# DOCUMENT RESUME


ED 115 204 IR 002 489

| | |
|---|---|
| AUTHOR | Endres, Frank L. |
| TITLE | SIMPLE: An Introduction. |
| INSTITUTION | Texas Univ., Austin. Project C-BE. |
| SPONS AGENCY | National Science Foundation, Washington, D.C. |
| REPORT NO | EP-33-10-17-74 |
| PUB DATE | 17 Oct 74 |
| NOTE | 23p.; For related documents see IR 002 463 and 464 |
| EDRS PRICE | MF-$0.76 HC-$1.58 Plus Postage |
| DESCRIPTORS | Algebra; *College Mathematics; Computer Assisted Instruction; *Computer Programs; Higher Education; Mathematics Education; Matrices; Programing; Programing Languages; Time Sharing |
| IDENTIFIERS | Interactive Computer Languages; Project C BE; *SIMPLE |



ABSTRACT

Symbolic Interactive Matrix Processing Language (SIMPLE) is a conversational matrix-oriented source language suited to a batch or a time-sharing environment. The two modes of operation of SIMPLE are conversational mode and programing mode. This program uses a TAURUS time-sharing system and cathode ray terminals or teletypes. SIMPLE performs all the standard operations on matrices, and eigenvalues, eigenvectors, remote plotting, and inquiry commands are also available. Self-paced instruction through TEACH commands allows the student to proceed at his own pace while maintaining full documentation. This paper is an introduction for the novice and a reference guide for the casual user. The SIMPLE Reference Manual provides a more rigorous definition for the experienced user. (CH)

# SIMPLE: AN INTRODUCTION

EP-33/10/17/74

Frank L. Endres
Department of Civil Engineering
The University of Texas at Austin





*The materials contained herein were supported by PROJECT C-BE under Grant GY-9340, "The Use of Computer-Based Teaching Techniques in Undergraduate Science and Engineering Education," from the National Science Foundation to The University of Texas at Austin, Drs. John J. Allan and J. J. Lagowski, Co-Directors.

2/3

4/5

TABLE OF CONTENTS

## PREFACE

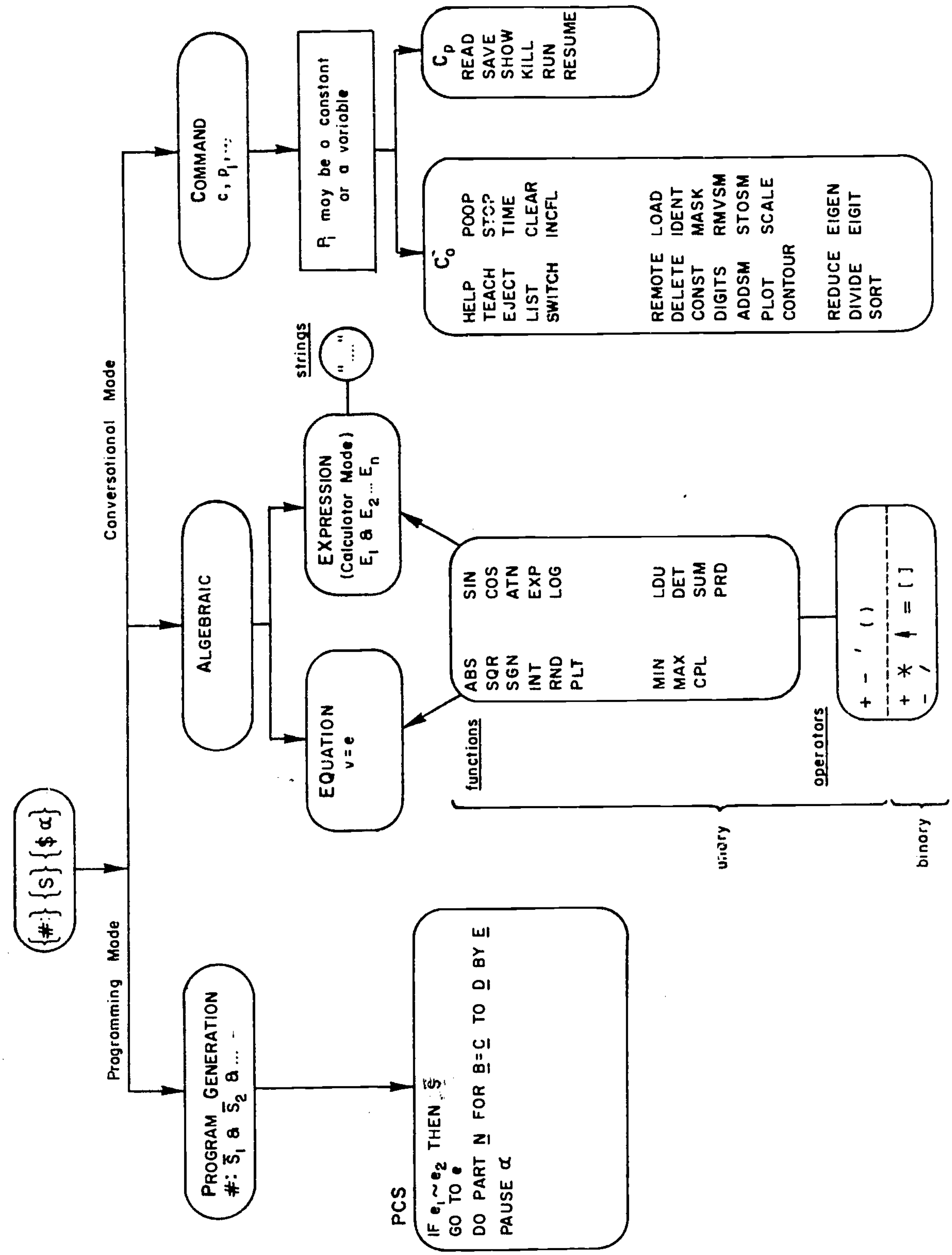
SIMPLE (Symbolic Interactive Matrix Processing LanguagE) is a conversational matrix oriented source language well suited to either a batch or a time-sharing environment.

