------------;

CHOICE STRING (C$)

',1,3,'.

KEYWORD STRING (K$)

GO TO ; CONTINUE

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>107</td>
</tr>
<tr>
<td>1</td>
<td>200</td>
<td>101</td>
<td>0</td>
<td>305</td>
</tr>
<tr>
<td>2</td>
<td>1404</td>
<td>106</td>
<td>117</td>
<td>801</td>
</tr>
<tr>
<td>3</td>
<td>400</td>
<td>909</td>
<td>801</td>
<td>29999</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 32

S 32

*PROCEDURAL*
/*STATEMENT */<---------------------->3-----;

*STOP *
*PREFIX *

1

*STATEMENT*<---------------------->2-STOP<------4----/A

|----- , -----|
| V *STEP * |

A/---AND-INDICATE-RESTART-LABELS-----*NUMBER*---^6-----;

CHOICE STRING (C$)
,1,2,3,4,5,6,

KEYWORD STRING (K$)
STOP AND INDICATE RESTART LABELS , ;

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>200</td>
<td>101</td>
<td>0</td>
<td>305</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>105</td>
<td>0</td>
<td>505</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>3701</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>628</td>
<td>117</td>
<td>905</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>3501</td>
<td>117</td>
<td>905</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>3701</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR  S 33

S 33

*PROC STMT*  /-2-DELAY-/
*DELAY*  /-1-*PREFIX*  *-
*STATEMENT*--<---------------------------------->--<---3-WAIT----->--------/>A

/-4------------------------
A/<--- *TIME*  /-6-OR>-UNTIL-*DESIGNATOR*----*TEST* -->---;
/-5-*VALUE*<---7----------------------------------------/

CHOICE STRING (CS)
,1,2,3,4,5,6,7,

KEYWORD STRING (K$)
DELAY WAIT OR UNTIL ;

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>107</td>
</tr>
<tr>
<td>1</td>
<td>200</td>
<td>101</td>
<td>0</td>
<td>305</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>106</td>
<td>0</td>
<td>705</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>705</td>
<td>0</td>
<td>735</td>
</tr>
<tr>
<td>4</td>
<td>1107</td>
<td>1506</td>
<td>200</td>
<td>104</td>
</tr>
<tr>
<td>5</td>
<td>200</td>
<td>103</td>
<td>0</td>
<td>1105</td>
</tr>
<tr>
<td>6</td>
<td>1104</td>
<td>1203</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>2201</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 34

S 34

*PROC STMT*

*/PREFIX */

*REPEAT * 1

*STEP *

*STATEMENT* -> 2 -> REPEAT-THROUGH -> *NUMBER* -> /A

*INTERNAL*

A/-<-3--FOR<- 6-*INTEGER*- >T I M E S-< 8-OR->-UNTIL--*EXTERNAL *--*COMPARISON*->-;

*DESIGNATOR* *TEST*

CHOICE STRING (C$)

,1,2,3,4,5,6,7,8,

KEYWORD STRING (K$)

REPEAT THROUGH FOR TIMES OR UNTIL:

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>1202</td>
<td>101</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>115</td>
<td>117</td>
<td>505</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>1604</td>
<td>0</td>
<td>905</td>
</tr>
<tr>
<td>4</td>
<td>1107</td>
<td>2906</td>
<td>0</td>
<td>104</td>
</tr>
<tr>
<td>5</td>
<td>200</td>
<td>400</td>
<td>2006</td>
<td>1305</td>
</tr>
<tr>
<td>6</td>
<td>200</td>
<td>113</td>
<td>2006</td>
<td>1305</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>3601</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>8</td>
<td>1104</td>
<td>2603</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 35

S 35

*PROC STMT*

/*PREFIX */

/*TERMINATE*/ 1

/*STATEMENT*/<--2-------------->TERMINATE<--4-------------->;

CHOICE STRING (C$)
1, 2, 3, 4,

KEYWORD STRING (K$)
TERMINATE SYSTEM ;