This document is not intended to be an instructional text nor a programmer's reference manual, but rather an introduction for the novice and a quick reference guide for the casual user. It does not attempt to tell the whole story and in some instances overstates constraints for the sake of simplicity and clarity.

Chapter 0 would normally be appendix A in most texts, but has been included initially in the hope of capturing the reader's interest and demonstrating SIMPLE's versatility.

The description of SIMPLE presented herein is basically non-technical and an informal approach is taken in hopes of enhancing initial comprehension. A forthcoming edition (SIMPLE - a Reference Manual) will provide a more rigorous definition for the experienced user. And SIMPLE - an Applications Manual will demonstrate the use of SIMPLE over a wide range of applications.

The author would like to express his gratitude for support received from the Department of Civil Engineering and NSF Grant GY-9340, project Computer Based Education (C-BE) of the National Science Foundation.



{#:} {S} {$α}
Programming Mode
Conversational Mode
PROGRAM GENERATION
#: $\bar{S}_1$ & $\bar{S}_2$ & ...
PCS
IF $e_1$~$e_2$ THEN $\bar{S}$
GO TO e
DO PART N FOR B=C TO D BY E
PAUSE α
ALGEBRAIC
EQUATION
v=e
EXPRESSION
(Calculator Mode)
$E_1$ & $E_2$ ... $E_n$
strings
" ... "
functions
ABS SIN
SQR COS
SGN ATN
INT EXP
RND LOG
PLT
MIN LDU
MAX DET
CPL SUM
PRD
operators
+ - ' ( )
+ * ↑ = [ ]
- /
unary
binary
COMMAND
c, $p_1$, ...;
$P_i$ may be a constant or a variable
$C_o$
HELP POOP
TEACH STOP
EJECT TIME
LIST CLEAR
SWITCH INCFL
REMOTE LOAD
DELETE IDENT
CONST MASK
DIGITS RMVSM
ADDSM STOSM
PLOT SCALE
CONTOUR
REDUCE EIGEN
DIVIDE EIGIT
SORT
$C_p$
READ
SAVE
SHOW
KILL
RUN
RESUME