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>1202</td>
<td>101</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>111</td>
<td>0</td>
<td>505</td>
</tr>
<tr>
<td>3</td>
<td>1104</td>
<td>1107</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>1901</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 41

S 41  *PROC *
     /*STMT */
     *ASSIGN* 1 *PREFIX* 
     *STMT ---<2------------->ASSIGN--*NAME ---<4-EQUAL-TO>---<6-*INTERNAL*-->; 
     *NAME *

CHOICE STRING (C$)
,1,2,3,4,5,6,

KEYWORD STRING (K$)
ASSIGN = EQUAL TO ;

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>1202</td>
<td>101</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>107</td>
<td>400</td>
<td>505</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>802</td>
<td>0</td>
<td>905</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>1009</td>
<td>0</td>
<td>905</td>
</tr>
<tr>
<td>5</td>
<td>200</td>
<td>102</td>
<td>2001</td>
<td>29999</td>
</tr>
<tr>
<td>6</td>
<td>200</td>
<td>400</td>
<td>2001</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 42

S 42 *PROC *

/*STMT */

*LET * 1 *PREFIX* \\ *INTERNAL* /3--- = ---\ *NUMERIC*
*EQUAL*<--2----------------->LET--*NAME *--<4-EQUAL-TO>--*FORMULA*--

*STMT *

CHOICE STRING (C$)
,1,2,3,4,

KEYWORD STRING (K$)
LET = EQUAL TO ;

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>1202</td>
<td>101</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>102</td>
<td>104</td>
<td>400</td>
<td>505</td>
</tr>
<tr>
<td>3</td>
<td>1105</td>
<td>502</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1105</td>
<td>709</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>200</td>
<td>110</td>
<td>1701</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 51

S 51

*PROC* *TIME*

*/-STMT*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/STATEMENT*/-EVERY--*VALUE*-

*/S'rtvlT*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-

*/PREFIX*/-EVERY--*VALUE*-
SYNTAX DATA FOR  S 52

S 52
  *PROC *
  /-*STMT *-
  *RELEASE * 1 *PREFIX* /-*S~MT*-*|
  *CONCURRENT*--<--2------------->RELEASE--<--3-----*NUMBER*--^-6--->--;
  *STATEMENT* 4
  \-------ALL-------/

CHOICE STRING (C$)
,1,2,3,4,5,6,

KEYWORD STRING (K$)
RELEASE ALL , ;

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>1202</td>
<td>101</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>108</td>
<td>0</td>
<td>505</td>
</tr>
<tr>
<td>3</td>
<td>200</td>
<td>117</td>
<td>0</td>
<td>905</td>
</tr>
<tr>
<td>4</td>
<td>1106</td>
<td>904</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1103</td>
<td>1401</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>1601</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 53

S 53

*PROC*

/*STMT*/

*PERFORM* 1 *PREFIX* \ *PROGRAM* 3

*PROGRAM*<-2----------->-PERFORM-*NAME* *----<4----------->--

*STMT*

CHOICE STRING (C$)

,1,2,3,4,

KEYWORD STRING (K$)

PERFORM ;

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>1202</td>
<td>101</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>108</td>
<td>150</td>
<td>505</td>
</tr>
<tr>
<td>3</td>
<td>124</td>
<td>14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>1001</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 54

S 54
*PROC*

*PERFORM 1 *PREFIX*

*SUBROUTINE* --<2-------------->--PERFORM--<4-------------->--SUBROUTINE--/A

*STATEMENT*

*SUBROUTINE* 6

/*PARAMETER*/

/A----*NAME*--<7-------------->----------------------------------~

CHOICE STRING (C$)
1, 2, 3, 4, 5, 6, 7,

KEYWORD STRING (K$)
PERFORM CRITICAL SYSTEM SUBROUTINE ;

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>1202</td>
<td>101</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>109</td>
<td>0</td>
<td>507</td>
</tr>
<tr>
<td>3</td>
<td>1104</td>
<td>909</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>2511</td>
<td>151</td>
<td>1105</td>
</tr>
<tr>
<td>5</td>
<td>1104</td>
<td>1807</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1207</td>
<td>140</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>3701</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 61