## 0.0 Example

```
TAURUS - 18 SEP 74  10.17.07
Q =CEAB123/NANCY/ABC
JOBNAME: CEAB123-20
  CHARGES YEAR-TO-DATE:  TIME $43.48  SUPPLIES $7.79
CC:
EXECPF 2450 EASIER
GO:
 END - EASIER
CC:
?SIMPLE
GO:
 FILE ALREADY ASSIGNED.
 END - DFN
 17 SEP 74 INTRODUCTORY MANUAL NOW AVALABLE
 10 SEP 74 **** SIMPLE VERSION 3.4 PUT ON LINE
 15 JUL 74 LATEST UPDATE TO TEACH OR POOP

      HELLO, THIS IS SIMPLE(3.4)
      TYPE HELP, FOR MORE INFORMATION
      THE TIME IS  18 SEP 74  10.17.42.
      ? $    0.1  THE CALCULATOR MODE  ****************************
      ? 2↑3 + 5
        =          13
      ? 4*ATN(1)                                    $ THIS IS PI
        =   3.1415926535898
      ? 355/113                                     $ THIS IS CLOSE
        =   3.1415929203540
      ? 22/7                                        $ BUT THIS WILL DO
        =   3.1428571428571
      ? EXP(1)                                      $ WOULD YOU BELIEVE "E"

        =   2.7182818284590
      ? $    0.2 SOME COMMANDS                  ****************************
      ? TIME
       THE TIME IS  19 SEP 74  10.20.45.
      ? LOAD A 2 2
       LETS LOAD A               (ROW-WISE)
           2 ROWS,      2 COLUMNS - HOW MANY VALUES PER LINE
      ? 4
       OK,       4 PER LINE IT IS
      ? 13 20 5 8
       ALLRIGHT, WE HAVE LOADED A
      ? TEACH LDU

              LDU(E)  - DECOMPOSITION FUNCTION, L,D,AND U ARE
                        COMPUTED SUCH THAT L*D*U=E,
                        L IS A UNIT LOWER TRIANGULAR MATRIX
                        U IS A UNIT UPPER TRIANGULAR MATRIX
                        D IS A DIAGONAL MATRIX
      ? TEACH DIVIDE

             DIVIDE,A,<L>,<D>,<U> - A IS SUBDIVIDED INTO L, D,
             AND U (A=L+D+U),
             L CONTAINS THE SUBDIAGONAL ELEMENTS,
             D THE DIAGONAL, AND U THE SUPERDIAGONAL ELEMENTS
      ?        DC = LDU(A)        $ DECOMPOSE A AND STORE IN DC
```

```
? TEACH IDENT

      IDENT,A,N - AN N BY N IDENTITY MATRIX,A, IS
                  CREATED
? IDENT I 2
? DIVIDE DC L,D,U
? (L+I)*D*(I+U)                                    $ LET'S CHECK

   SCALED BY    1E+01
        1       2
   1    1.30   2.00
   2     .50    .80
? 1/A                                              $ A INVERSE

   SCALED BY    1E+00
        1       2
   1    2.00 -5.00
   2   -1.25  3.25
? DET(A)                                           $ DETERMINENT OF A
  =              4
? TEACH CONST

      CONST,A,NR,NC,C - AN NR BY NC MATRIX,A,IS CREATED
      EACH ELEMENT OF A = C (A REAL NUMBER)
? CONST X 4,4 1.0                                  $ CREATE X ( ALL ONES )

? $ LETS PRODUCE A TABLE OF RANDOM NUMBERS BETWEEN 0 AND 10
?       X = RND( 10*X )     $ 10*RND(X) IS EQUIVILENT
? X                                                $ DISPLAY THE TABLE

   SCALED BY    1E+00
        1       2       3       4
   1    1.20   6.04    6.32    6.40
   2     .48   8.67    4.44    7.31
   3    6.97   8.87    4.23    5.14
   4    2.27   4.69    8.75    8.99
? SUM(X)/16                                        $ CHECK THE AVERAGE
  =   5.6733489806522
? TIME
THE TIME IS  18 SEP 74  10.28.23.
? $     0.3  THE PROGRAMMING MODE     **************************
? 1:               SQR = .5*( SQR + X/SQR )
? 2:               I = I + 1
? 3:          I & SQR
?       X = A
?       SQR = A
? RUN 3 TIMES
MATRICES INCOMPATIBLE 10S          15X3
ERROR IN LINE     2.0000 RETURNING CONTROL TO USER
?       I = 0 $ I FORGOT TO INITIALIZE I
```