S 61

*PROC *

*WHEN * 1 *PREFIX* 

*INTERRUPT*<--2---------->WHEN-INTERRUPT-*DESIGNATOR*--OCCURS------*/A

*STATEMENT*

*FUNCTION *

*STEP * 

*/GO-TO---*NUMBER*-----------------*/A

*PERFORM * 

A/><3-SUSPEND-INTERRUPTS-AND-><5-*SUBROUTINE*-AND-RETURN<-6-TO-*NUMBER*--;

*STATEMENT * 

CHOICE STRING (C$) 

,1,2,3,4,5,6,7,

KEYWORD STRING (K$) 

WHEN INTERRUPT OCCURS SUSPEND INTERRUPTS AND RETURN GO TO ;

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1202</td>
<td>101</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>1408</td>
<td>115</td>
<td>200</td>
<td>1607</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>2323</td>
<td>0</td>
<td>705</td>
</tr>
<tr>
<td>3</td>
<td>1107</td>
<td>5306</td>
<td>117</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>200</td>
<td>54</td>
<td>4111</td>
<td>1105</td>
</tr>
<tr>
<td>5</td>
<td>1107</td>
<td>5603</td>
<td>117</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>6091</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>507</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 62

S 62

PROC

/*STMT*/

DISABLE 1 PREFIX

INTERUPT--<2------------->DISABLE--<3--NUMBER--<6>-;

STATEMENT

CHOICE STRING (C$)

,1,2,3,4,5,6,

KEYWORD STRING (K$)

DISABLE ALL , ;

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>1202</td>
<td>101</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>108</td>
<td>0</td>
<td>505</td>
</tr>
<tr>
<td>3</td>
<td>200</td>
<td>117</td>
<td>0</td>
<td>905</td>
</tr>
<tr>
<td>4</td>
<td>1106</td>
<td>904</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1103</td>
<td>1401</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>1601</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 63

S 63

*PROC *
    /*STMT */
    *SUSPEND * 1 *PREFIX* \
    *INTERRUPT*----<-2---------->----SUSPEND-ALL----;
    *STATEMENT*

CHOICE STRING (C$)
,1,2,

KEYWORD STRING (K$)
    SUSPEND ALL;

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>1202</td>
<td>101</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>113</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 64

S 64

*PROC *

/*STMT */

*ENABLE  1 *PREFIX*

*INTERUPT*--->-2--------->-----ENABLE-ALL-----;

*STATEMENT*

CHOICE STRING (C$)

,1,2,

KEYWORD STRING (K$)

ENABLE ALL;

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>1202</td>
<td>101</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>112</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 65

S 65

*PROC *
*/STMT *

*ACTIVATE* 1 *PREFIX* \ *TABLE* \ *STMT *

<--2----->-ACTIVATE--*NAME* --><

CHOICE STRING (C$)
,1,2,3,4,5,6,7,3,8,

KEYWORD STRING (K$)
ACTIVATE ROW , ;

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>1202</td>
<td>101</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>109</td>
<td>400</td>
<td>507</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>1701</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>1084</td>
<td>0</td>
<td>1105</td>
</tr>
<tr>
<td>5</td>
<td>200</td>
<td>125</td>
<td>0</td>
<td>1505</td>
</tr>
<tr>
<td>6</td>
<td>200</td>
<td>113</td>
<td>0</td>
<td>1505</td>
</tr>
<tr>
<td>7</td>
<td>200</td>
<td>124</td>
<td>0</td>
<td>1505</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>1501</td>
<td>0</td>
<td>705</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR S 66

S 66

*PROC *
/*STMT */
*INHIBIT* 1 *PREFIX* \ 
*TABLE* *--<-2---------->INHIBIT-*NAME *--<-<
*STMT *

CHOICE STRING (C$)
,1,2,3,4,5,6,7,3,8

KEYWORD STRING (K$)
INHIBIT ROW , ;

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>1202</td>
<td>101</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>108</td>
<td>400</td>
<td>507</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>1601</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>904</td>
<td>0</td>
<td>1105</td>
</tr>
<tr>
<td>5</td>
<td>200</td>
<td>125</td>
<td>0</td>
<td>1505</td>
</tr>
<tr>
<td>6</td>
<td>200</td>
<td>113</td>
<td>0</td>
<td>1505</td>
</tr>
<tr>
<td>7</td>
<td>200</td>
<td>124</td>
<td>0</td>
<td>1505</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>1401</td>
<td>0</td>
<td>705</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR   E101

E101

*PROCE~UFAL* 1

*STEP* 1

*TIME* 2

*VERIF~Y* 3

*/-NUMBER*/ 4

*/-PREFIX*/ 5

*/-PREFIX*/ 6

*STATEMENT* |

*PREFIX* |

CHOICE STRING (C$) ,1,2,3,4,

KEYWORD STRING (K$) NULL

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>107</td>
</tr>
<tr>
<td>1</td>
<td>200</td>
<td>117</td>
<td>0</td>
<td>307</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>118</td>
<td>0</td>
<td>505</td>
</tr>
<tr>
<td>3</td>
<td>200</td>
<td>106</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR   E103

E103    /*INTERNAL*---------------------------------------------
         /*NAME* .
         /*TIME* 1/*DECIMAL*/ 3-DAY--*//*DECIMAL*/ 6-HR--/
*VALUE*<-2->*NUMBER*<-4->*DAYS--<-5->*NUMBER*<-8->*HOURS--/A
*      /*9->*HRS--/

         /*DECIMAL*/ /11-MIN--

         /*DECIMAL*/ /12-MINUTE--
         /*DECIMAL*/ /16-SEC--
B/<-20->*NUMBER*<-23->*MILLISECONDS--<-26->

         /21-MSEC------
         /25-GMT--

CHOOSE STRING (C$)
,1,2,5,10,15,20,25,27,3,4,5,25,26,27,6,7,8,9,10,25,26,27,11,12,13,14,15,25,26,27
,16,17,18,19,20,25,26,27,21,22,23,24,

KEYWORD STRING (K$)
DAYS HRS HOURS MINS MINUTES SECS SECONDS MSECS MILLISECONDS GMT CDT
## DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>122</td>
</tr>
<tr>
<td>1</td>
<td>200</td>
<td>400</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>2</td>
<td>900</td>
<td>108</td>
<td>0</td>
<td>2205</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>104</td>
<td>0</td>
<td>2612</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>105</td>
<td>0</td>
<td>2612</td>
</tr>
<tr>
<td>5</td>
<td>900</td>
<td>108</td>
<td>0</td>
<td>3709</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>603</td>
<td>0</td>
<td>4613</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>1005</td>
<td>0</td>
<td>4613</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>1006</td>
<td>0</td>
<td>4613</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>604</td>
<td>0</td>
<td>4613</td>
</tr>
<tr>
<td>10</td>
<td>900</td>
<td>100</td>
<td>0</td>
<td>5813</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1604</td>
<td>0</td>
<td>7013</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>2107</td>
<td>0</td>
<td>7013</td>
</tr>
<tr>
<td>13</td>
<td>100</td>
<td>2108</td>
<td>0</td>
<td>7013</td>
</tr>
<tr>
<td>14</td>
<td>100</td>
<td>1605</td>
<td>0</td>
<td>7013</td>
</tr>
<tr>
<td>15</td>
<td>900</td>
<td>108</td>
<td>0</td>
<td>8213</td>
</tr>
<tr>
<td>16</td>
<td>100</td>
<td>2904</td>
<td>0</td>
<td>9413</td>
</tr>
<tr>
<td>17</td>
<td>100</td>
<td>3407</td>
<td>0</td>
<td>9413</td>
</tr>
<tr>
<td>18</td>
<td>100</td>
<td>3408</td>
<td>0</td>
<td>9413</td>
</tr>
<tr>
<td>19</td>
<td>100</td>
<td>2905</td>
<td>0</td>
<td>9413</td>
</tr>
<tr>
<td>20</td>
<td>900</td>
<td>108</td>
<td>0</td>
<td>10613</td>
</tr>
<tr>
<td>21</td>
<td>100</td>
<td>4205</td>
<td>0</td>
<td>4910</td>
</tr>
<tr>
<td>22</td>
<td>100</td>
<td>4812</td>
<td>0</td>
<td>4910</td>
</tr>
<tr>
<td>23</td>
<td>100</td>
<td>4813</td>
<td>0</td>
<td>4910</td>
</tr>
<tr>
<td>24</td>
<td>100</td>
<td>4206</td>
<td>0</td>
<td>4910</td>
</tr>
<tr>
<td>25</td>
<td>100</td>
<td>6104</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>26</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>27</td>
<td>100</td>
<td>6504</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR  E104