20

```
? RESUME         $ THIS WILL RETURN CONTROL TO THE PROGRAM
  =             1

   SCALED BY   1E+01
        1      2
   1    .70  1.00
   2    .25   .45
  =              2

   SCALED BY   1E+00
        1      2
   1   4.15   5.77
   2   1.44   2.71
  =              3

   SCALED BY   1E+00
        1      2
   1   3.17   4.26
   2   1.06   2.10
 ? DIGITS 9
 ? RUN 3 MORE
  =              4

   SCALED BY   1E+00
          1            2
   1    3.00453807   4.00708604
   2    1.00177151   2.00276656
  =              5

   SCALED BY   1E+00
          1            2
   1    3.00000363   4.00000566
   2    1.00000142   2.00000221
  =              6

   SCALED BY   1E+00
          1            2
   1    3.00000000   4.00000000
   2    1.00000000   2.00000000
 ? A - SQR*SQR                                  $ LETS CK OUR RESULT

   SCALED BY   1E-10
          1            2
   1    1.42051704 -4.51109372
   2     .55479177 -1.76044068
                                                $ NOT BAD
 ?
 ? KILL ALL                    $ DELETE THE CURRENT PROGRAM
 ? $ LETS DO THAT AGAIN WITH A LITTLE DIFFERENT APPROACH
 ? 10:   IF I<=LIMIT THEN I=I+1 & SQR=.5*(SQR+X/SQR) & GO TO 10
 ? 20:   "ITERATIONS" & I & "SQR OF X" & SQR
 ?       LIMIT = 5
 ?  5:                I = 0 & SQR = X              $ INITIALIZE
 ? SHOW
 5.0000:               I = 0 & SQR = X              $ INITIALIZE
10.0000:    IF I<=LIMIT THEN I=I+1 & SQR=.5*(SQR+X/SQR) & GO TO 10

20.0000:    "ITERATIONS" & I & "SQR OF X" & SQR
```

```
? RUN

 VARABLE UNDEFINED O
 ERROR IN LINE     5.0000 RETURNING CONTROL TO USER
?  5:              I = 0 & SQR = X          $ INITIALIZE
? $ THATS A 0(ZERO) NOT AN O(OH)
? RESUME
  ITERATIONS
  =            6
  SQR OF X

   SCALED BY    1E+00
          1           2
   1   3.00000000  4.00000000
   2   1.00000000  2.00000000
? LIST
 NAME        ROWS  COLUMNS
 A              2    2
 DC             2    2
 L              2    2
 D              2    2
 U              2    2
 X              2    2
 LIMIT          1    1
 I              1    1
 SQR            2    2
 MAX OF  137 WORDS USED OUT OF A POSSIBLE   229 LEAVING     92
? TIME
 THE TIME IS  18 SEP 74  10.41.47.
? STOP

                     REQUESTED     USED
 FIELD LENGTH          050000     047644      (OCTAL)
 MATRIX STORAGE           229        137      (DECIMAL)
CC:
?PRINT
GO:
 ** 1 COPIES OF OUT RELEASED TO THE "TH" PRINTER **
 CC:
LO
ACCOUNT-RUN   LN-MIN  LN-COST   TM-SEC  TM-COST
CEAB123-20        24    $0.16   17.670    $1.27
```

22

## 1.0 Introduction

SIMPLE is a conversational interpretive system for performing a multitude of matrix operations. Its use is made possible by the TAURUS time-sharing system through the use of a remote CRT (Cathode-Ray terminal) or TTY (teletype). It is an easy-to-use tool --- even for beginners who have no experience with computers or a programming language, and only a minimal introduction to matrix calculations.

SIMPLE is an effective mechanism to introduce students in formal classes and others to time-sharing capabilities. Concepts involving matrix algebra are reinforced by the ability to get quick results through direct interaction with the computer. Self-paced instruction through SIMPLE's TEACH commands is a unique and rewarding feature allowing the student to proceed at his own rate, while insuring that the necessary documentation is always available and up-to-date.

SIMPLE performs all the standard operations on matrices along with a few not-so-standard operations. General matrix algebraic expressions are acceptable --- including parentheticals --- providing the user with generality and simplicity heretofore unattainable. Such things as eigenvalues and eigenvectors, remote plotting, and a group of inquiry commands are also available for the more mature user.

For clarity, we will describe SIMPLE assuming we are in an interactive environment, but SIMPLE can also be used via batch operation. There are two modes of operation: the conversational mode and the programming mode. When idling (waiting for your next request), SIMPLE is ready to receive one of three types of request:

1) a Command
2) an Algebraic statement (an equation or an expression)
3) an Edit (Programming Mode)

types 1) and 2) are conversational (i.e., action is requested immediately), while 3) is a programming mode request (the editing action is requested immediately but the statement will not be executed until later).

## 2.0 Fundamental Elements

Every statement in SIMPLE is composed of some combination of 3 independent parts.

<#> : <S> $ <comment>

These 3 parts (the statement number, the primary syntax, and the commenting field) are separated by SIMPLE's 2 primary delimiters.

## 2.1 Delimiters

The : separates the statement number from the rest of the statement and identifies it as a Programming Mode statement.

The $ identifies the end of the primary syntax; what follows is free-field commentary for documentation purposes.

The & has 2 purposes. It allows us to stack expressions in the Expression form of the Algebraic class and simplifies outputting (e.g., A & B & A+B). In a similar capacity, in the Programming Mode & allows us to stack primary syntax within the same line (e.g., 1: I = I+1 & GΘ TΘ I).

The ⓢ is not a character but is used to stand for the general separator and represents a character or group of characters as defined below. ⓢ may be a comma (,), any number of blanks ($\geq 1$) , or a comma preceded or followed by any number of blanks.

The " always occurs in pairs and is used to enclose an alphanumeric string (see 3.3).

## 2.2 Constants

A numeric constant may be a signed numeral in integer or real form with up to 10 significant digits. Constants may be expressed in scientific notation by using "En" which stands for "$\times 10^n$" (e.g., 1.3E5 means $1.3 \times 10^5$).

## 2.3 Variables

Variables in SIMPLE start with an alphabetic character and may be followed with up to 9 additional alphanumeric characters. A variable may be either a scalar or an array. Arrays may have one (a column vector) or two dimensions (a matrix).

A subscripted variable (<var> [<e>] or <var> [<e1>, <e2>]) is a special case of a scalar variable and is legal anywhere a variable is allowed, e.g., A[I] = 3*B[x,I*(J+2)]. See Appendix C No. 2,7.

## 2.4 Relations

Relations may or may not be functions. In SIMPLE there are 3 basic relational symbols:

< > =

These respectively mean less than, greater than, and equal. Any paired combination of these 3 is allowed, and "or" is the implied connector (e.g., < = means < or =, = < means = or <, >< means > or < (i.e., not equal)). These may be used with relational expressions in a SIMPLE IF-THEN statement (see 5.4).

Functions may be either operators or non-operators (hereafter referred to as intrinsic functions).

### Arithmetic Operators

Since variables in SIMPLE may be scalars or arrays their complete definition is slightly more involved than normal (for the complete definition use TEACH (see 4.0)). It should be stated, while some are primarily scalar operators (↑) and some are primarily array operators (', [ ]), no matter what the variable type, the operation is well-defined (assuming compatability). See appendix C, No. 1, 10, 11. The hierarchy of operations is as follows:

(1) [ ] - subscript evaluation
(2) ( ) - innermost parentheses next
(3) ' - transposition
(4) *,/,↑ - multiplication, division, and exponentiation
(5) +,- - addition and subtraction (including negation)
(6) = - assignment

### Intrinsic Functions

As was the case with SIMPLE operators, some functions are primarily scalar oriented and some array oriented, but all are well-defined.

All intrinsic functions have a 3 character name combined with a parenthesis pair enclosing any legal SIMPLE expression. A list is on page iii (use TEACH for more information). See Appendix C, No. 7.