E104

*COMPARISON*  
*TEST*  
+  

CHOOSE STRING (C$)
,1,2,

KEYWORD STRING (K$)

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>200</td>
<td>121</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>120</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR E105

E105

*OUTPUT

*EXCEPTIONS --<---3-RECORD--->---EXCEPTIONS------------------------/A

*TEXT

*CONSTANT

/DEFAULT TO SYSTEM MESSAGE

6- USING MESSAGES FROM --*INTERNAL*--

*EXTERNAL

*DESIGNATOR

/DEFAULT TO SYSTEM DEVICE

CHOICE STRING (C$)
,1,2,3,4,5,6,7,8,

KEYWORD STRING (K$)
DISPLAY PRINT WRITE EXCEPTIONS USING MESSAGES FROM TO

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>109</td>
</tr>
<tr>
<td>1</td>
<td>400</td>
<td>108</td>
<td>2811</td>
<td>909</td>
</tr>
<tr>
<td>2</td>
<td>400</td>
<td>906</td>
<td>2811</td>
<td>909</td>
</tr>
<tr>
<td>3</td>
<td>400</td>
<td>1507</td>
<td>2811</td>
<td>909</td>
</tr>
<tr>
<td>4</td>
<td>400</td>
<td>2106</td>
<td>2811</td>
<td>909</td>
</tr>
<tr>
<td>5</td>
<td>200</td>
<td>111</td>
<td>0</td>
<td>1305</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>3920</td>
<td>400</td>
<td>1305</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>5903</td>
<td>260</td>
<td>29999</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
**SYNTAX DATA FOR E106**

```
E106
     *COMPARISON* /3-THEN-
     /---< *---< \
1    \4-, --/
*VERIFY* /
*PREFIX*<---2-VERIFY---*EXTERNAL *---*COMPARISON*-----------/A
  *  * DESIGNATOR  TEST
*TIME*
5-WITHIN--*VALUE*-
A/---<--------------------------->---<

```

**CHOICE STRING (C$)**
,1,2,5,3,4,6,7,6,8,

**KEYWORD STRING (K$)**
THEN , VERIFY WITHIN ELSE AND

**DATA ARRAY**

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>200</td>
<td>104</td>
<td>0</td>
<td>705</td>
</tr>
<tr>
<td>2</td>
<td>1109</td>
<td>807</td>
<td>200</td>
<td>104</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>105</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>701</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>1507</td>
<td>103</td>
<td>709</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>2205</td>
<td>0</td>
<td>1505</td>
</tr>
<tr>
<td>8</td>
<td>200</td>
<td>105</td>
<td>2704</td>
<td>29999</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>511</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR E107

E107

DEVELOPMENT ONLY

-1-*DECIMAL*-<->-3-*DIMENSION*-->

*NUMBER* 4

*TIME*

*VALUE*

CHOICE STRING (C$)

1, 2, 3, 5, 3, 4,

KEYWORD STRING (K$)

AUTO

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>109</td>
</tr>
<tr>
<td>1</td>
<td>200</td>
<td>108</td>
<td>0</td>
<td>905</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>103</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>3</td>
<td>200</td>
<td>190</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>105</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR  E108

E108

TYPE A DECIMAL NUMBER IN THE FOLLOWING FORMAT:

+/-  XXXXX.YYYYYYEE+/--ZZZZZ

SIGNS, EXPONENT, DECIMAL, AND LEADING AND TRAILING ZEROS ARE NOT MANDATORY.