## 2.5 Expressions

Expressions in SIMPLE are somewhat analogous to algebraic expressions in mathematics, and are defined as follows. A constant, variable, or function is an expression. Furthermore, if x and y are expressions, then so are:

(x)  -x  x↑y  x*y  x/y  x+y  x-y  x'

Examples:

3.1  x  A[N]  3↑2  x+3*(A-SIN(x))

SQR( (x[1] - x[2]) ↑ 2 + (y[1] - y[2]) ↑ 2 )

Expressions are evaluated according to the higharchy of operations (2.4). When equal, they are processed from left to right.

## 4.0 The Command Class

The command class is second only to the calculator mode in simplicity.

< command > < parameter list >

or more specifically,

$$< \text{command} > \text{ⓢ}\ p_1\ \text{ⓢ}\ p_2\ \text{ⓢ} \cdots p_n$$

where ⓢ is our generalized separater (2.1) and $p_i$ represents a SIMPLE constant or variable ($n$ may be o , i.e., no parameter list).

The command class is composed of three groups. The most basic (HELP . . . CLEAR, see iii) are either informative (HELP, POOP, and TEACH) or general purpose (LIST, CLEAR, STØP, TIME, EJECT). Of these only TEACH has a parameter list.

TEACH ⓢ p

TEACH may be the most important command, for with it, the user may obtain information about all SIMPLE commands, functions (including operators), and programming syntax. Because of this, the user is referred to the list of commands on page iii, and instructed to use TEACH for more information. Note, TEACH with no parameters will TEACH everything.

The second group of commands ($C_p$, see iii) give the user the ability to manipulate program files (READ, SAVE, SHOW), edit files (KILL), and execute files (RUN, RESUME).

The third group contains a powerful assortment of matrix manipulation and numerical related commands (LOAD, DELETE, • • •, SØRT, • • •, EIGEN).

## 3.0 The Algebraic Class

The algebraic class of statements gives us the ability to manipulate variables in a mathematical sense and evaluate equations of the form $(v = e)$, and multiple expressions of the form $e_1$ & $e_2$ & . . . .

### 3.1 Equations (v = e)

v is either a regular or subscripted variable and e is a legal SIMPLE expression. A very general form of an equation is allowed. It may be a matrix equation, a scalar equation, or a combination as long as compatability requirements are met.

### 3.2 The "Calculator" Mode ($e_1$)

This can be thought of as a sepcial case of $v = e$, where we substitute "TTY" for v . In other words, after e is evaluated it is merely outputted (written to the output file if batch, displayed if interacting) as opposed to being stored in v . This is really just the degenerate form of the SIMPLE output mechanism below.

### 3.3 θutput (The "Expression" Mode $e_1$ & $e_2$ & . . .)

As described in 2.1 the delimiter & serves to separate expressions and alphanumeric strings (". . .") to be output.

Example: "A" & A & "B" & B & "A * B" & A * B

## 5.0 The Programming Mode (<#> : <s>)

This mode provides the user with the most powerful aspects of SIMPLE. It is through the programming mode that an internal program is created and/or modified. In connection with the program manipulation commands, it may be retrived, saved, displayed, deleted, or executed.

A statement in SIMPLE is of the programming mode if and only if it has a statement number followed by a colon (:). The syntax <s> which follows may be from the Algebraic class, the Command class (excluding $C_p$ , see page iii), or the Program Control Syntax (PCS, see 5.3). Statements may be compounded using the & .

Example: 1.3 : I = 1 & x = SIN(P) & GΘ TΘ 21

A SIMPLE internal program is executed by processing the statements in ascending order (unless diverted by a GΘ TΘ or a DΘ PART) until the end is reached or a PAUSE is encountered. See Appendix C, No. 5.

## 5.1 Statement Numbers

Every statement number has one of the following forms:

a.b .b a. or a where,

$1 \le a \le 9999$ and $0001 \le b \le 9999$

(i.e., $0 < \# < 10000$ and has at most 4 significant digits to the left or the right of the decimal point).

Examples: 1017.0302 .02 17. 100

## 5.2 Editing

Editing in SIMPLE is very straightforward. To insert a statement between statement a and c ($c > a$), simply preface the new statement with b , where $a < b < c$ . To replace statement number b , just preface the new statement with b .

## 5.3 Syntax

All program control syntactical entities (GΘ, TΘ, IF, THEN, DΘ, PART, FΘR, = , TO, BY, PAUSE) must be separated by blanks (a space).

PAUSE    PAUSE    <string>

When a PAUSE is encountered, control is returned to the user and <string> is output. (i.e., the RUN command iniates the execution of the internal program, and PAUSE terminates it (possible temporarily).

SIMPLE will automatically produce a PAUSE if a syntax or an execution error is encountered. The RESUME command enables the user to resume execution after the PAUSE.

GΘ TΘ    GO TO  <e>

When a GΘ TΘ is encountered <e> is evaluated and control transferred to the smallest statement number that is greater than or equal to <e> .

IF-THEN    IF $<e_1><\sim><e_2>$ THEN <S>

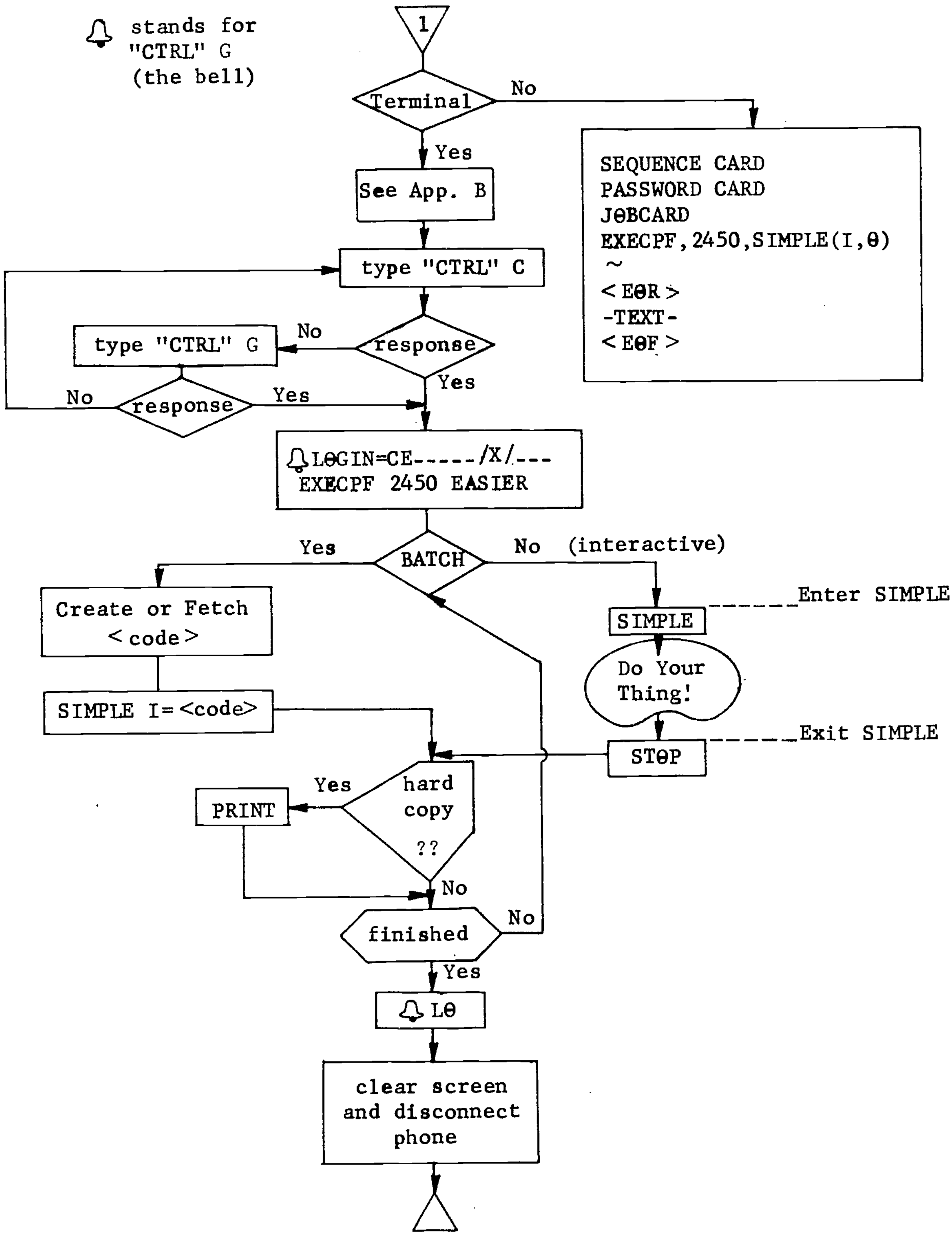
$\sim$ represents some combination of relationals (2.4). The expressions $<e_1>$ and $<e_2>$ are computed and then the relational expression $<e_1><\sim><e_2>$ is evaluated (either true or false). If true, <S> is processed. If false execution resumes with the next line.