ESCAPE TO ENTER.

CHOICE STRING (C$)

C$ NULL

KEYWORD STRING (K$)

K$ SPACE IN 1 IS SIGNIFICANT

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>600</td>
<td>101</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR  E111

E111

TEXT CONSTANT - TYPE A CHARACTER STRING.

ALL PRINTABLE ASCII CHARACTERS ARE VALID EXCEPT PARENTHESIS.
OPEN AND CLOSE PARENTHESIS ARE SUPPLIED BY THE PROGRAM.

CHOICE STRING (C$)
C$ IS NULL

KEYWORD STRING (K$)
()

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1101</td>
<td>102</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>800</td>
<td>0</td>
<td>303</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR E113

E113

TYPE AN INTEGER NUMBER - 1 TO 5 NUMERALS.
STEP NUMBER IS LIMITED TO 4 NUMERALS.
ESCAPE TO ENTER.

CHOICE STRING (C$)
C$ NULL

KEYWORD STRING (K$)
K$ SPACE IN 1 SIGNIFICANT

DATA ARRAY
ROW COL 0 COL 1 COL 2 COL 3
0 600 101 0 29999
SYNTAX DATA FOR E115

E115 NUMBER PATTERN - TYPE ONE OF THE FOLLOWING NUMBER PATTERNS

INTEGER  +/- ZZZZZ (Z=0 TO 9) =-32767 MIN TO +32767 MAX

BINARY  BZZZZZZZZZZZZZZZ (Z=0 OR 1) 16 DIGITS MAX

OCTAL  TZZZZZZZ (Z=0 TO 7) 177777 MAX

HEX  XZZZZZ (Z=0 TO 9 OR A TO F)

CHOICE STRING (C$)
C$ NULL

KEYWORD STRING (K$)
K$ NULL

DATA ARRAY
ROW  COL 0  COL 1  COL 2  COL 3
0  600  101  1  29999
SYNTAX DATA FOR E117

E117

|-------------------| 3 TYPE A MAX OF 4 NUMERALS |
|                  | | |
| *STEP  *         | *   |   |
| *NUMBER*------STEP------*NUMERAL*------ |

CHOICE STRING (C$)

,1,

KEYWORD STRING (K$)
STEP

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
<td>105</td>
<td>113</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR E118

E118

*TIME * /1-AFTER-* *FUNCTION * *TIME * /3-THEN-*
*PREFIX*--<  >-*DESIGNATOR*--IS---*VALUE*--< ---
*       * 2-WHEN--/ *       *       *       * 4-- , -/

CHOICE STRING (C$)
,1,2,3,4,

KEYWORD STRING (K$)
AFTER WHEN IS THEN ,

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>1</td>
<td>1405</td>
<td>106</td>
<td>250</td>
<td>1203</td>
</tr>
<tr>
<td>2</td>
<td>1405</td>
<td>705</td>
<td>250</td>
<td>1203</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>1505</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>2101</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>5</td>
<td>200</td>
<td>103</td>
<td>0</td>
<td>505</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR E120