DΘ PART    DΘ PART  N  FΘR  B = C  TO  D  BY  E

DΘ PART is basically a subroutine call although it can be used in much the same manner as a FΘRTRAN DΘ LΘΘP. B must be a variable (non-subscripted), while N , C , D , and E may be either constants, variables (regular or subscripted), or packed expressions (no embedded blanks). N is evaluated and truncated and then control is transferred to the group of statements that lie between N and N+1 (including N). (e.g., DΘ PART 4 would execute the group of statements with statement numbers from 4 to 4.9999).

Parameters may be deleted from the right giving us a total of 4 possible forms (use TEACH DΘ). See Appendix C, No. 4.

## 6.0 Using SIMPLE on TAURUS



The most general form of the SIMPLE control card macro is:

SIMPLE I = <fn> θ = <fn> C = <fn> P = <fn> T1 = <fn> T2 = <fn> FL = <#>

where I declares the input file, default = TTY

θ declares the output file, default = θUT

C declares the input copy file, default = IN

P declares the program copy file, default = PRθG

T1 declares the 1st data copy file, default = T1

T2 declares the 2nd data copy file, default = T2

FL declares the field length limit, default = 53000

SIMPLE I θ is equivalent to SIMPLE I = INPUT θ = θUTPUT which says that this is a standard batch run.

SIMPLE I = <fn1> θ = <fn2> . . .

is the form the user may use to batch a job from a remote terminal.

# APPENDIX A

## SIMPLE Character Set

| | |
|---|---|
| the English alphabet | A B · · · Z |
| the base ten digits | 0 1 2 3 · · · 9 |
| the relational symbols | = > < |
| the arithmetic operators | + - * / ↑ ' |
| the non-arithmetic operators | = ( ) [ ] |
| the delimiters | , . : $ & " |

APPENDIX B

# Accessing TAURUS[1]

The following assume the terminal is acoustically coupled. If the terminal is hardwired the user need only LØGIN.

A. Teletype

1) turn on-line
2) turn on
3) push TALK
4) dial the number
5) push DATA
6) hang up phone

B. Datapoint/Mini-T.E.C.

1) turn coupler on
2) put on remote
3) turn on
4) dial the number
5) place phone in coupler

[1] As of 1 September 1974 the number is 9-474-5011.

# APPENDIX C

## Notes

1) Parentheticals may be nested up to 9 deep.

2) Subscripts may not contain subscripted variables.

3) Variables and constants have at most 10 characters.

4) You cannot depart (DΘ - PART) to a depth $> 8$.

5) The maximum number of statements in an internal program is 30.

6) DELETE and CLEAR relate only to variable storage while, KILL <#> and KILL ALL relate to the internal program buffer.

7) Embedded blanks are not permitted between a function name and its parenthesis pair, nor between a variable and its subscript.

8) All intrinsic functions have 1 parameter and a 3 letter mnemonic.

9) All commands have 4 or more letters.

10) Beware: the hierarchy of $\uparrow$ is identical to $*$ and $/$.

$$a \uparrow b \uparrow c = (a \uparrow b) \uparrow c \ , \quad a * b \uparrow c = (a * b) \uparrow c \neq a * (b \uparrow c)$$

11) Beware: the hierarchy of -(negation) is the same as -(minus)

consequently, $-a \uparrow n \neq (-a) \uparrow n$ when $n$ is an even integer

(e.g. $-3 \uparrow 2 = -9 = -(3 \uparrow 2) \neq (-3) \uparrow 2 = 9$)