E120

*LIMIT * /-2-IS- \ /4-----/
*FORMULA*--<1-*NAME *---><--3-ARE--><--5-NOT-->--BETWEEN--->/A

* * * * *
/6--*QUANTITY*--AND--*QUANTITY*--
/A--<--7--*NUMBER*--AND--*NUMBER*

/8--*NAME*--AND--*NAME*

CHOICE STRING (C$)
,1,2,3,4,5,6,7,8,

KEYWORD STRING (K$)
IS ARE NOT BETWEEN AND

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>107</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>200</td>
<td>400</td>
<td>0</td>
<td>305</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>103</td>
<td>0</td>
<td>705</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>404</td>
<td>0</td>
<td>705</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>1208</td>
<td>0</td>
<td>1107</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>812</td>
<td>0</td>
<td>1107</td>
</tr>
<tr>
<td>6</td>
<td>1209</td>
<td>107</td>
<td>2004</td>
<td>107</td>
</tr>
<tr>
<td>7</td>
<td>1209</td>
<td>108</td>
<td>2004</td>
<td>108</td>
</tr>
<tr>
<td>8</td>
<td>1209</td>
<td>400</td>
<td>2004</td>
<td>400</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>0</td>
<td>29999</td>
<td>0</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR E121

E121
*RELATIONAL* *INTERNAL*
*FORMULA* *→<1-*NAME* *→* /A

*DECIMAL*
-12-*NUMBER*

*NUMBER*
-13-*PATTERN*

*TEXT*
-14-*CONSTANT*

*TEXT*
-15-*INTERNAL*

*NAME*
-16-*QUANTITY*

*STATE*
E121 CONTINUED

CHOICE STRING (C$)
,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,

KEYWORD STRING (K$)
IS ARE = NOT EQUAL TO GREATER THAN OR EQUAL TO LESS THAN OR EQUAL TO TEXT

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>109</td>
</tr>
<tr>
<td>1</td>
<td>200</td>
<td>400</td>
<td>0</td>
<td>307</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>103</td>
<td>0</td>
<td>719</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>404</td>
<td>0</td>
<td>719</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>802</td>
<td>0</td>
<td>2513</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>1409</td>
<td>0</td>
<td>2513</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>1013</td>
<td>0</td>
<td>2513</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>2313</td>
<td>0</td>
<td>2513</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>4810</td>
<td>0</td>
<td>2513</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>2325</td>
<td>0</td>
<td>2513</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>4822</td>
<td>0</td>
<td>2513</td>
</tr>
<tr>
<td>11</td>
<td>200</td>
<td>102</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>12</td>
<td>200</td>
<td>108</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>13</td>
<td>200</td>
<td>115</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>14</td>
<td>100</td>
<td>7005</td>
<td>111</td>
<td>29999</td>
</tr>
<tr>
<td>15</td>
<td>200</td>
<td>400</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>16</td>
<td>200</td>
<td>107</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR E150

E150

1. TEST
2. SAMPLE

PROG NAME

17. NEXT PAGE  18. NOT USED  19. RETURN  20. ENTER NEW NAME
E150 CONTINUED

CHOICE STRING (CS)
,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20.

KEYWORD STRING (K$)
()

<table>
<thead>
<tr>
<th>DATA ARRAY</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW</td>
<td>COL 0</td>
<td>COL 1</td>
<td>COL 2</td>
<td>COL 3</td>
</tr>
<tr>
<td>0</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>1</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>2</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>3</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>4</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>5</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>6</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>7</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>8</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>9</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>10</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>11</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>12</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>13</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>14</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>15</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>16</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>17</td>
<td>300</td>
<td>0</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>18</td>
<td>300</td>
<td>0</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>19</td>
<td>300</td>
<td>0</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>20</td>
<td>800</td>
<td>111</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR E190

E190

DIMENSION - TYPE IN A DIMENSION FROM TABLE.

CHOICE STRING (C$)

KEYWORD STRING (K$)

DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>600</td>
<td>101</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
SYNTAX DATA FOR E200

EXTERNAL DESIGNATOR TYPES

1. DISCRETE LOAD (E210)
2. DISCRETE RESPONSES (E220)
3. ANALOG LOAD (E230)
4. ANALOG RESPONSE (E240)
5. FUNCTION DESIGNATOR (E250)
6. SYSTEM DEVICE (E260)

CHOICE STRING (C$)
,1,2,3,4,5,6,

KEYWORD STRING (K$)
K$ NULL

DATA ARRAY
ROW  COL 0  COL 1  COL 2  COL 3
0     0     0     0     0     113
1     300   210   0     0     29999
2     300   220   0     0     29999
3     300   230   0     0     29999
4     300   240   0     0     29999
5     300   250   0     0     29999
6     300   260   0     0     29999
TYPICAL EXTERNAL DESIGNATOR LIST (ONE OF MANY TYPES)
EACH TYPE MAY HAVE MULTIPLE LISTS.

SYNTAX DATA FOR E240

EXTERNAL DESIGNATOR (ANALOG RESPONSE)
1. <ANALOG RESPONSE>
2. <0589 $MAIN BUS VOLTAGE$>

17. NEXT PAGE  18. DIFFERENT TYPE  19. RETURN  20. ENTER NEW <FD>
CHOICE STRING (C$)
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,

KEYWORD STRING (K$)
K$ NULL

<table>
<thead>
<tr>
<th>DATA ARRAY</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>400</td>
<td>1</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>400</td>
<td>1</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>400</td>
<td>1</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>400</td>
<td>1</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>400</td>
<td>1</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>400</td>
<td>1</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>400</td>
<td>1</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>400</td>
<td>1</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>400</td>
<td>1</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>400</td>
<td>1</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>400</td>
<td>1</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>400</td>
<td>1</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>400</td>
<td>1</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>400</td>
<td>1</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>400</td>
<td>1</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>400</td>
<td>1</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>300</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>300</td>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>300</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>800</td>
<td>111</td>
<td>0</td>
</tr>
</tbody>
</table>
### SYNTAX DATA FOR E400

**INTERNAL NAME TYPES**

1. SINGLE STATE  \( (E310) \)
2. STATE LIST  \( (E320) \)
3. STATE TABLE  \( (E330) \)
4. SINGLE QUANTITY  \( (E410) \)
5. QUANTITY LIST  \( (E420) \)
6. QUANTITY TABLE  \( (E430) \)

**CHOICE STRING (C$)**

,1,2,3,4,5,6.

**KEYWORD STRING (K$)**

#### DATA ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>113</td>
</tr>
<tr>
<td>1</td>
<td>300</td>
<td>310</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>2</td>
<td>300</td>
<td>320</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>3</td>
<td>300</td>
<td>330</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>4</td>
<td>300</td>
<td>410</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>5</td>
<td>300</td>
<td>420</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>6</td>
<td>300</td>
<td>430</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
TYPICAL INTERNAL NAME LIST (1 OF 12 TYPES)
EACH TYPE MAY HAVE MULTIPLE LISTS

SYNTAX DATA FOR  E310

E310  INTERNAL NAME (SINGLE STATE)
  1. (SINGLE STATE)
  2. (PWR ON RELAY STATUS)
  3. (ECS VALVE STATUS)
     4.
     5.
     6.
     7.
     8.
     9.
    10.
    11.
    12.
    13.
    14.
    15.

17. NEXT PAGE  18. DIFFERENT TYPE  19. RETURN  20. ENTER NEW (INTERNAL NAME)
E310 CONTINUED

CHOICE STRING (C$)
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,

KEYWORD STRING (K$)
K$ N,J,L

<table>
<thead>
<tr>
<th>DATA ARRAY</th>
<th>COL 0</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>1</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>2</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>3</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>4</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>5</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>6</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>7</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>8</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>9</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>10</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>11</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>12</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>13</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>14</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>15</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>16</td>
<td>400</td>
<td>1</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>17</td>
<td>300</td>
<td>0</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>18</td>
<td>300</td>
<td>200</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>19</td>
<td>300</td>
<td>0</td>
<td>0</td>
<td>29999</td>
</tr>
<tr>
<td>20</td>
<td>800</td>
<td>111</td>
<td>0</td>
<td>29999</td>
</tr>
</tbody>
</table>
LIST OF REFERENCES


